Advanced Function Presentation Consortium
Data Stream and Object Architectures

Font Object Content Architecture Reference

AFPC-0007-06

AFPConsortium
Advanced Function Presentation

Note:

Before using this information, read the information in "Notices" on page 203.

AFPC-0007-06
Seventh Edition (October 2015)

This edition applies to Font Object Content Architecture (FOCA). It is the first edition produced by the AFP Consortium™ (AFPC™) and replaces and makes obsolete the previous edition (S544-3285-05) published by IBM®. This edition remains current until a new edition is published. This publication also applies to any subsequent releases of Advanced Function Presentation™ (AFP™) products that use the FOCA architecture until otherwise indicated in a new edition.

Changes are indicated by a vertical bar to the left of the change. For a detailed list of changes, refer to "Changes in This Edition" on page xi.

Internet

Visit our home page: `http://www.afpcinc.org`

Preface

This book describes the functions and services associated with Font Object Content Architecture (FOCA). The FOCA architecture describes semantics and terminology for font objects used in a variety of environments. Syntax is provided for AFP system fonts used in the MO:DCA™ environment; refer to "AFP System Font Resource" on page 111. The syntax for printer fonts used in the IPDS™ environment is fully described in the *Intelligent Printer Data Stream™ Reference*.

Note: The FOCA architecture has been stabilized such that it can be fully used within AFP products and environments, but will not be extended. Many AFP products use either FOCA fonts, TrueType/OpenType fonts, or both.

Who Should Read This Book

This book describes the functions and services associated with the Font Object Content Architecture (FOCA). It is for systems programmers and other developers who need such information to develop or adapt a product or program to interoperate with other presentation products in an AFP environment.

This book is a reference, not a tutorial. It complements individual product publications, but does not describe product implementations of the architecture.

AFP data streams (MO:DCA and IPDS) provide the ability to manage and use a wide variety of font resources. These font resources can be classified into two major categories: *coded fonts* and *data-object fonts*. Coded fonts and their component parts are defined within the Font Object Content Architecture (FOCA) and are described in this book; data-object-fonts are defined elsewhere. A good description of data-object fonts and their component parts can be found in *How To Use TrueType and OpenType Fonts in an AFP System*.

AFP Consortium (AFPC)

The Advanced Function Presentation (AFP) architectures began as the strategic, general purpose document and information presentation architecture for the IBM Corporation. The first specifications and products go back to 1984. Although all of the components of the architecture have grown over the years, the major concepts of object-driven structures, print integrity, resource management, and support for high print speeds were built in from the start.

In the early twenty-first century, IBM saw the need to enable applications to create color output that is independent from the device used for printing and to preserve color consistency, quality, and fidelity of the printed material. This need resulted in the formation, in October 2004, of the AFP Color Consortium™ (AFPCC™). The goal was to extend the object architectures with support for full-color devices including support for comprehensive color management. The idea of doing this via a consortium consisting of the primary AFP architecture users was to build synergism with partners from across the relevant industries, such as hardware manufacturers that produce printers as well as software vendors of composition, work flow, viewer, and transform tools. Quickly more than 30 members came together in regular meetings and work group sessions to create the AFP Color Management Object Content Architecture™ (CMOCA™). A major milestone was reached by the AFP Color Consortium with the initial official release of the CMOCA specification in May 2006.

Since the cooperation between the members of the AFP Color Consortium turned out to be very effective and valuable, it was decided to broaden the scope of the consortium efforts and IBM soon announced its plans to open up the complete scope of the AFP architecture to the consortium. In June 2007, IBM's role as founding member of the consortium was transferred to the InfoPrint® Solutions Company, an IBM/Ricoh® joint venture. In February 2009, the consortium was incorporated under a new set of bylaws with tiered membership and shared governance resulting in the creation of a formal open standards body called the AFP Consortium (AFPC). Ownership of and responsibility for the AFP architectures was transferred at that time to the AFP Consortium.

Publication History

The FOCA Reference was first published by IBM in 1988 and has had several enhancements and updates since that time. The first six editions were published by IBM Corporation and the current edition was published by the AFP Consortium.

Note that the FOCA architecture has been stabilized such that it can be fully used within AFP products and environments, but will not be extended. Many AFP products use either FOCA fonts, TrueType/OpenType fonts, or both.

First Edition published by IBM Corporation

S544-3285-00 dated December 1988

Second Edition published by IBM Corporation

S544-3285-01 dated February 1990

Third Edition published by IBM Corporation

S544-3285-02 dated June 1993

This edition provides enhanced detail and clarifications:

- Information has been added to show the relationship between FOCA and the ISO® Font Information Interchange Standard (ISO/IEC 9541-1:1991). The concepts and parameters of FOCA can be mapped to, or transformed to, corresponding properties in the ISO font standard. Likewise, many of the concepts and properties defined by the ISO font standard can be mapped to, or transformed to, parameters of FOCA.
- Chapter 1 has been rewritten to introduce the family of IBM Presentation Architectures.
- Chapters 2–4 have been reorganized and edited to improve readability, arranging the information to allow the reader to move more freely from general introductory concepts to detailed architecture specifications.
- Chapter 5 has been expanded to include new FOCA parameters and to clarify how those parameters apply to different writing systems (e.g., top to bottom).
- Chapter 6 has been added to show the relationship between the font semantics defined in this document and the font syntax defined and interchanged by other architectures and products.
- Appendix A has been replaced, adding a reference summary of the relationship between IBM Font Architecture parameters and ISO/IEC 9541 Font Information Interchange standard properties.
- The Glossary has been extensively revised to include all terms applicable to the family of IBM Presentation Architectures.

Fourth Edition published by IBM Corporation

S544-3285-03 dated April 1996

This edition provides enhanced detail and the following major new additions:

- Syntax for AFP Host Fonts, previously contained in the AFP Host Font Data Stream Reference (IBM S544-3289-01), has been merged into chapter six.
- Outline Font support and associated syntax
- Miscellaneous technical clarifications and corrections (as marked)
- An appendix, which describes the pattern technologies supported by the FOCA architecture

Fifth Edition published by IBM Corporation

S544-3285-04 dated June 2000

This edition provides clarifications and additional detail to support the FOCA products that IBM has provided since 1996. Changes include:

- GCUID and UCS Presentation encoding scheme (basic Unicode support)
- 300 pel fixed metrics
- Metric Adjustment triplet for migrating from double-byte raster to outline fonts
- Fully Qualified Name (FQN) triplet for identifying and referencing fonts objects
- NOP structured field
- FND “Typeface Name” field changed to “Typeface Description”
- Typeface Name now allowed in FQN triplet on FND
- Default character support for outline fonts
- Triplet clarifications
- Space character clarifications
- Editorial clarifications

Sixth Edition published by IBM Corporation

S544-3285-05 dated June 2005

This edition provides clarifications and additional detailed description of the FOCA architecture. Changes include:

- Extended point-size range for relative metrics
- Variable-space-enable flag in the Code Page Control
- Pointsize terminology clarifications
- Relationship of the FOCA architecture to TrueType and OpenType fonts
- Unicode scalar values in code pages
- Improved syntax tables
- Editorial clarifications

How to Use This Book

This book is divided into seven chapters, with appendixes. Those readers who have little or no knowledge of fonts or font architecture concepts should read the introductory chapters first. Those readers who are experienced in using fonts in AFP implementations may wish to begin with "AFP System Font Resource" on page 111, and then use Chapter 5, "FOCA Parameters", on page 55 as a reference for parameter semantics. Those readers who are experienced in using fonts in other AFP product implementations may wish to begin with their product publications, and then use Chapter 5, "FOCA Parameters", on page 55 as a reference for parameter semantics.

- Chapter 1, "A Presentation Architecture Perspective", on page 1 introduces the Presentation Architecture framework that is covered in this book.
- Chapter 2, "Introduction to Fonts", on page 7 describes digitized fonts, text processing, font storage and accessing, font referencing, and FOCA font concepts.
- Chapter 3, "Referencing Fonts", on page 17 explains the relationship of fonts to the various processes in document production. Included are topics of font selection and substitution, font identification, and document fidelity.
- Chapter 4, "FOCA Overview", on page 33 explains, in more detail, FOCA font architecture concepts and character shape information.
- Chapter 5, "FOCA Parameters", on page 55 provides semantic descriptions of the parameters used in font resources, references, and queries.
- Chapter 6, "Font Interchange Formats", on page 111 provides information about the formats required for the interchange of font information.
- Chapter 7, "Compliance Requirements", on page 189 defines the requirements for compliance to the architecture.
- Appendix A, "AFP System Font Structured-Field and Triplet Summary", on page 191 provides tables of AFP system font data structures sorted by hexadecimal ID, with a page number reference to the full description of each data structure.
- Appendix B, "Mapping of ISO Parameters", on page 193 provides a summary cross-reference of all FOCA and ISO 9541 parameters.
- Appendix C, "Pattern Technology Information", on page 201 provides information about the various shape representation formats supported by FOCA.

The "Glossary" on page 205 defines those font terms used in this book, which might also be required by other presentation architectures.

Related Publications

Several other publications can help you understand the architecture concepts described in this book. AFP Consortium publications and a few other AFP publications are available on the AFP Consortium web site, www.afpcinc.org.

Table 1. AFP Consortium Architecture References

AFP Architecture Publication	Book Identification
AFP Programming Guide and Line Data Reference	S544-3884 (IBM)
Bar Code Object Content Architecture™ Reference	AFPC-0005
Color Management Object Content Architecture Reference	AFPC-0006
Font Object Content Architecture Reference	AFPC-0007
Graphics Object Content Architecture for Advanced Function Presentation Reference	AFPC-0008
Image Object Content Architecture Reference	AFPC-0003
Intelligent Printer Data Stream Reference	AFPC-0001
Metadata Object Content Architecture Reference	AFPC-0013
Mixed Object Document Content Architecture™ (MO:DCA) Reference	AFPC-0004
Presentation Text Object Content Architecture Reference	SC31-6803 (IBM)

Table 2. Additional AFP Consortium Documentation

AFPC Publication	Book Identification
AFP Color Management Architecture™ (ACMA™)	AFPC-0015
AFPC Company Abbreviation Registry	AFPC-0012
AFPC Font Typeface Registry	AFPC-0016
BCOCA™ Frequently Asked Questions	AFPC-0011
MO:DCA-L:The OS/2 PM Metafile (.met) Format	AFPC-0014
Presentation Object Subsets for AFP	AFPC-0002
Recommended IPDS Values for Object Container Versions	AFPC-0017

Table 3. AFP Font-Related Documentation

Publication	Book Identification
Character Data Representation Architecture Reference and Registry; please refer to the online version for the most current information: http://www-306.ibm.com/software/globalization/cdra/index.jsp	SC09-2190 (IBM)
Font Summary for AFP Font Collection	S544-5633 (IBM)
How To Use TrueType and OpenType Fonts in an AFP System	G544-5876 (IBM)
Technical Reference for Code Pages	S544-3802 (IBM)

Table 4. UP³I™ Architecture Documentation

Table 4 UP³I™ Architecture Documentation (cont'd.)

UP³I Publication	Book Identification
Universal Printer Pre- and Post-Processing Interface (UP³I) Specification	Available at www.afpcinc.org

Changes in This Edition

Changes between this edition and the previous edition are marked in green by a vertical bar (|) in the left margin.

This edition provides clarifications and additional detailed description of the FOCA architecture. Changes include:

- Editorial clarifications
- IBM product-specific information has been removed and the term AFP or FOCA used in place of IBM where appropriate
- Information about the AFP Consortium
- Style changes to make this book more consistent with the other AFPC publications

Note: The FOCA architecture has been stabilized such that it can be fully used within AFP products and environments, but will not be extended. Many AFP products use either FOCA fonts, TrueType/OpenType fonts, or both.

Contents

Figures

Tables

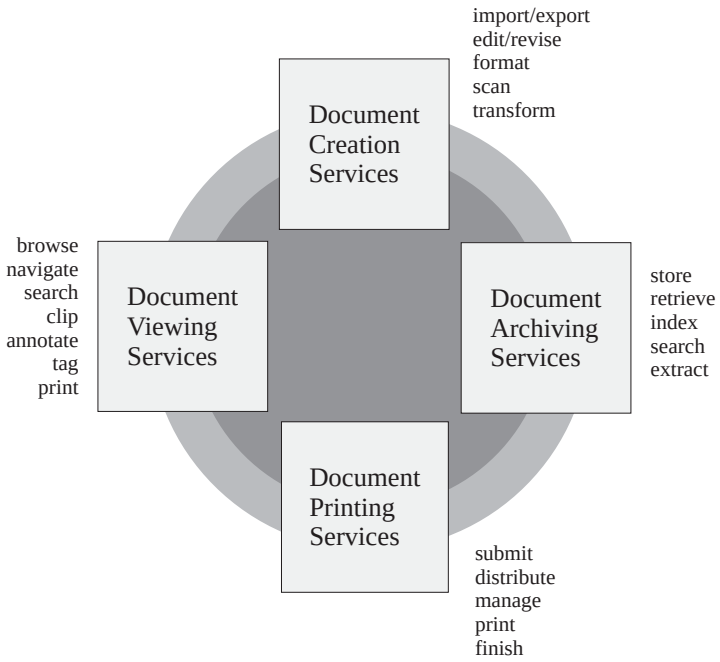
Chapter 1. A Presentation Architecture Perspective

This chapter provides a brief overview of Presentation Architecture.

The Presentation Environment

Figure 1 shows today's presentation environment.

Figure 1. Presentation Environment. The environment is a coordinated set of services architected to meet the presentation needs of today's applications.



The ability to create, store, retrieve, view, and print data in presentation formats friendly to people is a key requirement in almost every application of computers and information processing. This requirement is becoming increasingly difficult to meet because of the number of applications, servers, and devices that must interoperate to satisfy today's presentation needs.

The solution is a presentation architecture base that is both robust and open ended, and easily adapted to accommodate the growing needs of the open system environment. AFP presentation architectures provide that base by defining interchange formats for data streams and objects that enable applications, services, and devices to communicate with one another to perform presentation functions. These presentation functions might be part of an integrated system solution or they might be totally separated from one another in time and space. AFP presentation architectures provide structures that support object-oriented models and client/server environments.

AFP presentation architectures define interchange formats that are system independent and are independent of any particular format used for physically transmitting or storing data. Where appropriate, AFP presentation architectures use industry and international standards, such as the ITU-TSS (formerly known as CCITT) facsimile standards for compressed image data.

Architecture Components

AFP presentation architectures provide the means for representing documents in a data format that is independent of the methods used to capture or create them. Documents can contain combinations of text, image, graphics, and bar code objects in device-independent and resolution-independent formats. Documents can contain fonts, overlays, and other resource objects required at presentation time to present the data properly. Finally, documents can contain resource objects, such as a document index and tagging elements supporting the search and navigation of document data, for a variety of application purposes.

The presentation architecture components are divided into two major categories: *data streams* and *objects*.

Data Streams

A *data stream* is a continuous ordered stream of data elements and objects conforming to a given format. Application programs can generate data streams destined for a presentation service, archive library, presentation device, or another application program. The strategic presentation data stream architectures are:

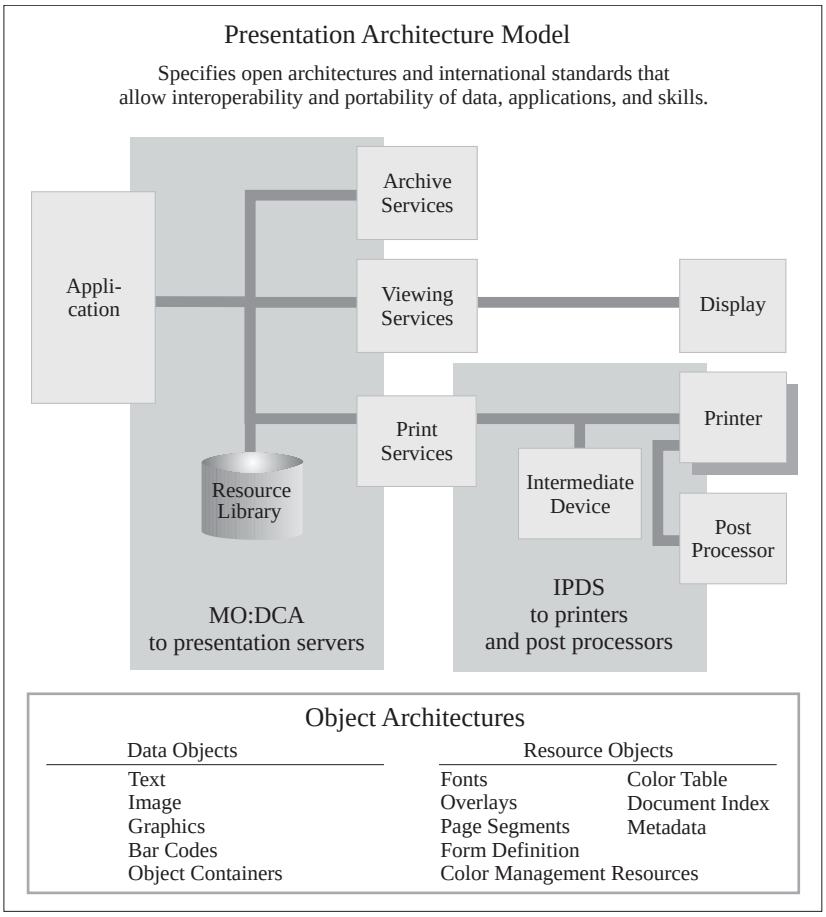
- *Mixed Object Document Content Architecture (MO:DCA)*
- *Intelligent Printer Data Stream (IPDS) Architecture*.

The MO:DCA architecture defines the data stream used by applications to describe documents and object envelopes for interchange with other applications and application services. Documents defined in the MO:DCA format can be archived in a database, then later retrieved, viewed, annotated, and printed in local or distributed systems environments. Presentation fidelity is accommodated by including resource objects in the documents that reference them.

The IPDS architecture defines the data stream used by print server programs and device drivers to manage all-points-addressable page printing on a full spectrum of devices from low-end workstation and local area network-attached (LAN-attached) printers to high-speed, high-volume page printers for production jobs, shared printing, and mailroom applications. The same object content architectures carried in a MO:DCA data stream can be carried in an IPDS data stream to be interpreted and presented by microcode executing in printer hardware. The IPDS architecture defines bidirectional command protocols for query, resource management, and error recovery. The IPDS architecture also provides interfaces for document finishing operations provided by pre-processing and post-processing devices attached to IPDS printers.

Figure 2 shows a system model relating MO:DCA and IPDS data streams to the presentation environment previously described. Also shown in the model are the object content architectures that apply to all levels of presentation processing in a system.

Figure 2. Presentation Model. This diagram shows the major components in a presentation system and their use of data stream and object architectures.



Objects

Documents can be made up of different kinds of data, such as text, graphics, image, and bar code. *Object content architectures* describe the structure and content of each type of data format that can exist in a document or appear in a data stream. Objects can be either *data objects* or *resource objects*.

A data object contains a single type of presentation data, that is, presentation text, vector graphics, raster image, or bar codes, and all of the controls required to present the data.

A resource object is a collection of presentation instructions and data. These objects are referenced by name in the presentation data stream and can be stored in system libraries so that multiple applications and the print server can use them.

All object content architectures (OCAs) are totally self-describing and independently defined. When multiple objects are composed on a page, they exist as peer objects, that can be individually positioned and manipulated to meet the needs of the presentation application.

The AFPC-defined object content architectures are:

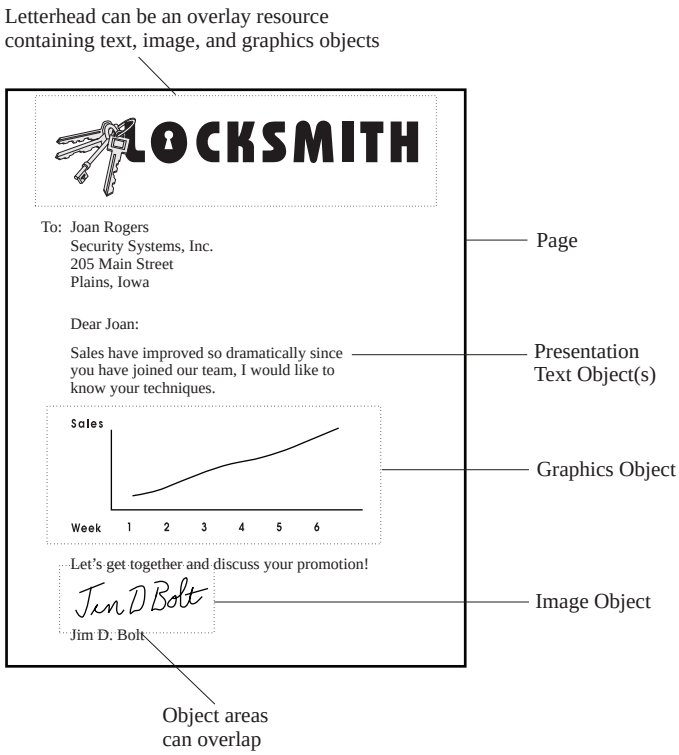
- *Presentation Text Object Content Architecture (PTOCA):* A data architecture for describing text objects that have been formatted for all-points-addressable presentations. Specifications of fonts, text color, and other visual attributes are included in the architecture definition.
- *Image Object Content Architecture (IOCA):* A data architecture for describing resolution-independent image objects captured from a number of different sources. Specifications of recording formats, data compression, color, and grayscale encoding are included in the architecture definition.
- *Graphics Object Content Architecture for Advanced Function Presentation (AFP GOCA):* A version of GOCA that is used in Advanced Function Presentation (AFP) environments. GOCA is a data architecture for describing vector graphics picture objects and line art drawings for a variety of applications. Specification of drawing primitives, such as lines, arcs, areas, and their visual attributes, are included in the architecture definition.
- *Bar Code Object Content Architecture (BCOCA):* A data architecture for describing bar code objects, using a number of different symbologies. Specification of the data to be encoded and the symbology attributes to be used are included in the architecture definition.
- *Font Object Content Architecture (FOCA):* A resource architecture for describing the structure and content of fonts referenced by presentation data objects in the document.
- *Color Management Object Content Architecture (CMOCA):* A resource architecture used to carry the color management information required to render presentation data.
- *Metadata Object Content Architecture (MOCA):* A resource architecture used to carry metadata in an AFP environment.

The MO:DCA and IPDS architectures also support data objects that are not defined by AFPC object content architectures. Examples of such objects are Tag Image File Format (TIFF), Encapsulated PostScript® (EPS), and Portable Document Format (PDF). Such objects can be carried in a MO:DCA envelope called an *object container*, or they can be referenced without being enveloped in MO:DCA structures.

In addition to object content architectures, the MO:DCA architecture defines envelope architectures for objects of common value in the presentation environment. Examples of these are *Form Definition* resource objects for managing the production of pages on the physical media, *overlay* resource objects that accommodate electronic storage of forms data, and *index* resource objects that support indexing and tagging of pages in a document.

Figure 3 on page 5 shows an example of an all-points-addressable page composed of multiple presentation objects.

Figure 3. Presentation Page. This is an example of a mixed-object page that can be composed in a device-independent MO:DCA format and can be printed on an IPDS printer.



Chapter 2. Introduction to Fonts

This chapter presents introductory information about fonts, digitized font structures, and how digitized fonts are used in information processing. The information in this chapter will aid the reader in understanding the Font Object Content Architecture (FOCA), but is not a required component of the architecture.

Fonts

A *font* is a set of graphic characters with similar design characteristics; that is, a font is a designer's concept of how a set of graphic characters should appear. *Graphic characters* (*glyphs* is the term used by the ISO/IEC 9541 *Font Information Interchange* standard) are letters, numerals, punctuation marks, ideograms, or symbols that appear in text.

Historically, a font was a collection of lead slugs containing the raised images of the characters. However, in electronic digital processing, the character images are digitized by transforming them into data or algorithms that are stored in a computer system. The character shape data is then used to display the images in the form of small dots on a presentation surface, for example, as picture elements (pels) on a display screen or as dots on a piece of paper.

Figure 4 shows how the pels or dots form a pattern that can be interpreted as a graphic character; it shows the letter *h* from a typeface family with the design characteristic of serifs. Such a graphic character could be presented on a presentation surface through any of several available presentation technologies, for example, wire matrix impact, laser fusion, ink jet transfer, or display phosphor illumination, all of which are able to approximate the form, style, and appearance intended by the font designer. The data or algorithm used to represent the character shape in digitized form on a computer system can be quite different from the pels or dots that appear on the presentation surface. The character shapes can be digitized as a matrix of binary bits, as a series of vectors, or as a set of second or third order polynomial equations.

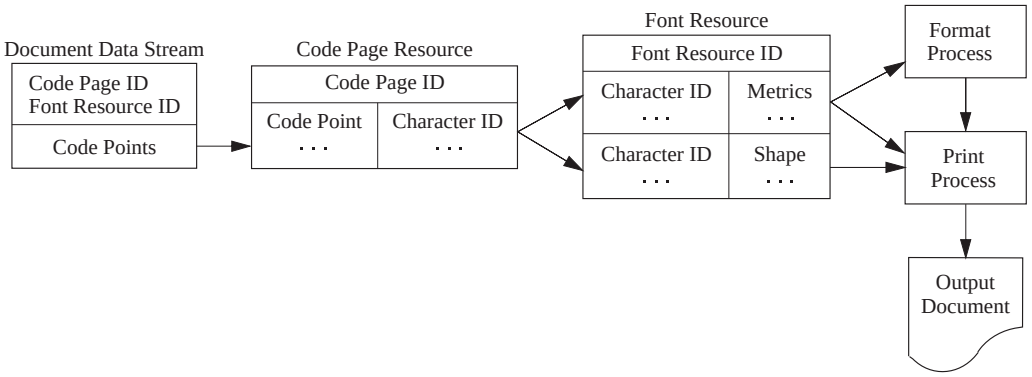
Figure 4. Representation of a Graphic Character

A digitized font contains not only the character shape information, but all of the information needed by an information processing system to format a character string and render the character shapes with a given presentation technology. Digitized fonts contain descriptive information used to identify the specific font resource, metric information used to position the character shapes, and the shape information used to render the character shapes.

In theory, a font could be defined to be the universal set of all the world's characters having the same design characteristics. In practice however, a digitized font resource contains a bounded set of characters having the same design characteristics. Further, multiple digitized font resources can exist having the same design characteristics, but each containing a different bounded set of characters. The set of characters contained in a font is determined by the font provider, depending on the language requirements for document processing.

Each graphic character in a font has a unique name, which is called the *graphic character identifier*. Each graphic character in a document data stream has a unique code, which is called a *code point*. A *code page* is a table that associates graphic character identifiers with code points (see Figure 5).

Figure 5. Relationship of Code Points to Graphic Characters



When documents are formatted and presented, the graphic characters in the document, which are usually encoded as code points of one or two bytes for each graphic character, are associated with the character information in a font through the code page. For an example, presentation of the word “the” requires the character shapes for the graphic characters *t, h,* and *e*. Each code point in the code page maps to a character identifier, which points to the font information, including metrics and shape. One code page is often sufficient to represent all the graphic characters in the document, although multiple code pages can be used.

Fonts are either *monospaced*, in which the horizontal space occupied by different characters is the same, or *typographic*, that is, *proportionally spaced fonts*, in which the character widths vary.

In typographic fonts, the font designer has adjusted the distance from one character to another to improve the visual flow of text by eliminating excess space. Adjusting space this way improves the readability and the appearance of a document.

For comparison, the letter *i* and the letter *m* are printed below 15 times in both a monospaced font and a typographic font. Note that in the monospaced font, both strings of characters are the same length, but in the typographic font, the string of *m*'s is longer than the string of *i*'s.

Monospaced iiiiiiiiiiiiiii
mmmmmmmmmmmmmmm

Typographic iiiiiiiiiiiiii
mmmmmmmmmmmmmm

Font Resources

The term *font resource* refers to the collection of code pages and other font information such as FOCA font parameters that are stored in the font library and used by data processing systems. The parameters define the attributes for a particular font. A parameter includes a name by which it can be referenced and a value that defines the attribute. See Chapter 5, “FOCA Parameters”, on page 55 for information about FOCA parameters, including their names, acceptable values, and usages.

Font resources normally reside in system storage; and the aggregate of stored font information is often called the *font library*. Font resources can also be resident in a presentation device or control unit. A font resource can be all or only a part of the information pertaining to a particular font.

Font References

A *font reference* is information that identifies a particular font resource. A font reference is embedded either in a document data stream or in an application program, which processes a document data stream and requests the use of a font resource. For example, a font reference could be the name of a font file or a list of font parameters. When font references are carried in an architected data stream or a program, they use the format, that is, the syntax, required by the document content architecture.

Note: Font references should not be confused with *data objects*, which use their own syntax when carried in architected data streams and use the data stream only as an envelope.

FOCA supports using font references by defining a precise set of parameters that specify a font resource or describe the font resource attributes. Each implementing product chooses the set of font resources it will support, and the set of characters contained in each of those font resources. For consistency when interchanging and presenting documents, all receiving sites, that is, processing applications and presentation devices, must have access to the same or equivalent font resources.

AFP Font Resources

An AFP font resource consists of font information for a specific font family in one or more styles and sizes originally developed by IBM. When FOCA font parameters are used, they support a broad range of IBM encoding schemes as well as the specific encoding supported by AFP products. For example, AFP products support a variety of encoding schemes for code pages and specific codes, which meet the processing requirements of specific geographic, language, or application environments. These encoding schemes identify the graphic characters of a document by their meaning, for instance, open parenthesis, or by their shape, for instance, left parenthesis.

The graphic characters in a font resource can be defined to support multiple code page encoding schemes, or they can correspond precisely to the code points of a specific code page encoding scheme. AFP font resources contain a large amount of font information, which allow products to access and process the resources in a variety of application environments.

The graphic characters in a font can be mapped to any of the various types of code page encoding schemes. However, the set of graphic characters defined within a font must be equal to or greater than the set of graphic characters defined in the code page. This ensures that all graphic characters in the encoding scheme are supported by the font and that font information is available at document formatting or document presentation time.

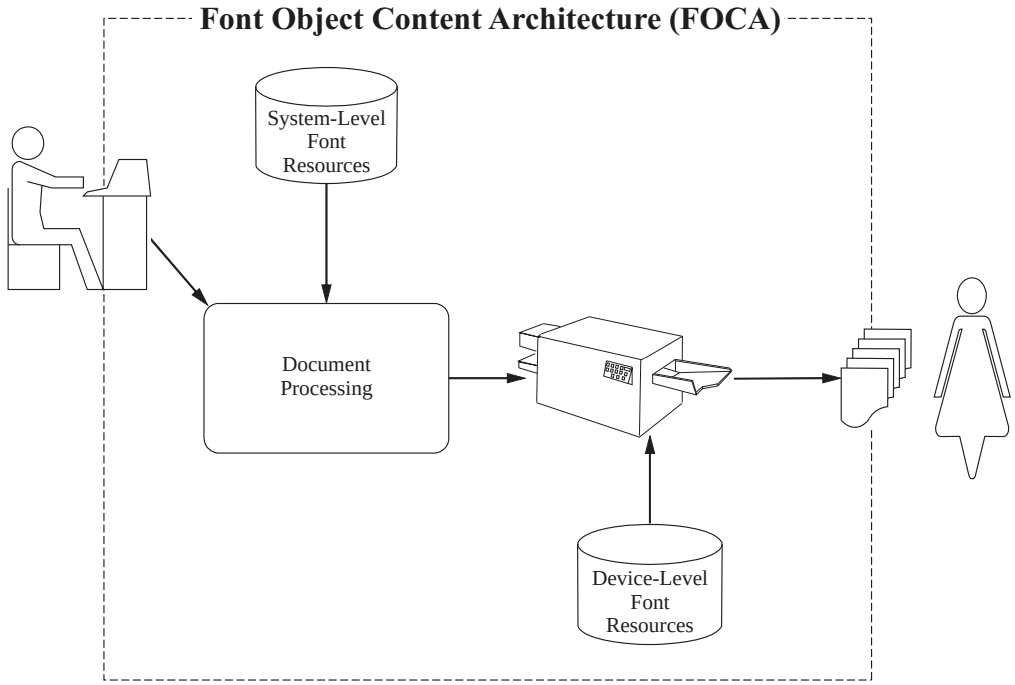
The following are the two types of AFP font resources:

- A *system-level font resource* is a font from which document processing applications can obtain formatting information that is resolution independent and from which device service applications can obtain device specific presentation information. A system-level font resource normally resides within a font library, which provides a common source of font information for all applications or devices.
- A *device-level font resource* is a device-specific font from which the presentation device, or family of presentation devices, can obtain the font information required to present the character images. A device-level font resource normally resides within a ROM cartridge[1], a loadable file on a disk or on a tape, or a loadable file from a host system.

1. A cartridge containing read-only memory

Figure 6 shows that FOCA supports all transactions as the user selects graphic characters and FOCA products draw on both levels of resources to produce the output.

Figure 6. FOCA Information Processing Environment



Digitized Font Structures

Digitized fonts are grouped (structured) according to the different function of the information that they provide. The following list identifies the parameter groups, the type of information provided, the function of the information, and the number of times, as a general practice, the information would occur in a typical digitized font resource:

Font-description parameters
: Provide font-descriptive information for referencing, that is, identifying and describing, fonts. Defined once per font resource.

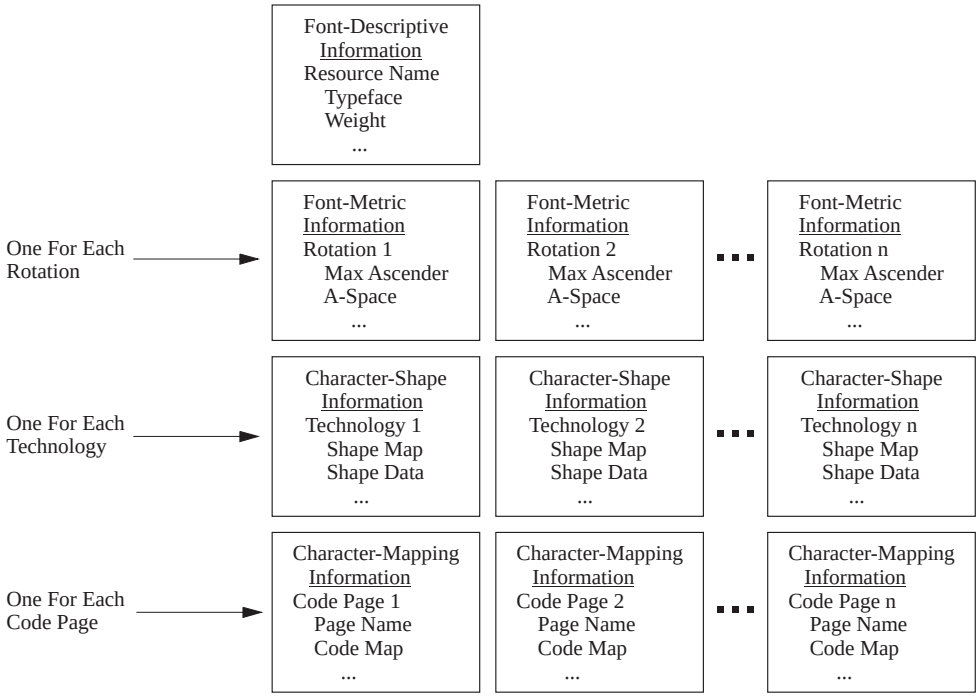
Font-metric parameters
: Provide font-metric information that states measurements for positioning the font and each character in the font. Repeated for each rotation supported.

Character-shape parameters
: Provide character-shape information for forming images of the character shapes on a presentation surface. Repeated for each shape representation technology supported.

Character-mapping parameters
: Provide character-mapping information for associating character identifiers with code points on code pages. Repeated for each code page supported.

Figure 7 shows an example of the organization of parameters in the library. The sections following Figure 7 describe each type of font information.

Figure 7. Font Information in a Library



Font-Descriptive Information

Font-descriptive information identifies and describes a font. The information includes, for example, resource name, family name, typeface name, weight, and supported sizes. The identification of a font resource for use by a system processor requires some or all of this information in the library to ensure access to the correct font.

Often, the originator of a document can identify a specific font resource by simply stating the name. But, in large systems that have numerous fonts and in distributed networks where available fonts are not known, listing each desired parameter and letting the system locate a resource that corresponds to the description might provide more accurate font selection. FOCA provides the common parameter definitions necessary to match descriptive information in a font reference to the definitions in a font resource.

Font-descriptive information should be grouped to identify and describe the font resource.

See "Font-Description Parameters" on page 56 for a description of each FOCA parameter that carries font-descriptive information.

Font-Metric Information

Font-metric information contains *metrics*, which is measurement information that defines the height, width, and space for a font or for each character in the font. Font-metric information also includes character-metric information. For example, a font resource might contain the information for the averages or maximums for all of the graphic characters as well as the measurements for each character. This information determines where a character will appear in a presentation space.

Furthermore, the presentation of graphic characters in a top-to-bottom writing mode requires different metric information than for left-to-right writing mode. Therefore, it is necessary to provide multiple groups of metric information for the various writing modes supported by the font resource. The font-metric information in FOCA

is grouped by character rotation (see “Character Rotation” on page 47 for the relationship between character rotation and non-Latin writing systems). The metrics can be expressed in the pel resolution of the presentation device that is used to present the font information or in a form that is resolution independent and applicable to all presentation devices.

Font-metric information must be defined for each rotation supported by the font resource. For good performance, the metric information should be grouped within a font resource.

See “Font-Metric Parameters” on page 74 for a description of each FOCA parameter that carries font-metric information and “Character-Metric Parameters” on page 91 for each FOCA parameter that carries character-metric information.

Character-Shape Information

Character-shape information enables the presentation device to create the image of the graphic character. The representation of character shapes in a font resource can be repeated for each supported shape representation technology (bitmaps, vectors, and conic sections). In theory, multiple device-specific representation technologies can be supported by a single font resource (for example, a font resource might include shape data for 240 pel, 300 pel and 600 pel bitmap representations), or a representation technology can be transformed from one to another (for example, vectors might be converted to 240-pel and 300-pel bitmap representations). In practice, most font resources contain the character shape information for a single representation technology.

Character-shape information can be defined once for all rotations supported by the font-metric information, or could be repeated in different font resources for each rotation. For good performance, the shape information should be defined once in a single font resource for all supported rotations.

See “Character-Shape Parameters” on page 96 for a description of each FOCA parameter that carries character-shape information.

Character-Mapping Information

Character-mapping information provides for the association of code points in a document to the appropriate graphic characters in a font. Character mapping requires knowledge about the techniques of document encoding, the identification of font characters, and the mapping of document character codes to font character identifiers.

The characters in a font resource can be defined to support the character content of one or more code pages. Using code pages minimizes processor storage and maximizes the document processing efficiency. That is, when the characters in a font resource are defined independently of the code page intended for support, many different documents, encoded with different code pages, can be presented using a single font resource.

Either a font resource contains the character identifiers and code points for mapping to each supported code page, or the code page definitions are stored separately from the font resource. If the first case is true and the font resource includes code page mappings, those mappings should be stored independently from the font-metric information and character-shape information (for each technology), except for character identifiers. The character-mapping information should be available for each supported code page.

See “Character-Mapping Parameters” on page 103 for a description of each FOCA parameter that carries character-mapping information.

Using Digitized Fonts

Digitized fonts are designed to be used by an application program that processes the font resource and creates output that the user can read when presented by a presentation device. The font resource must be produced and stored in the system before an application program can access and process it. The following sections describe the relationship of a font architecture to font production, font storage, and font processing.

Font Production

Font production occurs under a variety of circumstances, for example:

- An organization that specializes in font development using sophisticated design tools creates a new font family.
- An organization creates a special font for a collection of symbols to distribute to its branch offices.
- An individual modifies a few graphic characters in an existing font to make a company slogan fit on one line of a wallet-sized card.
- A developer of computers or text systems adapts an existing font to meet a customer's requirement for better performance.
- A computer assembles some font components as they are needed to create a document that includes both text and graphics.

Regardless of who or what initiates the font resource or its intended purpose, the font designer:

- Designs character shapes
- Converts the character shapes to a digital format and includes information for the presentation technology, such as bitmaps, vectors, or conic sections
- Defines the parameter values, such as height, width, and escapement point, for each character shape
- Assigns appropriate descriptive and identifying information, such as a graphic character identifier, to each character shape
- Creates any other information required by the application program

FOCA supports font production by providing definitions of the font parameters and by describing how they are used.

Font Storage

Font storage requires a method of addressing two factors:

Data availability
: Is the descriptive and measurement information available both to the applications that generate and format documents as well as to the devices that present the output?

Data ownership
: Is permission required to share or distribute this font information?

To create font resources that are more accessible for application use, the data can be stored as logical units rather than as a single, comprehensive font file. Logical units permit each system component or application program to access only those font elements they require, and might have the additional benefit of minimizing costly license payments that apply to other logical elements that the application doesn't require.

For example, a formatting application needs the font metric information that is available free to any using application, but does not require the licensed shape information that is required by a presentation device. many require font-metric information in addition. The following are some uses of font information as logical blocks:

- Network-distributed libraries that have database access communicate over the network.
- Host-system libraries that have shared data access or a resource-management program interface to any attached device or workstation application.
- Workstation-resident libraries or collections of files shared by users run in that or another workstation.
- Data compiled or linked to a certain editor has access only to the modules within that application program.
- Data resident within and available only to a specific presentation device is made available to the device by down-line loading or by exchangeable read-only storage.

FOCA supports font storage by precisely defining the set of font parameters required in each AFP environment so applications can implement FOCA by using only those logical structures or elements they require.

Font Processing

Font processing is the set of tasks performed when an application program accesses and uses a font resource. It includes three primary tasks:

Editing Creating and then modifying a data stream called a *revisable document*.

Formatting Using the revisable document to create a data stream, such as MO:DCA, called a *presentation document*.

Presenting Using the presentation document to create an *output document* that humans can read, such as a printed page.

Some systems require the user to request formatting as a separate step after editing, while other applications, such as a WYSIWYG[2] editor, combine these tasks into what appears to be a single process. However, without a command from the user, WYSIWYG systems also perform each task individually before presenting the results.

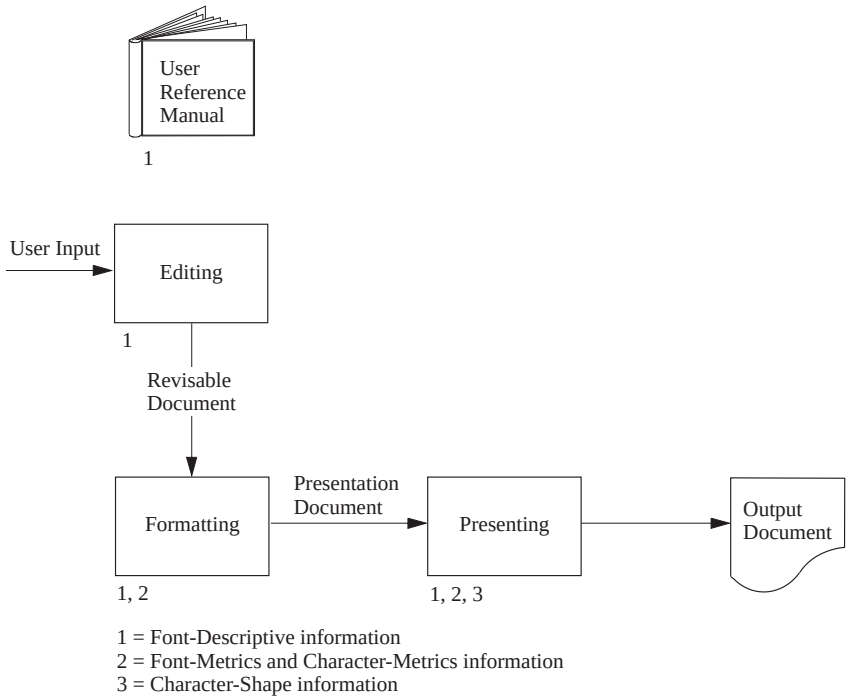
In the context of FOCA, font processing is accomplished by a *text processing system*, which is a type of editor or processing system that is the collection of hardware devices and software or firmware programs required to generate, modify, display, and print text on a presentation surface. The following are examples of text processing systems and components:

- Personal computers with a character display and an all-points-addressable (APA), dot-matrix printer
- Office workstations with a WYSIWYG editor, display, and an APA laser printer
- Graphic workstations with a high-resolution color display and software applications for rotating and scaling text to annotate figures
- Publishing workstations with preview devices that permit low-quality, fast-output preview of output and with software applications for page layout and composition
- Workstations in a network with distributed storage and distributed print systems

2. WYSIWYG is an acronym for “What you see is what you get”.

A processing system uses different types of font resource information when performing each of the three tasks. Figure 8 shows the tasks and indicates which type of information is used for each during font processing.

Figure 8. Font Resource Requirements for Font Processing Tasks



Editing

Editing is the task of creating or modifying the data stream that is the source for producing a document. Editing is performed by an application program or editor. The editor processes the font references, which must be consistent with the syntax specified by the document content architecture. The references use the FOCA parameters that provide font-descriptive information, that is, the references identify or describe the font. The references are also used to perform character mapping.

Formatting

Formatting is the task of determining where information is to appear in a presentation space. The formatter uses the FOCA parameters that contain font-metric information and character-metric information and position the graphic characters on the presentation surface.

The parameters include information about line and page breaks and how text flows around graphic and image objects in the document. Fonts provide only part of the information needed for character positioning. The formatter not only uses font parameters, but it also uses information contained in the document, such as paragraph breaks and other formatting instructions.

FOCA defines metrics for all characters in a font. If a development team creates or acquires a font resource for a presentation product, all formatters supported for that product must have access to the metrics for each character in that font.

Presenting

Presenting is the task of transforming the formatted information to a visible form on a presentation surface. The FOCA parameters that support presenting the document use character-shape information that images the character at its correct position.

The task of presenting creates the character shapes on a presentation surface at the position determined by the formatter. The presentation is performed by a hardware device, but it might be supported by software and hardware processes that provide the required position and shape information. The character-shape information is used only by the presentation device. Character-metric information used when presenting the font must correspond to the character-metric information used by the formatter.

FOCA supports presenting character shapes by providing support for device-specific techniques of character representation. FOCA also permits font producers and product implementers to make use of more generic representation technologies.

Font Processing Summary

The editing, formatting, and presenting tasks all use the FOCA font-descriptive information. During editing, the editor identifies (references) the font resource; during formatting, the formatter uses that font resource to locate and position the graphic characters, and during presenting, the presentation device uses that font resource to create the character shape with the proper position and metric information. The references can specify a specific font resource name or provide descriptive information about the font.

All three tasks are not necessarily performed on a single or integrated system. A workstation used for editing might be remote or detached from the system where formatting occurs. A document that is distributed for remote presentation might be formatted by the sender without knowing the devices or resources available for presentation. Many different document content architectures, document interchange data streams, and device data streams can be involved. In all cases, fonts must be referenced, graphic characters must be positioned, and character shapes must be presented. FOCA defines the necessary common and consistent font information that is needed for font processing.

Chapter 3. Referencing Fonts

When you create a document, you specify the font or fonts to be used whenever the document is presented. The application programs that process the document use the font references in the document to locate the required font resources for document formatting and presentation. System and presentation device libraries provide font resources, which have assigned names that must be associated with the font references contained in a document. The association of font references to font resources is not necessarily a one-to-one association based only on the name. The principal method of referencing fonts is by listing the applicable font-description parameters.

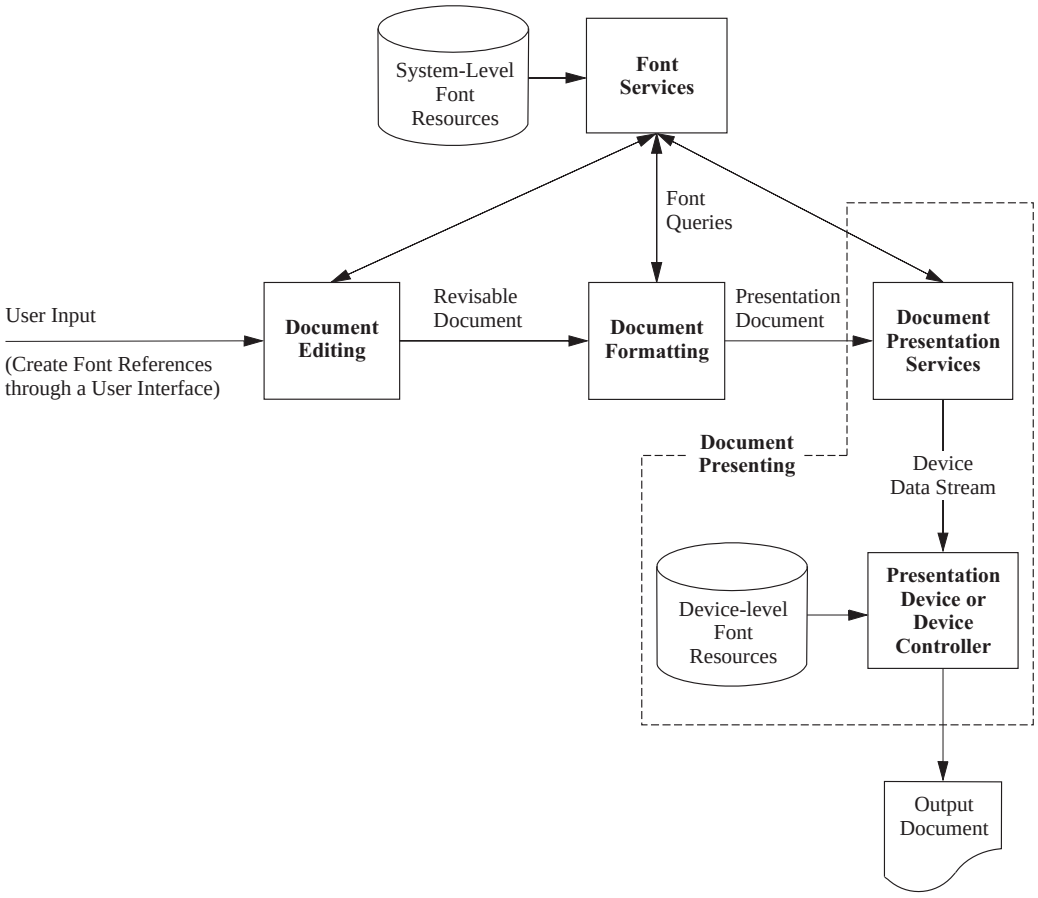
To enable text processing systems to locate and use the appropriate font resources, the font references must be made in a clear, consistent manner across all the systems, applications, and data streams that use the font resources. Then, if a document containing the font references is sent to other locations, the processors at those receiving locations can use the distinguishing characteristics or attributes to find the required font resource (perhaps filed under a different name), or to select an appropriate alternative. The following sections of this chapter describe:

- Understanding the font reference model
- Identifying fonts
- Selecting and substituting fonts

Understanding the Font Reference Model

Many components of a processing system interact to produce the final presentation of a document. The actual flow of the data stream through the system depends on both the configuration of the specific system and the intent or design of the user. Figure 9 illustrates the general flow of a document from creation to presentation. The figure assumes that the system, including the font resources, are in place when the user initiates a document. The sections following Figure 9 provide more detail for some of the separate transactions within the figure.

Figure 9. Font Reference Model General Flow



User Input

A user interacts with a text or graphics editor to produce a document that is in a revisable form and contains references to the desired fonts, as shown in Figure 10.

Figure 10. Creation of a Document

The user enters the references through the editor's user interface, which might present a list of the names of the available fonts or a list of the characteristic font attributes that enable the processing system to select a font. The document data stream produced by the editor can specify fonts in one of the following ways:

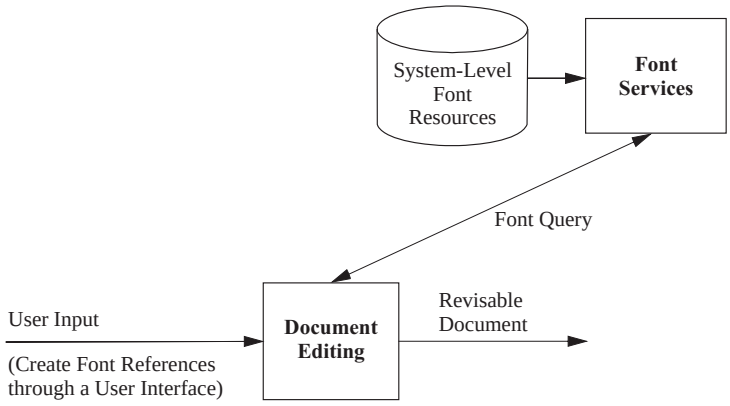
- Specifies the requested font by name
- Maps the font attributes selected by the user to a specific font name
- Reproduces the font attributes of the requested font from a separately stored list of attributes specified by the user
- Reproduces the font attributes of the requested font from the font resource specified by the user

Because editing the document creates the font references and does not include formatting the document, the user needs to record only the intended font as input (for example, *10 point IBM Sonoran Serif italic*).

Editor Determination of Font Availability

The application program or editor might provide the user with a list of available fonts or locate an available font that corresponds to the font information provided. To do this, as shown in Figure 11, an editor might use an interface that permits inquiries to a font services facility concerning the font resources available to the processing system. Another method might be for the editor to directly access the font resource information in system storage.

Figure 11. Editor Determination of Font Availability

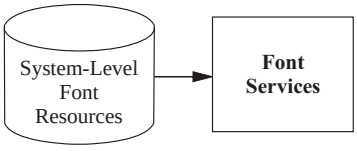


The font resources available to a system need not be physically resident on the user's workstation (distributed data base) or physically available in the form required for presentation (logical fonts or transformable resources). The editor queries font services about the availability of font resources.

Font Services Access to Font Information

The font services facility responds to editor queries and provides font information, as shown in Figure 12, by accessing system-level font resources available in the font library. The font references that the font services facility uses are different in form and content from those required by editing applications.

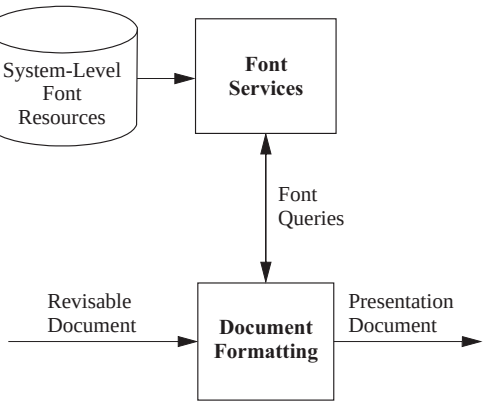
Figure 12. Font Services Access to Font Information



Formatter Access to Font Information

During formatting, the processing system must interpret the font references contained in the revisable document, locate the required font-metric information, and produce a formatted, final-form document. To do so, as shown in Figure 13, the formatter might use an interface that permits inquiries to a font services facility concerning the font resources available to the processing system. The formatter must generate a query to the font services facility by providing an appropriate font reference (which might be different from that issued during editing), and it will receive a response to the query.

Figure 13. Formatter Access to Measurement Information



The presentation document produced will contain references to the font resources used when formatting, and will often be more specific than those found in the revisable document. If the presentation document is sent to another location or system where the specified font resource is not available, it might be necessary to substitute another font for presentation of the document. Thus, font references contained in a presentation document should retain the user intent provided in the revisable document.

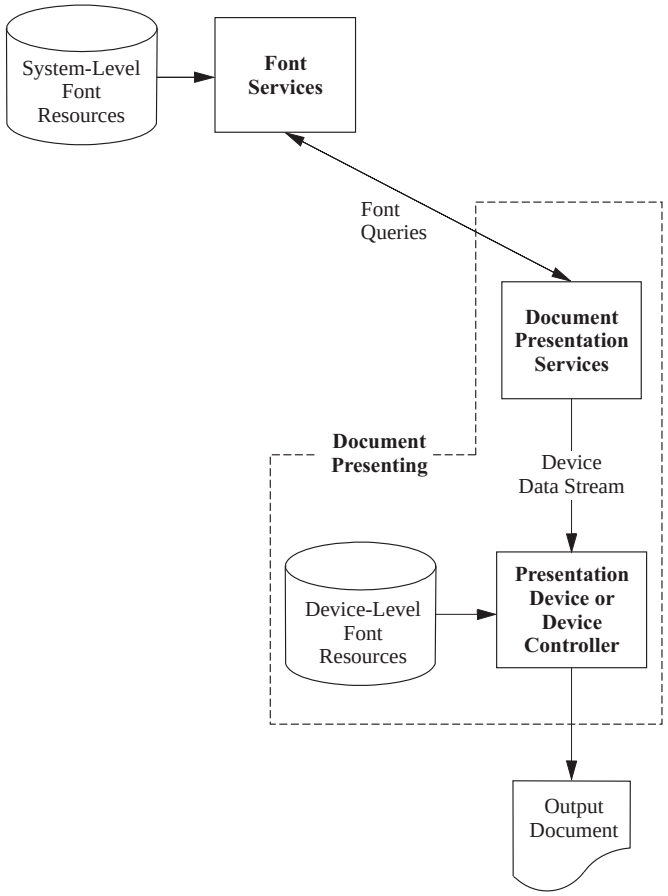
Presentation Services Access to Font Information

The presentation services provide several font-related functions that might include:

- Accessing and transforming the font information required by a device
- Down loading font information to a presentation device
- Determining available resident font support in the device
- Substituting one font resource for another

When processing presentation information, the processing system, as shown in Figure 14, must generate a query to the font services facility by providing an appropriate font reference. The font services facility responds by identifying the available font resources or the font and shape information that corresponds to the referenced font.

Figure 14. Presentation Services Access to Font Information



When processing presentation information, the processing system might also perform a query to the attached presentation devices to determine which resident fonts are available on the device, and it will receive font references that identify those fonts. The data stream for the device produced by the presentation services must contain references to the device-specific font resources needed to present the document, and it might include the required font-metric and character-shape information. Because the data stream for the device must contain specific references to device-specific font resources, the content and format of the font references will often be more specific than those required in the revisable or presentation document references.

Identifying Fonts

Each of the previously-defined process steps involves the use of font references to font resources. Each reference might require different information in different formats, and the different font resources referenced might be identified by different name formats. To fully support font referencing, the following items must be defined:

- System-level identifying characteristics
- Device-level identifying characteristics
- User input font reference
- Revisable document font reference
- Presentation document font reference
- Device data stream font reference

The general requirements for identifying font resources and referencing font resources are similar, although the specific requirements will vary by the type of installation, the type of implementation, and the point in the process. The specific content and format of a font resource name is defined by the implementing product specifications. When planning to identify font resources, the following factors must be considered:

- A font reference might be a list of font attributes, or it might be a system file name, a member name, or a resource-management name that is mapped to a system file name or member name.
- More than one font resource can correspond to a single font reference.
- More than one font reference can correspond to a single font resource.
- If a system uses raster font resources, different font resources might be necessary for each presentation device and each presentation size.
- If a system uses outline font resources, one font resource might provide the data for multiple font sizes, typeface variations, and presentation devices.
- The list of characters in a font resource might not match the list of characters in a code page.
- A user's intent should be distinguishable from system-default or architecture-default values.
- A user's font references should be as device-independent as possible to permit document distribution to a variety of sites and to permit document presentation on a variety of devices.

Fonts can be identified in different ways. One example is by using the descriptive parameters used in the Map Coded Font structured field in MO:DCA. Another example is by using the Global Resource Identifier, GRID, as used in MO:DCA and IPDS. Refer to the *Mixed Object Document Content Architecture Reference* and the *Intelligent Printer Data Stream Reference*.

The specific content and format of a font reference is defined by the appropriate document content, device service, or interface architecture. The following information is provided as a guideline for such use.

System-Level Font Resource

A system-level font resource is generally designed to be shared across multiple applications and presentation devices in support of multiple document and presentation data streams. System-level font resources will contain the metric and shape information for a large number of graphic characters, identifying them by unique global identifiers, independent of any single document encoding technique. The set of attributes used to identify a system-level font resource is more extensive than those used to identify a device-level font resource because of the metric and shape information used by editor, formatter, and presentation processes.

The following table identifies the AFP parameters and ISO properties (*properties* is the term used by the ISO/IEC 9541 *Font Information Interchange* standard) that should appear in a system-level font resource to support font referencing. Because locating a desired font resource requires matching the parameters of a font reference to corresponding parameters in the available font resources, this set of parameters represent the union of the parameters that should appear in any of the supported font references.

Table 5. AFP Parameters Mapped to ISO Properties for System-Level Font Resources

AFP Parameter	**ISO Property**
resource name	fontname
function set code	standardversion
typeface name	typeface
design source	dsnsource
data source	datasource
family name	fontfamily
posture class	posture
weight class	weight
width class	propwidth
primary GCSGID	incglyphcols
nominal vertical font size	dsnsize
minimum vertical font size	minsize
maximum vertical font size	maxsize
nominal horizontal font size	(not supported)
minimum horizontal font size	minanascale
maximum horizontal font size	maxanascale
font classification	dsngroup
structure class	structure
font type flags	escclass
character rotation	nomescdir
average escapement	avgescx/y
average lowercase escapement	avglcescx/y
average capital escapement	avgcapescx/y
figure space increment	tabescx/y
maximum baseline extent	maxfontext

The various methods of document encoding are addressed by a separate collection of code page (encoding vector) resources that map the supported code points to the font glyph identifiers. Differences in device resolution and imaging technology are addressed by the presentation process that transforms a system-level font resource and an appropriate code page resource into a device-level font resource.

Device-Level Font Resource

A device-level font resource is generally designed to be used by one device or family of related devices and a single document encoding technique. Device-level font resources should contain the metric and shape information for a single code page (encoding vector) and will most often identify the metric and shape information by the code point rather than by an independent glyph identifier (though it can use an attached mapping table to associate the code point to the glyph ID). The set of attributes used to identify a device-level font resource is less extensive than that used to identify a system-level font resource because the metric and shape information is used only by the presentation processes.

The following table identifies the AFP parameters and ISO properties that should appear in a device-level font resource to support font referencing. Because locating a desired font resource requires matching the parameters of a font reference to corresponding parameters in the available font resources, this set of parameters corresponds to the set of the parameters that should appear in a font reference in the device data stream.

Table 6. AFP Parameters Mapped to ISO Properties for Device-Level Font Resources

AFP Parameter	**ISO Property**
resource name	fontname
design source	dsnsource
data source	datasource
family name	fontfamily
posture class	posture
weight class	weight
width class	propwidth
primary GCSGID	incglyphcols
primary CPGID	(not supported)
nominal vertical font size	dsnsize
minimum vertical font size	minsize
maximum vertical font size	maxsize
nominal horizontal font size	(not supported)
minimum horizontal font size	minanascale
maximum horizontal font size	maxanascale
font classification	dsngroup
structure class	structure
font type flags	escclass
character rotation	nomescdir
average escapement	avgescx/y
average lowercase escapement	avglcescx/y

Table 6 AFP Parameters Mapped to ISO Properties for Device-Level Font Resources (cont'd.)

AFP Parameter	ISO Property
average capital escapement	avgcapescx/y
figure space increment	tabescx/y
maximum baseline extent	maxfontext
pattern technology identifier	glyphshapetech

User-Input Font Reference

A user-input font reference is generally designed to be used by people who might not be knowledgeable about font technology, but who wish to use different fonts in their document. Processing applications have a high degree of freedom in how they can present information to users and can hide most of the technical detail from the users, mapping the user's input to the specifics required by the data stream. The user-input font reference can be contained in a reference book containing examples and the corresponding font resource names that users type at their workstation. The font reference can be a workstation displayed list of available fonts from which users select the desired item, or it can be a series of pop-up pull-down menus from which users select various font characteristics (for example: bold, serif, 10 pt., italic) that are desired.

Because the font resources available at any one workstation might not be available at all other workstations in a distributed environment, the user-input font reference should be independent of specific resource names and should focus instead on getting as much information as possible about the user's desired intent (for example, highlight this phrase with the bold version of the current font). Whatever technique is used for user input, the user's selection must be translatable or mappable into the revisable-document font reference.

Revisable-Document Font Reference

A revisable-document font reference is generally designed to focus on the user's intent for text appearance, with little or no emphasis on specific font resources or font metrics. The document can then be sent to any location, formatted using the best available font resource (that satisfies the user's intent), and printed or displayed for the recipient's use.

Specification of the code page used for encoding the document is necessary at the revisable-document level, but the code page does not have to be linked to a font resource at this stage. The code page does not need to appear in the font reference (although it can if the document data stream architecture also uses the font reference to define the document encoding).

The following table identifies the AFP parameters and ISO properties that should appear in a revisable document font reference.

Table 7. AFP Parameters Mapped to ISO Properties for Revisable-Document Font References

AFP Parameter	ISO Property
resource name	fontname
design source	dsnsource
family name	fontfamily
posture class	posture
weight class	weight
width class	propwidth

Table 7 AFP Parameters Mapped to ISO Properties for Revisable-Document Font References (cont'd.)

AFP Parameter	ISO Property
vertical font size	dsnsize
font classification	dsngroup
structure class	structure
font type flags	escclass
character rotation	nomescdir

Presentation-Document Font Reference

A presentation-document font reference is generally designed to focus on the formatter's intent for page layout, while recalling the user's intent for text appearance. The presentation-document font reference should contain all of the revisable-document font reference information along with enough additional information so as to identify the metrics used for page layout. The document can then be sent to any location, and the required font can be located, or an alternate font that closely approximates the metrics can be substituted (see "Font Selection and Substitution" on page 28).

Specification of the glyph complement (graphic character set) that corresponds to the code page used for encoding the document is necessary at the presentation-document level, but the code page need not be linked to the font resource at this stage. The code page does not need to appear in the font reference (although it can if the document data stream architecture also uses the font reference to define the document encoding).

The following table identifies the AFP parameters and ISO properties that should appear in a presentation-document font reference.

Table 8. AFP Parameters Mapped to ISO Properties for Presentation-Document Font References

AFP Parameter	ISO Property
resource name	fontname
function set code	standardversion
typeface name	typeface
design source	dsnsource
data source	datasource
family name	fontfamily
posture class	posture
weight class	weight
width class	propwidth
primary GCSGID	incglyphcols
vertical font size	dsnsize
font classification	dsngroup
structure class	structure
font type flags	escclass
character rotation	nomescdir
average escapement	avgescx/y

Table 8 AFP Parameters Mapped to ISO Properties for Presentation-Document Font References (cont'd.)

AFP Parameter	ISO Property
average lowercase escapement	avglcescx/y
average capital escapement	avgcapescx/y
figure space increment	tabescx/y
maximum baseline extent	maxfontext

Device Data Stream Font Reference

A device data stream font reference is generally designed to focus on the presentation process's intent for glyph shape rendering, while providing enough reference information to identify the required font information under many different device-specific font resource names.

Specification of the glyph complement (graphic character set) and the corresponding code page used for encoding the document is necessary at the device data stream level.

The following table identifies the AFP parameters and ISO properties that should appear in a device data stream font reference.

Table 9. AFP Parameters Mapped to ISO Properties for Device Data Stream Font References

AFP Parameter	ISO Property
resource name	fontname
design source	dsnsource
data source	datasource
family name	fontfamily
posture class	posture
weight class	weight
width class	propwidth
primary GCSGID	incglyphcols
primary CPGID	(not supported)
vertical font size	dsnsize
font classification	dsngroup
structure class	structure
font type flags	escclass
character rotation	nomescdir
average escapement	avgescx/y
average lowercase escapement	avglcescx/y
average capital escapement	avgcapescx/y
figure space increment	tabescx/y
maximum baseline extent	maxfontext
pattern technology identifier	glyphshapetech

Font Selection and Substitution

When an application process encounters a font reference, it is necessary for the process to locate a font resource that corresponds to that reference. *Font selection* is the process of locating a font resource that corresponds to the font reference or that satisfies all the information contained in the font reference. *Font substitution* is the process of locating an alternate font resource that approximates the font resource, because no font resource could be found that met the selection criteria.

Font selection is most simply performed by having an exact match of a font resource name to a font reference that contains the font resource name. But, in a distributed processing environment, the number and combination of font resource names is without limit, with each font supplier and system supplier having their own naming conventions and each variation in font resource content requiring a unique name. The chances of an exact match for a font resource name diminishes as the number of users in a distributed environment increases.

Font substitution can be as simple as using a system or device default font (for example, substituting fixed pitch Courier whenever the requested font cannot be found), but that is often disappointing to the user. Font selection or substitution is most effective when all of the identifying characteristics contained in a font reference are intelligently compared to the identifying characteristics contained in all of the available font resources, until an exact or best match is located.

The specific data formats and processing steps required for intelligent font substitution is outside the scope of this architecture (that is, the data formats and the processing steps must be defined by implementations and document content architectures based on their specific user requirements). However, the following sections provide guidelines for various selection and substitution situations.

Identifying User Intent

The document data stream should contain information about a user's intent for presentation. The success of font substitution is described in terms of fidelity. Fidelity is described in the ANSI and ISO user requirements for fonts as the ability to faithfully replicate a document with an equivalent presentation.

There are five classes of fidelity, which are described in order of increasing fidelity:

Content fidelity

Permits a user to identify the characters used in the document, or to identify a character that cannot be presented (a character can be substituted when a specified character is not available in the font).

Format fidelity

Preserves line and cell identity.

A user must be able to identify the basic parts of the logical document structure, such as associated values, forms of emphasis, lines, paragraphs, and so on.

Areas can be reproduced using fonts that change line-ending position, such as wrap-around lines or tab-stop positions.

Layout fidelity

Maintains the integrity of line endings, column endings, and page endings.

Layout fidelity maintains the relationship between areas. Enough font information must be provided to achieve these results.

Appearance fidelity

Requires a document, or area, to appear as specified by the user.

Substitution of visually equivalent fonts is permitted.

Absolute fidelity

Requires a document to appear precisely as specified by the user.

No substitution of fonts is permitted.

Document Editing

The document-editing process provides the interface between the user input and the revisable document data stream output. The user input can be in a variety of forms, and it is necessary for the document editing process to convert that user input to a form that is appropriate for the revisable-document font reference.

Different techniques are available for creating the revisable-document font reference, depending on the form of the user input and processing environment. The preferred method is to take the information provided by the user input and select an available font resource that satisfies the user's intent, and then copy the required font parameters from the selected font resource into the corresponding parameters of the revisable-document font reference. The resource name alone does not provide enough information to locate the font resource on another system, if the resource name is not globally unique or if the specified font resource is not available.

The process of selecting an available font resource that satisfies the user's intent should be performed by prioritizing the user inputs and searching for the best matching font resource. If the user simply designates an available font resource, the editing process has no information about which parameters are most important to the user. Ideally, the user interface allows the user to specify which font attributes are most important and which are less important. In the absence of such information, selection should occur by beginning with the most general attributes (finding the set of all available matching font resources) and then working down to the most specific attributes (eliminating resources from the initial set until a single resource is found that satisfies the greatest number of attributes).

For the document editing process, the following default order should be used for font selection:

1. Content-level fidelity is of primary importance to the processing system and should take precedence over all other fidelity levels. The selected font resource must contain the glyphs needed to render the code points contained in the document data stream.
   a. Glyph Collection (GCSGID)
2. Format- and layout-level fidelity is the next most important to the processing system and should take precedence over the appearance-level fidelity.
   a. Vertical font size
   b. Escapement direction
   c. Escapement class
3. Appearance-level fidelity is of primary interest to the user at the time of document creation (the user assumes that the processing system will display and print proper glyphs at the right location on the page).
   a. Structure class
   b. Posture class
   c. Weight class
   d. Width class
   e. Font classification
   f. Family name
   g. Design source

Document Formatting

The document-formatting process provides the interface between the revisable-document data stream input and the presentation-document data stream output. The revisable-document input provides the user's appearance intent, for the document formatting process to locate an appropriate font resource, to format the presentation-document, and to create a presentation-document font reference.

The document formatting process should take the information provided by the revisable document font reference, select an available font resource that satisfies the user's intent, and then copy the required font parameters from the selected font resource into the corresponding parameters of the presentation-document font reference. Ideally, the revisable-document font reference will designate which font attributes are most important to the user and which are less important. In the absence of such information, selection should occur by beginning with the most general attributes (finding the set of all available matching font resources) and then working down to the most specific attributes (eliminating resources from the initial set until a single resource is found that satisfies the greatest number of attributes).

For the document formatting process, the following default order should be used for font selection. Note that this order is the same as for the document editing process, because formatting has not yet occurred.

1. Content-level fidelity is of primary interest to the formatting process and should take precedence over all other fidelity levels. The selected font resource must contain the glyphs needed to render the code points contained in the document data stream.
   a. Glyph Collection (GCSGID)
2. Satisfying the user's format- and layout-level fidelity is the next most important task of the document formatting process.
   a. Vertical font size
   b. Escapement direction
   c. Escapement class
3. Satisfying the user's appearance-level fidelity is the third most important task of the document-formatting process.
   a. Structure class
   b. Posture class
   c. Weight class
   d. Width class
   e. Font classification
   f. Family name
   g. Design source

Document Presentation

The document-presentation process provides the interface between the presentation-document data stream input and the device data stream output. The presentation-document input provides the user's and formatter's intent, for the document-presentation process to locate an appropriate system or device font resource. It might be necessary for the document-presentation process to transform a system-font resource into a device-font resource, and down load the device-font resource to the appropriate presentation device; or it might be necessary for the document-presentation process to query the presentation device to determine availability of the appropriate device-resident font resource.

The document-presentation process should take the information provided by the presentation-document font reference and select an available font resource that satisfies the user and formatting process intent. Ideally, the presentation-document font reference will designate which font attributes are most important to the user and formatting process, and which are less important. In the absence of such information, selection should occur by beginning with the most general attributes (finding the set of all available matching font resources) and then working down to the most specific attributes (eliminating resources from the initial set until a single resource is found that satisfies the greatest number of attributes).

For the document-presentation process, the following default order should be used for font selection:

1. Content-level fidelity is of primary importance and should take precedence over all other fidelity levels. The selected font resource must contain the glyphs needed to render the code points contained in the document data stream.
   a. Glyph Collection (GCSGID)
2. Format- and layout-level fidelities takes precedence over appearance-level fidelity after the document has been formatted. At this level of fidelity the line, column, and page breaks must occur where the formatting process indicated.
   a. Vertical font size
   b. Escapement direction
   c. Escapement class
   d. Maximum extents
   e. Average escapement
   f. Average lowercase escapement
   g. Average capital escapement
   h. Tabular escapement (figure space increment)
3. Satisfying the user's appearance-level fidelity is the third most important task of the document-presentation process.
   a. Structure class
   b. Posture class
   c. Weight class
   d. Width class
   e. Font classification
   f. Family name
   g. Design source

Chapter 4. FOCA Overview

The Font Object Content Architecture (FOCA) supports presentation of character shapes by defining their characteristics, which include Font-Description information for identifying the characters, Font-Metric information for positioning the characters, and Character-Shape information for presenting the character images. Presenting a graphic character on a presentation surface requires that you communicate this information clearly to rotate and position characters correctly on the physical or logical page. For example, you can:

- Rotate a physical page within a print system
- Rotate and move a logical page to new locations on a physical page
- Rotate and position character shapes anywhere within a logical page

Note: FOCA does not address the orientation of pages on devices, logical pages on a physical page, or lines of text on a logical page.

By defining FOCA parameters, you enable the system to separate font information from physical and logical page information as the system identifies, positions, and presents characters. This chapter gives an overview of FOCA by defining the terms you need when you use FOCA parameters, including terms for:

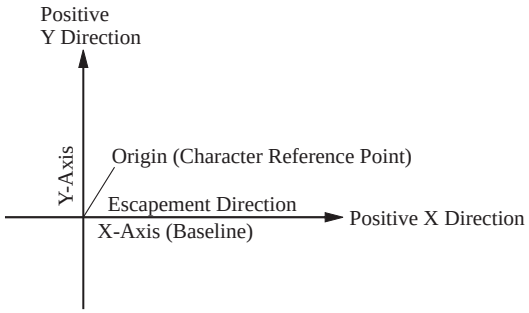
- The character coordinate system, including units of measure and direction within the coordinate system
- Graphic characters
- Character-set metrics
- Design metrics

The chapter concludes with some general recommendations when using FOCA to design fonts.

Character Coordinate System

FOCA positions character shapes within an orthogonal *character coordinate system*, as shown in Figure 15.

Figure 15. Character Coordinate System



FOCA locates font or character measurements in this system as *distances* between points. The point where the axes intersect is called the *origin* or *character reference point* (see "Character Reference Point" on page 37 for a specific example). All the positioning and rotating measurements of a character are relative to that point; that is, as a character is rotated, the character coordinate system does not rotate.

The x-axis of the character coordinate system is coincident with the *character baseline* (see "Character Baseline" on page 37). Positive x-axis direction is the *escapement direction*, as shown in Figure 15. Positive y-axis direction is 90°; counterclockwise or 270° clockwise from the positive x-axis direction.

Note: The default escapement direction is left-to-right along the x-axis of the character coordinate system.

The following sections describe how you specify measurement and direction units within the character coordinate system.

Units of Measure

A font designer specifies the type of measurements used in defining a font as either fixed units, such as inches or centimeters, or relative units, which are dimensionless.

Fixed units apply only to a single or limited number of devices, while relative units allow a font to be used by multiple devices. For example, a 240-pel per inch device requires 30 pels to represent 1/8 inch, but a 300-pel per inch device cannot represent 1/8 inch with a whole number of pels (300/8 = 37.5). When using fixed units for a presentation, the system evaluates the following:

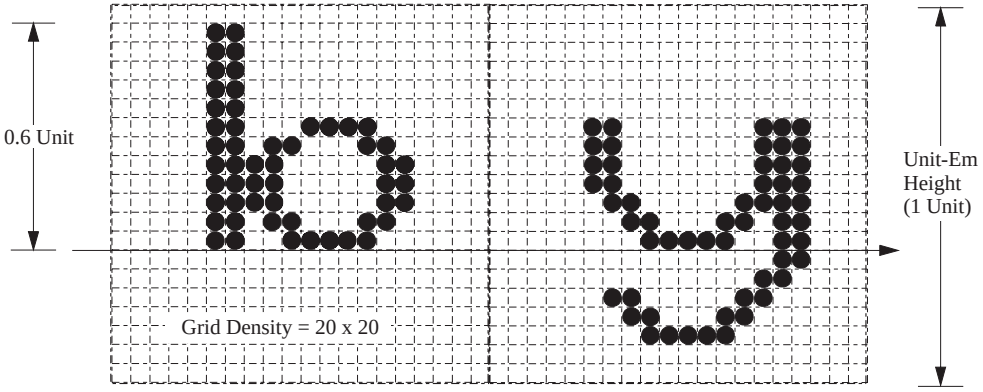
- Can the specified length unit be accepted?
- Can the specified length unit be converted for use?
- Should some error-recovery action be taken?

If the fixed units for a designated font cannot be accepted or converted, the system might be unable to supply an acceptable font for the selected presentation device, and it might take an error-recovery action. If the designated font was designed using relative units, the system scales or proportions the font measurements to accommodate the presentation device.

Unit-Em

FOCA expresses relative units as a decimal fraction of a *Unit-Em*. A Unit-Em is a unit of one that corresponds to the height of the design space, which is also the nominal vertical font size. You specify this measurement in the Nominal Vertical Font Size parameter. Figure 16 shows a relative unit expressed as the Unit-Em. See "Nominal Vertical Font Size" on page 68 for information about the Nominal Vertical Font Size parameter.

Figure 16. Relative Unit as the Unit-Em



For practical purposes, the Unit-Em must be expressed in some fixed unit of measure. In the printing industry, the *point*, 1/72 inch, is customarily used. In order to find the fixed value of any of the relative metrics expressed as fractions of a Unit-Em, the fraction must be multiplied by the Unit-Em point value.

Unit-Base

A Unit-Base specifies a one-byte code that represents the length of the measurement base. Allowable code values for the Unit-Base are from 0 to 2 and represent the following:

Table 10. Unit-Base Values

Values	Unit-Base
0	Ten inches
1	Ten centimeters
2	Relative units (measurement base = one Unit-Em)
3-255	Reserved

Units Per Unit-Base

The *Units per Unit-Base* specifies the number of units in the measurement base. Allowable values for the Units per Unit-Base are from 1 to 32,767.

In order to avoid the use of decimal fractions, the Units per Unit-Base is customarily multiplied by the Unit-Em value in points where the point value of the Unit-Em is the Unit-Base. The relative metrics must then also be multiplied by the same factor. Furthermore, a value of 1000 is usually taken for the Units per Unit-Base multiplier. For example, a Unit-Em value of 20 points becomes 20,000 units when multiplied by a Units per Unit-Base value of 1000. A character height of 0.6 Unit-Em, as in Figure 16 on page 34, then becomes 12,000 units (0.6 * 20 * 1000).

Calculating the Units of Measure

Units of Measure is calculated by dividing the value of the measurement base (as specified by the Unit-Base parameter) by the Units per Unit-Base value, according to the following formula:

```
Units of Measure = measurement base/Units per Unit-Base
```

Resolution is the reciprocal of Units of Measure. The following examples show how different resolutions can be specified.

Example 1: An example of fixed-value measurements.

```
A resolution of 240 pels per inch is specified as:
            Unit-Base = 0 (measurement base = 10 inches)
            Units per Unit-Base = 2400

       therefore:

            Units of Measure = 10 inch/2400 = 1/240 inch
            Resolution = 1/Unit of Measure = 240 units per inch
```

Example 2: An example of relative-value measurements.

```
        A resolution of 20 divisions per Unit-Em is specified as:
            Unit-Base = 2 (measurement base = 1 Unit-Em)
            Units per Unit-Base = 20

        therefore:

            Units of Measure = measurement base/Units per Unit-Base
                             = 1(Unit-Em)/20
            Resolution = 1/Unit of Measure = 20 Units per Unit-Em
```

With relative units of measure, font dimensions can be scaled to any desired size by multiplying the specified dimension by the desired vertical font size. For example, if the height of a character image is expressed as 12 relative units when the Unit-Em value is 20 points, the character height as a relative value is 0.6 (12/20 = 0.6). At any other Unit-Em value, such as 12 points, the character height is the relative value multiplied by the Unit-Em value: 0.6 x 12 = 7.2 points.

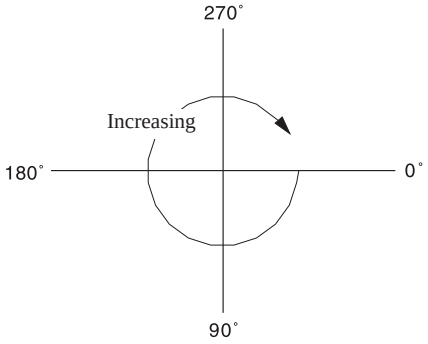
Units of Direction

In FOCA, *units of direction*, as shown in Figure 17, are specified as two integer values, which represent degrees and minutes:

Degrees Integers from 0 to +359.
Minutes Integers from 0 to +59.

Increasing values, as shown in Figure 17, indicate increasing clockwise directions.

Figure 17. Directions



Character Concepts

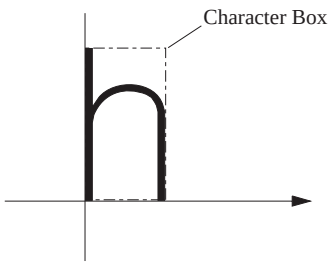
This section defines the terms and concepts used to describe individual graphic characters in FOCA.

Character Boxes

FOCA uses a concept in which each character shape is defined to be within a *character box*, which is a conceptual rectangle. When character shapes are presented, the character boxes are positioned and can be rotated.

A character box is a conceptual rectangle having two sides parallel to the character baseline and two sides perpendicular to the character baseline. A character box circumscribes a graphic character. We differentiate between boxes having no extra space surrounding the character, and boxes having extra space, by the terms bounded and unbounded, respectively. Figure 18 shows a bounded character box.

Figure 18. Bounded Character Box for the Latin Letter 'h'



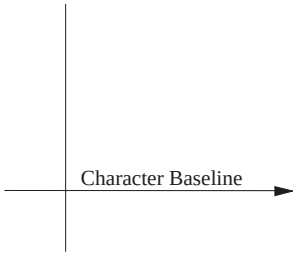
In this architecture, unless noted otherwise, the character boxes are bounded-boxes. The four sides of the box just touch the character shape.

Character boxes are positioned and rotated using the Font-Metric information for positioning provided in "Font-Metric Parameters" on page 74.

Character Baseline

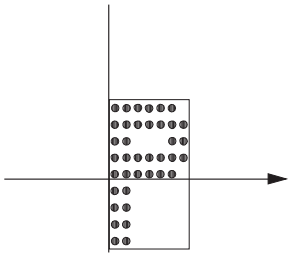
All characters in a font are aligned on an imaginary line called the *character baseline*. The character baseline corresponds to the x-axis of the character coordinate system. The use of the character baseline ensures that successive characters—even from different fonts—are properly aligned. Figure 19 shows the character baseline.

Figure 19. Character Baseline



If character shapes are presented using pel addressing, all character measurements represent distances between the pels. No pels fall on the baseline. Figure 20 shows a figure presented in pels and its relationship to the baseline.

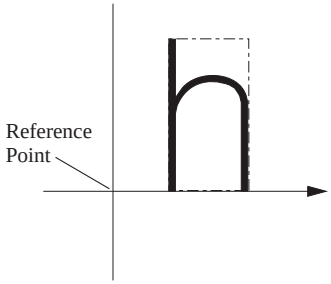
Figure 20. A Character Shape Presented in Pels



Character Reference Point

The *character reference point* as shown in Figure 21, corresponds to the origin of the character coordinate system. It coincides with the presentation position when the character is presented on the presentation surface.

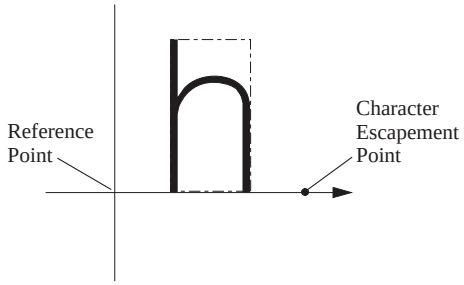
Figure 21. Character Reference Point



Character Escapement Point

The *character escapement point*, as shown in Figure 22, is the point where the next character reference point is usually positioned.

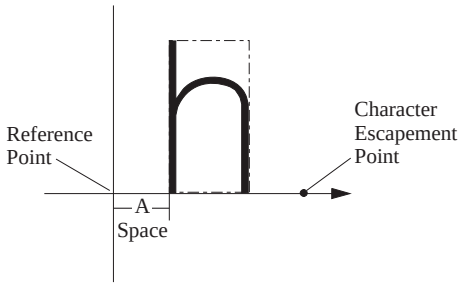
Figure 22. Character Escapement Point



A-space

A-space, as shown in Figure 23, is the distance from the character reference point to the least positive character coordinate system x-axis value of the character shape.

Figure 23. A-space



The baseline is coincident with the character coordinate system's x-axis. A-space can be positive, zero, or negative:

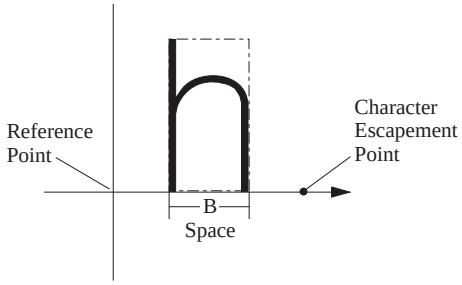
- A positive value specifies that the character reference point lies before the leftmost edge of a character box.
- A zero value specifies that the reference point coincides with the leftmost edge of the character box.
- A negative value specifies that the character reference point lies after the leftmost edge of a character box.

  Using negative A-space is a technique to implement kerning (kerning is described in "Kerning" on page 40).

B-space

B-space, as shown in Figure 24, is the distance between the character coordinate system x-axis values of the two extremities of a character shape.

Figure 24. B-space

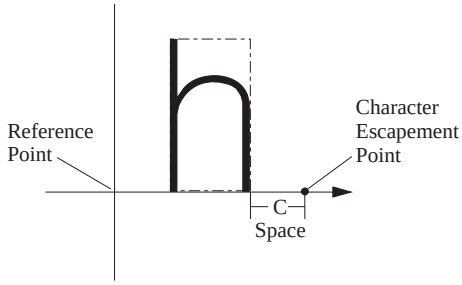


For a bounded character box, the B-space is equal to the width of the character box. Negative or zero values of B-space have no meaning.

C-space

C-space, as shown in Figure 25, is the distance, measured in the inline (positive x-axis) direction, from the character's most positive x-axis coordinate value in the character coordinate system, to the character escapement point.

Figure 25. C-space



C-space can be positive, zero, or negative:

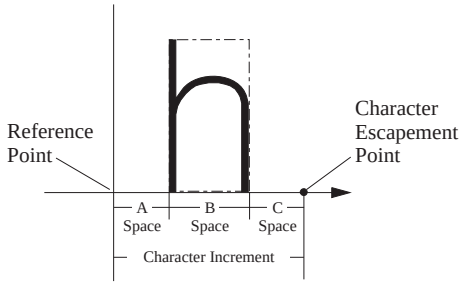
- A positive value specifies that the escapement point lies after the right-hand edge of a character box.
- A zero value indicates that the escapement point coincides with the right-hand edge of a character box.
- A negative value specifies that the escapement point lies before the right-hand edge of a character box.

  A technique to implement kerning is the use of negative C-space.

Character Increment

The *character increment* is the distance from the character reference point to the character escapement point. Character increment, as shown in Figure 26, is the sum of the A-space, the B-space, and the C-space.

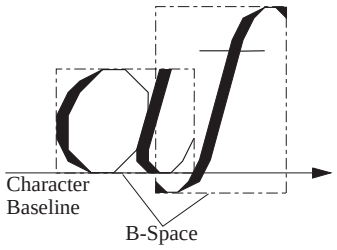
Figure 26. Character Increment



Kerning

The spacing between adjacent characters can be reduced so that the characters overlap, which is called *kerning*. Kerning is done by making the A-space or C-space negative. Figure 27 illustrates the kerning of the characters *a* and *f*.

Figure 27. An Example of Kerning



Pair Kerning

The kerning of two characters can be varied by adjusting the increment between them. Using this adjustment, the two characters can be positioned closer together, further apart, or made to overlay one another.

Pair kerning can be implemented by specifying pairs of characters to be kerned and by specifying the increment between them, depending on the tightness or looseness of the fit desired. See Chapter 5, "Kerning Pair Character 1" on page 76, "Kerning Pair Character 2" on page 77, and "Kerning Pair X-Adjust" on page 77.

Ascender Height

Ascender height is the distance from the character baseline to the top of the character box. A negative ascender height signifies that all of the graphic character is below the character baseline. Figure 28 shows the ascender height.

Figure 28. Ascender Height

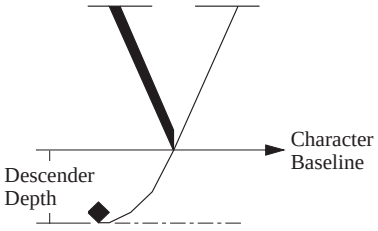
For a character rotation other than 0°, “ascender height” loses its meaning when the character is lying on its side or is upside down with respect to normal viewing orientation. For the general case, Ascender Height is the character's most positive y-axis value.

For bounded character boxes, for a given character having an ascender, ascender height and baseline offset are equal.

Descender Depth

Descender depth is the distance from the character baseline to the bottom of a character box. A negative descender depth signifies that all of the graphic character is above the character baseline. Figure 29 shows the descender depth.

Figure 29. Descender Depth



Baseline Extent

Baseline extent is the space parallel to the character baseline that can be used to place characters. Figure 30 shows the baseline extent with a subscript character, Figure 31 on page 42 with a superscript character, and Figure 32 on page 42 with a character on the baseline.

Note: A negative value for baseline extent is treated as zero.

Figure 30. Baseline Extent: Subscript Character

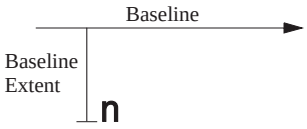


Figure 31. Baseline Extent: Superscript Character

Figure 32. Baseline Extent: Character on the Baseline

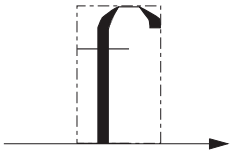
Baseline Offset

The *baseline offset* is the perpendicular distance from the character baseline to the top of a character box. This value is positive if any part of the character box is above the character baseline, and it is negative if all of the character box is below the character baseline. If the baseline offset is zero, the top of the character box is at the baseline.

Slope

The *slope* is the posture (or incline) of the main strokes in the graphic characters in a font and is measured clockwise from the vertical. Slope is specified in degrees by a font designer. Figure 33 shows the letter *f* with 0° slope.

Figure 33. Slope 0°



If the characters are upright or if they have a slight forward slant, the slope is small. If the characters have a backward slant, the slope is large. Slope is usually 0° for upright fonts and 17.5° for italic fonts. Figure 34 shows the letter *f* with 17.5° slope.

Figure 34. Slope 17.5°

Font Concepts

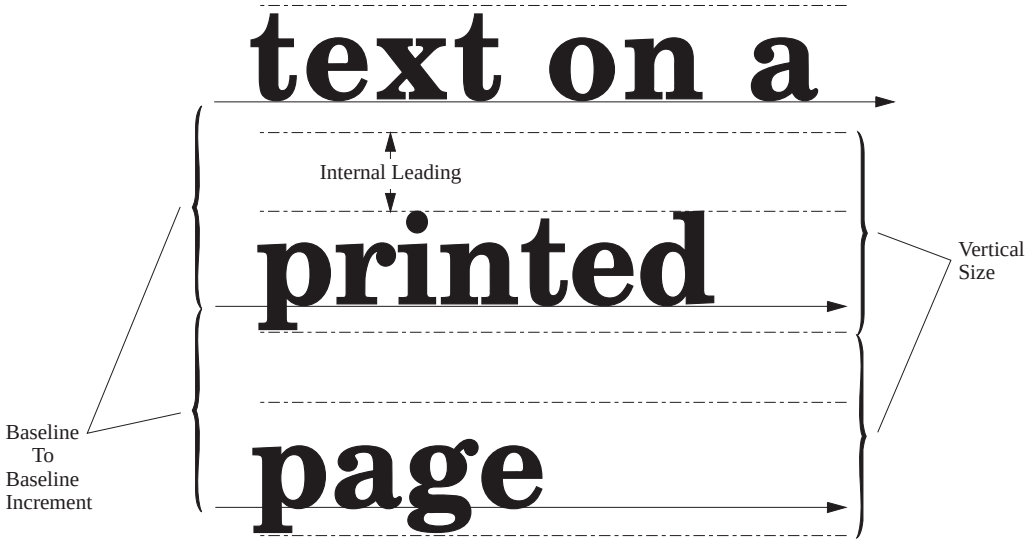
This section defines the terms and concepts used to describe fonts as they apply to all the characters in an entire font in FOCA.

Vertical Size

The size of a font can be characterized by the vertical size of its characters. *Vertical size* represents the nominal baseline-to-baseline increment that includes the vertical size of the set of all characters and the designer's recommendation for the internal leading (space above or below the characters). Vertical size is illustrated in Figure 35.

Vertical size is also known as body size, point size, em-height.

Figure 35. An Illustration of Vertical Size and Internal Leading



Horizontal Font Size

The size of a uniformly-spaced font can be characterized by the increment of its characters. *Horizontal Font size* is measured parallel to the baseline when character rotation is 0°.

Horizontal font size is also known as character width and character pitch.

Cap-M Height

Cap-M height is the average height of the uppercase characters in a font. This value is specified by the designer of a font and is usually the height of the uppercase *M*.

X-Height

X-height is the nominal height above the baseline (ignoring the ascender) of the lowercase characters in a font and is usually the height of the lowercase letter *x*. The value of X-height is specified by a font designer.

Internal Leading

Internal leading is the total amount of space above and below the text character shapes, which provides an aesthetically pleasing interline spacing. The value of this parameter usually equals the difference between the vertical font size and the baseline extent for text characters in the font. Internal leading is illustrated in Figure 35 on page 43.

Note: Special characters that extend from baseline to baseline (the backslash, the forward slash, and characters used for drawing boxes) are not included in this calculation.

Fonts are normally designed with some nominal amount of white space above or below the character images. When placing more text on a page by using less space between lines, this value can be used to determine how little space between lines can be used without overwriting lines of text.

External Leading

External leading is the amount of white space—in addition to the internal leading—that can be added to the interline spacing without degrading the aesthetic appearance of a font. This value is usually specified by a font designer. External leading is illustrated in Figure 36.

Figure 36. An Illustration of External Leading

Maximum Ascender Height

The *maximum ascender height* is the maximum height attained by any graphic character in a font from the character baseline to the top of the character box.

A value for the maximum ascender height is specified for each supported rotation of the character. This value can be positive, zero, or negative.

- A positive value specifies that some of the graphic characters in a font extend above the character baseline.
- A value of zero specifies that the top of at least one character box is at the baseline, and no character box extends above the baseline.
- A negative value specifies that all the graphic characters in a font lie completely below the character baseline, as in the case where the characters are subscripts.

Maximum Descender Depth

The *maximum descender depth* is the maximum depth attained by any graphic character in a font from the character baseline to the bottom of the character box.

The value for the maximum descender depth is specified for each supported rotation of the character. This value can be positive, zero, or negative.

- A positive value specifies that some of the graphic characters in a font extend below the baseline.
- A value of zero specifies that the bottom of at least one character box is at the baseline, and no character box extends below the baseline.
- A negative value specifies that all the graphic characters in a font lie completely above the character baseline, as in the case where the graphic characters are superscripts.

Maximum Baseline Extent

Maximum baseline extent is the sum of the most positive baseline offset of any individual character in the font and the most positive descender depth of any individual character in the font.

Note: A negative value for maximum baseline extent is treated as zero.

Superscripts and Subscripts

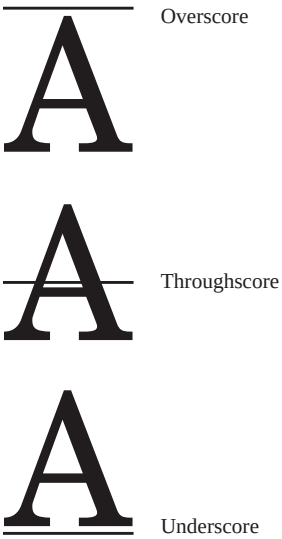
Font designers might recommend the size and position of the subscript and superscript characters in a font.

If a font does not include all of the required subscript and superscript characters for an application, an application process can use the information to create the required characters in the size and position recommended by the font designer. If the characters of the font have a slope (italic or oblique characters), the reference point for superscript and subscript characters is normally adjusted to compensate for the angle of the slope.

Overscores, Throughscores, and Underscores

When graphic characters are presented, a horizontal bar can also be presented over, through, or under the graphic character. Bars over the character are called overscores, bars through the character are called throughscores, bars under the character are called underscores. Figure 37 shows an overscore, a throughscore, and an underscore.

Figure 37. Overscore, Throughscore, and Underscore



Recommendations for Overscores and Throughscores

Designer recommendations for overscores and throughscores in a font can be specified as follows:

- As a displacement of the top of the score stroke-width from the character baseline:
  - A positive value specifies that the top of the overscore or throughscore stroke-width is positioned above the character baseline.
  - A value of zero specifies that the top of the overscore or throughscore stroke-width is coincident with the character baseline.
  - A negative value specifies that the top of the overscore or throughscore stroke-width is positioned below the character baseline.
- As the width (thickness) of each overscore or throughscore stroke for implementation by the processing system

Recommendations for Underscores

Designer recommendations for underscores in a font can be specified as follows:

- As a displacement of the top of the score stroke-width from the character baseline (see "Underscore Position" on page 88):
  - A positive value specifies that the top of the underscore stroke-width is positioned below the character baseline.
  - A value of zero specifies that the top of the stroke-width is coincident with the character baseline.
  - A negative value specifies that the top of the underscore stroke-width is positioned above the character baseline.
- As the width (thickness) of each underscore for implementation by the processing system

Non-Latin Language Support

This font architecture fully supports all requirements for national language support that were known while the architecture was being developed, including multidirectional text and multi-byte document encoding. It is, however, the responsibility of implementing products to provide the necessary collections of font information required to support the national language variations required by their product. That is, the font architecture provides for the definition of metric information for support of multiple character rotations, but the implementing product is responsible for providing the character positioning information for those rotations.

The following sections describe how the concepts of the architecture, which are most often described in terms of the left-to-right Latin writing system, are applied to non-Latin writing systems.

Character Rotation

The principle distinctive characteristic of non-Latin writing systems is presentation of text in a top-to-bottom direction or in a right-to-left direction. FOCA addresses this characteristic by permitting the font designer to provide metric information for multiple character rotations in a single font resource. The 0° and 180° character rotations are normally used for horizontal writing and the 90° and 270° character rotations are normally used for vertical writing.

Character rotation is specified in degrees and minutes. A font designer must specify at least one rotation for the characters in a font and—depending on product and user requirements—more angles of rotation might be necessary.

Positioning values might change for each rotation, which is shown in Figure 38 through Figure 41 on page 48.

The A-space, B-space, and C-space for a character vary for each rotation. These font metrics are specified by the font designer.

Figure 38. 0° Rotation

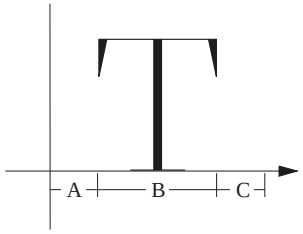


Figure 39. 90° Rotation

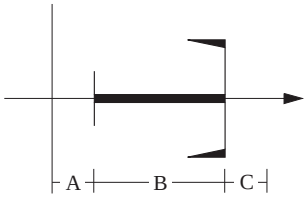


Figure 40. 180° Rotation

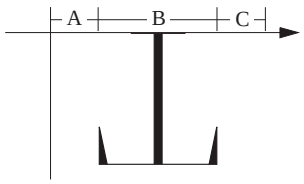
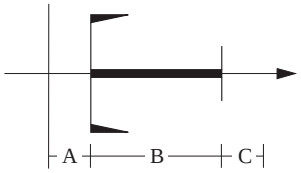


Figure 41. 270° Rotation

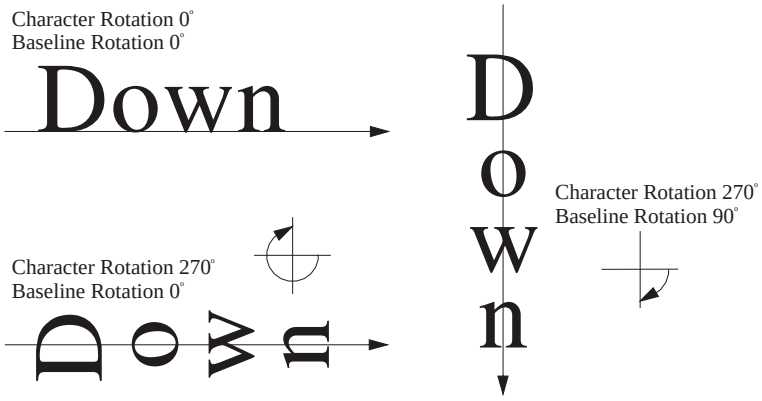


Rotated Baseline and Character Boxes

If text is to be read top-to-bottom as in traditional Kanji, the baseline is rotated 90°, and the characters are presented rotated 270°. The presentation of baseline rotated 90°, and characters boxes rotated 270° is illustrated in Figure 42.

Note: Rotation of characters relative to the baseline is defined by FOCA. Rotation of the baseline is defined outside of FOCA.

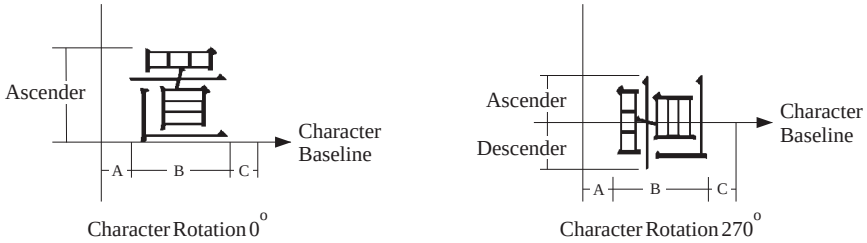
Figure 42. Rotating Baseline and Characters. (Baseline rotated 90° and Characters rotated 270°)



Eastern Writing Systems

In the Eastern writing systems, such as Japanese Kanji, graphic characters can be presented in either the left-to-right direction (using font metric information for characters rotated 0°, see Figure 43) or the top-to-bottom direction (using font metric information for characters rotated 270°, see Figure 43). Thus, font designers must either specify metric information for two character rotations in one font resource or create two font resources, one for each character rotation.

Figure 43. Two Rotations of a Kanji Character



The following diagrams illustrate in a general way how the character and font concepts of this chapter apply to Japanese Kanji. Specific details are provided with each parameter definition of Chapter 5, “FOCA Parameters”, on page 55.

Figure 44. 0° Character Rotation for Horizontal Writing

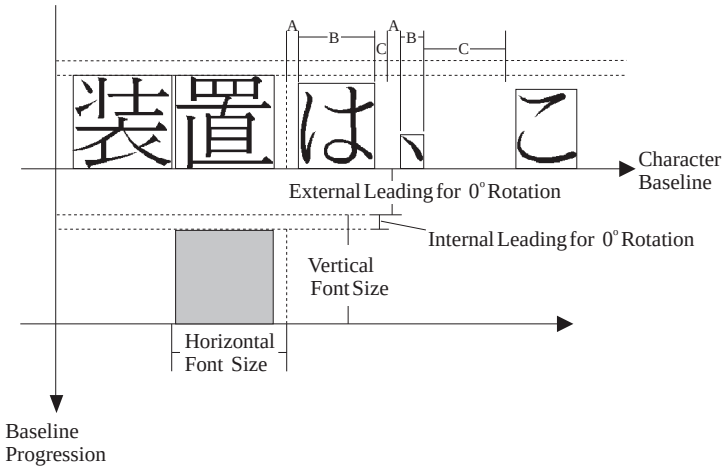
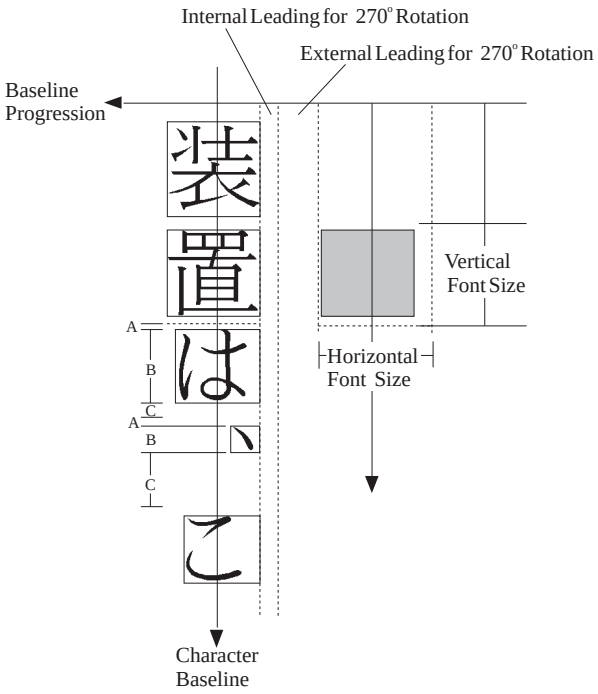
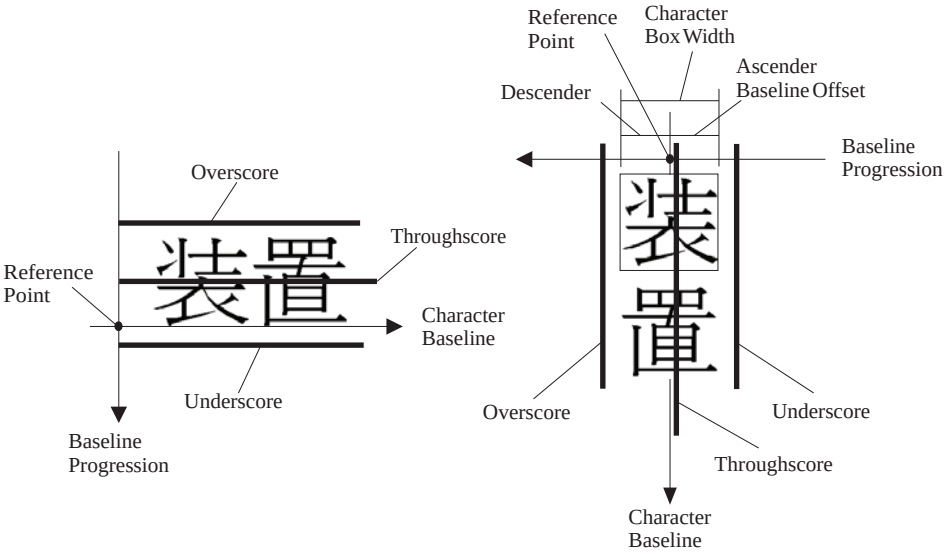


Figure 45. 270° Character Rotation for Vertical Writing



The terminology for overscore, underscore, and throughscore were defined with Latin text in mind, but apply to Eastern text as shown in Figure 46.

Figure 46. Example of Score Positioning



Middle Eastern Writing Systems

The Hebrew alphabet consists of 27 characters. Five of these characters are final forms but in a font are treated like any other character. Uppercase and lowercase letters do not exist in Hebrew.

Figure 47. Example of Hebrew Text

Hebrew is written from right-to-left but the design of each font is performed with the same rules as for Latin fonts. Slanted fonts are permitted to both sides. Usually the oblique or italic fonts are slanted to the right in order to avoid conflict with the italic Latin fonts. For text containing only Hebrew fonts the correct typographic design should be an inclination to the left, according to the reading direction.

Kerning could be used in some cases but it is not necessary in order to obtain a correct font.

The Hebrew code pages are bilingual Latin/Hebrew code pages to permit mixed text in the same environment. In a normal business environment the Hebrew text will generally be mixed with Latin text. The numbers and punctuation marks are the same as in the Latin font.

For this reason the Hebrew fonts should match the following general rules in order to provide consistent page presentation.

- Hebrew characters can be divided into two major categories, wide and thin characters. The wide Hebrew characters (for example, alef and shin) should have similar width values as the wide Latin characters (for example, H and W) contained in the same bilingual font.
- The height of the Hebrew characters should be approximately 1/3 lower than the difference between the uppercase and lowercase for Latin. The result should be a font slightly lower than the uppercase Latin font.
- The space character used for Hebrew should be the same as for the matching Latin font.
- Hebrew characters with descenders will use all the permitted descender space for better readability.
- The typical vertical stem width of the Latin characters in the same font should equal the typical horizontal stem of the Hebrew characters. Modern Hebrew sans-serif fonts can have the same width for the horizontal and vertical parts of the characters.
- In case of a low resolution device, the Hebrew fonts are usually designed with the same height as the Latin uppercase in order to improve readability.

Non-AFP Architecture Support

FOCA is only one of many font architectures that exist in the world of information processing. Many companies have their own architecture for internal consumption, while others use industry, national, or international standard architectures. The following sections, along with specific details provided in association with each of the FOCA parameters in Chapter 5, “FOCA Parameters”, on page 55, provide the information necessary to transform FOCA information between other widely accepted font architectures.

ISO 9541-1 Font Architecture

The ISO/IEC 9541-1 *Information technology—Font information interchange: Part 1: Architecture* document provides the semantic definition of a comprehensive set of properties (parameters) for use in creating font resources or font references for interchange at the global level. The specific syntactic representation of that

information is defined in ISO/IEC 9541-2 *Information technology—Font information interchange: Part 2: Interchange formats*.

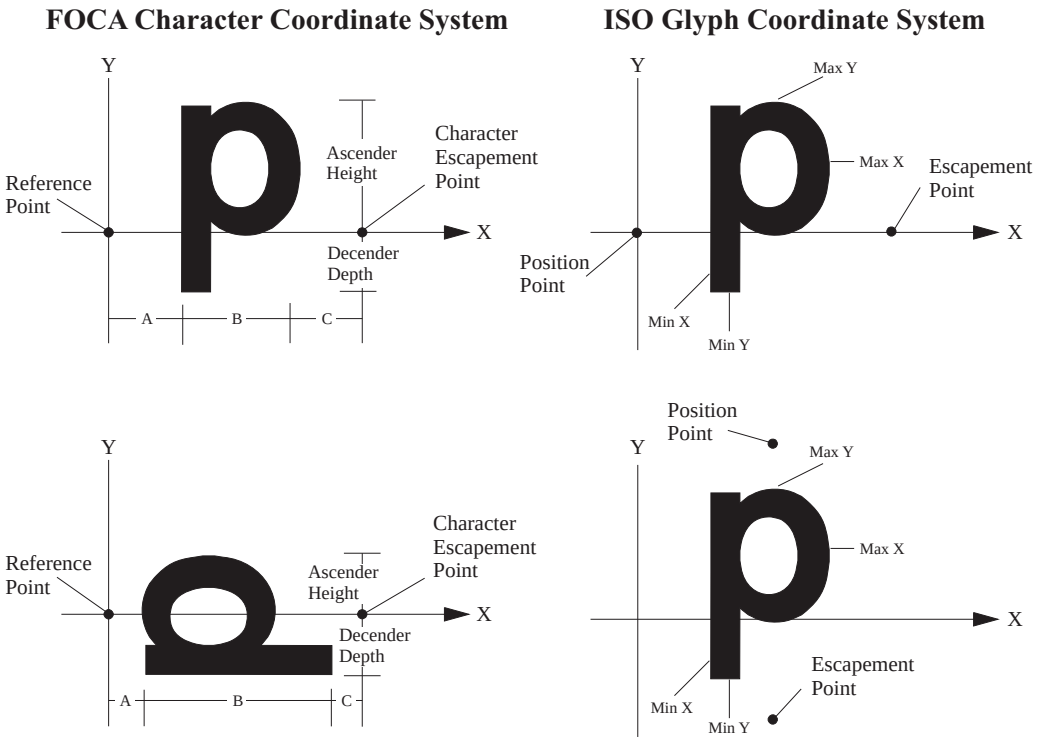
The following sections describe the general terminology and coordinate system differences between the ISO and AFP font architectures, and the global naming concepts used in the ISO standard to assure globally unique naming of information technology objects (for example, typeface names, family names, and resource names).

Coordinate System

ISO/IEC 9541-1 describes font resource information in terms of x and y positions within a Cartesian glyph coordinate system. This method of describing font metric information differs from the FOCA use of distances or offsets in a similar Cartesian coordinate system. In addition, the ISO standard permits specification of the glyph positioning point anywhere in the glyph coordinate system, while FOCA requires the positioning point to remain fixed at the origin. This difference permits the ISO glyphs to always appear upright in the glyph coordinate for different writing modes, while the AFP method requires rotation of the character in the coordinate system to achieve the same result. The two systems are equivalent, but require transformation of metric information when going from one system to the other.

Figure 48 illustrates in a general way how the FOCA character coordinate system and the ISO glyph coordinate system relate to each other, for left-to-right and top-to-bottom writing systems. Specific details for transforming from one system to the other are provided with each applicable parameter definition of Chapter 5, “FOCA Parameters”, on page 55.

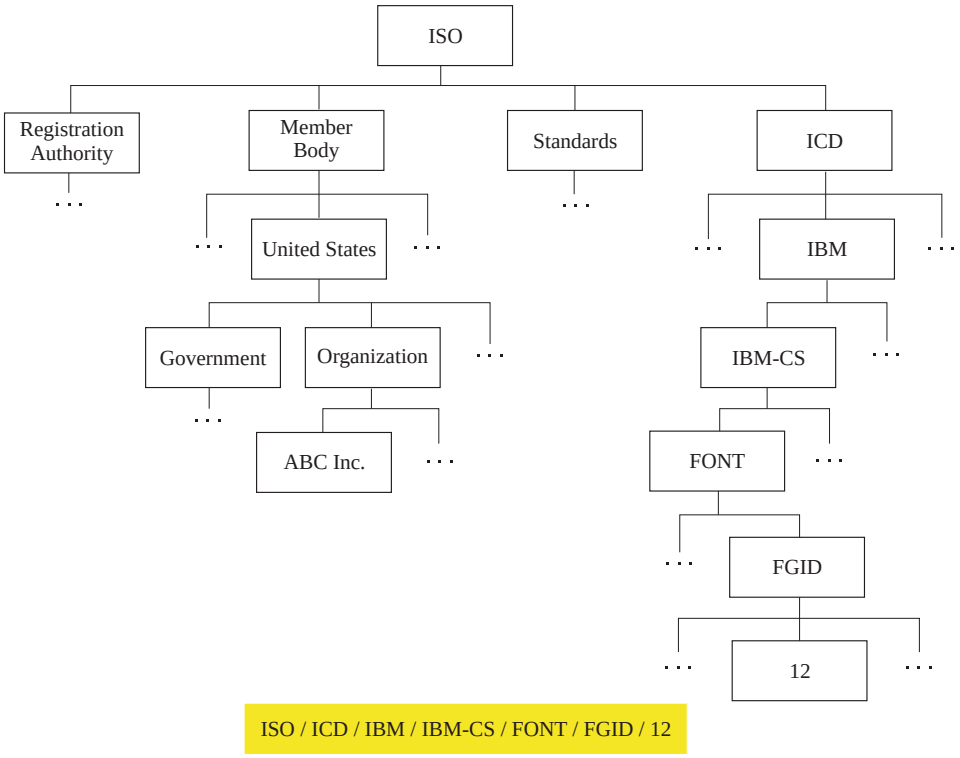
Figure 48. FOCA and ISO Coordinate System Relationship



Global Naming

ISO uses a hierachically structured naming tree, rooted at the international level, for naming information objects (see Figure 49). The organization of the tree and its sublevel branches permit organizations to graft in existing internally defined names without serious incontinuity or redefinition.

Figure 49. ISO Hierarchical Naming Tree



The structure permits organizations to define their own local names for entities and to then use those names in progressively broader environments by prepending owning branches of the naming tree. Thus, an AFP Font Typeface Global ID might be identified by only the local identifier 12 if the context of its use is known, or it might be identified by the name FONT/FGID/12 within an IBM operating environment if the context of its use is not known, or it might be identified by the name ISO/ICD/IBM/IBM-CS/FONT/FGID/12 within the global environment if the context of its use is not known.

ISO/IEC 9541 structured names for the representation of font information is in two forms: ASN.1 numeric identifiers and SGML clear text names. The IBM FONTS structured-name prefix using ASN.1 is 1-3-18-0-1. Using SGML it is ICD0018/IBM/ /IBM-CS/FONTS.

Chapter 5. FOCA Parameters

Digital data processing requires that you create data structures using a defined set of parameters and parameter values organized in a specific format. FOCA provides a precise set of parameter definitions and architected values or ranges of values you can use to create font resources and font references.

This chapter describes the conventions for defining FOCA parameters. Then, it defines the available FOCA parameters, dividing them into the following five categories:

- Font-description parameters
- Font-metric parameters
- Character-metric parameters
- Character-shape parameters
- Character-mapping parameters

Defining FOCA Parameters

This section describes three parameter formats, the parameter types available for each format, and byte and bit numbering conventions.

Parameter Formats

FOCA supports a variety of parameter formats, that is, types of syntax. The choice of format depends on the environment where you want to use the font resource. The following are three common formats supported by FOCA. See Chapter 6, “Font Interchange Formats”, on page 111 for more detailed information about the formatting standards necessary to implement FOCA.

Fixed-Format

A fixed-format parameter is defined by its position and length within a string of fixed-format parameters. The variable name and its associated meaning is implied by the position of the parameter in the string. The position and length assigned to each fixed-format parameter is defined in supporting documentation or elsewhere in the resource object. FOCA support of fixed-format parameters requires a precise definition of parameters and the type of values supported.

Self-Identifying

A self-identifying parameter has a set of fields that identify the parameter, specify its length, and specify its values. The definition of these fields is specified by the implementing product or architecture documentation.

Clear-Text

A clear-text parameter has two fields, which are separated by a delimiter. The first field contains a character string that is the name of the parameter and the second field contains a character string that represents the value of the parameter. The delimiter character and the set of characters permitted is defined in the implementing product or architecture documentation.

Parameter Types

You define a FOCA parameter by identifying it and assigning it a value. The remaining sections of this chapter list the names of the parameters and show **Parameter type =** as requiring a value in the form of one of the following data types:

Character string — A character string parameter value is any user, system, or font-supplier defined name. It is composed of alphanumeric characters and can be any specified length. An example of a character string is the font family name *Sonoran Serif*.

Unless otherwise specified (see "Graphic Character Set Global Identifier" on page 61), the default set of graphic characters for character string data is the set of graphic characters specified by IBM Graphic Character Set 103. That set consists of 94 characters: uppercase and lowercase A–Z, the numerals 0–9, and the 32 special characters + < = > $ ` ^ ~ # % & * @ [\ ] { | } ! " ' ( ) , _ - . / : ; ?

Unless otherwise specified (see "Code Page Global Identifier" on page 103), the default encoding of the graphic characters for character string data is the IBM Interchange Code Page for the environment in which the resource is being used.

Because font objects can be interchanged among environments, all syntactic representations of font resources and font references should explicitly specify the Graphic Character Set Global ID and the Code Page Global ID used for character string encoding.

Flag A flag parameter value indicates a *binary flag* and refers to a field interpreted as a binary bit, which has a binary condition value of on (1), or off (0). Each flag bit can occur in a string of binary flag bits as an independent variable, the setting of which does not affect the setting of another flag bit in the same binary bit string.

Number A number parameter value uses real numbers or integer numbers to represent count or magnitude.

Some quantities such as counts must be positive, but others such as measurements can be either positive or negative. Numbers are assumed positive unless otherwise stated.

Code A code parameter value is a collection of architected choices. In general, parameters having the code parameter type use a code (integers, letters, or acronyms) to identify the architected choices. Specific interchange formats might use different code assignments to identify the choices in the list or set, but those assignments must be unambiguously mappable to the choices defined in this architecture. For reference purposes, this architecture assigns integer codes to each of the architected choices.

Undefined A undefined parameter value is not defined by the architecture.

Byte and Bit Numbering

The byte and bit numbering conventions used in this publication follow those used within IBM System 370 Principles of Operation.

Byte numbering During both transmission and storage, data is viewed as a long horizontal string. For most operations, access to data is left to right. The string of data is divided into units of eight bits called bytes, which are the basic units of all data.

Each byte is identified by a positive integer, which is the address (offset) of that byte within the data. Adjacent byte locations are identified by consecutive integers starting with 0, which represent consecutive addresses.

Bit numbering Bytes are divided into eight bits. The bits are numbered, left to right, from 0 to 7. The four bits on the left are sometimes referred to as the *high-order* and the four bits on the right as the *low-order* bits. The bit numbers are not storage addresses: only bytes can be addressed. To change the value of individual bits in a byte, it is necessary to perform operations on the entire byte.

Font-Description Parameters

This section lists and describes the descriptive parameters required to identify a font, select the appropriate font for formatting, and locate the specified font for presentation. In general, most of the parameters defined in this section will be used in both font resources and font references (the process of locating a referenced font resource requires matching the parameter values in the font reference to the parameter values in the available

font resources). Where there are exceptions, the parameter definition will distinguish between font resource usage and font reference usage.

Average Weighted Escapement

The Average Weighted Escapement parameter specifies the arithmetic average of the escapement of all, or some subset of, the characters in a font. The escapement value for each character is weighted by its anticipated frequency of use.

The average weighted escapement is computed as follows: each character increment is multiplied by its frequency of use, the products of this multiplication are totaled, and the total is divided by 1000 times the number of characters for which this average is being computed. This parameter is a descriptive attribute of the font that specifies character spacing that is used when comparing fonts for font selection or substitution. When comparing fonts, units of measure, character content, and frequency-of-use values for each font must be known.

When computing a weighted average, based on some subset of the font characters (such as the lowercase Latin alphabet), the frequency of use value for the desired characters should be set as required and set to zero for all other characters. The computation of weighted average is calculated by using 1000 times the number of characters in the subset, not the number of characters in the font.

If an implementation compares fonts using this value, it must either strictly control the set of frequency values applied to the font characters to attain this value, or use a single frequency table to compute the values for each font before making the comparison. Fonts that are interchanged among different environments might have different character content or frequency-of-use values.

To aid in the comparison of Latin-based fonts, a default set of frequency-of-use values is provided in the definition of the Frequency of Use parameter. This set of values can be used explicitly to compute the average weighted escapement, or can be used implicitly by providing a nonzero value of average weighted escapement (using the 27 default frequency-of-use values) without providing any frequency-of-use values.

Parameter type = *number*
Synonyms = *average character width*
Transformation to Eastern Writing systems
This parameter does not apply to Eastern writing systems.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Modal Properties property list.

Cap-M Height

The Cap-M Height parameter specifies the height above the baseline for uppercase character shapes. Cap-M height is the nominal height of the uppercase characters and is usually equal to the height of the uppercase letter *M*. The cap-M height value is specified by a font designer.

Parameter type = *number*
Transformation to Eastern Writing systems
This parameter should be set to the character box height for Eastern writing systems.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to *capheight* (Capitol Height). It is expressed as a relative rational number.

Character Rotation

The Character Rotation parameter specifies the rotation of the character box relative to the character baseline. Refer to "Units of Direction" on page 36 for an explanation of character rotation. A user selects various writing modes by specifying the appropriate character rotation.

FOCA permits the definition of character shapes that can be used for all rotations. A given character shape can normally be rotated or translated to any position in the presentation space by using a variety of techniques. However, to maintain the character spacing specified by a font designer, the information for character positioning relative to the baseline must be specified for each required rotation of the character image.

Parameter type = *number*
Synonyms = *font character rotation*
Transformation to Eastern Writing systems
This parameter should be set to 0° or 270° for horizontal or vertical writing respectively.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to *nomescdir* (nominal escapement direction), and to *nomwrmode* (Nominal Writing Mode). The *nomescdir* is expressed as a rational angle, measured counterclockwise from the positive x-axis. For both horizontal and vertical writing, the AFP Character Rotation and ISO Escapement Direction values are the same. The *nomwrmode* property is expressed as the global name of the corresponding nominal escapement direction (for example, ISO/IEC 9541-1/ /left-to-right).

Comment

The Comment parameter allows the creator or the user of the font resource to make comments. The content of this parameter must be non-processable information and is ignored by any processing implementation.

Parameter type = *character string*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
This parameter can be expressed as a *non-iso-property*, wherever the *non-iso-property* is permitted.

Design General Class (ISO)

The Design General Class parameter specifies the ISO (International Standards Organization) Font Standard General Classification of the font family design. This parameter is intended for use in selecting an alternate font when the requested font is not available. The General Class parameter is the least specific, the Subclass parameter more specific, and the Specific Group parameter the most specific of the Design Class parameters.

Parameter type = *number*
Synonyms = *design class*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to the class subfield of *dsngroup* (Design Group). The ISO property is a code in the range of 0 to 255.

Design Specific Group (ISO)

The Design Specific Group parameter specifies the ISO (International Standards Organization) Font Standard Specific Group of the font family design. This parameter is intended for use in selecting an alternate font when the requested font is not available. The General Class parameter is the least specific, the Subclass parameter more specific, and the Specific Group parameter the most specific of the Design Class parameters.

Parameter type = *number*
Synonyms = *design class*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to the specific group subfield of *dsngroup* (Design Group). The ISO property is a code in the range of 0 to 255.

Design Subclass (ISO)

The Design Subclass parameter specifies the ISO (International Standards Organization) Font Standard Subclass of the font family design. This parameter is intended for use in selecting an alternate font when the requested font is not available. The General Class parameter is the least specific, the Subclass parameter more specific, and the Specific Group parameter the most specific of the Design Class parameters.

Parameter type = *number*
Synonyms = *design class*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to the subclass subfield of *dsngroup* (Design Group). The ISO property is a code in the range of 0 to 255.

Em-Space Increment

The Em-Space Increment parameter specifies typographic space that corresponds to the space between sentences. An Em-Space Increment is a formatting dimension that traditionally equals the vertical font size. This value normally has a relative value of one (equal to the Unit-Em; see "Units of Measure" on page 34).

Parameter type = *number*
Synonyms = *em increment*
Transformation to Eastern Writing systems
This parameter does not apply to Eastern Writing systems.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to the difference between the *px,py* (Positioning Point X,Y) and the *ex,ey* (Escapement Point X,Y) values for the Em-space glyph, if it occurs in the subject font resource. The ISO values are expressed as relative rational numbers with an x component, a y component, or both.

Extension Font

The Extension Font parameter indicates that this font resource was designed to be an extension (contains user-defined characters) to another font (a base font containing a set of general-use characters).

Parameter type = *flag*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Description property list.

Family Name

The Family Name parameter specifies the common name for a font design. A font family includes all typeface variations of the font design. The font family name should correspond to the family designation as it appears in the appropriate product documentation.

The font-family name is specified by a font designer, for example, *Sonoran Serif*.

Parameter type = *character string*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to *fontfamily* (Font Family). It is expressed as a character string value.

Font Local Identifier

The Font Local Identifier parameter specifies a numeric identifier assigned temporarily to a font resource within the context of another object. The scope of the identifier is limited in time and space to the data stream in which the font resource is being carried or referenced for use.

This parameter provides a short, unique tag by which the font object can be locally identified for reference between functional entities. It is used in font references or font maps contained within a document data stream.

Parameter type = *number*
Synonyms = *font local ID, font LID*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter represents a temporary identifier in time and space, and generally should not be required in any interchange environment. If used, it should be expressed as a *non-iso-property* in the Font Description property list.

Font Typeface Global Identifier

The Font Typeface Global Identifier parameter (usually called Font Global Identifier, FGID) specifies the unique number assigned to the font typeface. The Font Typeface Global ID numbers that are supported are specified in product documentation. Font Typeface Global IDs 1 through 65,279 are reserved for assignment by AFP.

Parameter type = *number*
Synonyms = *font global identifier, FGID, registry identifier, font-standard identifier, typeface global identifier, typeface identifier*
Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Description property list.

Font Use Code

The Font Use Code parameter specifies the intended use of the graphic characters in a font. This parameter permits font designers to specify their intent for fonts.

Parameter type = *code*
Valid choices:

- **0** No font use intent
- **1** Image symbol set for text in a graphics object
- **3** Pattern symbol set in a graphics object
- **4** Marker symbol set in a graphics object
- **5** High resolution indicator in a graphics object

Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Description property list.

Graphic Character Set Global Identifier

The Graphic Character Set Global Identifier (GCSGID) parameter specifies the number assigned to a graphic character set, which identifies the set of graphic characters contained in the font resource. The Graphic Character Set Global ID numbers that are supported, are specified in product documentation. Graphic Character Set Global IDs 1 through 65,279 are reserved for assignment by IBM.

Parameter type = *number*
Synonyms =*graphic-character-set ID, graphic-character-set name, character-set name, character complement, character-collection name, collection name*
Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to a single *incglyphcols* (Included Glyph Collections) field of the *glyphcomp* (Glyph Complement) property. The ISO property uses the full ISO structured name, and any transform to the ISO format should prepend the appropriate name prefix (see "Global Naming" on page 53). The ISO

glyphcomp property permits specification of one or more included or excluded collections, and one or more included or excluded glyphs.

Hollow Font

The Hollow Font parameter specifies that the graphic characters of the font appear with only the outer edges of the strokes. If the Hollow font flag is off (0), the graphic characters do not have a hollow appearance. If this flag is on (1), the graphic characters do have a hollow appearance.

Parameter type = *flag*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to the Outline code (2) of the *structure* (Structure Code) property. The ISO property is a code, identifying two different structures (*solid* and *outline*).

Italics

The Italics parameter specifies that the graphic characters are designed with a clockwise incline. If this flag is off (0), the graphic character shapes have no clockwise italic design. If this flag is on (1), the character shapes have a clockwise italic design.

Parameter type = *flag*
Transformation to Eastern Writing systems
This parameter does not apply to Eastern writing systems.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to the italic value (4) of the ISO *posture* (Posture Code) property. The ISO property is a code, identifying five different posture combinations (upright, forward italic, backward italic, forward oblique, and backward oblique).

Kerning Pair Data

The Kerning Pair Data parameter specifies that kerning pair data is available in the font resource. If this flag is off (0), no kerning pair data is available in the font resource. If this flag is on (1), kerning pair data is available for one or more character rotations.

Parameter type = *flag*
Transformation to Eastern Writing systems
This parameter does not apply to Eastern writing systems.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter indicates the presence or absence of kerning pair data in the font resource. In ISO the presence or absence of kerning pair data is indicated by the presence or absence of the *peas* (Pairwise Escapement Adjusts) data itself. This AFP flag can be represented as a *non-iso-property* in the Font Description property list.

Maximum Horizontal Font Size

The Maximum Horizontal Font Size parameter specifies the maximum horizontal size for scaling, as specified by a font designer. This value generally corresponds to the maximum character escapement value of a space character (SP010000) in a font of the Nominal Vertical Font Size. The value is used to calculate a maximum scaling ratio, which can be applied to the width of all characters of the font.

This parameter only occurs in font resources and does not occur in font references. The parameter is used to determine the permitted maximum for shape manipulation.

This parameter should be expressed in fixed measurement units, not relative measurement units, and it should correspond to the physical size of the font when presented on an output medium that is to be viewed at a normal reading distance of 14 inches.

This parameter is often used for graphic-display fonts or for Asian languages where scaling can be different in the horizontal and vertical directions.

The maximum horizontal font size is specified as the maximum horizontal increment to which the space character (as a representative character of all characters in the font) can be scaled, given the vertical size as specified by the nominal vertical font size parameter. The following two examples show how the scaling ratio can be derived and used.

Example 1:

Assume a fixed-pitch font in which all characters (including the space character) have a character increment of 1/12th inch (6 points), a nominal vertical font size of 10 points, and the font designer specifies a maximum horizontal font size of 12 points.

```
Nominal Vertical Font Size = 10 Points
Nominal Horizontal Font Size = 6 Points
Maximum Horizontal Font Size = 12 Points
```

The maximum scaling ratio for all characters in this case is 2/1 (12 points divided by 6 points). If the font is vertically scaled to 22 points, the nominal horizontal font size scales to 13.2 points (22/10 * 6). The maximum character increment for each of the font's characters is 26.4 points (2/1 * 13.2).

Example 2:

Assume a proportional font in which all character increments are different and in which the space character has a character increment of 4 points at a nominal vertical font size of 12 points, the letter A has a character increment of 8 points at a nominal vertical font size of 12 points, and a font designer specified maximum horizontal font size of 8 points.

```
Nominal Vertical Font Size = 12 Points
Nominal Horizontal Font Size = 4 Points
Maximum Horizontal Font Size = 8 Points
```

The maximum scaling ratio for all characters in this case is 2/1 (8 points divided by 4 points). If the font is vertically scaled to 36 points, the nominal horizontal font size scales to 12.0 points (36/12 * 4). The maximum character increment for the letter *A* becomes 48 points (2/1 * 8 * 36/12).

Parameter type = ***number***
Synonyms = ***maximum character width, maximum space-character width, maximum horizontal size, maximum horizontal point size***
Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to the writing-mode-dependent *maxanascale* (Maximum Anamorphic Scale) property. The ISO property is expressed as a rational number corresponding to the ratio of the maximum to

nominal (for example, 2/1) size in the escapement direction, while the AFP parameter is expressed as a maximum horizontal size from which the ratio is computed.

Maximum Vertical Font Size

The Maximum Vertical Font Size parameter specifies the maximum vertical size for scaling purposes as specified by a font designer.

This parameter only occurs in font resources and does not occur in font references. The parameter is used to determine the permitted maximum for shape manipulation.

The Maximum Vertical Font Size parameter should be expressed in fixed measurement units not relative measurement units, and it should correspond to the physical size of the font when presented and viewed at a normal reading distance of 14 inches.

This parameter is an indicator of the maximum valid size for the character-metrics values in this font. It is not necessarily the maximum size that character images can be scaled. For font metrics that cannot or should not be scaled, this parameter should be the same as the value of the Nominal Vertical Font Size parameter.

Parameter type = *number*
Synonyms *maximum font vertical point size, maximum design size, maximum point size, maximum body size, maximum size*
Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to *maxsize* (Maximum Design Size). The ISO property is expressed as a rational number in millimeters.

Measurement Units

The Measurement Units parameter contains four values that specify the unit base in the X direction, the unit base in the Y direction, the units per unit base in the X direction, and the units per unit base in the Y direction, respectively. Refer to "Units of Measure" on page 34 for information about the units of measure. These values specify how the sizes of the components of the font are expressed.

For font resources (minimum, nominal, and maximum vertical font size), values must always be expressed in fixed units of measure, which permits correct interpretation of minimum and maximum scaling. Relative units can be used for all other character measurements. For consistency, specify all relative measurement units in 1/1000th of a Unit-Em. The default measurement unit, to be used for all rational numbers in the absence of this parameter, is 1/1440th of an inch.

This parameter can occur in font references or font resources. If the value is different in the font reference from that found in a font resource, an application must normalize any associated metric values before performing a compare.

Parameter type = *two codes and two numbers*
Valid unit-base code choices:

- **0** Ten inches
- **1** Ten centimeters
- **2** Relative units

Synonyms = *base units, relative units*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to *relunits* (Relative Units). This parameter could be expressed as a *non-iso-property* in the Font Description property list.

MICR Font

The MICR Font parameter indicates that this font resource was designed for use in Magnetic Ink Character Recognition (MICR) applications.

Parameter type = *flag*
Transformation to Eastern Writing systems
This parameter does not apply to Eastern writing systems.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Description property list.

Minimum Horizontal Font Size

The Minimum Horizontal Font Size parameter specifies the minimum horizontal size for scaling, as specified by a font designer. The value of the parameter corresponds to the minimum character escapement value of a space character (SP010000) in a font of the Nominal Vertical Font Size. The value is used to calculate a minimum scaling ratio, which can be applied to the width of all characters of the font. This parameter is often used for fonts in graphic displays or in Asian languages where scaling is permitted in horizontal and vertical directions.

This parameter only occurs in font resources and does not occur in font references. The parameter is used to determine the permitted minimum for shape manipulation.

This parameter should be expressed in fixed measurement units, not relative measurement units. It should correspond to the physical size of the font when presented to be viewed at a normal distance of 14 inches.

The minimum horizontal font size is specified as the minimum horizontal increment to which the space character (as a representative character of all characters in the font) can be scaled, given the vertical size as specified by the nominal vertical font size parameter. The following two examples show how the scaling ratio can be derived and used.

Example 1:

Assume a fixed-pitch font in which all characters (including the space character) have a character increment of 1/12th inch (6 points), a nominal vertical font size of 10 points, and the font designer specifies a minimum horizontal font size of 3 points.

```
Nominal Vertical Font Size = 10 Points
Nominal Horizontal Font Size = 6 Points
Minimum Horizontal Font Size = 3 Points
```

The minimum scaling ratio for all characters in this case is 1/2 (3 points divided by 6 points). If the font is vertically scaled to 22 points, the nominal horizontal font size scales to 13.2 points (22/10 * 6). The minimum character increment for each of the font's characters is 6.6 points (1/2 * 13.2).

Example 2:

Assume a Proportional Font in which all character increments are different and in which the space character has a character increment of 4 points at a nominal vertical font size of 12 points, the letter *A* has a character increment of 8 points at a nominal vertical font size of 12 points; and a font designer specified minimum horizontal font size of 2 points.

```
Nominal Vertical Font Size = 12 Points
Nominal Horizontal Font Size = 4 Points
Minimum Horizontal Font Size = 2 Points
```

The minimum scaling ratio for all characters in this case is 1/2 (2 points divided by 4 points). If the font is vertically scaled to 36 points, the nominal horizontal font size scales to 12.0 points (36/12 * 4). The minimum character increment for the letter *A* becomes 12 points (1/2 * 8 * 36/12).

Parameter type = *number*
Synonyms = *minimum character width, minimum space character width, minimum horizontal size, minimum horizontal point size, minimum anamorphic scaling*
Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to the writing-mode-dependent *minanascale* (Minimum Anamorphic Scale) property. The ISO property is expressed as a rational number corresponding to the ratio of the minimum to nominal (for example, 1/2) size in the escapement direction, while the AFP parameter is expressed as a minimum horizontal size from which the ratio is computed.

Minimum Vertical Font Size

The Minimum Vertical Font Size parameter specifies the minimum vertical size for scaling purposes as specified by a font designer.

This parameter only occurs in font resources and does not occur in font references. The parameter is used to determine the permitted minimum for shape manipulation.

The parameter should be expressed in fixed measurement units not relative measurement units. It should correspond to the physical size of the font when presented to be viewed at a normal distance of 14 inches.

This parameter is an indicator of the minimum valid size for the character-metrics values in this font. It is not necessarily the minimum size that character images can be scaled. For font metrics that cannot or should not be scaled, this parameter should be the same as the value of the Nominal Vertical Font Size parameter.

Parameter type = *number*
Synonyms = *minimum font vertical point size, minimum design size, minimum point size, minimum body size, minimum size*
Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to *minsize* (Minimum Design Size). The ISO property is expressed as a rational number in millimeters.

Nominal Character Slope

The Nominal Character Slope parameter specifies the slope (*stem incline* is the term used in the ISO/IEC 9541 *Font Information Interchange* standard) of the graphic characters of this font. A value of zero for this parameter indicates a positive direction parallel to the vertical axis. Increasing values indicate increasing clockwise directions.

An upright font normally has a character slope of 0°. An italic font normally has a character slope of 17.5° (seventeen degrees and thirty minutes). If nominal appearance of the character has a back slant, the angle is large.

This parameter can occur in font resources or font references. If it occurs in a font resource, it is used to specify the base reference for shape manipulation. If it occurs in a font reference, it is used to specify the desired character slope. A processing application might either select a font resource having the same Nominal Character Slope or might perform shape manipulation (within the range of Minimum to Maximum Character Slope) to obtain the desired result.

Parameter type = *number*
Synonyms = *character slope, nominal font character slope*
Transformation to Eastern Writing systems

This parameter does not apply to Eastern writing systems.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to *postureangle* (Posture Angle). The ISO property is expressed as a rational angle, measured counterclockwise from the positive x-axis, while the AFP value is measured clockwise from the positive y-axis.

Nominal Horizontal Font Size

The Nominal Horizontal Font Size parameter specifies the nominal horizontal size for scaling. This parameter corresponds to the character escapement value of the space character (SP010000) expressed in fixed units of measure.

This parameter can occur in font resources and font references. If it occurs in a font resource, it is used to specify the base reference for shape manipulation. If it occurs in a font reference, it has the same semantic as "Specified Horizontal Scale Factor" on page 70.

The Nominal Horizontal Font Size parameter should be expressed in fixed measurement units (not relative measurement units) and should correspond to the physical size of the font when presented on an output medium that is to be viewed at a normal reading distance of 14 inches.

This parameter specifies the primary font-size indicator for fixed-pitch typewriter fonts. Historically, fixed-pitch fonts were measured in characters per inch and represented with a size value for character width. Typographic font size can be represented using the character width if a standard representative character (the space character) is chosen as the basis for comparison.

Parameter type = *number*
Synonyms = *space character width, horizontal size, horizontal font size, horizontal point size, nominal horizontal point size*
Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Description property list.

Nominal Vertical Font Size

The Nominal Vertical Font Size parameter specifies the vertical size (*design size* is the term used in the ISO/IEC 9541 *Font Information Interchange* standard) of the font as specified by a font designer. The Nominal Vertical Font Size parameter specifies the nominal size for which the character-metric values of this font are defined.

This parameter can occur in font resources and font references. If it occurs in a font resource, it is used to specify the base reference for shape manipulation. If it occurs in a font reference, it has the semantic of specifying the desired size of the font.

The Nominal Vertical Font Size parameter should be expressed in fixed measurement units (not relative measurement units) and should correspond to the physical size of the font when presented on an output medium that is to be viewed at a normal reading distance of 14 inches.

Parameter type = *number*
Synonyms = *nominal point size, font vertical point size, design size, point size, body size, nominal size*
Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to *dsnsize* (Design Size). The ISO property is expressed as a rational number in millimeters.

Overstruck Font

The Overstruck Font parameter specifies that the graphic characters of the font appear as though over-struck by another graphic character (often a hyphen graphic character). If this flag is off (0), the graphic characters in the font are not overstruck. If this flag is on (1), they are overstruck.

Note: This is not the same as the Throughscore parameter (see "Throughscore Position" on page 87).

Parameter type = *flag*
Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This characteristic should be represented by a unique font family name. This AFP parameter could also be expressed as a *non-iso-property* in the Font Description property list.

Proportional Spaced

The Proportional Spaced parameter specifies that the character increments for each graphic character in the font resource might vary. If this flag is off (0), the font is monospaced (all characters of the font have the same character increment). If this flag is on (1), the font is proportionally spaced (some characters of the font have different character increments).

Parameter type = *flag*
Transformation to Eastern Writing systems
This parameter does not apply to Eastern writing systems.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to the proportional value (2) of the modal *escclass* (Escapement Class) property. The ISO property is a code, identifying two different escapement classes (*monospaced* and *proportional*).

Private Use

The Private Use parameter specifies that some or all of the data contained in this font resource is privately owned or protected by a licensing agreement. If this flag is off (0), the font is considered to be in the public domain and can be modified, interchanged, or captured for use by other users; if this flag is on (1), the font is considered to be privately owned, or subject to a licensing agreement, and should not be modified, interchanged, or captured for use by any other users.

Parameter type = *flag*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Description property list.

Resource Name

The Resource Name parameter identifies a resource object by a character string name. This parameter should be a short name by which the resource object can be identified uniquely from among any other resource object in a system or distributed data environment. Assignment and management of resource name uniqueness depends upon the system environment in which the resource objects are used.

Parameter type = *character string*
Synonyms = *resource tag*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to *fontname* (Font Resource Name). The ISO property uses the full ISO structured name, and any transform to the ISO format should prepend the appropriate name prefix (see "Global Naming" on page 53).

Specified Horizontal Font Size

The Specified Horizontal Font Size parameter specifies the horizontal font size indicated by the document creator or originator. The Specified Horizontal Font Size is specified in 20ths of a point (1440ths of an inch).

This parameter only occurs in a font reference. A processing application might either select a font resource having the same Nominal Horizontal Font Size or might scale an outline font resource (within the range of Minimum to Maximum Horizontal Font Size) to obtain the desired result.

Parameter type = *number (registered number)*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Description property list.

Specified Horizontal Scale Factor

The Specified Horizontal Scale Factor parameter specifies uniform or anamorphic scaling of the graphic characters and their associated metric information (for example, Character Increment). The value corresponds to the numerator in a ratio consisting of the Specified Horizontal Scale Factor divided by the Specified Vertical Font Size. The Specified Horizontal Scale Factor is specified in 20ths of a point (1440ths of an inch), and must be a positive integer, greater than or equal to one.

This parameter only occurs in a font reference. If the value specified is equal to the Specified Vertical Font Size, uniform vertical and horizontal scaling of the graphic characters occurs. If the value specified is greater or less than the Specified Vertical Font Size, the graphic characters and their corresponding metric values are stretched or compressed in the horizontal direction relative to the vertical direction by the ratio indicated.

Parameter type = *number*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO font resource architecture. This parameter, which appears only in data stream font references, is used to express the desired anamorphic scaling to be applied to the graphic characters in a font resource. The AFP Specified Horizontal Scaling Factor, if used for selection and scaling of an ISO font resource, should define a ratio that is within the range of the ISO Minimum and Maximum Anamorphic Scale Factor values.

Specified Vertical Font Size

The Specified Vertical Font Size parameter specifies the vertical font size indicated by the document creator or originator. The Specified Vertical Font Size is specified in 20ths of a point (1440ths of an inch).

This parameter only occurs in a font reference. A processing application might either select a font resource having the same Nominal Vertical Font Size or might scale an outline font resource (within the range of Minimum to Maximum Vertical Font Size) to obtain the desired result.

Parameter type = *number (registered number)*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to *dsnsize* (Design Size). The ISO property is expressed as a rational in units of millimeters. This property, used as a specified font size, should only occur in a font reference.

Transformable Font

The Transformable Font parameter specifies that the pattern data of this font resource is expressed using algorithmic techniques that permit transformation of the graphic character shapes (for example, scaled to different sizes). If this flag is off (0), the font will not be transformed. If this flag is on (1), the font can be transformed.

Parameter type = *flag*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Description property list.

Typeface Name

The Typeface Name parameter specifies the common name of the typeface. The typeface name is a common name that the typeface is usually known by. It is usually some combination of family, posture, width, and weight. The typeface name is assigned by a font designer, and it should correspond to the common name of the font as it appears in the product documentation. An example of a typeface name is *Sonoran Sans Serif Roman bold condensed*.

Parameter type = *character string*
Synonyms = *facename*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to *typeface* (Typeface Name). The ISO property is expressed as a character string message (intended as an informational property, not for match string compares).

Underscored Font

The Underscored Font parameter specifies that the graphic character pattern data of this font resource contain underscores as part of the character shape. If this flag is off (0), the graphic character patterns in the font are not underscored. If this flag is on (1), they are underscored.

Parameter type = *flag*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Description property list. This font characteristic could be represented by a unique font family name.

Uniform Character Box Font

The Uniform Character Box Font parameter specifies that the raster (bit-mapped) pattern data for all the graphic characters of the font resource are of the same size. This parameter is only valid if the "Pattern Technology Identifier" on page 100 indicates that the pattern data is of a bit map technology. If this flag is off (0), the graphic character pattern boxes vary in size. If this bit is on (1), all the character boxes in the font are of uniform height and width, and the height and width for each box are taken from the Maximum Character Box Height and Maximum Character Box Width parameters, respectively.

The Character Box Height and the Character Box Width parameters specify the size of nonuniform character boxes.

Parameter type = *flag*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Description property list.

Weight Class

The Weight Class parameter indicates the visual weight (degree or thickness of strokes) of the collection of graphic characters in the font resource. These values are assigned by a font designer, and the visual effect is not defined in FOCA.

Parameter type = *code*
Valid choices:

1	Ultralight
2	Extralight
3	Light
4	Semilight
5	Medium (normal)
6	Semibold
7	Bold
8	Extrabold
9	Ultrabold

Synonyms = *weight, weight-class indicator, font-weight-class indicator*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to *weight* (Weight Code).

Width Class

The Width Class parameter indicates a relative change from the normal aspect ratio (width-to-height ratio) as specified by a font designer for the character shapes in a font. Although every character in a font might have a different numeric aspect ratio, each character in a font of normal width has a relative aspect ratio of one. When a new type style is created with a different width class (either by a font designer or by some automated means) the relative aspect ratio of the characters in the new font is some percentage greater or less than those same characters in the normal font. It is this difference that this parameter specifies.

The font designer assigns a width class designation for each design variation of a particular typeface. However, if a font design is to be varied by automated means, percentage changes are allowed from normal to each of the width class values. For uniformity, when AFP fonts are varied by automated means, a percentage change from normal is assigned to each of the width class values defined above.

The font designer normally assigns the width class values, and the corresponding percentage difference from normal might not apply. Comparing the designated percentage values with the aspect-ratio differences for several designed fonts, the ratios varied by as much as 50 percent from the designated percentage.

Note: It is not accepted practice to vary the width class of a font by automated means. The appearance characteristics of a font can be severely altered by changing stroke width and changing the aspect ratio.

Parameter type = *code*
Valid choices:

1 Ultracondensed, which is 50 percent of normal
2 Extracondensed, which is 62.5 percent of normal
3 Condensed, which is 75 percent of normal
4 Semicondensed, which is 87.5 percent of normal
5 Normal (medium), which is 100 percent of normal
6 Semiexpanded, which is 112.5 percent of normal
7 Expanded, which is 125 percent of normal
8 Extraexpanded, which is 150 percent of normal
9 Ultraexpanded, which is 200 percent of normal

Synonyms = *width, width-class indicator, font-width-class indicator, aspect ratio, width-height ratio, proportionate width, proportion*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to *propwidth* (Proportionate Width Code).

X-Height

The X-Height parameter specifies the height of the body (not including the ascender) of lowercase graphic characters above the character baseline. This value is assigned by a font designer. This parameter divided by the Cap-M height calculates the ratio of X height to Cap-M height, which provides a useful selection or substitution characteristic of the font design.

Parameter type = *number*
Synonyms = *x height*
Transformation to Eastern Writing systems

This parameter does not apply to Eastern writing systems.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to *lcheight* (Lower Case Height). The ISO property is expressed as a relative rational number.

Font-Metric Parameters

This section lists and describes those parameters that apply to all of the font characters. The parameters provide information about a font that can be used for document formatting and document presentation. Generally, these parameters and the Character-Metric Parameters are repeated for each character rotation supported by a font resource. Most of the parameters defined in this section will be used in font resources and will not be used in font references (the metric information is primarily used by applications for formatting and shape presentation). Where there are exceptions, the parameter definition will distinguish between font resource and font reference usage.

Default Baseline Increment

The Default Baseline Increment parameter specifies the nominal distance between character reference points in the vertical direction (90 degrees to the character baseline) recommended by a font designer. This parameter represents the baseline increment to use for this font, and it is specified by a font designer. This value is normally greater than or equal to the nominal vertical font size, and it is often equal to the sum of the nominal vertical font size and the internal leading.

Parameter type = *number*
Transformation to Eastern Writing systems

For vertical writing, this parameter is equal to the distance between the baselines of two columns of graphic characters.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to *minlinesp* (Minimum Line Spacing). The ISO property is expressed as a relative rational number.

External Leading

The External Leading parameter specifies the amount of white space, in addition to the vertical font size increment, that can be added to the interline spacing without degrading the aesthetic appearance of a font. The value of this parameter is usually specified by a font designer; it cannot be derived from other parameters in a font. If fonts are designed with a very minimal amount of additional space above and below the character images, additional space should be added between presentation lines to provide pleasing text appearance. The value of this parameter might represent a font designer's recommendation for this additional leading (interline spacing).

Parameter type = *number*

Transformation to Eastern Writing systems

For vertical writing, this parameter corresponds to any supplemental value the font designer recommends for extending the distance between the baselines of two columns of graphic characters.

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Modal Properties property list.

Figure Space Increment

The Figure Space Increment parameter specifies the character increment used for numerals. A figure space is a formatting measure that sometimes equals the character increment of the numeric characters graphics of a font. The font designer specifies the figure space.

If the numerals all have equal character increments, the value for this parameter can be derived from an analysis of the character-increment parameters for the numeric characters of the font. If the numerals do not have equal character increments, and the designer of the current font has not declared a value, the value of the figure space increment cannot be determined.

Parameter type = *number*

Transformation to Eastern Writing systems

This parameter does not apply to Eastern Writing systems.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to the *tabescx* (Tabular Escapement X) and *tabescy* (Tabular Escapement Y) properties. The ISO values are expressed as relative rational numbers with an x component, a y component, or both.

Internal Leading

The Internal Leading parameter specifies the nominal amount of white space above and below the character shapes of the font that provides a nominal interline spacing for text that is aesthetically pleasing. This value is specified by a font designer. Fonts are usually designed with some nominal amount of white space above and below the character shapes. To compress more text onto a page, characters can be presented with less space between lines. The value of this parameter can be used to calculate the reduction in line spacing that can be made without overwriting lines of text.

The value of this parameter is usually the difference between the vertical font size and the maximum baseline extent for text characters of the font, but special characters that might extend beyond the normal extent of the text character shapes are not included.

Parameter type = *number*
Transformation to Eastern Writing systems
For vertical writing, the value of this parameter corresponds to the difference between the Nominal Font Horizontal Size and the Maximum Baseline Extent for the 270° Character Rotation.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Modal Properties property list.

Kerning

The Kerning parameter is a flag that indicates whether any of the character metric parameters contain negative values that permit the character images to kern. If this flag is off (0), there are no negative A-space or C-space values. If this flag is on (1), A-space values or C-space values can be negative.

Parameter type = *flag*
Transformation to Eastern Writing systems
This parameter does not apply to Eastern writing systems.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Modal Properties property list.

Kerning Pair Character 1

The Kerning Pair Character 1 parameter specifies the first character in a pair of characters for kerning. The space between the two characters can be adjusted by the amount specified in the Kerning Pair X-Adjust parameter. The character identifiers to be used are specified in the appropriate product documentation.

This parameter, the Kerning Pair Character 2 parameter, and the Kerning Pair X-Adjust parameter permit adjusting the space between two specified characters, which can be used for defining kerning, presenting mathematical formula, making composite characters, or making any other space adjustment that can be required.

Parameter type = *character string*
Transformation to Eastern Writing systems
This parameter does not apply to Eastern writing systems.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to *gname* (Glyph Name). In the ISO architecture, Kerning pair data is associated with the individual glyph metrics, thus the first glyph of the kerning pair is the *gname* associated with the *glyphmetrics* property list in which the associated kerning pair data is located.

Kerning Pair Character 2

The Kerning Pair Character 2 parameter specifies the second character in a pair of characters for kerning. The space between the two characters can be adjusted by the amount specified in the Kerning Pair X-Adjust parameter. The character identifiers to be used are specified in the implementing product documentation.

This parameter, the Kerning Pair Character 1 parameter, and the Kerning Pair X-Adjust parameter permit adjusting the space between two specified characters, which can be used for defining kerning, presenting mathematical formula, making composite characters, or making any other space adjustment that can be required.

Parameter type = *character string*

Transformation to Eastern Writing systems

This parameter does not apply to Eastern writing systems.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to the global name component (*gname*) of the *peax* (Pairwise Escapement Adjust X) and *peay* (Pairwise Escapement Adjust Y) property, which are contained within the *pean* (Pairwise Escapement Adjust Name) property list. The ISO architecture permits multiple, named kerning pair techniques to exist for any given glyph (for example, Loose Pair, Touching Pair, Sectored Pair, and Class Pair). The AFP kerning pair data should be represented under the global name: ISO(1) ICD(3) IBM (18) IBM-CS(0) FONTS(1) KPAIR(3) (encoded according to the appropriate ASN.1 or SGML interchange syntax).

Kerning Pair X-Adjust

The Kerning Pair X-Adjust parameter specifies the required escapement adjustment in the x direction for the character pair specified in the Kerning Pair Character 1 and Kerning Pair Character 2 parameters that are associated with this parameter. Letter pair kerning specifies two characters and the space adjustment to be made between them. This parameter specifies the adjustment in the x direction, which can be positive or negative. By using this adjustment, the two characters can be positioned closer together, farther apart, or overlaying one another.

Parameter type = *number*

Synonyms = *pairwise escapement x-adjust, adjustment X*

Transformation to Eastern Writing systems

This parameter does not apply to Eastern writing systems.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to the adjustment component of the *peax* (Pairwise Escapement Adjust X) property. The value is expressed as a relative rational number.

Maximum Ascender Height

The Maximum Ascender Height parameter specifies the maximum ascender height of any graphic character in a font for a specific character rotation. This value is for a specific rotation of the character and is repeated for each rotation supported.

Maximum ascender height is the maximum height attained by any character shape from the character baseline to the top of the character box.

A negative value for this parameter specifies that all graphic characters in the font are completely below the character baseline, as with subscripts.

The value of maximum ascender height can be derived from an analysis of the graphic-character shape information for a font.

Parameter type = *number*
Synonyms = *maximum ascender*
Transformation to Eastern Writing systems

For vertical writing, this parameter is equal to the maximum distance attained by any character shape from the character baseline to the right-hand edge of the character box.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to maximum x, or maximum y value of the *maxfontext* (Maximum Font Extents) value-list. If the AFP Character Rotation is 0°, the Maximum Ascender Height value should be expressed as the maximum y ISO value. If the AFP Character Rotation is 270°, the Maximum Ascender Height value should be expressed as the maximum x ISO value. The value is expressed as a relative rational number.

Maximum Baseline Extent

The Maximum Baseline Extent parameter specifies the space parallel to the character baseline that can be used to place characters. If the maximum baseline offset and the maximum descender depth are positive, their sum is the maximum baseline extent.

If no character in the font extends below the character baseline, the baseline extent is equal to the maximum ascender height.

If no character in the font extends above the character baseline, the maximum baseline extent equals the maximum descender depth. Baseline extent is used to determine the space required by characters at a boundary such as the edge of a presentation space.

Parameter type = *number*
Synonyms = *maximum base-line extension*
Transformation to Eastern Writing systems

For vertical writing, this parameter is equal to the Maximum Ascender Height plus the Maximum Descender Depth (assuming the vertical baseline runs through the character box).

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter is not required for export and can be derived from the *maxfontext* (Maximum Font Extents) property list on import. This parameter could be expressed as a *non-iso-property* in the Modal Properties property list.

Maximum Baseline Offset

The Maximum Baseline Offset parameter specifies the maximum distance of any character in a font from the character baseline to the upper edge of the character box after it has been rotated as specified in the Character Rotation parameter. If any part of the character box is above the character baseline, the baseline offset is the perpendicular distance from the character baseline to the edge of the character box that is above and farthest from the character baseline. When the complete character box is below the character baseline, the baseline offset is the perpendicular distance from the character baseline to the edge of the character box that is nearest to the character baseline. The maximum baseline offset is the greatest of the absolute values of the baseline offset values.

If the Uniform Baseline Offset bit parameter is on (1), this parameter specifies a uniform baseline offset for all characters for a given character rotation.

The Maximum Baseline Offset parameter positions the character box vertically from the character baseline after rotation. If each graphic character has its own baseline offset, this parameter contains the maximum baseline offset for any character in a font, and the amount of the baseline offset for each character is specified by the Baseline Offset parameter.

Parameter type = *number*
Synonyms = *uniform baseline offset*
Transformation to Eastern Writing systems

For vertical writing, this parameter is equal to the Maximum Ascender Height parameter. In the above semantic, “right of” should be substituted for above, and “left of” should be substituted for below.

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter is not required for export and can be derived from the *maxfontext* (Maximum Font Extents) property list on import. This parameter could be expressed as a *non-iso-property* in the Modal Properties property list.

Maximum Character Box Height

The Maximum Character Box Height parameter specifies the height of uniform character boxes or the maximum height of variable character boxes, depending on the value of the Uniform Character Box parameter.

If the Uniform Character Box parameter is off (0), this parameter specifies the maximum height of any character box in a font, and the Character Box Height parameter specifies the height of each character box in a font. This parameter can be used to determine if the character, when positioned, fits in the presentation space.

Parameter type = *number*
Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter is not required for export and can be derived from the *maxfontext* (Maximum Font Extents) property list on import. This parameter could be expressed as a *non-iso-property* in the Modal Properties property list.

Maximum Character Box Width

The Maximum Character Box Width parameter specifies the width of uniform character boxes or the maximum width of variable character boxes, depending on the value of the Uniform Character Box parameter.

If the Uniform Character Box parameter is off (0), the Maximum Character Box Width value specifies the width of the widest character box in the font, and the Character Box Width parameter specifies the width of each individual character box. This parameter can be used to determine if the character, when positioned, fits in the presentation space.

Parameter type = *number*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter is not required for export and can be derived from the *maxfontext* (Maximum Font Extents) property list on import. This parameter could be expressed as a *non-iso-property* in the Modal Properties property list.

Maximum Character Increment

The Maximum Character Increment parameter specifies the maximum character increment for all characters of a font. If the Uniform Character Increment bit parameter is on (1), the character increment for all characters, for a given character rotation, in a font is specified by this parameter. Otherwise, the character increment is specified for each character by the Character Increment parameter.

For uniform increment fonts, this parameter specifies the increment from one graphic character to the next. If the graphic characters have proportional increments, this parameter is the maximum character increment for any character in a font, and the character increment for each character is specified by the Character Increment parameter.

Parameter type = *number*
Synonyms = *uniform character increment*
Transformation to Eastern Writing systems
For vertical writing, this parameter is the maximum character increment for the 270° character rotation metrics.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter is not required for export and can be derived from the *px*,*py*, *ex* and *ey* properties of the *glyphmetrics* property list on import. This parameter could be expressed as a *non-iso-property* in the Modal Properties property list.

Maximum Descender Depth

The Maximum Descender Depth parameter specifies the maximum descender depth of all graphic characters in a font for a specific character rotation. This maximum descender value applies to a specific rotation of the character and is repeated for each rotation supported.

Maximum descender depth is the maximum depth attained by any graphic character in a font, from character baseline to bottom of the character box. A negative descender depth specifies that all graphic characters in a font are completely above the character baseline, for example, as when all graphic characters are superscripts.

Parameter type = *number*
Synonyms = *maximum descender*
Transformation to Eastern Writing systems

For vertical writing, this parameter is equal to the maximum distance attained by any character shape from the character baseline to the left-hand edge of the character box.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to minimum x, or the minimum y value of the *maxfontext* (Maximum Font Extents) value-list. If the AFP Character Rotation is 0°, the Maximum Descender Depth value should be expressed as the minimum y ISO value. If the AFP Character Rotation is 270°, the Maximum Descender Depth value should be expressed as the minimum x ISO value. The value is expressed as a relative rational number.

Maximum Lowercase Ascender Height

The Maximum Lowercase Ascender Height parameter specifies the maximum ascender height of the lowercase graphic characters (a–z) in a font for a character rotation of 0°. The value of this parameter is a descriptive attribute of a font that characterizes the appearance of the typeface design. This value is used for comparing different font typefaces for alternate font selection or font substitution. This parameter can be used to determine where floating accent characters should be positioned.

The value of maximum lowercase ascender height can be derived from an analysis of the graphic character shape information of a font.

Parameter type = *number*
Synonyms = *lowercase ascent*
Transformation to Eastern Writing systems

This parameter does not apply to Eastern writing systems.

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Modal Properties property list.

Maximum Lowercase Descender Depth

The Maximum Lowercase Descender Depth parameter specifies the maximum descender depth of the lowercase graphic characters (a–z) in a font for a character rotation of 0°. The value of this parameter is a descriptive attribute of a font, characterizing the appearance of the typeface design. The value is used for comparing different font typefaces for alternative font selection or font substitution. This parameter can be used to determine where floating accent characters should be positioned.

The value of Maximum Lowercase Descender Depth can be derived from an analysis of the graphic character shape information for a font.

Parameter type = *number*
Synonyms = *lowercase descent*
Transformation to Eastern Writing systems

This parameter does not apply to Eastern writing systems.

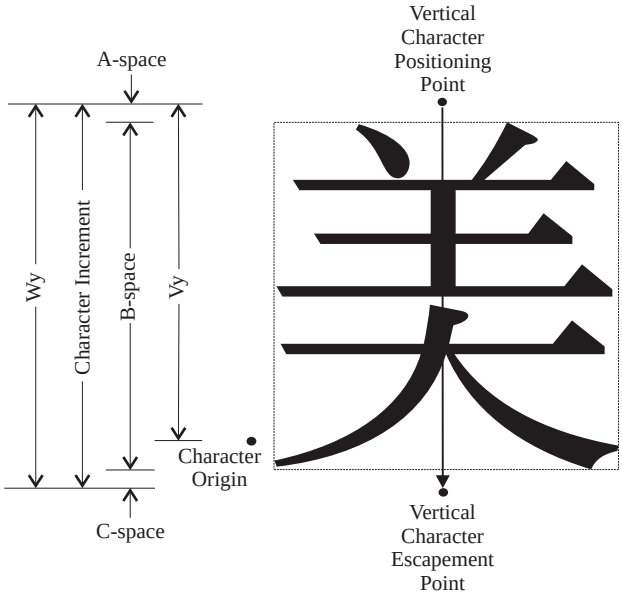
Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Modal Properties property list.

Maximum V(y)

The Maximum V(y) parameter is the maximum of all the Adobe® ATM V(y) values returned for the characters in a given CID font character set. The v(y) value is the y coordinate of the distance from the character origin to the character positioning point. For horizontal writing modes, the character origin and the character positioning point are normally coincident.

Figure 50. Example of V(y) and W(y) Parameters. This character means “beauty”.



Parameter type = *number*
Synonyms = *none*
Transformation to Eastern Writing systems

The definition of this parameter is writing-system independent, though the specific values will be different depending on the design origin of the character and the location of the positioning point relative to that origin.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to the maximum of all P(y) values in the font character set. The ISO property is expressed as a relative rational number.

Maximum W(y)

The Maximum W(y) parameter is the maximum of all the Adobe ATM W(y) values returned for the characters in a given CID font character set. The W(y) value is the y coordinate of the distance from the character positioning point to the character escapement point. For horizontal writing modes, the character positioning point and the character escapement point are normally on the same horizontal line.

Parameter type = *number*
Synonyms = *Vertical Character Increment*
Transformation to Eastern Writing systems

The definition of this parameter is writing-system independent, though the specific values will be different depending on the location of the character positioning point and the character escapement point.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to the maximum, absolute magnitude of all (P(y)-E(y)) values in the font character set. The ISO property is expressed as a relative rational number.

Minimum A-space

The Minimum A-space parameter specifies either the most negative or the least positive A-space value for this font. If the Uniform A-space parameter is off (0), the value of the Minimum A-space parameter specifies the minimum A-space for all characters in a font. Otherwise, the parameter specifies the uniform A-space for all characters for a given character rotation.

The minimum A-space value can be used to determine if the character extends outside the margin at the beginning of a line.

Parameter type = *number*
Transformation to Eastern Writing systems

For vertical writing, this parameter is equal to the least A-space value for the 270° character rotation metrics.

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter is not required for export and can be derived from the *px*, *py*, *minex*, *maxey* properties of the *glyphmetrics* property list on import. This parameter could be expressed as a *non-iso-property* in the Modal Properties property list.

Nominal Character Increment

The Nominal Character Increment parameter specifies the most commonly repeated character increment for all characters of a font. This value permits storage and performance improvements in processing by using the nominal value as a default, rather than searching for and processing highly repetitive character increments (especially useful for Asian fonts, which have a large number of characters with the same character increment, while still having some characters that can be half-width or proportionally spaced).

Application note: CID-Keyed outline fonts (type X'1E') can be used with either single-byte code pages or double-byte code pages and can contain proportional, full-width, and half-width characters. When such a font contains a mixture of character sizes the Uniform Character Increment flag should be set to B'0' (the font is not uniform) and the Nominal Character Increment parameter should contain the most commonly repeated character increment. A character increment can be specified in the FNI for each character in the font, but to decrease the size of the font, all characters that use the Nominal Character Increment can be omitted from the FNI. To determine the character increment for a particular character in a CID-Keyed outline font, an application program should attempt to find an FNI repeating group entry for the character; if one is not found, the Nominal Character Increment should be used. However, this can be inefficient when the font is used with some double-byte code pages (such as, Asian code pages whose characters are all of the same width) because most characters in the font do not have an FNI entry. All double-byte raster fonts sections X'45'–X'FE' have this characteristic. Therefore, for processing efficiency, the following algorithm can be used:

- When a CID-Keyed outline font is used with a double-byte code page, and the code page uses the Double-Byte EBCDIC Presentation encoding scheme, the Nominal Character Increment can be used as the character increment for all characters in the range X'4500'–X'FEFF'. The character increment should be obtained from the FNI for all other characters. Note that for IBM-supplied CID-Keyed fonts, characters in the range X'4100'–X'44FF' also have a uniform character increment and therefore the Nominal Character Increment can be used for these characters, but since the FOCA architecture allows characters in this range to be proportional, fonts from other sources might not adhere to this convention.
- When a CID-Keyed outline font is used with any other code page, the FNI should first be searched for a Character Increment, and if not found the Nominal Character Increment should be used.

Parameter type = *number*

Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter is not required for export and can be derived from the *px*, *py*, *ex* and *ey* properties of the *glyphmetrics* property list on import. This parameter could be expressed as a *non-iso-property* in the Modal Properties property list.

Space Character Increment

The Space Character Increment parameter specifies the default value of the character increment to be used with the space character that is identified by the Space Character Code Point parameter. The value of this parameter is the character increment for the code point that corresponds to the space character. For double-byte fonts, no other specification of the space-character increment is available. For single-byte fonts, the space character increment can be specified in the Space Character Increment parameter or in the font-character metrics information for the space character identifier. The space character increment is the value normally used for the space between words in a sentence.

Parameter type = *number*
Synonyms = *default variable space increment*
Transformation to Eastern Writing systems

For vertical writing, this value corresponds to the space character increment for the 270° character rotation.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to the difference between the *px,py* (Positioning Point X,Y) and the *ex,ey* (Escapement Point X,Y) values for the Normal Space glyph, if it occurs in the subject font resource. The ISO values are expressed as relative rational numbers with an x component, a y component, or both.

Subscript Vertical Font Size

The Subscript Vertical Font Size parameter specifies a font designer's recommended vertical font size for subscript characters associated with this font. If a font does not include all of the required subscript characters for an application, and the application can substitute characters by anamorphically scaling the characters in a font or by substituting characters from another font, this parameter specifies the recommended vertical font size for those subscript characters.

Parameter type = *number*
Synonyms = *subscript y size*
Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to the *vsscaley* (Variant Script Scale Y) property of the *vscript* (Variant Script) property list. The ISO property is expressed as a rational anamorphic scaling of the Design Size. The ISO architecture permits multiple, named variant scripts (for example, Left-vscript, Right-vscript, Ruby-vscript). The AFP subscript data should use the global name ISO/IEC 9541-1/ /RIGHT-VSCRIPT, represented in accordance with the ASN.1 or SGML interchange syntax. The ISO standard uses the concept: right of the alignment line when facing in the escapement direction.

Subscript X-Axis Offset

The Subscript X-Axis Offset parameter specifies a font designer's recommended vertical offset from the character baseline to the character baseline for subscript characters associated with this font. Values are expressed as a positive offset below the character baseline.

If a font does not include all of the required subscript characters for an application, and the application can substitute characters by anamorphically scaling the characters of a font or by substituting characters from another font, this parameter specifies the recommended vertical distance below the character baseline for those subscript characters.

Parameter type = *number*
Synonyms = *subscript x offset*
Transformation to Eastern Writing systems

For vertical writing, this parameter does not apply.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to the *vsoffsetx* (Variant Script Offset X) property of the *vscript* (Variant Scripts) property list (see "Subscript Vertical Font Size" on page 85 for variant script naming). The ISO property is expressed as a relative rational number.

Superscript Vertical Font Size

The Superscript Vertical Font Size parameter specifies a font designer's recommended vertical font size for superscript characters associated with this font. If a font does not include all of the required superscript characters for an application, and the application can substitute characters by anamorphically scaling the characters of a font or by substituting characters from another font, this parameter specifies the recommended vertical size for those superscript characters.

Parameter type = *number*
Synonyms = *superscript y size*
Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to the *vsscaley* (Variant Script Scale Y) property of the *vscript* (Variant Script) property list. The ISO property is expressed as a rational anamorphic scaling of the Design Size. The ISO architecture permits multiple, named variant scripts (for example, Left-vscript, Right-vscript, Ruby-vscript). The AFP superscript data should use the global name ISO/IEC 9541-1/ /LEFT-VSCRIPT, represented in accordance with the ASN.1 or SGML interchange syntax. The ISO standard uses the concept: left of the alignment line when facing in the escapement direction.

Superscript X-Axis Offset

The Superscript X-Axis Offset parameter specifies a font designer's recommended vertical offset from the character baseline to the superscript character baseline associated with this font. Values for this parameter are expressed as a positive offset above the character baseline.

If a font does not include all of the required superscript characters for an application, and the application can substitute characters by anamorphically scaling the characters of a font or by substituting characters from another font, this parameter specifies the recommended vertical distance above the character baseline for those superscript characters.

Parameter type = *number*
Synonyms = *superscript x offset*
Transformation to Eastern Writing systems
For vertical writing, this parameter does not apply.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to the *vsoffsetx* (Variant Script Offset X) property of the *vscript* (Variant Scripts) property list (see "Superscript Vertical Font Size" on page 86 for variant script naming). The ISO property is expressed as a relative rational number.

Throughscore Position

The Throughscore Position parameter specifies the recommendation of a font designer for drawing throughscores for a font. This parameter is specified as the perpendicular distance from the character baseline to the top of the throughscore stroke width. The stroke is parallel to the baseline. A value of zero means that the top of the throughscore stroke is coincident with the baseline. A negative position specifies that the throughscore stroke is below the character baseline.

Absence of the Throughscore Position parameter indicates that no throughscore position recommendation is given. An implementation should use zero as a data stream default.

Parameter type = *number*
Synonyms = *strikeout position, score position*
Transformation to Eastern Writing systems
For vertical writing, this parameter is equal to the distance between the center line and the vertical score line running through the graphic characters.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to the *scoreoffsetx* (Score Offset X) and the *scoreoffsety* (Score Offset Y) properties of the Scores property list, except that the ISO property is measured to the center of the score. The ISO property is expressed as a relative rational number in the x or y direction. The ISO architecture permits multiple, named scores (for example, Rightscore, Leftscore, and Throughscore). The AFP throughscore data should use the global name ISO/IEC 9541-1/ /THROUGHSCORE, represented in accordance with the ASN.1 or SGML interchange syntax.

Throughscore Width

The Throughscore Width parameter specifies the recommendation of a font designer for the width (thickness) of the throughscore for a font. A value of zero for this parameter indicates no font-designer recommendation is available.

Absence of the Throughscore Width parameter indicates that no throughscore width recommendation is given. An implementation should use the height of the character box of the underscore character as a data stream default.

Parameter type = *number*
Synonyms = *strikeout size, score thickness*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to the *scorethick* (Score Thickness) property of the Scores property list (see "Throughscore Position" on page 87). The ISO property is expressed as a relative rational number.

Underscore Position

The Underscore Position parameter specifies the recommendation of a font designer for drawing underscores for a font. This parameter is specified as the perpendicular distance from the character baseline to the top of the underscore stroke width. The stroke is parallel to the baseline. A value of zero means that the top of the underscore stroke is coincident with the baseline.

Absence of the Underscore Position parameter indicates that no underscore position recommendation is given. An implementation should use 75 relative units (75/1000 of an Em) as a data stream default.

A negative position specifies that the underscore stroke is above the character baseline.

If the Underscore font parameter is off (0), the character shapes of the font do not contain underscore strokes.

Parameter type = *number*
Synonyms = *font underscore position, score position*
Transformation to Eastern Writing systems
For vertical writing, this parameter is equal to the distance between the center line and the vertical score line running to the left. An implementation should use 75 relative units (75/1000 of an Em) from the edge of the Maximum Character Box (75 relative units plus 1/2 the Maximum Character Box Width in relative units) as a data stream default.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to the *scoreoffsetx* (Score Offset X) and the *scoreoffsety* (Score Offset Y) properties of the Scores property list, except that the ISO property is measured to the center of the score. The ISO property is expressed as a relative rational number. The ISO architecture permits multiple, named scores (for example, Rightscore, Leftscore, and Throughscore). The AFP underscore should use the global Name ISO/IEC 9541-1/ /RIGHTSCORE, represented in accordance with the ASN.1 or SGML interchange syntax. ISO uses the concept: right of the baseline when facing in the escapement direction.

Underscore Width

The Underscore Width parameter specifies the recommendation of a font designer for the width (thickness) of underscores for a font. A value of zero for this parameter indicates no font-designer recommendation is available.

Absence of the Underscore Width parameter indicates that no underscore width recommendation is given. An implementation should use the height of the character box of the underscore character as a data stream default. If the character box height of the underscore character is not specified for a font, or is not available to the process, the data stream default value should be equal to 50 relative units (50/1000 of an Em).

If the Underscore font parameter is off (0), the character shapes of a font do not contain underscore strokes.

Parameter type = *number*
Synonyms = *underscore size, font underscore width, score thickness*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
This parameter corresponds to the *scorethick* (Score Thickness) property of the Scores property list (see "Underscore Position" on page 88). The ISO property is expressed as a relative rational number.

Uniform A-space

The Uniform A-space parameter specifies that a uniform amount of A-space has been removed from all raster (bit mapped) patterns for the font. This parameter is valid only if the "Uniform Character Box Font" on page 72 indicates uniform character boxes. If this flag is off (0), the Minimum A-space parameter specifies the smallest A-space for all characters in the font. If this flag is on (1), the Minimum A-space parameter specifies a uniform A-space value for all characters in the font.

Parameter type = *flag*
Transformation to Eastern Writing systems
For vertical writing, this parameter represents the A-space for 270° character rotation metrics.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Modal Properties property list.

Uniform Baseline Offset

The Uniform Baseline Offset parameter specifies that all raster (bit mapped) patterns for the font resource have a common offset from the baseline to the top of the pattern box. If this flag is off (0), the baseline offset might differ for each character; if this flag is on (1), the Maximum Baseline Offset parameter specifies a uniform baseline offset for all characters in the font.

Parameter type = *flag*
Transformation to Eastern Writing systems
For vertical writing, this parameter represents the Uniform Baseline Offset for 270° character rotation metrics.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Modal Properties property list.

Uniform Character Increment

The Uniform Character Increment parameter specifies that all character increments for the font resource are the same (see also "Proportional Spaced" on page 69). If this flag is off (0), the graphic characters increment is proportional; if this flag is on (1), the Maximum Character Increment parameter specifies a uniform character increment for all characters in the font.

Parameter type = *flag*

Transformation to Eastern Writing systems

For vertical writing, this parameter represents the Uniform Character Increment for 270° character rotation metrics.

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Modal Properties property list.

Character-Metric Parameters

This section lists and describes those metric parameters that apply to individual characters of a font. The parameters provide specific information about each character for document formatting and document presentation. These parameters are repeated for each character in a font resource and for each rotation of those characters.

A-space

The A-space parameter specifies the distance from the character reference point to the least positive character coordinate system x-axis value of the character shape. The value of this parameter can be positive, zero, or negative.

- A positive value means that the A-space and the character reference point lie in the escapement direction before the start of B-space.
- An A-space value of zero means that there is no space preceding the character shape.
- A negative A-space value means that the A-space and the character reference point lie after the beginning of B-space. Negative A-space is used in kerning.

The value of this parameter can be used to compute a character increment.

Parameter type = *number*

Transformation to Eastern Writing systems

For vertical writing, this parameter is equal to the distance between the character reference point and the least positive character coordinate system x-axis value of the 270° rotated character shape.

Transformation to ISO/IEC 9541 font architecture

No equivalent property exists in the ISO architecture, but this parameter corresponds to the distance between the *px,py* (Position Point X and Y) and the closest *ext* (Extents) property value along the escapement direction line. To convert the AFP A-space value to the corresponding ISO value, the ISO *px* and *py* values should first be set to 0,0 (the AFP reference point is at the origin of the coordinate system). If the AFP Character Rotation is 0°, the *minx* (Minimum X Extent) should be set equal to the A-space value (with appropriate unit-of-measure conversions). If the AFP Character Rotation is 270°, the *maxy* (Maximum Y Extent) should be set equal to the negative of the A-space value (with appropriate unit of measure conversions). The ISO property is expressed as a relative rational number.

Ascender Height

The Ascender Height parameter specifies the height of the topmost mark of a graphic character. If the ascender height is negative, the graphic character lies completely below the character baseline. For example, subscripts have a negative ascender height.

Parameter type = *number*

Transformation to Eastern Writing systems

For vertical writing, this parameter is equal to the distance between the character reference point and the most positive character coordinate system y-axis value of the 270° rotated character shape.

Transformation to ISO/IEC 9541 font architecture

No equivalent property exists in the ISO architecture, but this parameter corresponds to the distance between the *px,py* (Position Point X and Y) and a projection of the left extent (when facing in the escapement direction) on a line passing through the *px,py* point, oblique to the escapement direction line. To convert the AFP Ascender Height value to the corresponding ISO value, the ISO *px* and *py* values should first be set to 0,0 (the AFP reference point is at the origin of the coordinate system). If the AFP

Character Rotation is 0°, the *maxy* (Maximum Y Extent) should be set equal to the Ascender Height value (with appropriate unit-of-measure conversions). If the AFP Character Rotation is 270°, the *maxx* (Maximum X Extent) should be set equal to the Ascender Height value (with appropriate unit of measure conversions). The ISO property is expressed as a relative rational number.

Baseline Offset

The Baseline Offset parameter specifies the distance from the character baseline to the topmost edge of a character box. If the Uniform Baseline Offset parameter is on (1), the Maximum Baseline Offset parameter specifies the uniform offset for all the font's graphic characters, and this parameter value can be omitted. If the Uniform Baseline Offset parameter is off (0), the Maximum Baseline Offset parameter specifies the maximum offset for all the font's graphic characters, and this parameter specifies the baseline offset for each of the font's graphic characters.

The baseline offset value is positive when any part of the character box is positioned above the character baseline, and it is negative when the complete character box is positioned below the character baseline. If any part of the character box is above the character baseline, the baseline offset is the distance from the character baseline to the edge of the character box that is parallel, above, and farthest from the character baseline. If the complete character box is below the character baseline, the baseline offset is the distance from the character baseline to the edge of the character box that is nearest and parallel to the character baseline.

For raster fonts, the top edge of the top of the character box usually corresponds to the origin of the character-shape information, and the baseline offset is useful in positioning the character image.

Parameter type = *number*
Transformation to Eastern Writing systems

For vertical writing, this parameter is equal to the Ascender Height. In the above semantic, “right of” should be substituted for above, and “left of” should be substituted for below.

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter is not required for export and can be derived from the *ext* (Extents) property list on import. This parameter could be expressed as a *non-iso-property* in the Glyph Properties property list.

B-space

The B-space parameter specifies the width of the (bounded) character box. The value of this parameter can be used to compute a character increment.

Parameter type =*number*
Transformation to Eastern Writing systems

For vertical writing, this parameter is equal to the distance between the most positive and least positive character coordinate system x-axis value of the 270° rotated character shape.

Transformation to ISO/IEC 9541 font architecture

No equivalent property exists in the ISO architecture, but this parameter corresponds to the distance between the *minx* and *maxx* (Min and Max X Extent), or the *miny* and *maxy* (min and Max Y Extent) property value, depending on the escapement direction. To convert the AFP B-space value to the corresponding ISO value, the ISO *minx* and *maxy* values should first be computed (see “A-space” on page 91). If the AFP Character Rotation is 0°, the *maxx* (Maximum X Extent) should be set equal to the *minx* value plus the AFP B-space value (with appropriate unit-of-measure conversions). If the AFP Character Rotation is 270°, the *miny* (Minimum Y Extent) should be set equal to the *maxy* value plus the negation of the B-space value (with appropriate unit-of-measure conversions). The ISO property is expressed as a relative rational number.

Character Box Height

The Character Box Height parameter specifies the height of the character box for a graphic character. If the Uniform Character Box parameter is on (1), the uniform character box height for all of the font's graphic characters is specified by the Maximum Character Box Height parameter and this parameter value is ignored. If the Uniform Character Box parameter is off (0), the Maximum Character Box Height parameter specifies the maximum character box height for all the font's graphic characters, and this parameter specifies the character box height for each of the font's graphic characters.

Parameter type = *number*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter is not required for export and can be derived from the difference between the *miny* and *maxy* (Minimum Y and Maximum Y Extents) of the Glyph Properties property list on import. This parameter could be expressed as a *non-iso-property* in the Glyph Properties property list.

Character Box Width

The Character Box Width parameter specifies the width of the character box for a character. If the Uniform Character Box parameter is on (1), the uniform character box width for all of the font's graphic characters is specified by the Maximum Character Box Width parameter and this parameter value is ignored. If the Uniform Character Box parameter is off (0), the Maximum Character Box Width parameter specifies the maximum character box width for all the font's graphic characters, and this parameter specifies the character box width for each of the font's graphic characters.

The width of the character box normally corresponds to the value of the B-space parameter.

Parameter type = *number*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter is not required for export and can be derived from the difference between the *minx* and *maxx* (Minimum X and Maximum X Extents) of the Glyph Properties property list on import. This parameter could be expressed as a *non-iso-property* in the Glyph Properties property list.

Character Increment

The Character Increment parameter value is the algebraic sum of the A-space, the B-space and the C-space values for a character shape. Using a different value can result in displeasing aesthetic effects.

If the Uniform Character Increment parameter is on (1), the uniform character increment for all of the font's graphic characters is specified by the Maximum Character Increment parameter and this parameter is ignored. If the Uniform Character Increment parameter is off (0), the Maximum Character Increment parameter specifies the maximum character increment for all the font's graphic characters, and this parameter specifies the character increment for each of the font's graphic characters.

Parameter type = *number*
Synonyms = *character escapement, escapement*
Transformation to Eastern Writing systems

For vertical writing, this parameter is equal to the distance between the reference point and the escapement point for the 270° rotated character shape.

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter is not required for export and can be derived from the difference between the *px,py* (Position Point X and Y) property value and the *ex,ey* (Escapement Point X and Y) property value on import. This parameter could be expressed as a *non-iso-property* in the Glyph Properties property list.

C-space

The C-space parameter specifies the width of the space from the (bounded) character box to the escapement point. This parameter can be positive, zero, or negative.

- A positive value means that the C-space lies (in the escapement direction) after B-space.
- A C-space value of zero means that there is no space following the character shape.
- A negative value means that C-space lies (in the escapement direction) before the end of B-space.

The value of this parameter can be used to compute a character increment.

Parameter type = *number*
Transformation to Eastern Writing systems

For vertical writing, this parameter is equal to the distance between the character escapement point and the least positive character coordinate system x-axis value of the 270° rotated character shape.

Transformation to ISO/IEC 9541 font architecture

No equivalent property exists in the ISO architecture. This parameter is not required for export, and can be computed from ISO glyph properties on import. This parameter could be represented as a *non-iso-property* in the Glyph Properties property list.

Descender Depth

The Descender Depth parameter specifies the descender depth of a graphic character. A negative descender depth specifies that the graphic character lies completely above the character baseline; superscript graphic characters are an example.

Parameter type = *number*

Transformation to Eastern Writing systems

For vertical writing, this parameter is equal to the distance between the character reference point and the most negative character coordinate system y-axis value of the 270° rotated character shape.

Transformation to ISO/IEC 9541 font architecture

No equivalent property exists in the ISO architecture, but this parameter corresponds to the distance between the *px,py* (Position Point X and Y) and a projection of the right extent (when facing in the escapement direction) on a line passing through the *px,py* point, oblique to the escapement direction line. To convert the AFP Descender Depth value to the corresponding ISO value, the ISO *px* and *py* values should first be set to 0,0 (the AFP reference point is at the origin of the coordinate system). If the AFP Character Rotation is 0°, the *miny* (Minimum Y Extent) should be set equal to the Descender Depth value (with appropriate unit-of-measure conversions). If the AFP Character Rotation is 270°, the *minx* (Minimum X Extent) should be set equal to the Descender Depth value (with appropriate unit of measure conversions). The ISO property is expressed as a relative rational number.

Graphic Character Global Identifier

The Graphic Character Global Identifier parameter specifies the registered identifier of a graphic character. Unless otherwise specified, the default encoding is EBCDIC and the default length is 8 bytes.

The IBM Graphic Character Global IDs that are supported are specified in product documentation.

Parameter type = *character string*

Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to *gname* (Glyph Name). The ISO property uses the full ISO structured name, and any transform to the ISO format should prepend the appropriate name prefix (see "Global Naming" on page 53). AFP graphic characters, used in font resources, should be represented under the global name prefix ISO(1) ICD(3) IBM(18) IBM-CS(0) FONTS(1) GCGID(2), encoded according to the appropriate ASN.1 or SGML interchange syntax.

Character-Shape Parameters

This section lists and describes those parameters required for presentation of the character shapes. The information does not include those parameters required for positioning the characters in the presentation space. These parameters are repeated for each technology supported by the font resource. Most of the parameters defined in this section are used in font resources and are not used in font references (the character shape information is primarily used to present the character shape on the presentation surface). Where there are exceptions, the parameter definition will distinguish between font resource and font reference usage.

Design Resolution X

The Design Resolution X parameter specifies the intended presentation resolution in the x direction for this character shape representation technology. Transformations of a character shape from one technology, or device specific format, to another requires knowledge about the resolution of the target devices. For example, the transformation of an image from a resolution of 240 by 240 pels per inch to 60 by 72 per inch requires a different transformation from that to a resolution of 300 by 300 pels per inch. An outline-representation technique might only need the output-device resolution, but a raster representation needs both the design resolution and the output-device resolution.

Parameter type = *number*
Synonyms = *x device resolution*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Glyph Shapes property list.

Design Resolution Y

The Design Resolution Y parameter specifies the intended presentation resolution in the y direction for this character-shape-representation technology. Transformation of a character shape from one technology, or device specific format, to another requires knowledge about the resolution of the target devices. For example, transformation of a 240 by 240 resolution image to a 60 by 72 resolution image requires a different transformation from that to a 300 by 300 resolution image. An outline representation technique might only require knowledge of the output-device resolution, but a raster representation requires knowledge of both the design resolution and the output-device resolution.

Parameter type = *number*
Synonyms = *y device resolution*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Glyph Shapes property list.

Linkage Code

The Linkage Code parameter specifies whether or not the character IDs in the CMAP file are linked to the character IDs in the Name Map file. CMAP files that are not linked to the Name Map file should only be used with the code page identified by the CPGID parameter.

Parameter type = *code*
Valid Choices:

- **0** Linked
- **1** Unlinked

Synonyms = *none*
Transformation to Eastern Writing systems

This parameter is used most often with eastern writing systems, but is not restricted to any particular writing system.

Transformation to ISO/IEC 9541 font architecture

No equivalent property exists in the ISO architecture.

Object Type

The Object Type parameter provides a method of identifying various objects that can be imbedded in a font resource. The objects identified by this parameter are most often non-AFP data objects which are architected by other companies or organizations. The format specification for those objects must be obtained from the defining source.

Parameter type = *code*
Valid Choices:

- **0** No information
- **1** CMAP file
- **5** CID file
- **6** PFB file
- **7** AFM file
- **8** File Name Map

Synonyms = *none*
Transformation to Eastern Writing systems

This parameter is used most often with eastern writing systems, but is not restricted to any particular writing system.

Transformation to ISO/IEC 9541 font architecture

No equivalent property exists in the ISO architecture.

Pattern Data

The Pattern Data parameter specifies the pattern data for this character-shape representation technique. The data might represent any of the character representation techniques and is formatted according to the definition of the pattern technology and compression algorithm.

Parameter type = *undefined data*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Glyph Shapes property list.

Pattern Data Alignment Code

The Pattern Data Alignment Code parameter specifies the alignment of the beginning of each character's pattern data (see also Pattern Data Alignment Value). The code values assigned to this parameter represent the exponent of the base 2 corresponding to the byte alignment. For example, a code value of 2 means raise the base, 2, to the second power to get the Pattern Data Alignment Value of 4 bytes, a full word.

Use of the Pattern Data Alignment parameter allows flexibility in aligning pattern data in storage to permit use of different computing systems with differing abilities in the degree of fineness in addressing storage.

Parameter type = *code*
Valid choices:

0	One-byte Alignment
1	Two-byte Alignment
2	Four-byte Alignment
3	Eight-byte Alignment

Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Glyph Shapes property list.

Pattern Data Alignment Value

The Pattern Data Alignment Value parameter specifies the byte alignment of the beginning of each character's pattern data (see also Pattern Data Alignment Code). The value assigned to this parameter is used as a multiplier of the Pattern Data Offset to determine the corresponding byte offset.

Use of the Pattern Data Alignment Value parameter allows flexibility in aligning pattern data in storage to permit use of different computing systems with differing abilities in the degree of fineness in addressing storage.

Parameter type = *number*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Glyph Shapes property list.

Pattern Data Count

The Pattern Data Count parameter specifies the total quantity of shape data, expressed in number of bytes. The data count does not include any header or trailer information that can be used in an interchange format to identify the shape data.

Parameter type = *number*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Glyph Shapes property list.

Pattern Data Offset

The Pattern Data Offset parameter is used in conjunction with the Pattern Data Alignment Value parameter or the Pattern Data Alignment Code parameter to calculate the actual byte offset of a character's shape data from the beginning of the pattern data (see "Pattern Data" on page 98). The byte offset of the pattern data is the Pattern Data Offset number multiplied by the alignment value defined by the Pattern Data Alignment Value parameter or the Pattern Data Alignment Code parameter.

For example, if the Pattern Data Alignment is four-byte alignment, the actual pattern data offset, in bytes, is the Pattern Data Offset number multiplied by 4. If the Pattern Data Offset number is 97 and the Pattern Data Alignment is four-byte alignment, the actual pattern data offset is 97 * 4 = 388 bytes.

The value of this parameter is the actual data offset divided by the alignment value defined by the Pattern Data Alignment Value or the Pattern Data Alignment Code parameter. In the preceding example, the offset value of 97 is the actual offset (388) divided by the alignment value of 4: 388 / 4 = 97.

Parameter type = *number*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Glyph Shapes property list.

Pattern Technology Identifier

The Pattern Technology Identifier parameter specifies the technologies for the font graphic patterns for this font. Pattern Technology is defined by implementing font products and is documented in the font product documentation.

This parameter is used in both font resources and font references.

Parameter type = *code*
Valid choices:

5	Laser Matrix N-Bit Wide Horizontal Sections
30	CID Keyed Outline Font Technology
31	Type 1 PFB Outline Font Technology

Synonyms = *shape technology, font-shape technology*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Glyph Shapes property list.

Precedence Code

The Precedence Code parameter specifies whether or not a particular object is the primary object of its type in the resource, or if it is an auxiliary (alternate) object of its type in the resource.

Implementation Note: When the Object Type is 5 (CID Font, "Object Type" on page 97), and the Font Character Set is a base font ("Extension Font" on page 60), the precedence code will be set to 0 (primary). The precedence code for CID fonts will be set to 1 (auxiliary) if this is an extension font. When the Object Type is 1 (CMap File), the precedence code will be set to 0 (primary) if this CMap is the first CMap to be used for the current CPGID and Writing Mode. The precedence code for CMaps will be set to 1 (auxiliary) if this CMap is pointed to by another CMap.

Parameter type = *code*
Valid Choices:

0 Primary
1 Auxiliary

Synonyms = *none*
Transformation to Eastern Writing systems

This parameter is used most often with eastern writing systems, but is not restricted to any particular writing system.

Transformation to ISO/IEC 9541 font architecture

No equivalent property exists in the ISO architecture.

Shape Pattern Index

The Shape Pattern Index parameter specifies an index into the repeating group in the Pattern Data Offset parameter that corresponds to the graphic character associated with this parameter. The index values defined by this parameter allow using the same pattern data information for multiple character rotations. Some pattern data can be unique for different rotations, and other pattern data can be used for multiple rotations.

Parameter type = *number*
Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Glyph Shapes property list.

Writing Direction Code

The Writing Direction Code parameter specifies the nominal direction in which characters of the font are written or read by the end user.

Implementation Note: This field is only used with Object Type 1 (CMap, "Object Type" on page 97). When the value of this field is 1 or 2 (horizontal or vertical), the CMap is intended to be used only for the specified writing direction, but when the value of this field is 3 (both), the CMap is writing direction independent and may be used for either vertical or horizontal writing.

Parameter type = *code*
Valid Choices:

- **0** No information
- **1** Horizontal (left to right or right to left)
- **2** Vertical
- **3** Both Vertical and Horizontal

Synonyms = *writing mode, character rotation*
Transformation to Eastern Writing systems

The definition of this parameter is writing-system independent, though the specific values will be different depending on the direction specified.

Transformation to ISO/IEC 9541 font architecture

This parameter corresponds to the Writing Mode property. The ISO property is a named value.

Character-Mapping Parameters

This section lists and describes the parameters required to associate (map) the code points to the graphic character identifiers for a code page. Most of the parameters defined in this section are used in font resources and are not used in font references. The character mapping information is primarily used by applications to associate code points in a document to the graphic character information in a font resource. Where there are exceptions, the parameter definition will distinguish between font resource and font reference usage.

Code Page Description

The Code Page Description parameter provides a descriptive title or short description of the code page. The value assigned to this parameter should be the same as that assigned by the IBM Code Page registration authority when registering the code page, though any descriptive character string is permitted.

Parameter type = *character string*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Resource property list.

Code Page Global Identifier

The Code Page Global Identifier (CPGID) parameter specifies the number assigned to a code page. The code page numbers that are supported are specified in product documentation.

Code Page Global IDs 1 through 65,279 are reserved for assignment by IBM.

This parameter is used in both font resources and font references.

Parameter type = *number*
Synonyms = *code page, code-page identifier, code-page-standard identifier*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Resource property list.

Code Point

The Code Point parameter specifies the value of the integer sequence assigned to a graphic character in an ordered list of control and graphic characters. The code point numbers assigned depend on the code-page definition. The code page definition can be specified by a user, or as specified in the appropriate product documentation.

The Code Point parameter consists of a one-byte binary number representing a graphic character in a list of 256 potential control and graphic characters. If the Number of Code Points Available parameter specifies that the available code points exceed 256, it is necessary that this parameter is used with the Section Number parameter.

Parameter type = *number*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Resource property list.

Encoding Scheme

The Encoding Scheme parameter is a three-part, two-byte, hexadecimal number defined in the *Character Data Representation Architecture (CDRA) Reference*, which identifies the scheme used to code graphic character data. The first part is a one-digit hexadecimal number (1/2 byte) corresponding to the Basic Encoding Structure. The second part is a one-digit hexadecimal number (1/2 byte) corresponding to the Number of Bytes per Code Point, and the third part is a two-digit hexadecimal number (1 byte) corresponding to the Code Extension Method.

Font resources and font references do not require the full set of Encoding Scheme parameter values supported by CDRA. The valid set of choices required for FOCA support are identified below.

ASCII is the American Standard Code for Information Interchange, and EBCDIC is the Extended Binary Coded Decimal Interchange Code. ISO is the International Standards Organization, which uses both a seven-bit binary code and an eight-bit binary code to represent characters. UCS is the ISO universal coded character set intended to contain most of the graphic characters used in languages and scripts throughout the world; the fixed two-byte version of UCS (UCS-2) is commonly known as Unicode. UCS Presentation is a subset of UCS that contains only code points that can be directly mapped to a single glyph.

Parameter type = *code*
Valid choices:

Basic Encoding Structure:
- **0** No specified organization
- **2** IBM-PC Data
- **6** EBCDIC Presentation
- **8** UCS Presentation (an all glyph scheme based on UCS)

Number of Bytes per Code Point:
- **1** Fixed Single-byte
- **2** Fixed Double-byte

Code Extension Method:
- **00** No specified extension

Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Resource property list.

Graphic Character Identifier Type

The Graphic Character Identifier Type parameter identifies the naming source and the method used to identify graphic characters.

Note: Values 0, 1, and 4 are not used in FOCA fonts.

Parameter type = *code*
Valid choices:

0 No character identification defined
1 ISO Registered EBCDIC Glyph ID
2 IBM Registered EBCDIC GCGID
3 Font Specific ASCII Character Name
4 Font Specific Binary Glyph Index
5 CMap Binary Code Point

Note:

- An ISO Registered EBCDIC Glyph ID is the leaf element of a hierarchical structured name of a glyph registered by the Association for Font Information Interchange in accordance with the registration procedures of ISO/IEC 10036. The leaf element is a character string, representing a decimal number in the range of 1 to (2**32)-1, with no leading zeros. The ISO Registered Glyph IDs identified by this Graphic Character Identifier Type are expressed in EBCDIC encoding, with a length defined by the Graphic Character Identifier Length parameter. The full structured name is: "ISO/IEC 10036/RA/ /GLYPH::nnnn", where nnnn is the leaf element. Contact the Association for Font Information Interchange, ISO/IEC 10036 Glyph Registrar, P.O. Box 33683, Northglenn, Colorado 80233-0683 USA, for their most recent listing of the world's glyphs (for example, *International Glyph Register, Volume 1: Alphabetic Scripts and Graphic Symbols*).
- An IBM Registered EBCDIC GCGID is a character string, using the uppercase letters *A–Z* and the numbers *0–9*. The IBM GCGIDs identified by this Graphic Character Identifier Type are expressed in EBCDIC encoding, with the length of the GCGID string being 4 bytes or 8 bytes, depending on the value of the Graphic Character Identifier Length parameter. IBM GCGIDs are published in implementing product documentation, such as the *Technical Reference for IBM Expanded Core Fonts*, S544-5228.
- A Font Specific ASCII Character Name is a character string, using the lowercase letters *A–Z* and the numbers *0–9*. The graphic character names identified by this Graphic Character Identifier Type are expressed in ASCII encoding, with a variable length (the lesser of 128 or the value of the Graphic Character Identifier Length parameter). These graphic character names might be registered by a single font or system supplier, or might be a collection of graphic character names from multiple font or system suppliers. They must be unique within a font resource, but not necessarily unique across multiple font resources.
- A Font Specific Binary Glyph Index is a binary string, representing an integer in the range of 1 to (2**16)-1. The glyph index values identified by this Graphic Character Identifier Type may have a length of one or two bytes, depending on the length defined by the Graphic Character Identifier Length parameter. The assigned index number is font dependent, assigned by the creator of the font, and is published in the implementing font product documentation. These glyph index values might be defined by a font or system supplier, or might be dynamically created by a document processing system. They must be unique within a font resource, but not necessarily unique across multiple font resources.
- A CMap Binary Code Point is a binary string, representing a character code as defined by an Adobe Type 0 CMap file. The values identified by this Graphic Character Identifier Type may have a length of 1, 2, or 4 bytes, depending on the length defined by the Graphic Character Identifier Length parameter. The assigned values are defined by the creator of the Adobe Type 0 CMap file, but is normally based on a national or internationally approved character coding standard, or a

widely used industry coding standard. Use of this parameter must identify the Adobe Type 0 CMap file to which this method of identifying Graphic Character Identifiers is applicable.

Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Description property list.

Graphic Character GID Length

The Graphic Character Identifier Length parameter specifies the length of the graphic character identifier. The length is specified as an even number of bytes. Unless otherwise specified, the default length is eight bytes.

Parameter type = *number*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Description property list.

Invalid Coded Character

The Invalid Coded Character parameter specifies that the associated coded graphic character is not valid and should not be used for processing. If this flag is off (0), the coded graphic character is valid. If this flag is on (1), the coded graphic character is not valid.

Parameter type = *flag*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Resource property list.

No Increment

The No Increment parameter specifies that the character increment for the corresponding coded character should be ignored. If this flag is off (0), the coded character is incrementing. If this flag is on (1), the coded character is non-incrementing.

Parameter type = *flag*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Resource property list.

No Presentation

The No Presentation parameter specifies that the corresponding coded character should be ignored. If this flag is off (0), the coded character is presenting. If this flag is on (1), the coded character is non-presenting.

Parameter type = *flag*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Resource property list.

Number of Coded Graphic Characters Assigned

The Number of Coded Graphic Characters Assigned parameter specifies the number of graphic characters (code points in one or more sections) that have been assigned in a code page.

Parameter type = *number*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Resource property list.

Section Number

The Section Number parameter specifies the section number of a multibyte code page. A code page section number is the first byte of a two-byte coded character. The number of valid sections depends on the encoding scheme being used. The encoding scheme is defined in "Encoding Scheme" on page 104. If the encoding scheme is double-byte EBCDIC, values 1–64 and 255 are not permitted. See the *Character Data Representation Architecture Reference and Registry* for other encoding scheme restrictions.

This parameter is used in both font resources and font references.

Parameter type = *number*
Synonyms = *ward, section, coded-font section*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Resource property list.

Space Character Code Point

The Space Character Code Point parameter specifies the code point assigned as the space character; for a double-byte code page, the space character is identified with the space character section parameter followed by the space character code point parameter.

The space character section and space character code point parameters can be assigned any value, but the values normally used in FOCA fonts are shown in the following table:

Table 11. Space Character Code Point Used in FOCA Fonts

Basic Encoding Structure	Single Byte	Double Byte
IBM-PC Data	X'2020'	Not used
EBCDIC Presentation	X'4040'	X'4040'
UCS Presentation (Latin)	Not applicable	X'0020'
UCS Presentation (ideographic, full-width space)	Not applicable	X'3000'

The following characteristics apply to space characters:

- The increment for this character can be changed.
- The No Presentation flag in the Graphic Character Use Flags parameter equals 1.

Parameter type = *number*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Resource property list.

Space Character Section Number

The Space Character Section Number parameter specifies the section number for the code point of the space character.

Parameter type = *number*
Synonyms = *space character ward number*
Transformation to Eastern Writing systems
This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture
No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Resource property list.

Unspecified Coded Character Identifier

The Unspecified Coded Character Identifier parameter specifies the registered identifier for the graphic character information that is used whenever a font object does not contain the graphic character information for the graphic character identifier associated with a code point, or whenever a code point has no associated graphic character identifier. The default value for this parameter is the space character as specified by either the Space Character Increment parameter or by the character positioning information for that character in the font, depending on the implementation technique selected.

A coded character is specified by assigning a graphic character to a code point in the code page. For any code point in a code page that does not have an explicitly specified graphic character identifier, the unspecified coded character is implicitly specified. An unspecified coded character is not of necessity an invalid coded character.

Parameter type = ***character string (named graphic character global identifier)***
Transformation to Eastern Writing systems

This parameter is writing-system independent.

Transformation to ISO/IEC 9541 font architecture

No equivalent parameter exists in the ISO architecture. This parameter should be expressed as a *non-iso-property* in the Font Resource property list.

Chapter 6. Font Interchange Formats

Font Information Interchange, for the purpose of this chapter, is defined to be the transfer of font information between or among processing systems, software processes, and hardware devices. You can transfer font information between any two systems, processes, or devices as long as both parties recognize the same format. The font information that you interchange can be a complete font resource, selected components of a font resource, or reference information about a font resource. For example, a host-based application might only require the metric information for formatting a document, while a printer might require both the metric and shape information for printing that same document. The document itself might only contain a reference to the font resource, since the resource might already exist on both the host and printer.

The font information might be interchanged on transportable media (for example, magnetic or optical disk, diskette, or tape), or through a communications network. The formats permitted for FOCA information interchange depend on the scope of the interchange. If the scope of the interchange is among a known set of products, with all sending and receiving products recognizing a commonly defined font information format, then those products might use any agreed upon private font resource format. If the scope of the interchange is not known and the capabilities of receiving products is not known, then those products must use a recognized public font information format.

The parameter definitions in FOCA permit parameters to be expressed in a variety of formats. See "Parameter Formats" on page 55 for a brief description of three common formats supported by FOCA. In addition, the parameter definitions in FOCA include transformation information required for support of public interchange using formats defined in the ISO/IEC 9541 *Font Information Interchange* standard.

This chapter specifies the public font information interchange formats supported by this architecture.

AFP System Font Resource

The format for font resource data, to be loaded and managed by Advanced Function Presentation (AFP) software, is defined by the following syntax specification. The syntax specification is divided into three sections, defining objects, structured fields, and triplets. The first section describes the three AFP font resource objects: coded fonts, code pages, and font character sets. Each AFP font resource object is composed of self-identifying structured fields, which are unique to that resource object, and self-identifying triplets that can occur in multiple structured fields and/or resource objects. Structured fields and triplets contain a set of parameters (defined by the architecture) in an architected structure of fixed and/or variable length value fields. The objects, structured fields, and triplets in each of the following three sections are arranged in alphabetic order to aid the reader in locating the information. The order, location, and quantity of structured fields and triplets within an object is defined by the architecture, and must occur as specified, except where variation is specifically permitted by the architecture.

Objects

Coded Font

A *coded font* is an AFP font resource object that associates AFP font character set objects with AFP code page objects. The metric and shape information for each of the characters defined by the font is contained in the font character set, and the information for mapping the code points used in the document data stream to the graphic character global identifiers used in the font character set is contained in the code page. A coded font resource does not contain either the font character set object or the code page object, but only references those objects by name. If the font character set referenced by the coded font contains scalable outline shape data, the coded font object might also include the desired vertical font size and horizontal scale factor.

The structured fields that compose the coded font must appear in the following sequence (Figure 51):

1. **Begin Coded Font (BCF)**

   Identifies the object as a Coded Font, and identifies this particular object by name.

2. **Coded Font Control (CFC)**

   Provides control information that is used to manage and interpret the data contained in the other structured fields of the object.

3. **Coded Font Index (CFI)**

   Identifies one or more Font Character Set and Code Page object pairs, and associated size information for outline font scaling.

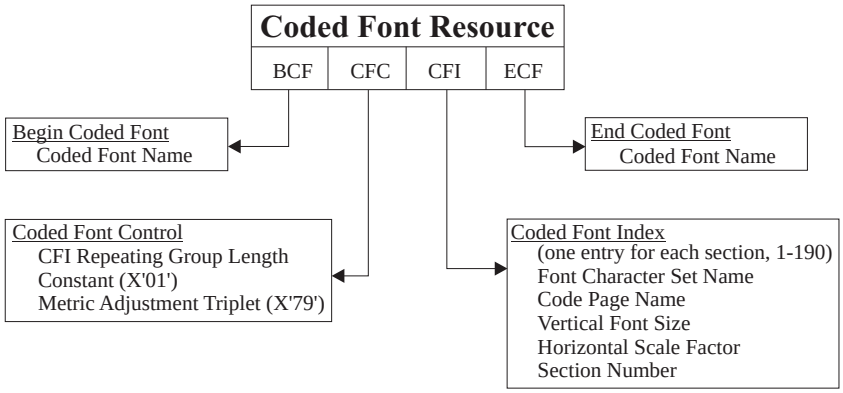
4. **End Coded Font (ECF)**

   Identifies the end of the Coded Font object.

5. **No Operation (NOP)**

   Provides a means to carry comment information. One or more NOP structured fields can appear between any two structured fields within a coded font.

Figure 51. Structured Fields for Coded Fonts



A single-byte coded font has only one section and associates one font character set with one single-byte code page. Code point values X'00' to X'FF' may be used to identify up to 256 characters. A single-byte coded font section is always identified as X'00' in the Coded Font Index structured field.

A double-byte coded font may consist of multiple sections to permit identification of all the characters in a writing system such as Kanji. Two bytes are needed to identify each character. The first byte identifies the section, while the second byte identifies the code point (character) within that section. The number of characters that can be identified with a double-byte coded font is equal to the number of code points allowed in a section (256 maximum) multiplied by the number of sections allowed (256 maximum).

Double-byte Section Numbering

Raster Coded Fonts: Sections are numbered from X'41' to X'FE' and are identified in the Coded Font Index (CFI) structured field. The maximum number of code points that can be defined in each section is 256 (code points X'00' to X'FF').

Outline Coded Fonts: Up to 256 sections are permitted in a double-byte code page, depending on the encoding system specified (190 maximum for EBCDIC encoding). When specifying a double-byte outline coded font, the CFI section number field is set to X'00', indicating all sections of the associated double-byte code page object.

Code Page

A code page is a font resource object that associates graphic character global IDs (GCGIDs) to code points. A single-byte code page or a double-byte code page section can define up to 256 code points. A double-byte code page can define up to 65,536 code points. Flags for each code point show if the character is valid, printable, and incrementing.

If a code point is not specified in the code page, the *default character* from the Code Page Control (CPC) structured field is printed when that code point is used, depending on code page flags and server-specified data check values. Only one graphic character ID (GCGID) can be assigned per code point. However, the same GCGID can be assigned to several code points.

The structured fields that compose the code page must appear in the following sequence (Figure 52 on page 114):

1. **Begin Code Page (BCP)**

   Identifies the object as a Code Page, and identifies this particular object by name.

2. **Code Page Descriptor (CPD)**

   The CPD provides the IBM registered global identifiers for the code page and its associated graphic character set.

   Application Note: The CPD was optional in the past (code page could be identified by its resource name), but is now required. Existing applications should not be impacted by the presence of this structured field (should ignore the field). AFP products which support MO:DCA MCF/2 GRID referencing of coded fonts and/or which support the IPDS download of code page resources might require the data contained in the CPD structured field. Some code pages might contain the structured field but not the global identifiers needed by these products. In such cases, the product might not have sufficient information to continue processing.

3. **Code Page Control (CPC)**

   The CPC gives the default character ID information for the code page. The default character is used when a code point is referenced in text and no character ID has been assigned to that code point in the Code Page Index (CPI) structured field.

4. **Code Page Index (CPI)**

   The CPI matches character IDs and code points. If a code point is not matched with a character ID, the code point used will be the default character ID specified in the CPC.

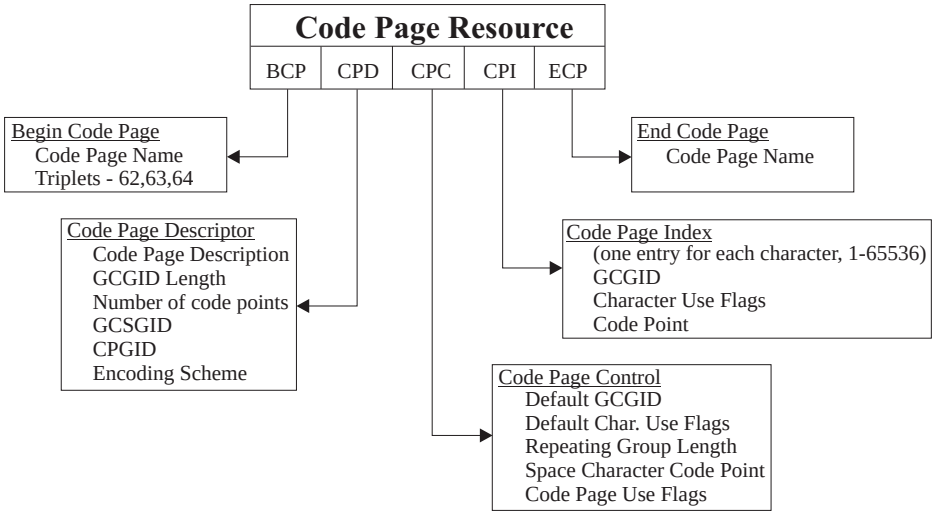
5. **End Code Page (ECP)**

   Identifies the end of the Code Page object.

6. **No Operation (NOP)**

   Provides a means to carry comment information. One or more NOP structured fields can appear between any two structured fields within a code page.

Figure 52. Structured Fields for Code Pages



Note: Up to 256 repeating groups can be used in a single-byte code page or in a double-byte code page section. Successive CPI structured fields are used to define each double-byte code page section. Each repeating group and CPI is sorted according to the Sort Order flag in the CPC structured field.

Font Character Set

A font character set is a font resource object that contains descriptive and metric information for the whole font, and metric and shape information for each character identifier in the font.

The structured fields that compose the font character set must appear in the following sequence (Figure 53 on page 116):

1. **Begin Font (BFN)**

   Identifies the object as a Font Character Set, and identifies this particular object by name.

2. **Font Descriptor (FND)**

   The FND describes characteristics common to all characters, such as size and stroke thickness.

3. **Font Control (FNC)**

   The FNC specifies defaults and other information about the font character set.

4. **Font Patterns Map (FNM)**

   For each character, the FNM specifies the raster pattern box size and the address of the character raster pattern in the Font Patterns (FNG) structured fields. This structured field is omitted for fonts that do not contain shape data (raster or outline).

5. **Font Orientation (FNO)**

   The FNO includes character rotation, maximum or uniform values, and other font data. Fonts have one, two, three, or four FNOs (one per rotation).

6. **Font Position (FNP)**

   The FNP specifies the common metrics. Fonts have one, two, three, or four FNPs (one per rotation).

7. **Font Index (FNI)**

   For each character in a raster font, the FNI points to an entry in the FNM structured field and provides character positioning information. For outline fonts, the FNI defines the character increment for each character whose character increment is different from that specified in the FNO for that rotation. The FNI

for a given character rotation is specified in the appropriate repeating group of the FNO structured field. Fonts have one, two, three, or four FNIs (one per rotation).

8. **Font Name Map (FNN)**

   The FNN maps the AFP character names contained in the FNI structured field to the outline font technology character names contained in the FNG structured field. This structured field is omitted from fonts that do not have outline fonts included in their FNGs.

9. **Font Patterns (FNG)**

   The FNG contains the character raster pattern or outline font data. This structured field is omitted for Metric Only Fonts (MOFs) that do not contain shape information.

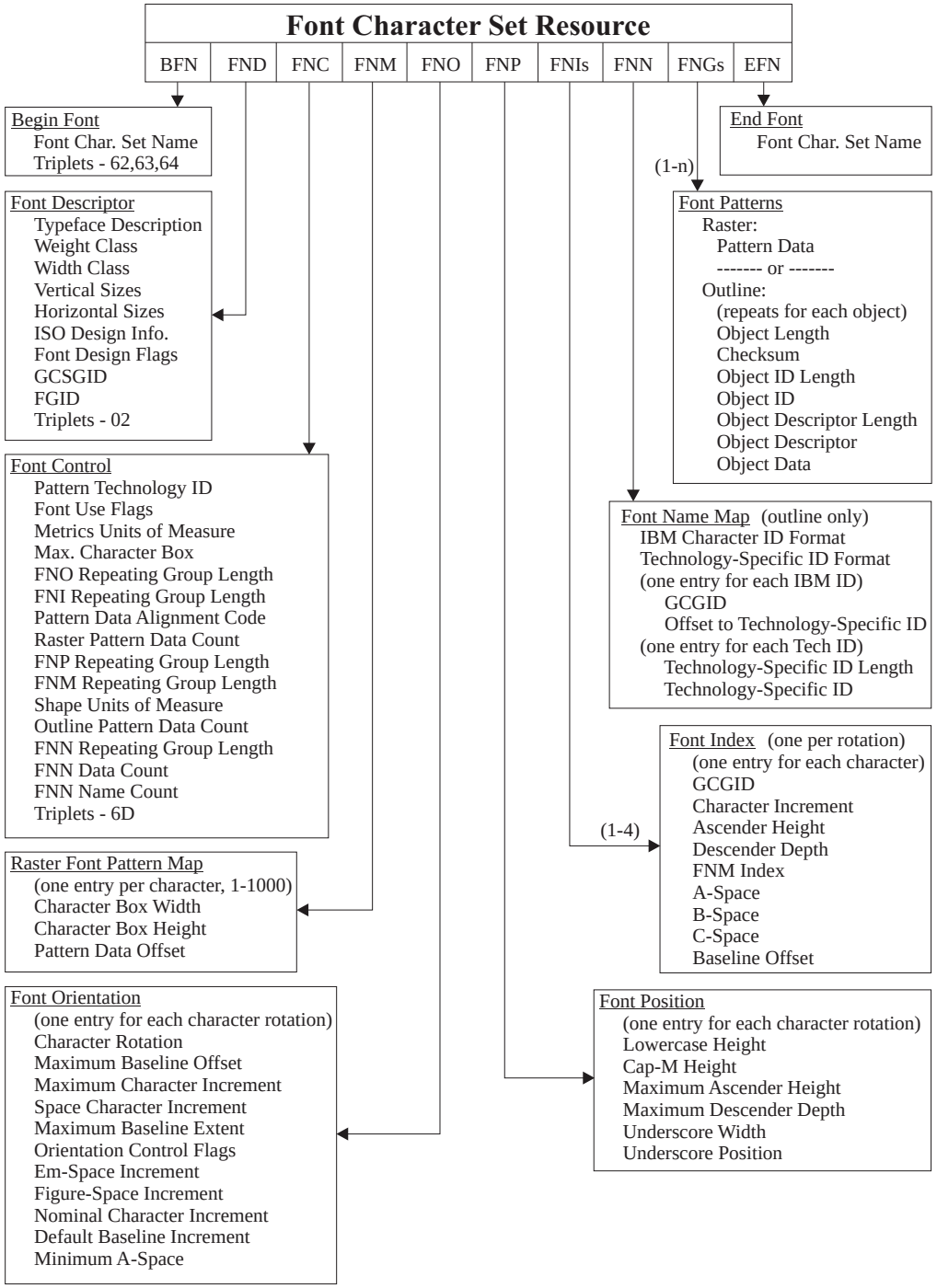
10. **End Font (EFN)**

    Identifies the end of the Font Character Set object.

11. **No Operation (NOP)**

    Provides a means to carry comment information. One or more NOP structured fields can appear between any two structured fields within a font character set.

Figure 53. Structured Fields for Font Character Sets

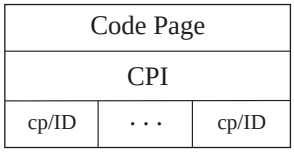


Associating Character Identifiers

With Code Points

The Code Page Index (CPI) structured field in the code page has repeating groups of parameters. Each repeating group pairs a character ID with a code point.

Figure 54. Associating Character Identifiers with Code Points



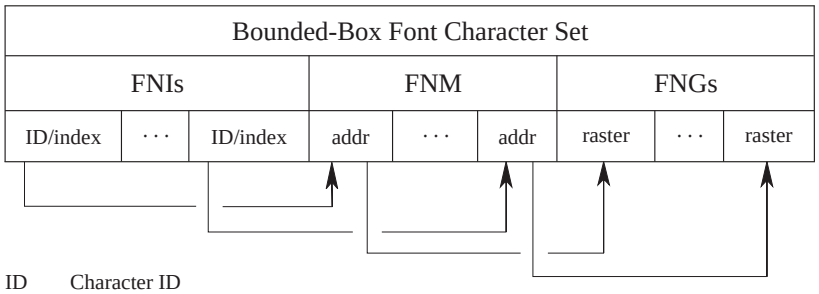
cp Code Point
ID Character ID

With Raster Pattern Addresses

A font can have one, two, three, or four Font Indexes (FNIs), one for each of the four possible character rotations. Each FNI in the font character set has repeating groups of parameters. Each repeating group pairs a character ID with a raster pattern address. The following list and diagram show the relationship of the character IDs and the raster pattern addresses.

- The FNI parameter in the Font Orientation (FNO) structured field (for the current character rotation) specifies the FNI to be used.
- Each repeating group in the FNI pairs a character ID and an index into the Font Patterns Map (FNM) structured field.
- The FNM contains the addresses of the character raster patterns.
- The raster patterns are in the Font Patterns (FNG) structured fields of the font character set. The character identifiers in the FNI are used to locate each character raster pattern.

Figure 55. Associating Character IDs with Raster Pattern Addresses



ID Character ID
addr Raster Pattern Address

Single-Byte Raster Coded Font Summary

- Each character in the text string is represented by a one-byte code point (for example, X'C1' for the uppercase character A).
- The character ID that is paired with the code point in the code page is matched with the same character ID in the font character set.
- The character ID in the font character set is, in turn, paired with the character raster pattern for the character.

Double-Byte Raster Coded Font Summary

- Each character in the text string is represented by a two-byte code (for example, X'41C1' for section 65 uppercase character A). The first byte shows the correct section (code page/font character set pair). The second byte is the code point.
- The character ID that is paired with the code point in the code page is matched with the same character ID in the font character set.
- The character ID in the font character set is, in turn, paired with the character raster pattern for the character.

With Outline Fonts

When the FNC pattern technology field indicates an outline font is included in the FNG structured field, the FNG is preceded by a Font Name Map (FNN) structured field which provides a map between the outline font character names and the IBM graphic character identifiers (GCGID). The Font Patterns Map (FNM) is omitted from fonts that have outline fonts included in the FNG.

The FNN is divided into three sections. The first section described the encoding scheme used in the second and third sections. The second section contains IBM character names (GCGID) and the offset of the corresponding outline font character name in the third section. The second section is sorted so the GCGIDs are in ascending order. The third section has outline font character names. Each outline font character name in third section is preceded by a length field giving the length of the following outline font character name.

Outline Single and Double-byte Coded Font Summary

- Mapping for single-byte and double-byte outline coded fonts are each treated the same. That is, it makes no difference how many code points are present in the code page object. The FNN structured field provides the necessary name mapping information.
- A character's outline representation name is paired with a character's IBM GCGID using the Font Name Map.

Structured Fields

Resource objects contain structured fields. In some environments, each structured field is preceded by a X'5A' carriage control character, which is *not* part of the structured field.

Each structured field must have an introducer and usually has parameter bytes that contain control information or raster patterns. A structured field can also include padding bytes.

Structured Field Components

	X'5A'	Structured Field Introducer	Structured Field Parameters	Structured Field Padding
Number of bytes allowed	1	8 to 263	0 to 32,759	0 to 32,759

In this publication, the structured fields are presented alphabetically, not in required sequence.

For each structured field, the following information is included:
- Purpose
- Meaning and allowed values for the parameters
- Contents of constant parameters

The section heading for each structured field gives its abbreviation, the three-byte hexadecimal code identifier, and the name of the structured field.

The contents of each structured field are presented in a table, and additional detailed descriptions follow the table. Only the structured field parameters are described. The X'5A', the introducer, and the optional padding values are not included in these descriptions.

For all structured fields, the following conventions apply:

- The byte offset of each structured field parameter is given.
- The first byte in the data field (after the structured field introducer) is numbered X'00' (byte 0).
- A number not preceded by X (hexadecimal) or B (binary) is a decimal number.
- Bit numbering follows the EBCDIC convention, with bit 0 being the most significant bit.
- A range of permitted values is provided for each parameter. The specified range is the architected range, not that used by any one implementation. If the parameter is a constant or a reserved parameter, the single value, not a range, is provided in the Range column of the tables. The purpose of the constant might not be given. A reserved parameter need not be checked; however, such parameters should be set to the specified value (if one is given) or to 0 (zero).

  Note: Fields marked "Reserved" are for future use. If bytes beyond the structured field bytes described here are specified, results are unpredictable.
- Some structured fields contain data in repeating groups. For example, in a Code Page Index (CPI) structured field, each repeating group specifies a character ID, a set of flags, and a code point. In this publication, numbering of the repeating groups starts with zero (0).
- The structured field parameter positions are fixed; however, some structured fields have optional parameters. To maintain the fixed position requirements, all optional fields before the last one used must be coded.
- Whenever a range is given, the "to" is inclusive. For example, the range "1 to 6" is the same as "1, 2, 3, 4, 5, 6".
- A mandatory or optional indicator is provided with each parameter to indicate whether or not the parameter field must be provided; not that the parameter value must be nonzero.

The structured field tables have a column called **Type**. The column contains one of the following notations, notation meanings are shown in the **Comments** column.

Table 12. Notations in the Type Column in Structured Field Tables

Type	Meaning
BITS	Binary flags, where each binary bit is assigned a specific meaning.
CHAR	Alphanumeric characters whose code points are defined in the EBCDIC code page 500. For example, the characters ABC1 can be represented as X'C1C2C3F1'. The characters that can be used are a restricted set in this code page. The characters that can be used are identified in the registered IBM graphic character set for international data processing (GCSGID 103). Characters that can be used within fields with data type CHAR are shown in Figure 56 on page 121.
CODE	A numeric parameter in which each value is assigned a specific meaning.
SBIN	A numeric parameter. A one-byte parameter can contain a value in the range of -128 to +127. For example, a decimal +19 would be X'13' and a decimal -19 would be X'ED'. Negative values are in twos complement form, for example: • X'7FFF' = +32,767 • X'8000' = -32,768
UBIN	A numeric parameter. A one-byte parameter can contain a value of decimal 0 to 255 (X'00' to X'FF').
UNDF	An undefined value. The content of this parameter value is not defined by this architecture.
Note: Signed and unsigned binary fields are shown in hexadecimal.	

Figure 56. EBCDIC Code Page 500 With Character Set 103

GCSGID = 103, CPGID = 500

Hex Digits 1st → 2nd ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	– SP100000						{ SM110000	} SM140000	\ SM070000	0 ND100000
-1			/ SP120000		a LA010000	j LJ010000	~ SD190000		A LA020000	J LJ020000		1 ND010000
-2					b LB010000	k LK010000	s LS010000		B LB020000	K LK020000	S LS020000	2 ND020000
-3					c LC010000	l LL010000	t LT010000		C LC020000	L LL020000	T LT020000	3 ND030000
-4					d LD010000	m LM010000	u LU010000		D LD020000	M LM020000	U LU020000	4 ND040000
-5					e LE010000	n LN010000	v LV010000		E LE020000	N LN020000	V LV020000	5 ND050000
-6					f LF010000	o LO010000	w LW010000		F LF020000	O LO020000	W LW020000	6 ND060000
-7					g LG010000	p LP010000	x LX010000		G LG020000	P LP020000	X LX020000	7 ND070000
-8					h LH010000	q LQ010000	y LY010000		H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9				` SD130000	i LI010000	r LR010000	z LZ010000		I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	[SM060000	] SM080000		: SP130000								
-B	. SP110000	$ SC030000	, SP080000	# SM010000				| SM130000				
-C	< SA030000	* SM040000	% SM020000	@ SM050000								
-D	(SP060000	) SP070000	_ SP090000	' SP050000								
-E	+ SA010000	; SP140000	> SA050000	= SA040000								
-F	! SP020000	^ SD150000	? SP150000	" SP040000								

A structured field introducer begins each structured field.

Table 13. Structured Field Introducer

Offset	Type	Name	Range	Meaning	M/O
0–1	UBIN	Length	8–32,767	Length of Structured Field	M
2–4	CODE	SFID	See Below	Structured Field Identifier	M
5	BITS	SFFlags	See Below	Control Flags	M
6–7	UBIN	Sequence Number	1–32,767	The number of the structured field in the object.	M
8	UBIN	Extension Length	1–255	Length of extension data	O
9–*n*	UNDF	Extension Data		Up to 254 bytes of data	O
x–*y*	UNDF	Padding Data		Up to 32,759 bytes of data	O

Bytes Description

0–1 Length

A two-byte count field that specifies the length of the structured field. The length value can range from 8 to 32,767. The count includes the length itself, the structured field introducer parameters, and the structured field data contents, including padding bytes when present. The count does not include the X'5A' control character.

Note: Some platforms include structured fields in a larger platform-specific record by surrounding the structured field with additional bytes (for example, the X'5A' prefix). This can result in a record length greater than 32,767 if the structured field length is 32,767. Such a record length can be misinterpreted as a negative number if the length is treated as SBIN. To ensure portability of print files, it is recommended that the maximum structured field length be limited to X'7FF0' = 32,752, which avoids such record length issues on the known platforms.

2–4 Identifier

A three-byte value that identifies the type of structured field; the identifiers (in hexadecimal) are as follows:

Table 14. Structured-Field Identifiers

Field Name Abbreviation	Assigned Number	Full Field Name
BCF	X'D3A88A'	Begin Coded Font
BCP	X'D3A887'	Begin Code Page
BFN	X'D3A889'	Begin Font
CFC	X'D3A78A'	Coded Font Control
CFI	X'D38C8A'	Coded Font Index
CPC	X'D3A787'	Code Page Control
CPD	X'D3A687'	Code Page Descriptor
CPI	X'D38C87'	Code Page Index
ECF	X'D3A98A'	End Coded Font
ECP	X'D3A987'	End Code Page

Table 14 Structured-Field Identifiers (cont'd.)

Field Name Abbreviation	Assigned Number	Full Field Name
EFN	X'D3A989'	End Font
FNC	X'D3A789'	Font Control
FND	X'D3A689'	Font Descriptor
FNG	X'D3EE89'	Font Patterns
FNI	X'D38C89'	Font Index
FNM	X'D3A289'	Font Patterns Map
FNN	X'D3AB89'	Font Names **(Outline Fonts Only)**
FNO	X'D3AE89'	Font Orientation
FNP	X'D3AC89'	Font Position
NOP	X'D3EEEE'	No Operation

5 Flags

A one-byte field that specifies the value of optional structured-field indicators.

Bit 0 Extension Indicator. Shows whether a structured field introducer extension is present.
- **B'0'** No extension is present.
- **B'1'** An extension is present.

Bit 1 B'0'

Bit 2 Segmentation Indicator. Shows whether a structured field has been divided into multiple segments; required when there is more data than will fit in a single structured field.
- **B'0'** No segmentation is present.
- **B'1'** Segmentation is present.

Bit 3 B'0'

Bit 4 Padding Indicator. Shows whether the structured field includes padding bytes (up to 32,759).
- **B'0'** No padding is present.
- **B'1'** Padding is present.

Bits 5–7 B'000'

6–7 Sequence Number

A two-byte correlation field that applications can use to label specific structured fields so the fields can be located more easily in a data stream. Applications define the sequence-numbering conventions used. Presentation services software does not validate sequence numbering; therefore, applications can use any numbering convention.

Note: The MO:DCA architecture does not use the sequence number field and recommends that it be set to zero for structured fields defined in the MO:DCA Reference.

8 Extension Length

This parameter is valid only if bit 0 of byte 5 is set to B'1'. The first byte of the extension field must specify its length. The extension field length may be from 1 to 255, counting the length byte itself.

9–*n* Extension Data

If present, the extension data contains specific orders, parameters, and data appropriate for the given structured field. The maximum length of the remainder of the structured field is 32,759 minus the length in byte 8.

x–y Padding

If bit 4 (padding flag) is on in the flag byte, padding bytes have been added to the structured field. Structured field padding must include the padding length value. There are two methods:

1. Specify the length in the last padding byte
2. Specify the length in the last three padding bytes; the last byte must be X'00' and the two preceding bytes specify the padding length.

The method to use depends on the number of padding bytes:

Number of Padding Bytes	Method Used
1 or 2	1
3 to 255	1 or 2
256 to 32,759	2

BCF – D3A88A – Begin Coded Font

The Begin Coded Font (BCF) structured field begins a coded font object.

Length (2 bytes)	**X'D3A88A'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (0 or 8 bytes)

The data for the BCF structured field is as follows:

Offset	Type	Name	Range	Meaning	M/O
0–7	CHAR	CFName		Coded Font Name	O

Bytes Description

0–7 Coded Font Name

If used, it is suggested that the coded font name be the same name used to reference the object. Refer to the Map Coded Font (MCF) in the *Mixed Object Document Content Architecture Reference*.

Character string names are encoded using CPGID 500, GCSGID 103 (see "Parameter Types" on page 55). Other architectures or implementations may permit more or less characters than are contained in GCSGID 103. For interchange purposes, CPGID=500, GCSGID=961 is recommended; this code page contains the following characters:

- A–I (code points X'C1'–X'C9')
- J–R (code points X'D1'–X'D9')
- S–Z (code points X'E2'–X'E9')
- 0–9 (code points X'F0'–X'F9')
- $, #, and @ (code points X'5B', X'7B', and X'7C' respectively)

Note: If a Begin Coded Font (BCF) structured field has a name, the corresponding End Coded Font (ECF) structured field name should match. However, in an ECF structured field, no name or a null name (any name with X'FFFF' in the first two bytes) will match any name in the BCF structured field.

BCP – D3A887 – Begin Code Page

The Begin Code Page (BCP) structured field begins a code page object.

Length (2 bytes)	**X'D3A887'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (0, 8, or 10–32,759 bytes)

The data for the BCP structured field is as follows:

Offset	Type	Name	Range	Meaning	M/O
0–7	CHAR	CPName		Code Page Name	O
8–*n*	UNDF	Triplets	See Below	Self Defining Triplets	O

Bytes Description

0–7 Code Page Name

If used, it is suggested that the code page name be the same name used to reference the object. See bytes 8–15 of the Coded Font Index (CFI) repeating group in this document, or the Map Coded Font (MCF) in the *Mixed Object Document Content Architecture Reference*.

Character string names are encoded using CPGID 500, GCSGID 103 (see "Parameter Types" on page 55). Other architectures or implementations may permit more or less characters than are contained in GCSGID 103. For interchange purposes, CPGID=500, GCSGID=961 is recommended; this code page contains the following characters:

- A–I (code points X'C1'–X'C9')
- J–R (code points X'D1'–X'D9')
- S–Z (code points X'E2'–X'E9')
- 0–9 (code points X'F0'–X'F9')
- $, #, and @ (code points X'5B', X'7B', and X'7C' respectively)

Note: If a BCP structured field has a name, the corresponding End Code Page (ECP) structured field name must match. In an ECP structured field, no name or a null name (any name with X'FFFF' in the first two bytes) matches any name in the BCP structured field.

8–*n* Triplets

The following triplets may only occur once:

X'62', Type 0 "X'62' – Local Date and Time Stamp Triplet" on page 178 (not permitted when X'62' Type 3 is present)

X'62', Type 1 Retired (Local RMARK Date and Time Stamp triplet)

X'62', Type 3 Local Revision Date and Time Stamp triplet (not permitted when X'62' Type 0 is present)

X'63', Type 1 "X'63', Type 1 – CRC Resource Management Triplet" on page 180

X'63', Type 2 Retired ("X'63', Type 2 – Font Resource Management Triplet" on page 181)

X'64' "X'64' – Object Origin Identifier Triplet" on page 183

These triplets provide additional data used to manage this code page. Print-server software uses this information to determine whether or not to download a resource or to activate a resident resource. Programs, such as IBM Remote PrintManager™, or a printer with resident resources, use this information for:

- Quick screening of resources
- Identification of identical objects from different network sources
- Notification when an object has been changed

The following triplet can occur multiple times:

X'65' “X'65' – User Comment Triplet” on page 184

The triplets can occur in any order.

BFN – D3A889 – Begin Font

The Begin Font (BFN) structured field begins the font character set object.

Length (2 bytes)	**X'D3A889'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (0, 8, or 10–32,759 bytes)

The data for the BFN structured field is as follows:

Offset	Type	Name	Range	Meaning	M/O
0–7	CHAR	CSName		Font Character Set Name	O
8–*n*	UNDF	Triplets	See Below	Self Defining Triplets	O

Bytes Description

0–7 Font Character Set Name

If used, it is suggested that the font character set name be the same name used to reference the object. See bytes 0–7 of the Coded Font Index (CFI) repeating group in this document, or the Map Coded Font (MCF) of the *Mixed Object Document Content Architecture Reference*.

Character string names are encoded using CPGID 500, GCSGID 103 (see "Parameter Types" on page 55). Other architectures or implementations may permit more or less characters than are contained in GCSGID 103. For interchange purposes, CPGID=500, GCSGID=961 is recommended; this code page contains the following characters:

- A–I (code points X'C1'–X'C9')
- J–R (code points X'D1'–X'D9')
- S–Z (code points X'E2'–X'E9')
- 0–9 (code points X'F0'–X'F9')
- $, #, and @ (code points X'5B', X'7B', and X'7C' respectively)

Note: If a BFN structured field has a name, the corresponding EFN structured field name should match. However, in an EFN structured field, no name or a null name (any name with X'FFFF' in the first two bytes) matches any name in the BFN structured field.

8–*n* Triplets

The following triplets may only occur once:

X'62', Type 0 "X'62' – Local Date and Time Stamp Triplet" on page 178 (not permitted when X'62' Type 3 is present)

X'62', Type 1 Retired (Local RMARK Date and Time Stamp triplet)

X'62', Type 3 Local Revision Date and Time Stamp triplet (not permitted when X'62' Type 0 is present)

X'63', Type 1 "X'63', Type 1 – CRC Resource Management Triplet" on page 180

X'63', Type 2 Retired ("X'63', Type 2 – Font Resource Management Triplet" on page 181)

X'64' "X'64' – Object Origin Identifier Triplet" on page 183

These triplets provide additional data used to manage this font character set. Print-server software uses this information to determine whether or not to download a resource or to activate a resident resource. Programs, such as IBM Remote PrintManager, or a printer with resident resources, use this information for:

- Quick screening of resources
- Identification of identical objects from different network sources
- Notification when an object has been changed

The following triplet can occur multiple times:

X'65' “X'65' – User Comment Triplet” on page 184

The triplets can occur in any order.

CFC – D3A78A – Coded Font Control

The Coded Font Control (CFC) structured field specifies the length of the repeating group in the Coded Font Index (CFI) structured field.

Length (2 bytes)	**X'D3A78A'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (2 or 17 bytes)

The data for the CFC structured field is as follows:

Offset	Type	Name	Range	Meaning	M/O
0	UBIN	CFIRGLen	X'19'	CFI Repeating Group Length	M
1	UBIN	Retired	X'01'	Retired Parameter	M
2–end		Triplet		Metric Adjustment triplet (X'79')	O

Bytes **Description**

0 CFI Repeating Group Length

This is a control parameter, used to manage the data structures. The value contained in this parameter defines the length of the repeating group used in the Coded Font Index (CFI) structured field.

1 Constant

This is a retired parameter which must be set to a constant value of X'01'. No significance should be attached to the value by any using application, but any font generator should set the value as indicated.

2 to end Triplet

An optional Metric Adjustment triplet can be specified to provide adjustment values for certain font metrics. Refer to "X'79' – Metric Adjustment Triplet" on page 186 for a description of these adjustment values.

CFI – D38C8A – Coded Font Index

The Coded Font Index (CFI) structured field names the font character sets and code pages for the font. The structured field contains a set of parameters, defined as a repeating group. The length of the repeating group is defined in the Coded Font Control structured field. The number of repeating groups in the structured field can be determine by dividing the length of the CFI structured field (minus the length of the structured field introducer) by the length of the CFI repeating group. The repeating groups are sorted in ascending order based on the section identifier.

Single-Byte and Double-byte Outline

There is only one section for a single-byte font, or a double-byte outline font; therefore, there is only one repeating group.

Double-Byte Raster Coded Fonts

There can be 190 sections (X'41' to X'FE'). Each section requires its own repeating group. The repeating groups are sorted in ascending order based on the section identifier.

Length (2 bytes)	**X'D38C8A'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (25*[number of sections] bytes)

The data for the CFI structured field consists of a series of repeating groups. Each repeating group is defined as follows:

Offset	Type	Name	Range	Meaning	M/O
0–7	CHAR	FCSName		Font Character Set Name	M
8–15	CHAR	CPName		Code Page Name	M
16–17	UBIN	SVSize	0–65,535	Specified Vertical Font Size, in 20ths of a point (1440ths of an inch)	M
18–19	UBIN	SHScale	0–65,535	Specified Horizontal Scale Factor, in 20ths of a point (1440ths of an inch)	M
20–23	UNDF		X'00000000'	Reserved	M
24	UBIN	Section	X'00' X'41'–X'FE' X'FF'	Section Number Single-byte Double-byte Raster reserved	M

Bytes **Description**

0–7 Font Character Set Name

See "Resource Name" on page 69.

The member name of the font character set can be any value except null (any name that contains X'FFFF' in the first two bytes) or eight blanks (X'4040 4040 4040 4040').

Character string names are encoded using CPGID 500, GCSGID 103 (see "Parameter Types" on page 55). Other architectures or implementations may permit more or less characters than are contained in GCSGID 103. For interchange purposes, CPGID=500, GCSGID=961 is recommended; this code page contains the following characters:

- A–I (code points X'C1'–X'C9')
- J–R (code points X'D1'–X'D9')
- S–Z (code points X'E2'–X'E9')
- 0–9 (code points X'F0'–X'F9')
- $, #, and @ (code points X'5B', X'7B', and X'7C' respectively)

Raster Fonts

IBM's naming convention, which is not part of the FOCA architecture, includes a two-character prefix of *C0* followed by one to six non-null characters. IBM presentation services software uses the entire name to locate the font character set to be downloaded. Some presentation services software and other Advanced Function Presentation (AFP) programs interpret font names using the two-character prefix convention.

Outline Fonts

IBM's naming convention, which is not part of the FOCA architecture, includes a two-character prefix of *CZ* followed by one to six non-null characters.

8–15 Code Page Name

See "Resource Name" on page 69.

The member name of the code page can be any value except null (any name that contains X'FFFF' in the first two bytes) or eight blanks (X'4040 4040 4040 4040').

Character string names are encoded using CPGID 500, GCSGID 103 (see "Parameter Types" on page 55). Other architectures or implementations may permit more or less characters than are contained in GCSGID 103. For interchange purposes, CPGID=500, GCSGID=961 is recommended; this code page contains the following characters:

- A–I (code points X'C1'–X'C9')
- J–R (code points X'D1'–X'D9')
- S–Z (code points X'E2'–X'E9')
- 0–9 (code points X'F0'–X'F9')
- $, #, and @ (code points X'5B', X'7B', and X'7C' respectively)

16–17 Specified Vertical Font Size

See "Specified Vertical Font Size" on page 71.

This field is intended for outline font support, when no other size information is available in the font reference (for example, MCF GRID or Font Descriptor). For raster fonts, this field is not used. A nonzero value, when no other scaling information is available, indicates the size to which an outline font shall be scaled, expressed in 20ths of a point (1440ths of an inch). For example, 10.8 points is specified as 216. A nonzero value, when other scaling information is available, shall be ignored. A zero value indicates that no size information has been specified.

Note: The IPDS and MO:DCA architectures limit the range of the vertical font size to 0–32,767. It is recommended that this smaller range also be used in the CFI structured field.

18–19 Specified Horizontal Scale Factor

See "Specified Horizontal Scale Factor" on page 70.

This field is intended for outline font support, when no other size information is available in the font reference (for example, MCF GRID or Font Descriptor). For raster fonts, this field is not used. A nonzero value is only to be used in conjunction with the Vertical Font Size in bytes 16–17 of this same structured field to determine the horizontal scaling ratio to be applied to the characters of the font. The value is expressed in 20ths of a point (1440ths of an inch). A nonzero value, when any other size information is available (for example, a vertical or horizontal size value expressed in the MCF GRID or Font Descriptor) shall be ignored. A zero value indicates that no horizontal scaling information has been specified.

Note: The IPDS and MO:DCA architectures limit the range of the horizontal scale factor to 0–32,767. It is recommended that this smaller range also be used in the CFI structured field.

20–23 Reserved

24 Section Number

See "Section Number" on page 107.

Single-Byte

Must be X'00' to identify a single-byte section.

Double-Byte Raster Coded Fonts

Must be X'41' to X'FE' to identify a double-byte section.

This section number is the first byte of the two-byte code that identifies the character in a coded font and in text.

Double-Byte Outline Coded Fonts

Must be X'00' to identify all double-byte sections.

CPC – D3A787 – Code Page Control

The Code Page Control (CPC) contains information about the code page.

Length (2 bytes)	**X'D3A787'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (13 or 17 bytes)

The data for the CPC structured field is as follows:

Offset	Type	Name	Range	Meaning	M/O
0–7	CHAR	DefCharID		Default Graphic Character Global ID	M
8	BITS	PrtFlags	See Below	Default Character Use Flags	M
9	CODE	CPIRGLen	 X'0A' X'0B' X'FE' X'FF'	CPI Repeating Group Length: Single-byte Code Page Double-byte Code Page Single-byte Code Page including Unicode scalar values Double-byte Code Page including Unicode scalar values	M
10	UBIN	VSCharSN	X'00'–X'FF'	Space Character Section Number	M
11	UBIN	VSChar	X'00'–X'FF'	Space Character Code Point	M
12	BITS	VSFlags	See Below	Code Page Use Flags	M
Zero or one Unicode scalar value for the default character ID (only present when byte 9 contains either X'FE' or X'FF') in the following format:					
+ 0–3	UBIN	Unicode scalar value	X'00000000' – X'FFFFFFFF' (excluding surrogates)	Unicode scalar value to be mapped to the default character GCGID in bytes 0–7	O

Bytes Description

0–7 Default Graphic Character Global ID

See "Graphic Character Global Identifier" on page 95.

The character ID of the character used when no character is assigned to a code point. The default character must be named in bytes 0–7 of a repeating group in the Font Index (FNI) structured field.

This parameter is EBCDIC encoded using CPGID 500 and GCSGID 103.

Note: The letters in a character ID are *case-sensitive*. Therefore LA020000, La020000, lA020000, and la020000 denote four different characters. IBM font products use uppercase alphanumeric character IDs based on IBM Corporate Standards. While other naming schemes can be used, it is important to use a consistent naming scheme across all font objects in a system; therefore, the GCGID scheme used by IBM font products is recommended.

Double-Byte Raster Coded Fonts

This parameter must be set the same for all sections of a double-byte raster coded font.

Note: In IPDS environments, the default Graphic Character Global ID is used with raster fonts for unassigned code points (for which there is no CPI repeating group). IPDS raster fonts and double-byte font sections must be fully populated; therefore, the default GCGID is used to fill in the holes between defined characters.

In an IPDS outline font, the code page does not have to be fully populated, and the IPDS code page does not always contain a default Graphic Character Global ID (older IPDS outline-font printers do not support the default GCGID parameters). All code points that do not have a code-point entry are assigned the default GCGID, if one is present in the IPDS command that begins the code page. If a default GCGID is not provided in the IPDS command that begins the code page, each code point without a code-point entry is assigned the *undefined, non-printing, incrementing* processing flags, and if printing is continued as part of an error continuation action, the space character code point is used instead of the default graphic character ID.

Therefore, to ensure consistent results on all IPDS printers, it is good practice to set the Default Graphic Character Global ID to the same GCGID as that used by the space character so that this code page can be used to print with both raster and outline font character sets. Also, when you are using an older IPDS outline-font printer that does not support the default GCGID parameter, if a default GCGID other than the space character is required for a code page to be used with an outline font, that code page should be fully populated by using the desired default GCGID to fill in the holes in the code page.

Refer to the *Intelligent Printer Data Stream Reference* for a description of the IPDS font objects.

8 Default Character Use Flags

Bit 0 Invalid Coded Character

When this flag bit is set to B'1' (on), the character is an invalid coded character (see "Invalid Coded Character" on page 106). When this flag bit is set to B'0' (off), the character is a valid coded character.

Bit 1 No Presentation

When this flag bit is set to B'1' (on), the character is not to be printed (see "No Presentation" on page 107). When this flag bit is set to B'0' (off), the character is to be printed.

Bit 2 No Increment

When this flag bit is set to B'1' (on), the print position is not to be incremented (see "No Increment" on page 106). When this flag bit is set to B'0' (off), the print position is to be incremented.

Bits 3–7 Reserved

Note: An attempt to print a character whose invalid flag bit is on (B'1') will cause a printer data check unless that printer data check is suppressed by the print-server software.

9 CPI Repeating Group Length

The length of the CPI repeating group is dependent on the length of the code point as defined by the Encoding Scheme parameter in the CPD Structured Field. A single-byte code point would result in a CPI length of X'0A', while a double-byte code point would result in a length of X'0B'.

When GCGID-to-Unicode mappings are provided (values X'FE' or X'FF'), the length of each CPI repeating group can vary depending on how many Unicode scalar values are associated with each code point. Valid values for the CPI repeating group length include:

X'0A' Single-byte code point, repeating group length is X'0A' bytes

X'0B' Double-byte code point, repeating group length is X'0B' bytes

X'FE' Single-byte code point, repeating group length varies per code point depending on how many Unicode scalar values are listed

X'FF' Double-byte code point, repeating group length varies per code point depending on how many Unicode scalar values are listed

Note: Some FOCA products do not support the values X'FE' or X'FF'.

10–11 Space Character Section Number and Code Point

See "Space Character Section Number" on page 108, and "Space Character Code Point" on page 108.

If the code page represents a single-byte encoding scheme, the space character code point is indicated in the first byte and is repeated in the second byte. If the code page represents a double-byte encoding scheme, the space character section number is indicated in the first byte and the code point in the second byte.

Double-Byte Raster Coded Fonts

This value must be the same for all sections.

Although the space character parameters can be assigned any value, the typically used values are shown in the following table:

Basic Encoding Structure	Single Byte	Double Byte
IBM-PC Data	X'2020'	Not used
EBCDIC Presentation	X'4040'	X'4040'
UCS Presentation (latin)	Not applicable	X'0020'
UCS Presentation (ideographic, full-width space)	Not applicable	X'3000'

The character shape assigned to this code point is not printed. The increment for the space character can be changed by a PTOCA Set Variable Space Character Increment (SVI) control sequence. Refer to *Presentation Text Object Content Architecture Reference* for SVI information.

12 Code Page Use Flags

Bit 0 Sort Order

When this flag bit is set to B'1' (on), the CPI repeating groups are sorted in ascending code point order. When this flag bit is set to B'0' (off), the CPI repeating groups are sorted in ascending character ID order.

Bit 4 Variable Space Enable

This flag is used to enable variable spacing. When the flag is B'1', the PTOCA variable space increment is used whenever the code point for the variable space character is encountered. Some AFP products, such as DCF, use variable spacing for text justification and therefore require this flag to be B'1'.

When the flag is B'0', all PTOCA SVI control sequences are ignored and the space character is printed as defined in the accompanying font whenever the variable space code point is encountered. If a code page is built that has this flag as B'0', it should not be used with AFP products that require the variable space function.

Double-Byte Raster Coded Fonts

This flag bit must be set the same for all sections.

Optional Unicode mapping entry for the default character ID

To allow code pages that contain user-defined characters (that is, those characters that have not been registered with IBM and assigned a GCGID value) to be used with TrueType/OpenType fonts, the default character ID can be mapped to a Unicode scalar value. This function is selected by specifying either X'FE' (for a single-byte code page) or X'FF' (for a double-byte code page) in the CPI repeating group length field (CPC byte 9).

Note: When CPC byte 9 is X'0A' or X'0B', the Unicode scalar value field is not present. When CPC byte 9 is X'FE' or X'FF', the Unicode scalar value field can be present, but is not required.

+ 0–3 Unicode scalar value

A Unicode scalar value is any Unicode code point except high-surrogate and low-surrogate code points. In other words, the ranges of values from X'00000000' to X'0000D7FF' and from X'0000E000' to X'0010FFFF' inclusive. Any other value is an ill-formed Unicode value (as of Unicode Standard 4.1); in the future, values above X'0010FFFF' might be added to the valid Unicode range.

The Unicode scalar value and the GCGID should identify the same character and should be in all TrueType/OpenType fonts used with this code page.

CPD – D3A687 – Code Page Descriptor

The Code Page Descriptor (CPD) structured field describes the code page.

Length (2 bytes)	**X'D3A687'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (42 or 44 bytes)

The data for the CPD structured field is as follows:

Offset	Type	Name	Range	Meaning	M/O
0–31	CHAR	CPDesc		Code Page Description	M
32–33	UBIN	GCGIDLen	8	Graphic Character GID Length	M
34–37	UBIN	NumCdPts	1–65,535	Number of Coded Graphic Characters Assigned	M
38–39	UBIN	GCSGID	X'0000' X'0001'–X'FFFE' X'FFFF'	Graphic Character Set GID: No GCSGID is assigned GCSGID No GCSGID is assigned	M
40–41	UBIN	CPGID	X'0000' X'0001'–X'FFFE' X'FFFF'	Code Page Global Identifier No CPGID is assigned CPGID No CPGID is assigned	M
42–43	UBIN	EncScheme	See Below	Encoding Scheme	O

Bytes Description

0–31 Code Page Description

See “Code Page Description” on page 103.

The character string assigned to this field is intended to aid the end user, who might need to edit the code page, in identifying the set of characters represented by the code page. The name or title used should correspond to the name assigned by the IBM registration authority when the code page was registered. Though it may be any name or title that has meaning to the creator or editor of the code page. Unless otherwise specified, character string data is encoded as CPGID 500, GCSGID 103 (see “Parameter Types” on page 55).

32–33 Graphic Character GID Length

See “Graphic Character GID Length” on page 106.

This is the length of the IBM registered GCGID (AFP uses the eight-character identifier format), or a user-assigned GCGID.

34–37 Number of Coded Graphic Characters Assigned

See “Number of Coded Graphic Characters Assigned” on page 107.

The number of assigned code points in the code page that equals the number of Code Page Index (CPI) repeating groups.

38–39 Graphic Character Set Global Identifier

See “Graphic Character Set Global Identifier” on page 61.

This is the IBM registered GCSGID, or it may be a user defined GCSGID number from the reserved number space X'FF00' to X'FFFE'.

40–41 Code Page Global Identifier

See "Code Page Global Identifier" on page 103.

This is the IBM registered CPGID, or it may be a user defined CPGID number from the reserved number space X'FF00' to X'FFFE'.

42–43 Encoding Scheme

See "Encoding Scheme" on page 104.

This parameter identifies the code page as either EBCDIC-Presentation encoded, IBM-PC-Data (ASCII) encoded, or UCS-Presentation encoded. It also specifies the code points as either fixed single-byte values or fixed double-byte values. The following values are used:

X'0000'	No encoding scheme specified
X'0100'	Single-byte, encoding not specified
X'0200'	Double-byte, encoding not specified
X'2100'	Single-byte IBM-PC Data
X'6100'	Single-byte EBCDIC Presentation
X'6200'	Double-byte EBCDIC Presentation
X'8200'	Double-byte UCS Presentation

Note: Even though this field is optional, the encoding scheme should be specified to allow proper interpretation of the code page. This field must be specified for a UCS Presentation code page.

CPI – D38C87 – Code Page Index

In a series of repeating groups, the Code Page Index (CPI) associates character IDs with code points. Each repeating group specifies a character ID, a set of flags, and a code point. The repeating groups may be sorted in ascending order by character IDs or by code point, depending on the Sort Order flag in the Code Page Control structured field. The default sort order is by ascending character ID order. One repeating group is required, but as many as 65,536 repeating groups are allowed. The maximum number of repeating groups is determined by the maximum number of code points in the code page: 256 for single-byte code pages, 256 for double-byte code page sections, and 65,536 for double-byte code pages.

For processing efficiency, it is required that all code points in a single CPI correspond to the same double-byte section. That is, each double-byte section shall begin a new CPI structured field; the segmentation flag is not used. For performance and storage efficiency, it is required that double-byte code pages be sorted in ascending code point order. For migration consistency with the existing product base, it is required that single-byte code pages be sorted in ascending character ID order (this includes those code pages which were defined for use with double-byte raster fonts; each code page object represented one double-byte code page section).

Note: The same GCGID could be assigned to multiple code points, therefore, “ascending order” includes multiple GCGIDs of the same value but with different code point assignments. No specific ordering is required within such a group of equal GCGIDs.

Each code point not included in this structured field is associated with the default character specified in the Code Page Control (CPC) structured field.

Length (2 bytes)	**X'D38C87'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data

The data for the CPI structured field consists of a series of repeating group entries. Each repeating group is divided as follows, with the length of the group specified by the CPI Repeating Group Length parameter (byte 9) in the CPC structured field; the length is 10 bytes for a single-byte code page and 11 bytes for a double-byte code page.

When a GCGID-to-Unicode mapping is provided in the CPI (that is, when CPC byte 9 contains either X'FE' or X'FF'), the code point field is followed by a Unicode mapping entry consisting of a count parameter and one or more Unicode scalar values. To allow for combining characters, each code point in the code page can be mapped to a different number of Unicode scalar values.

Offset	Type	Name	Range	Meaning	M/O
0–7	CHAR	GCGID		Graphic Character Global ID	M
8	BITS	PrtFlags	See Below	Graphic Character Use Flags	M
9 or 9–10	UBIN	CodePoint	0–65,535	Code Point	M
Zero or one Unicode mapping-entry in the following format:					
+ 0	UBIN	Count	X'00' –X'FF'	Number of Unicode scalar values	O
Zero or more Unicode scalar-value entries (depending on count value) in the following format:					
++ 0–3	UBIN	Unicode scalar value	X'00000000' – X'FFFFFFFF' (excluding surrogates)	Unicode scalar value to be mapped to the GCGID value and code point value	O

Bytes **Description**

0–7 Graphic Character Global ID

See "Graphic Character Global Identifier" on page 95.

Identifies the character for the code point (byte 9) of this repeating group. The identifier must match bytes 0–7 of a repeating group in the Font Index (FNI) structured field, or an error is reported at print time.

This parameter is EBCDIC encoded using CPGID 500 and GCSGID 103.

Note: The letters in a character ID are *case-sensitive*. Therefore LA020000, La020000, lA020000, and la020000 denote four different characters. IBM font products use uppercase alphanumeric character IDs based on IBM Corporate Standards. While other naming schemes can be used, it is important to use a consistent naming scheme across all font objects in a system; therefore, the GCGID scheme used by IBM font products is recommended.

8 Graphic Character Use Flags

Bit 0 Invalid Coded Character

When this flag bit is set to B'1' (on), the character is an invalid coded character (see "Invalid Coded Character" on page 106). When this flag bit is set to B'0' (off), the character is a valid coded character.

Bit 1 No Presentation

When this flag bit is set to B'1' (on), the character not to be printed (see "No Presentation" on page 107). When this flag bit is set to B'0' (off), the character is to be printed.

Bit 2 No Increment

When this flag bit is set to B'1' (on), the print position is not to be incremented (see "No Increment" on page 106). When this flag bit is set to B'0' (off), the print position is to be incremented.

Bits 3–7 Reserved for future use

Note: An attempt to print a character whose invalid flag bit in on (B'1') will cause a printer data check unless that printer data check is suppressed by the print-server software..

9 or 9–10 Code Point

See "Code Point" on page 103.

The code point for the character. The number of bytes in the code point is determined by the Encoding Scheme parameter in the CPD structured field (the default code point length is 1). An exception condition exists if more than one repeating group contains the same code point, in which case, processing should be terminated (unexpected print results would occur).

Optional Unicode mapping entry

To allow code pages that contain user-defined characters (that is, those characters that have not been registered with IBM and assigned a GCGID value) to be used with TrueType/OpenType fonts, each code point can be mapped to one or more Unicode scalar values. This function is selected by specifying either X'FE' (for a single-byte code page) or X'FF' (for a double-byte code page) in the CPI repeating group length field (CPC byte 9). In this case, each repeating group entry must contain a count value in byte +0 that specifies the number of Unicode scalar values (bytes ++0–3) to follow. To allow for combining characters, each code point in the code page can be mapped to a different number of Unicode scalar values.

+ 0 Count

This parameter specifies the number of Unicode scalar values to follow. A value of zero indicates that there are no Unicode scalar values mapped to this code point.

++ 0–3 Unicode scalar value

A Unicode scalar value is any Unicode code point except high-surrogate and low-surrogate code points. In other words, the ranges of values from X'00000000' to X'0000D7FF' and from X'0000E000' to X'0010FFFF' inclusive. Any other value is an ill-formed Unicode value (as of Unicode Standard 4.1); in the future, values above X'0010FFFF' might be added to the valid Unicode range.

The Unicode scalar value and the GCGID should identify the same character and should be in all TrueType/OpenType fonts used with this code page.

Note: When multiple Unicode scalar values are specified for a code point, some presentation devices will present the multiple characters in a device-defined manner. When the device supports the combining character capability of Unicode, combining characters are presented correctly; but for simpler devices, only the first Unicode scalar value might be presented or the combining characters might be presented one after the other or all positioned at a single place (but not necessarily aligned correctly).

For example, suppose code point X'DC' (an EBCDIC ü character) is mapped to two Unicode scalar values: X'0075' (u) and X'0308' (combining diaeresis). Some devices will correctly present these two Unicode scalar values as “ü”; however, other devices present characters in a one-to-one fashion and might present the code point as “u¨”. In some cases, this problem can be avoided by mapping to a single, already-combined Unicode scalar value (if such a Unicode value is registered), such as X'00FC' (ü).

ECF – D3A98A – End Coded Font

The End Coded Font (ECF) structured field ends the coded font object.

Length (2 bytes)	**X'D3A98A'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (0 or 8 bytes)

The data for the ECF structured field is as follows:

Offset	Type	Name	Range	Meaning	M/O
0–7	CHAR	CFName		Coded Font Name	O

Bytes Description

0–7 Coded Font Name

The End Coded Font (ECF) structured field name, if supplied, must match the corresponding Begin Coded Font (BCF) structured field name. In an ECF structured field, no name or a null name (any name with X'FFFF' in the first two bytes) matches any name in the BCF structured field.

ECP – D3A987 – End Code Page

The End Code Page (ECP) structured field ends the code page object.

Length (2 bytes)	**X'D3A987'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (0 or 8 bytes)

The data for the ECP structured field is as follows:

Offset	Type	Name	Range	Meaning	M/O
0–7	CHAR	CPName		Code Page Name	O

Bytes Description

0–7 Code Page Name

The ECP structured field name, if supplied, must match the corresponding BCP structured field name. In an ECP structured field, no name or a null name (any name with X'FFFF' in the first two bytes) matches any name in the BCP structured field.

EFN – D3A989 – End Font

The End Font (EFN) structured field ends the font character set object.

Length (2 bytes)	**X'D3A989'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (0 or 8 bytes)

The data for the EFN structured field is as follows:

Offset	Type	Name	Range	Meaning	M/O
0–7	CHAR	CSName		Font Character Set Name	O

Bytes Description

0–7 Token Name

The EFN structured field name, if supplied, must match the corresponding BFN structured field name. In an EFN structured field, no name or a null name (any name with X'FFFF' in the first two bytes) matches any name in the BFN structured field.

FNC – D3A789 – Font Control

The Font Control (FNC) structured field provides defaults and information about the font character set.

Length (2 bytes)	**X'D3A789'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (22, 28, 32, 35, 42, or 46 bytes)

The data for the FNC structured field is as follows:

Offset	Type	Name	Range	Meaning	M/O
0	UBIN	Retired	X'01'	Retired Parameter	M
1	CODE	PatTech	X'05' X'1E' X'1F'	Pattern Technology Identifier: Laser Matrix N-bit Wide CID Keyed font (Type 0) PFB (Type 1)	M
2	UNDF	Reserved	X'00'	Reserved for future use	M
3	BITS	FntFlags	See Below	Font Use Flags	M
4	CODE	XUnitBase	X'00' X'02'	Fixed metrics, 10 inches Relative metrics	M
5	CODE	YUnitBase	X'00' X'02'	Fixed metrics, 10 inches Relative metrics	M
6–7	UBIN	XftUnits	X'0960' X'0BB8' X'03E8'	240 pels per inch 300 pels per inch 1000 units per em	M
8–9	UBIN	YftUnits	X'0960' X'0BB8' X'03E8'	240 pels per inch 300 pels per inch 1000 units per em	M
10–11	UBIN	MaxBoxWd	0–32,767	Maximum Character Box Width	M
12–13	UBIN	MaxBoxHt	0–32,767	Maximum Character Box Height	M
14	UBIN	FNORGLen	X'1A'	FNO Repeating Group Length	M
15	UBIN	FNIRGLen	X'0A' X'1C'	FNI Repeating Group Length	M
16	CODE	PatAlign	X'00' X'02' X'03'	Pattern Data Alignment Code: 1-Byte Alignment 4-Byte Alignment 8-Byte Alignment	M
17–19	UBIN	RPatDCnt	X'0' to X'7FFFFF'	Raster Pattern Data Count	M
20	UBIN	FNPRGLen	X'16'	FNP Repeating Group Length	M
21	UBIN	FNMRGLen	X'00' X'08'	FNM Repeating Group Length: No Raster Data Raster Data	M
22	CODE	ResXUBase	X'00'	Shape resolution, 10 inches	O
23	CODE	ResYUBase	X'00'	Shape resolution, 10 inches	O
24–25	UBIN	XfrUnits	X'0000' X'0960' X'0BB8'	No shape resolution provided 240 pels per inch 300 pels per inch	O

Offset	Type	Name	Range	Meaning	M/O
26–27	UBIN	YfrUnits	X'0000' X'0960' X'0BB8'	No shape resolution provided 240 pels per inch 300 pels per inch	O
28–31	UBIN	OPatDCnt	X'0A' to X'FFFFFFFF'	Outline Pattern Data Count	O
32–34	UNDF	Reserved	X'000000'	Reserved for future use	O
35	UBIN	FNNRGLen	X'0C'	FNN Repeating Group Length	O
36–39	UBIN	FNNDCnt	X'02' to X'FFFFFFFF'	FNN Data Count	O
40–41	UBIN	FNNMapCnt	0–65,535	FNN Name Count	O
42–*nn*		Triplets	See Below	Self-Defining Triplets	O

Bytes **Description**

0 Retired; must be X'01'.

1 Pattern Technology Identifier

See "Pattern Technology Identifier" on page 100.

The valid values for AFP printing are:

X'05' Laser Matrix N-bit Wide
X'1E' Composite Adobe Type 0
X'1F' Type 1 Font Printer File Binary (PFB)

2 Reserved

3 Use Flags

Bit 0 Identifies whether normal or Magnetic Ink Character Recognition (MICR) printing is to be done. See "MICR Font" on page 65.

B'0' Normal printing
B'1' MICR printing

Bit 1 Identifies whether this is a complete font resource or an extension to another font resource. See "Extension Font" on page 60.

B'0' Complete Font
B'1' Extension Font

Bits 2–3 B'00', (reserved)

Bit 4 (Retired)

This bit was used by some implementations to determine whether the baseline offset should be shifted by one pel or not. The original V1 unbounded box architecture counted pels instead of the spaces between pels, thus no font could have a zero base- line offset. If this bit is 0 (default), and the baseline offset is negative, most implementations will shift the baseline offset value up by one pel. If this bit is 1, some implementations will not alter the baseline offset value. This processing does not apply to fonts using relative units of measure.

Bit 5 B'0', (reserved)

Bit 6 Uniform Character Box Font

Shows whether the raster pattern is variable or uniform for all the characters in the font character set. See "Uniform Character Box Font" on page 72.

B'0' The raster pattern size varies. The individual character raster pattern box size is specified in bytes 10–13 of the Font Index (FNI) structured field repeating groups. The maximum width and height of the individual character raster pattern boxes is specified in bytes 10–13 of this structured field.

B'1' The raster pattern size is uniform for all the characters. The uniform raster pattern box size is specified in bytes 10–13 of this structured field.

Double-Byte Raster Coded Fonts

Sections X'41' to X'44' can have either uniform or variable size raster pattern boxes. Sections X'45' to X'FE' must have uniform raster pattern boxes (bit must be B'1').

Bit 7 B'0', (reserved)

4–9 Font-Metrics Units of Measure

See "Units of Measure" on page 34.

4 X Unit Base

X'00' Base is fixed at 10 inches
X'02' Base is relative

5 Y Unit Base

X'00' Base is fixed at 10 inches
X'02' Base is relative

Note: Bytes 4 and 5 must have the same value.

Double-Byte Raster Coded Fonts

Must be the same for all font character sets in the same coded font.

6–7 X Units per Unit Base

X'0960' (2400) 240-pel resolution, fixed base
X'0BB8' (3000) 300-pel resolution, fixed base
X'03E8' (1000) Relative base for Pattern Technology Identifiers X'1E' and X'1F'

8–9 Y Units per Unit Base

X'0960' (2400) 240-pel resolution, fixed base
X'0BB8' (3000) 300-pel resolution, fixed base
X'03E8' (1000) Relative base for Pattern Technology Identifiers X'1E' and X'1F'

Notes:

1. Bytes 6–7 and 8–9 must have the same value.
2. While 240-pel and 300-pel resolutions are the only fixed resolutions defined for AFP system fonts, some AFP products support additional resolutions such as 480 pel and 600 pel. In particular, many IPDS printers will accept raster fonts at any pel resolution and automatically convert them to the print-head resolution (support for "all resolutions in the range X'0001'-X'7FFF' " is indicated in the IPDS printer's OPC reply).

   Therefore, it is possible to use raster fonts built at resolutions other than 240 pel and 300 pel, but this can cause problems in some circumstances. For example, some printers do not support additional resolutions and some print-server software limits support to the two FOCA-specified resolutions. Also, print quality can suffer since printers must convert the

font resolution into the print-head resolution. A few IPDS printers are optimized for 240-pel and 300-pel resolutions, but not necessarily optimized for other values.

For example, a resolution of 240 pels-per-inch by 240 pels-per-inch is represented by 0,0,2400,2400 and a relative base is represented by 2,2,1000,1000.

If optional bytes 22–27 are not provided or contain binary zeroes, then for a fixed-metric font, bytes 4–9 specify both the metric resolution and the shape resolution. For a relative-metric font, no shape resolution is provided in this case. This could be a "metrics-only" font. See Table 15 on page 151 for the relationship of these bytes to bytes 22–27.

Note: Use the following formula to convert between fixed-metric values (in pels) and relative-metric values:

$$\text{fixed value} = \frac{\text{relative value} * \text{point size} * \text{fixed resolution}}{72 * \text{relative units per unit base}}$$

10–13 Maximum Character Box Size

A 0° character rotation is assumed.

If bit 6 of byte 3 is set to B'0' (variable character box size), these values are the maximum width and maximum height of all the raster pattern boxes in the font character set.

If bit 6 of byte 3 is set to B'1' (uniform character box size), these values are the uniform raster pattern box size.

Double-Byte Raster Coded Fonts

Must be the same for all font character sets in the high sections X'45' to X'FE' of the same coded font.

Outline fonts

An outline font will have an box width of zero and a box height of zero.

10–11 Maximum Character Box Width

See "Maximum Character Box Width" on page 80.

The raster pattern box width in pels, minus 1 (the first column of pels is column 0).

Note: Some server software refers to this value as x-extent in some messages.

12–13 Maximum Character Box Height

See "Maximum Character Box Height" on page 79.

The raster pattern box height in pels, minus 1 (the first row is numbered as row 0).

Note: Some server software refers to this value as y-extent in some messages.

Note: When the FNC pattern technology identifier contains X'1E' or X'1F' bytes 10–13 are set to X'00000000'.

14 FNO Repeating Group Length

This is a control parameter, used to manage the data structures. The value contained in this parameter defines the length of the repeating group used in the Font Orientation (FNO) structured field. The value is a constant, set to X'1A'.

15 FNI Repeating Group Length

This is a control parameter, used to manage the data structures. The value contained in this parameter defines the length of the repeating group used in the Font Index (FNI) structured field.

For raster technology fonts, this value must be X'1C'. For outline technology fonts, this value may be X'0A' or X'1C'.

16 Raster Pattern Data Alignment

See "Pattern Data Alignment Code" on page 98.

The boundary alignments for the raster patterns. The codes X'00', X'02', and X'03' indicate one-byte, four-byte, and eight-byte alignments, respectively. The choice is left to the font designer for a font containing raster patterns. To derive actual pattern data addresses, the pattern data address in bytes 4–7 in the Font Patterns Map (FNM) repeating groups must be multiplied by 1, 4, or 8, respectively.

Note: If the font contains no raster patterns, this byte must be set to X'00'.

17–19 Raster Pattern Data Count

See "Pattern Data Count" on page 99.

The total number of data bytes for all the Font Patterns (FNG) structured fields in this font character set, when the pattern technology identifier is set to X'05'. Must be set to X'000000' if the font does not contain raster patterns, or the pattern technology identifier is not X'05'.

20 FNP Repeating Group Length

This is a control parameter, used to manage the data structures. The value contained in this parameter defines the length of the repeating group used in the Font Position (FNP) structured field. The value is a constant, set to X'16'.

21 FNM Repeating Group Length This is a control parameter, used to manage the data structures. The value contained in this parameter defines the length of the repeating group used in the Font Position (FNP) structured field. The value is set to X'00' if the font contains outline font data in the FNGs. Otherwise it contains X'08'.

22–27 Shape Resolution Units of Measure

See "Units of Measure" on page 34.

This parameter is optional for raster fonts, but is required if the FNC structured field includes outline font descriptive information in bytes 28–41.

22 X Unit Base

X'00' Base is fixed at 10 inches

23 Y Unit Base

X'00' Base is fixed at 10 inches

24–25 X Units per Unit Base

X'0960' (2400) 240-pel resolution, fixed base
X'0BB8' (3000) 300-pel resolution, fixed base
X'0000' No value provided

26–27 Y Units per Unit Base

X'0960' (2400) 240-pel resolution, fixed base
X'0BB8' (3000) 300-pel resolution, fixed base
X'0000' No value provided

Note: Bytes 24–25 and 26–27 must have the same value.

For example, a resolution of 240 pels-per-inch by 240 pels-per-inch is represented by 0,0,2400,2400, a 300 pels-per-inch by 300 pels-per-inch is represented by 0,0,3000,3000, and an outline font is represented by 0,0,0000,0000.

28–31 Outline Pattern Data Count

See "Pattern Data Count" on page 99.

Length, in bytes, of the font pattern data when the pattern technology identifier is set to X'1E' or X'1F'. This parameter is used only when the pattern technology identifier is not equal to X'05'.

32–34 Reserved for future use; must be set to a constant X'000000'.

35 FNN Repeating Group Length

This is a control parameter, used to manage the data structures. The value contained in this parameter defines the length of the repeating group used in the Font Name Map (FNN) structured field. Used when the pattern technology identifier is not equal to X'05'.

36–39 FNN Data Count

This is a control parameter used to manage the data structures and specify the number of data bytes in the FNN structured fields. This field is included when the FNGs contain outline font data; when the pattern technology identifier is equal to X'1E', or X'1F'.

40–41 FNN IBM Name Count

This is a control parameter used to manage the data structures and specify the number of IBM character names (GCGIDs) mapped to outline font character names in the FNN structured fields. This field is included when the FNGs contain outline font data; when the pattern technology identifier is equal to X'1E', or X'1F'.

42–end Triplets

See "X'6D' – Extension Font Triplet" on page 185.

If this is an extension font character set (byte 3, bit 1 = B'1'), there must be one extension triplet. The extension triplet contains the GCSGID for the base font. If the triplet is provided, all positional fields must be included in the FNC structured field.

Table 15. Relationship Between FNC Unit of Measure and Font Resolution Fields

Bytes 4–5	Bytes 22–23	Bytes 24–25	Bytes 26–27	Bytes 6–9 Identify	Bytes 24–27 Identify
X'0000'	Not Present	Not Present	Not Present	Fixed metric and fixed raster shape base	Not Present
X'0000'	X'0000'	X'0000'	X'0000'	Fixed metric and fixed raster shape base	Not Used
X'0000'	X'0000'	X'0960' or X'0BB8'	X'0960' or X'0BB8'	Fixed metric and fixed raster shape base	Not Used
X'0202'	Not Present	Not Present	Not Present	Relative metric base	No raster shape data for this font
X'0202'	X'0000'	X'0000'	X'0000'	Relative metric base	No raster shape data for this font
X'0202'	X'0000'	X'0960' or X'0BB8'	X'0960' or X'0BB8'	Relative metric base	Fixed raster shape base

FND – D3A689 – Font Descriptor

The Font Descriptor (FND) specifies the overall characteristics of a font character set.

Length (2 bytes)	**X'D3A689'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (80 or 85–588 bytes)

The data for the FND structured field is as follows:

Offset	Type	Name	Range	Meaning	M/O
0–31	CHAR	TypeFcDesc		Typeface Description	M
32	CODE	FtWtClass	 X'01' X'02' X'03' X'04' X'05' X'06' X'07' X'08' X'09'	Weight Class Ultralight Extralight Light Semilight Medium (normal) Semibold Bold Extrabold Ultrabold	M
33	CODE	FtWdClass	 X'01' X'02' X'03' X'04' X'05' X'06' X'07' X'08' X'09'	Width Class Ultracondensed Extracondensed Condensed Semicondensed Medium (normal) Semiexpanded Expanded Extraexpanded Ultraexpanded	M
34–35	UBIN	MaxPtSize	 0–720 0–32,767	Maximum Vertical Size, in 10ths of a Point: Fixed Relative	M
36–37	UBIN	NomPtSize	 0–720 0–32,767	Nominal Vertical Size, in 10ths of a Point: Fixed Relative	M
38–39	UBIN	MinPtSize	 0–720 0–32,767	Minimum Vertical Size, in 10ths of a Point: Fixed Relative	M
40–41	UBIN	MaxHSize	0–X'7FFE'	Maximum Horizontal Size, in 20ths of a point (1440ths of an inch)	M
42–43	UBIN	NomHSize	0–X'7FFE'	Nominal Horizontal Size, in 20ths of a point (1440ths of an inch)	M
44–45	UBIN	MinHSize	0–X'7FFE'	Minimum Horizontal Size, in 20ths of a point (1440ths of an inch)	M
46	CODE	DsnGenCls	0–255	Design General Class (ISO)	M
47	CODE	DsnSubCls	0–255	Design Subclass (ISO)	M
48	CODE	DsnSpcGrp	0–255	Design Specific Group (ISO)	M
49–63	UBIN	Reserved	X'00...00'	Reserved for future use	M
64–65	BITS	FtDsFlags	See Below	Font Design Flags	M

Offset	Type	Name	Range	Meaning	M/O
66–75	UBIN	Reserved	X'00...00'	Reserved for future use	M
76–77	UBIN	GCSGID	X'0000' X'0001'–X'FFFE' X'FFFF'	Graphic Character Set GID: No information is given Registered GCSGID No information is given	M
78–79	UBIN	FGID	X'0000' X'0001'–X'FFFE' X'FFFF'	Font Typeface GID (FGID): No information is given Registered FGID No information is given	M
80–*nn*		Triplets	See Below	Self Defining Triplets	O

Bytes Description

0–31 Typeface Description

This parameter (which was formerly called “Typeface Name”) contains descriptive information including the Family Name and, when appropriate, the supported language complement. Examples include “Helvetica Cyrillic Greek”, “Times New Roman”, and “Letter Gothic Latin1”.

See “Family Name” on page 60.

This parameter is EBCDIC encoded using CPGID 500 and GCSGID 103.

32 Weight Class

See “Weight Class” on page 72.

33 Width Class

See “Width Class” on page 73.

34–35 Maximum Vertical Font Size

See “Maximum Vertical Font Size” on page 64.

For outline fonts, this parameter contains the maximum recommended font size expressed in tenths of a point. Otherwise it contains the actual raster font size, expressed in tenths of a point and is the same value as contained in bytes 36–37. The range of values permitted in this field are: 0 to 720 for fixed metrics and 0 to 32,767 for relative metrics. Originally, the range was 0 to 720 for relative metrics; some products use this smaller range (support for values larger than 720 for relative metrics is optional). A value of X'0000' indicates that no size is specified.

36–37 Nominal Vertical Font Size

See “Nominal Vertical Font Size” on page 68.

For outline fonts, this parameter contains the nominal presentation size, expressed in tenths of a point. Otherwise it contains the actual raster font size, expressed in tenths of a point. The range of values permitted in this field are 0 to 720 for fixed metrics and 0 to 32,767 for relative metrics. Originally, the range was 0 to 720 for relative metrics; some products use this smaller range (support for values larger than 720 for relative metrics is optional). A value of X'0000' indicates that no size is specified.

A point is a unit of measure. There are 12 points to a pica, and about 72 points to an inch.

38–39 Minimum Vertical Font Size

See “Minimum Vertical Font Size” on page 66.

For outline fonts, this parameter contains the minimum recommended font size, expressed in tenths of a point. Otherwise it contains the actual raster font size, expressed in tenths of a point and is the same value as contained in bytes 36–37. The range of values permitted in this field are: 0 to 720 for fixed metrics and 0 to 32,767 for relative metrics. Originally, the range was 0 to 720 for relative metrics;

some products use this smaller range (support for values larger than 720 for relative metrics is optional). A value of X'0000' indicates that no size is specified.

40–41 Maximum Horizontal Font Size

See "Maximum Horizontal Font Size" on page 63.

For outline fonts, this parameter contains the maximum recommended horizontal size, expressed in 20ths of a point (1440ths of an inch). Otherwise it contains the actual raster size, expressed in 20ths of a point (1440ths of an inch) and is the same value as contained in bytes 42–43. The range of values permitted in this field are: 0 to X'7FFE'.

42–43 Nominal Horizontal Font Size

See "Nominal Horizontal Font Size" on page 67.

For outline fonts, this parameter contains the nominal horizontal font size, expressed in 20ths of a point (1440ths of an inch). Otherwise it contains the actual raster size, expressed in 20ths of a point (1440ths of an inch).

For uniformly-spaced character sets, this is the value of the variable space increment converted to 20ths of a point (1440ths of an inch).

For mixed-pitch character sets, this is the value of the 12-pitch space converted to 20ths of a point (1440ths of an inch).

For proportionally spaced character sets, this is one-third of the point size, converted to 20ths of a point (1440ths of an inch).

The horizontal font size is included as the width portion of a GRID (global resource identifier).

The GRID is an eight-byte identifier in a MO:DCA Map Coded Font structured field used to identify a font by global identifiers rather than external file names. The GRID is defined in *Mixed Object Document Content Architecture Reference*. A grid contains the following two-byte fields in the order shown:

1. GCSGID (graphic character set global identifier) from the font code page
2. CPGID (code page global identifier) from the font code page
3. FGID (font global identifier)
4. Font width in 1/1440-inch units

44–45 Minimum Horizontal Font Size

See "Minimum Horizontal Font Size" on page 65.

For outline fonts, this parameter contains the minimum recommended horizontal size, expressed in 20ths of a point (1440ths of an inch). Otherwise it contains the actual raster size, expressed in 20ths of a point (1440ths of an inch) and is the same value as contained in bytes 42–43. The range of values permitted in this field are: 0 to X'7FFE'.

46–48 ISO Font Typeface Design Classification

See "Design General Class (ISO)" on page 58, "Design Subclass (ISO)" on page 59, and "Design Specific Group (ISO)" on page 59.

These three bytes represent the first, second, and third levels of a hierarchical typeface design grouping system, defined by the International Organization for Standardization (ISO), see ISO/IEC 9541-1, Annex A, "Typeface Design Grouping". Byte 46 is the least specific classification, while byte 48 is the most specific.

49–63 Reserved

64–65 Design Flags

Definitions of the terms in these flags are as follows:

Bit 0 B'0' Not italic
B'1' Italic
A type style with characters that slant. The slant is generally upward to the right. See "Italics" on page 62.

Bit 1 B'0' Not underscored
B'1' Underscored
A font character set in which the raster pattern for each graphic character includes an underscore as toned pels as well as the graphic character itself. See "Underscored Font" on page 72.

Bit 2 B'0'

Bit 3 B'0' Solid
B'1' Hollow
A type style in which the interior of the character is not toned. See "Hollow Font" on page 62.

Bit 4 B'0' Not overstruck
B'1' Overstruck
A type style in which one graphic character pattern (usually an overscore character) is merged with each graphic character pattern. The overstrike raster pattern is included as toned pels in the raster pattern for each graphic character. See "Overstruck Font" on page 68.

Bits 5–15 B'00000000000'

66–75 Reserved

76–77 Graphic Character Set Global Identifier (GCSGID)

See "Graphic Character Set Global Identifier" on page 61.

This is the GCSGID of the font character set. It might represent a set of characters that is either smaller or larger than the set of characters contained in the code page associated with this font character set in a coded font reference. When creating a coded font reference, the GCSGID that represents the smaller of the two sets of characters (font character set or code page character set) should be used.

78–79 Font Typeface Global Identifier (FGID)

See "Font Typeface Global Identifier" on page 61.

The FGID is included as part of the GRID.

80–end Triplets

"X'02' – Fully Qualified Name Triplet" on page 177.

There can be zero, one, or two Fully Qualified Name triplets specified in this field. Each Fully Qualified Name triplet must be of a different type; the valid types include:

X'07' Family name
X'08' Typeface name

FNG – D3EE89 – Font Patterns

The Font Patterns (FNG) structured field carries the character shape data (raster patterns or outline data) for a font character set. When there is more shape data than can fit in a single FNG structured field, a series of consecutive FNG structured fields are used to carry the character shape data; the data can be split across FNG structured fields at any byte boundary; the segmentation flag is not used.

The FNG structured field is omitted for fonts that do not contain shape information; these are called Metric Only Fonts (MOFs).

Length (2 bytes)	**X'D3EE89'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (1–32,759 bytes)

The data for the FNG structured field depends on the font technology.

For raster fonts, the data for the FNG structured field is as follows:

Offset	Type	Name	Range	Meaning	M/O
0–*n*	UNDF	PatData	See Below	Up to 32,759 bytes of pattern data	M

Bytes Description

0–*n* Pattern Data

See "Pattern Data" on page 98.

Notes:

1. The first pel of each character raster pattern must start on a one-byte, four-byte, or eight-byte boundary. See "Pattern Data Alignment Code" on page 98.
2. Multiple FNGs can be used to contain the shape data in a font. Each FNG can contain up to 32,759 bytes of data.

For outline fonts, the data for the FNG structured field is as follows:

Offset	Type	Name	Range	Meaning	M/O
0–*n*	UNDF	PatData	See Below	Up to 32,759 bytes of pattern data	M

Bytes Description

0–*n* Pattern Data

See "Pattern Data" on page 98.

Notes:

1. The outline font data is copied intact from the source outline font into the FNG.
2. Multiple FNGs are used to contain the shape data in a font. Each FNG can contain up to 32,759 bytes of data.
3. A raster pattern or outline font can cross into the next FNG.

When outline font data is included in the FNG the repeating group used in the FNG is as follows:

Offset	Type	Name	Range	Meaning	M/O
0–3	UBIN	OFLLen	X'0A' to X'FFFFFFFF'	Length of the outline font object	M
4–7	UBIN	Checksum	See Below	Checksum value for the technology-specific object data	M
8–9	UBIN	TIDLen	X'0002'– X'FFFF'	Technology-specific object identifier length	M
10–*n*	UBIN	TIDName	See Below	Technology-specific object identifier	O
n+1 to *n*+2	UBIN	ODescLen	X'0002'– X'FFFF'	Technology-specific object descriptor length	O
n+3 to *m*	UNDF	ObjDesc	See Below	Technology-specific object descriptor	O
m+1 to end		ObjData	See Below	Technology-specific object data	O

Bytes Description

0–3 Length of the Outline Font Object

The length of this repeating group, which contains an identifiable outline font object, including this field.

4–7 Checksum

Checksum for the technology-specific object data included in this repeating group.

The checksum applies only to the object data portion (bytes *m*+1 to end) of the technology-specific object. To calculate the checksum, all of the bytes of the object data are considered as a continuous sequence of bytes. The object data is then mapped to an array containing four unsigned bytes.

The first four bytes of object data are placed into the array as follows:

- The first byte of object data becomes byte 0 of the array (most significant byte).
- The second byte of object data becomes byte 1 of the array.
- The third byte of object data becomes byte 2 of the array.
- The fourth byte of object data becomes byte 3 of the array (least significant byte).

The remaining bytes of the object data are added on a byte by byte basis to the values contained in the array, ignoring all carry bits. The mapping of the remaining object data is done such that the fifth byte is added to the value in array position 0, the sixth byte to array position 1, the seventh byte to array position 2, the eighth byte to array position 3, the ninth byte to array position 0, and so forth, until all data has been processed. When all object data has been processed, the checksum is the unsigned, 32-bit integer created from the four-byte array.

Note: The following code fragment is shown to illustrate the algorithm:

```
uchar checksum_partial[4]={0,0,0,0};
short index=0;
ulong checksum;
uchar singlebyte;
while (1)
   {
    singlebyte=fgetc(pfb data file);
    if (end of pfb data file)  break;
    checksum_partial[index] = checksum_partial[index] + singlebyte;
    index = index + 1;
    if (index = 4) index = 0;
   }
checksum = *(ulong *)&checksum_partial[0];
```

8–9 Technology-specific object identifier length

Length of the technology-specific object identifier plus the length of this field.

10–*n* Technology-specific Object Identifier

Character string name of the object, as defined by the technology owner.

***n*+1 to *n*+2** Technology-specific Object Descriptor Length

Length of the Technology-specific Object Descriptor plus the length of this field. The existence of this field (and the associated descriptor) is dependent on the outline font Pattern Technology Identifier of the FNC Structured Field.

If the Pattern Technology Identifier is X'1F' (outline font, Type 1 PFB), this field and the associated descriptor data is omitted. If the Pattern Technology Identifier is X'1E' (outline font, CID Indexed), this field and the associated descriptor data is as described.

Note: If additional outline font technologies are supported in the future, it is expected that these descriptor length and descriptor data fields will be used to describe the FNG data objects.

n*+3 to *m Technology-specific Object Descriptor: Description of the Technology-specific Object. The content of this field is as described below:

Byte *n*+3: Object Type
- 0 = no information or N/A
- 1 = CMap File
- 2–4 = (reserved)
- 5 = CID file
- 6 = PFB file
- 7 = AFM file
- 8 = Filename Map File (eg. ATMDATA.DAT)
- 9–255 = (reserved)

See "Object Type" on page 97, and "Linkage Code" on page 97.

If the Object Type is 0, 2–4, or 9–255, the Technology-specific Object Descriptor Length is determined by the generator. Any data that occurs in the remainder of this field is not defined by this architecture and should be ignored.

If the Object Type is 1 (CMAP), the Technology-specific Object Descriptor Length is ten (two byte-length, one object-type, and seven descriptor bytes) and shall be as described below:

Byte *n*+4 Precedence Code: 0 = Primary, 1 = Auxiliary

Note: Auxiliary objects are ignored, unless referenced within another technology-specific object.

Byte *n*+5 Linkage Code: 0 = Linked, 1 = Unlinked

Byte *n*+6: Writing Direction Code:
- 0 = no information or N/A
- 1 = horizontal
- 2 = vertical
- 3 = both vertical and horizontal
- 4–255 = (reserved)

Byte *n*+7 to *n*+8: IBM GCSGID of the CMAP

Byte *n*+9 to *n*+10: IBM CPGID of the CMAP

If the Object Type is 5 (CID), the Technology-specific Object Descriptor Length is eight (two byte-length, one object-type, and five descriptor bytes) and shall be as described below:

Byte *n*+4 Precedence Code: 0 = Primary, 1 = Auxiliary

Note: Auxiliary objects are ignored, unless referenced within another technology-specific object.

Byte *n*+5 to Byte *n*+6: Maximum V(y) value for all characters in the CID font

Byte *n*+7 to Byte *n*+8: Maximum W(y) value for all characters in the CID font

If the Object Type is 6 (PFB), 7 (AFM), or 8 (File Name Map), the Technology-specific Object Descriptor Length is three (two byte-length and one object-type bytes), and no other descriptor information is provided.

***m*+1 to end** Technology-specific object data

The data associated with this object, in the format defined for the outline font technology vendor.

FNI – D38C89 – Font Index

The MO:DCA Structured Field Introducer (SFI) segmentation flag is used for DBCS outline font FNI structured fields. This allows more than one FNI per rotation. DBCS outline fonts can have several thousand characters, so the implementation of SFI segmentation is imperative for DBCS outline fonts.

For each character in a raster font, the Font Index (FNI) describes specific characteristics and points to an entry in the Font Patterns Map (FNM) structured field.

For outline fonts, there is no Font Patterns Map (FNM) structured field, so the Font Index (FNI) does not include the FNM Index. The FNM Index should be set to zero and ignored.

A font character set can have one, two, three, or four FNI structured fields, one for each rotation value. The FNI to be used for a given rotation is specified in the appropriate repeating group of the character Font Orientation (FNO) structured field.

For outline font technology X'1E' (CID Keyed), if the repeating group for a given Graphic Character GID is not included in the FNI structured field, the “Nominal Character Increment” in the FNO structured field should be used as the character increment.

Note: For storage and processing efficiency, if a large number of characters in the font character set share the same character increment, the “Nominal Character Increment” in the FNO structured field may be set to that nominal value and the repeating group for all characters having that same nominal value may be omitted from the FNI structured field.

Length (2 bytes)	**X'D38C89'** (3 bytes)	X'00', X'20'	X'0000' (2 bytes)	Structured Field Data ([10 or 28]*[number of characters] bytes)

Note: There is one FNI repeating-group entry for each character in a font. When the font contains a larger number of characters than can fit in a single 32K-byte long FNI structured field (for example, a DBCS font), the FNI data can be carried in a series of consecutive, segmented FNI structured fields. In this case, each FNI structured field except for the last FNI in the series has the segmentation-indicator flag (bit 2) set to B'1'.

An FNI consists of one or more repeating groups, sorted into ascending order by the character IDs (bytes 0–7). The length of each repeating group is specified in byte 15 of the FNC structured field. A length of 28 is used for all raster fonts; outline fonts can use a length of 10 (when the optional parameters are not specified) or a length of 28. Each repeating group is divided as follows:

Offset	Type	Name	Range	Meaning	M/O
0–7	CHAR	GCGID	See Below	Graphic Character Global Identifier	M
8–9	UBIN	CharInc	 0–255 0–32,767	Character Increment Fixed Relative	M
10–11	SBIN	AscendHt	 ± 256 ± 32,767	Ascender Height Fixed Relative	O
12–13	SBIN	DescendDp	 ± 256 ± 32,767	Descender Depth Fixed Relative	O
14–15	UBIN	Reserved	X'0000'	Reserved for future use	O
16–17	UBIN	FNMCnt	See Below	FNM Index	O

Offset	Type	Name	Range	Meaning	M/O
18–19	SBIN	ASpace	± 256 ± 32,767	A-Space Fixed Relative	O
20–21	UBIN	BSpace	0–256 0–32,767	B-space Fixed Relative	O
22–23	SBIN	CSpace	± 256 ± 32,767	C-Space Fixed Relative	O
24–25	UBIN	Reserved	X'0000'	Reserved for future use	O
26–27	SBIN	BaseOset	± 256 ± 32,767	Baseline Offset Fixed Relative	O

Bytes Description

0–7 Graphic Character Global Identifier

See "Graphic Character Global Identifier" on page 95.

This identifier is used in bytes 0–7 of one or more repeating groups in the Code Page Index (CPI) structured field to assign a code point to the character.

An error exists if the same character ID occurs in more than one repeating group in a Font Index (FNI).

This parameter is EBCDIC encoded using CPGID 500 and GCSGID 103.

Note: The letters in a character ID are *case-sensitive*. Therefore LA020000, La020000, lA020000, and la020000 denote four different characters. IBM font products use uppercase alphanumeric character IDs based on IBM Corporate Standards. While other naming schemes can be used, it is important to use a consistent naming scheme across all font objects in a system; therefore, the GCGID scheme used by IBM font products is recommended.

8–9 Character Increment

See "Character Increment" on page 94.

The number of pels or relative units by which the current print position changes in the inline (print) direction when the character is printed. This value generally equals the sum of the A-space, B-space, and C-space of the character. The value X'0000' indicates no change.

This value is used only if bit 7 of byte 12 of the Font Orientation (FNO) structured field is B'0' (maximum increment) and the character is one of the following:

- The default character and bit 2 of byte 8 of the Code Page Control (CPC) structured field is set to B'0' (the current print position is incremented by the default character increment).
- Any other character and bit 2 of byte 8 of the repeating group for the character in the Code Page Index (CPI) structured field is set to B'0' (the current print position is incremented by this character increment).

If bit 7 of byte 12 of the Font Orientation (FNO) structured field is B'1' (uniform increment), this parameter should have the same value as bytes 6 and 7 of that structured field.

Note: This value is used in positioning the character. For outline fonts, the FNI character increment need only be specified if the character increment is different from the "Nominal Character Increment" in the FNO structured field.

10–11 Character Ascender

See "Ascender Height" on page 91.

The number of pels or relative units between the topmost toned pel and the character baseline. This value is negative for characters having their topmost toned pel below the character baseline.

For positive values, the topmost pel is included in the count. For negative fixed-metric values, the topmost pel is also included in the count; zero is not a valid value. For negative relative-metric values, the topmost pel is not included in the count; zero means the top of the topmost pel is immediately below the baseline.

Note: This measurement is not the height of an ascender in the typographic terms, that is, the portion of certain lowercase characters that rises above the main body.

12–13 Character Descender

See "Descender Depth" on page 95.

The number of pels or relative units between the bottom of the bottommost toned pel and the character baseline. This value is negative for characters having the bottom of their bottommost toned pel above the character baseline.

For positive values, the bottommost pel is included in the count. For negative fixed-metric values, the bottommost pel is also included in the count; zero is not a valid value. For negative relative-metric values, the bottommost pel is not included in the count; zero means the bottom of the bottommost pel is immediately above the baseline.

Note: This measurement is not the depth of a descender in typographic terms, that is, the portion of certain characters that extends below the main body.

14–15 Reserved

A constant value of X'0000' that is reserved for future use.

16–17 FNM Index

This is a control parameter for managing the structured field content. The value represents the index number of the repeating group in the Font Patterns Map (FNM) structured field that describes this character. The index is based on 0 and is from 0 to the number of FNM structured field repeating groups minus 1.

Note: This field does not have meaning when the FNGs have no pattern data (for example, Metrics Only Fonts), or if the font contains outline font technology (X'1E' or X'1F') data.

18–19 A-Space

See "A-space" on page 91.

The number of pels or relative units measured in the inline (print) direction, from the current print position to the near edge of the toned pel box (that is, up to but not including the first toned pel of the character).

A negative A-space means that some part of the toned pel box extends before the current print position when this character is printed (left kerning).

20–21 B-Space

See "B-space" on page 92.

The measurement of toned pels (inclusive) taken in the inline (print) direction of a given character (that is, the width for 0 and 180 degrees rotation or the height for 90 and 270 degrees rotation in pels or relative units of the toned pel box.

22–23 C-Space

See "C-space" on page 94.

The number of pels or relative units measured in the inline (print) direction from the far edge of the toned pel box (that is, from but not including the toned pel furthest from the character reference point) to the character escapement point.

A negative C-space value means that the next print position overlaps some part of the toned pel box when the character is printed (right kerning).

24–25 Reserved

A constant value of X'0000' that is reserved for future use.

26–27 Baseline Offset

See "Baseline Offset" on page 92.

The number of pels or relative units between the top of the raster-pattern box and the character baseline.

For positive values, the topmost pel or relative unit is included in the count. For negative fixed-metric values, the topmost pel is also included in the count; zero means that the presentation services software sets the character's "not print" flag on. For negative relative metric values, the relative units representing the topmost pel are not included in the count; zero means the relative unit representing the top of the topmost pel is immediately below the baseline.

Notes:

1. This value is used in positioning the character.
2. The presentation services software increments negative fixed metric values by +1 as it builds the IPDS Load Font Index command.

FNM – D3A289 – Font Patterns Map

The Font Patterns Map (FNM) structured field describes some characteristics of the font character patterns. Each pattern is described in a separate repeating group. This structured field is omitted if the font does not contain raster patterns.

Length (2 bytes)	**X'D3A289'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (8*[number of patterns] bytes)

The repeating groups are sorted by the Pattern Data Address field and must be in the same order as the characters in the Font Patterns (FNG) structured field. The FNM can contain up to 1000 repeating groups. The length of each repeating group (8 bytes) is specified in byte 21 of the FNC structured field. Each repeating group is divided as follows:

Offset	Type	Name	Range	Meaning	M/O
0–1	UBIN	CharBoxWd	 0–255 0–32,767	Character Box Width Fixed Relative	M
2–3	UBIN	CharBoxHt	 0–255 0–32,767	Character Box Height Fixed Relative	M
4–7	UBIN	PatDOset	X'00' to X'FFFFFFFF'	Pattern Data Offset	M

Bytes Description

0–1 Character Box Width

See “Character Box Width” on page 93.

The width of the raster pattern box in pels, minus 1 (the first column is specified as 0; a 0° character rotation is assumed.

Note: Some server software refers to this value as x-extent in some messages. The term x-extent has a different meaning for characters than it does for other resources, such as page overlays.

2–3 Character Box Height

See “Character Box Height” on page 93.

The height of the raster pattern box in pels, minus 1 (the first row is specified as 0; a 0° character rotation is assumed.

Note: Some server software refers to this value as y-extent in some messages. The term y-extent has a different meaning for characters than it does for other resources, such as page overlays. See default baseline increment in the Font Orientation (FNO) structured field.

4–7 Pattern Data Offset

See “Pattern Data Offset” on page 100.

The start of the raster pattern for this character. This value multiplied by 1, 4, or 8 according to byte 16 (pattern data address alignment) in the Font Control (FNC) structured field is the offset, in bytes, to the first pel of the character raster pattern contained in the collection of character patterns from all of the Font Patterns (FNG) structured fields.

Note: In an AFP environment, the following rules apply:

- Pattern data offsets must be in ascending order.
- Each offset must be greater than the previous offset.
- Raster patterns must not overlap.

FNN – D3AB89 – Font Name Map

The Font Name Map is used to map IBM character names to the character names in outline fonts.

Length (2 bytes)	**X'D3AB89'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (2 or 16–32,759 bytes)

The data for the FNN structured field is divided into three sections. The first section identifies the encoding scheme used in the the second and third sections FNN fields. Multiple physical records might be required to contain all the data in a logical FNN.

Offset	Type	Name	Range	Meaning	M/O
0	CODE	IBM format	X'02'	IBM character ID format: EBCDIC GCGID	M
1	CODE	Technology format	X'03' X'05'	Technology-specific character ID format: Font-specific ASCII character name, used with Type 1 PFB fonts CMAP binary code point, used with CID-keyed fonts	M

Bytes Description

0 IBM format

See "Graphic Character Identifier Type" on page 105.

This parameter defines the type of graphic character identifiers used in the first of the two repeating groups of this structured field. Most often, these will be the IBM GCGIDs which occur in code page objects.

1 Technology-specific format

See "Graphic Character Identifier Type" on page 105.

This parameter defines the type of graphic character identifiers used in the second of the two repeating groups of this structured field. Most often, these will be the character identifiers which occur in the font character set.

The second section of the structured field is made of repeating groups containing IBM GCGIDs and indices to the outline font technology character names. The third section of the structured field contains the outline font technology character names and the length of each mapped name. The Font Control structured field gives the number (bytes 40–41) of IBM GCGIDs included in this structured field. The IBM GCGIDs are sorted in ascending order. Duplicate GCGIDs are not allowed. The Font Control structured field gives the repeating group length (byte 35) for the FNN repeating group. The repeating group, containing zero or more entries, has the following format:

Offset	Type	Name	Range	Meaning	M/O
0–7	UNDF	GCGID	See Below	Graphic Character Global ID	O
8–11	UBIN	TSOffset	X'0' to X'FFFFFFFF'	Technology-specific identifier offset	O

Bytes Description

0–7 Graphic Character Identifier

See "Graphic Character Global Identifier" on page 95.

This parameter defines the graphic character identifier; most often, this will be the IBM GCGID.

This parameter is EBCDIC encoded using CPGID 500 and GCSGID 103.

Note: The letters in a character ID are *case-sensitive*. Therefore LA020000, La020000, lA020000, and la020000 denote four different characters. IBM font products use uppercase alphanumeric character IDs based on IBM Corporate Standards. While other naming schemes can be used, it is important to use a consistent naming scheme across all font objects in a system; therefore, the GCGID scheme used by IBM font products is recommended.

8–11 Technology-specific identifier offset

This is a control parameter that manages the structured field content. This parameter defines the offset into the second of the two repeating groups of this structured field where the Technology-specific identifier will be found. It contains a count of the number of bytes from the first data byte of the first FNN structured field to the technology-specific identifier which maps to the GCGID named in this repeating group.

The third FNN section is variable length. The length of each member of the third section is defined in the first byte of each member of the data area. The data area format for outline fonts with pattern technology identifier X'1E' or X'1F' is:

Offset	Type	Name	Range	Meaning	M/O
0	UBIN	TSIDLen	2–128	Technology-specific identifier length	O
1–*n*	UNDF	TSID	See Below	Technology-specific identifier	O

Bytes Description

0 Technology-specific identifier length

This is a control parameter that manages the structured field content. This parameter defines the length of the character identifier, plus the length of this field; from 2 to 128 bytes.

1–*n* Technology-specific identifier

This parameter defines the Technology-specific identifier; the type of identifier is determined by byte 1 of the first section of this structured field.

FNN example

Table 16. FNN Example

Offset	Meaning	Value	Comments
0	Graphic Character Identifier Type	X'02'	The second section will be represented in EBCDIC
1	Graphic Character Identifier Type	X'03'	The third section will be represented in Adobe ASCII
End first section			
Start second section			
2–9	GCGID	LA010000	GCGID for lowercase 'a'
10–13	Offset to technology-specific identifier corresponding to LA010000	X'0000005F'	95
14–21	GCGID	LA020000	GCGID for uppercase 'A'
22–25	Offset to technology-specific identifier corresponding to LA020000	X'00000061'	97

Table 16 FNN Example (cont'd.)

Offset	Meaning	Value	Comments
26–33	GCGID	LB010000	GCGID for lowercase 'b'
34–37	Offset to technology-specific identifier corresponding to LB010000	X'0000005D'	93
38–45	GCGID	LB020000	GCGID for uppercase 'B'
46–49	Offset to technology-specific identifier corresponding to LB020000	X'00000069'	105
50–57	GCGID	SP010000	GCGID for space character
58–61	Offset to technology-specific identifier corresponding to SP010000	X'00000063'	99
62–69	GCGID	ZX020000	GCGID for imaginary character
70–73	Offset to technology-specific identifier corresponding to ZX020000	X'0000004A'	74
End second section			
Start third section			
74	Technology-specific identifier length	X'13' (19)	Section three items are not required to have a specific order
75–92	Technology-specific identifier	Imaginary Character	
93	Technology-specific identifier length	2	Section three items are not required to have a specific order
94	Technology-specific identifier	b	
95	Technology-specific identifier length	2	Section three items are not required to have a specific order
96	Technology-specific identifier	a	
97	Technology-specific identifier length	2	Section three items are not required to have a specific order
98	Technology-specific identifier	A	
99	Technology-specific identifier length	6	Section three items are not required to have a specific order
100–104	Technology-specific identifier	space	
105	Technology-specific identifier length	2	Section three items are not required to have a specific order
106	Technology-specific identifier	B	
End third section			

FNO – D3AE89 – Font Orientation

Each repeating group in the Font Orientation (FNO) structured field applies to *one* character rotation of a font character set. There can be one, two, three, or four repeating groups.

Length (2 bytes)	**X'D3AE89'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (26*[number of repeating groups] bytes)

The length of each repeating group (26 bytes) is specified in byte 14 of the FNC structured field. Each repeating group is divided as follows:

Offset	Type	Name	Range	Meaning	M/O
0–1	UBIN	Reserved	X'0000'	Reserved for future use	M
2–3	UBIN	CharRot	X'0000' X'2D00' X'5A00' X'8700'	Character Rotation: 0 degrees 90 degrees 180 degrees 270 degrees	M
4–5	SBIN	MaxBOset	± 256 ± 32,767	Maximum Baseline Offset Fixed Relative	M
6–7	UBIN	MaxCharInc	0–255 0–32,767	Maximum Character Increment Fixed Relative	M
8–9	UBIN	SpCharInc	0–255 0–32,767	Space Character Increment Fixed Relative	M
10–11	UBIN	MaxBExt	0–255 0–32,767	Maximum Baseline Extent Fixed Relative	M
12	BITS	OrntFlgs	See Below	Orientation Control Flags	M
13	UBIN	Reserved	X'00'	Reserved for future use	M
14–15	UBIN	EmSpInc	0–255 0–32,767	Em-Space Increment Fixed Relative	M
16–17	UBIN	Reserved	X'0000'	Reserved for future use	M
18–19	UBIN	FigSpInc	0–255 0–32,767	Figure Space Increment Fixed Relative	M
20–21	UBIN	NomCharInc	0–32,767	Nominal Character Increment Relative	M
22–23	UBIN	DefBInc	0–65,535	Default Baseline Increment	M
24–25	SBIN	MinASp	± 255 ± 32,767	Minimum A-Space Fixed Relative	M

Bytes Description

0–1 Reserved

Constant X'0000'; this field is reserved for future use.

2–3 Character Rotation

See "Character Rotation" on page 58.

Identifies the character rotation applicable to this repeating group of the FNO structured field.

4–5 Maximum Baseline Offset

See "Maximum Baseline Offset" on page 79.

If bit 6 of byte 12 (flags) is set to B'1' (uniform), this parameter is the uniform baseline offset in pels or relative units for all the characters in this rotation. This means that all the characters have the same baseline offset.

If bit 6 of byte 12 (flags) is set to B'0' (maximum), this parameter is the maximum baseline offset in pels or relative units for all characters in this rotation. The baseline offset for each character is specified in bytes 26 and 27 of the Font Index (FNI) structured field repeating group.

Double-Byte Raster Coded Fonts

Must be the same for all font character sets in the high sections X'45' to X'FE' of the same coded font.

6–7 Maximum Character Increment

See "Maximum Character Increment" on page 80.

If bit 7 of byte 12 (flags) is set to B'1' (uniform), this parameter is the uniform character increment in pels or relative units for all the characters in this rotation. This means that all the characters have the same character increment.

If bit 7 of byte 12 (flags) is set to B'0' (maximum), this parameter is the maximum character increment in pels or relative units for all characters in this rotation. The character increment for each character is specified in bytes 8 and 9 of the Font Index (FNI) structured field repeating group.

Double-Byte Raster Coded Fonts

Must be the same for all font character sets in the high sections X'45' to X'FE' of the same coded font.

8–9 Space Character Increment

See "Space Character Increment" on page 85.

The increment, in pels or relative units, to be used when a space character code point appears in the text and no Set Variable Space Character Increment (SVI) has preceded the code point. The value is usually equal to the width of the space character (character ID SP010000). The value is also called word space.

10–11 Maximum Baseline Extent

See "Maximum Baseline Extent" on page 78.

This measurement, in pels or relative units, is equal to the maximum baseline offset for any character in the font character set plus the distance from the baseline to the bottom edge of the character box for any character in the font character set. The value must be equal to or greater than the maximum baseline offset, that is, negative values are ignored.

Note: This value is used to determine if the characters are positioned off the page.

For uniform character boxes (FNC flag bit 6 on), the maximum baseline extent value is equivalent to either the height or width of the character box (before being extended to meet byte boundary requirements) depending on the character rotation.

Double-Byte Raster Coded Fonts

Must be the same for all font character sets in the high sections X'45' to X'FE' of the same coded font.

12 Control Flags

Bits 0–2 Font Index Number

The value is used to select the Font Index (FNI) structured field for the repeating group for this character rotation.

The value B'000' selects the first FNI, B'001' the second FNI, and so on.

Bit 3 B'0'; reserved

Bit 4 Kerning (see "Kerning Pair Data" on page 62)

- **B'0'** No kerning data
- **B'1'** Kerning data

Bit 5 Uniform A-space (see "Uniform A-space" on page 89)

Shows whether the value in bytes 24 and 25 is a minimum or uniform A-space for this character rotation:

- **B'0'** Minimum A-space
- **B'1'** Uniform A-space

Double-Byte Raster Coded Fonts

Must be set to B'1' for sections X'45' to X'FE'.

Bit 6 Uniform Baseline Offset (see "Uniform Baseline Offset" on page 89)

Shows whether the value in bytes 4 and 5 is a maximum or uniform baseline offset for this character rotation:

- **B'0'** Maximum baseline offset
- **B'1'** Uniform baseline offset

Double-Byte Raster Coded Fonts

Must be set to B'1' for sections X'45' to X'FE'.

Bit 7 Uniform Character Increment (see "Uniform Character Increment" on page 90)

Shows whether the value in bytes 6 and 7 is a maximum or uniform character increment for this character rotation:

- **B'0'** Maximum character increment
- **B'1'** Uniform character increment

Double-Byte Raster Coded Fonts

Must be set to B'1' for sections X'45' to X'FE'.

13 Reserved

14–15 Em Space Increment

See "Em-Space Increment" on page 59.

The character increment, in pels or relative units, of an em space. An em is the square whose sides are formed by the value of the point size; for example, 9 points at 240-pel by 240-pel resolution is 30 pels, a 9-point em at that resolution is 30 pels wide and 30 pels high.

16–17 Reserved

18–19 Figure Space

See "Figure Space Increment" on page 75.

The character increment, in pels or relative units, for numerals 0 through 9. A figure space is usually equal to the width of the numeric space character (character ID SP310000).

20–21 Nominal Character Increment

See "Nominal Character Increment" on page 84 for a description of this parameter and how it can be used efficiently.

This parameter is only applicable to outline font technology X'1E'. Its value should be ignored for all other technologies. This value should be set to either the uniform character increment or the nominal character increment to be used if no FNI character increment value is provided.

22–23 Default Baseline Increment

See "Default Baseline Increment" on page 74.

The default increment, in pels or relative units, between character baselines. A text formatter can use this value when a line spacing value is not specified. For example, the following conversion from lines-per-inch to pels can be helpful:

Table 17. Sample Lines-Per-Inch to Pel Conversions

Lines-per-inch	240 Pels	300 Pels
12	20	25
10	24	30
8	30	37.5
6	40	50

24–25 Minimum A-space

See "Minimum A-space" on page 83.

If bit 5 of byte 12 (flags) is set to B'1' (uniform), this parameter is the uniform A-space in pels or relative units to be used instead of the individual character A-space to position raster pattern boxes. This value can be positive or zero, but not negative.

If bit 5 of byte 12 (flags) is set to B'0' (minimum), this value, in pels or relative units, is the largest left kern for all the characters in this rotation.

Double-Byte Raster Coded Fonts

Must be the same for all font character sets in the high sections X'45' to X'FE' of the same coded font.

FNP – D3AC89 – Font Position

The Font Position (FNP) structured field describes the common characteristics of all the characters in a font character set. Four repeating groups, one for each character rotation, are allowed. The order of these groups must correspond to the order of the repeating groups in the Font Orientation (FNO) structured field, each of which specifies a separate character rotation.

Length (2 bytes)	**X'D3AC89'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (22*[number of repeating groups] bytes)

The length of each repeating group (22 bytes) is specified in byte 20 of the FNC structured field. Each repeating group is divided as follows:

Offset	Type	Name	Range	Meaning	M/O
0–1	UBIN	Reserved	X'0000'	Reserved for future use	M
2–3	SBIN	LcHeight	 ± 256 ± 32,767	Lowercase Height Fixed Relative	M
4–5	SBIN	CapMHt	 ± 256 ± 32,767	Cap-M Height Fixed Relative	M
6–7	SBIN	MaxAscHt	 ± 256 ± 32,767	Maximum Ascender Height Fixed Relative	M
8–9	SBIN	MaxDesDp	 ± 256 ± 32,767	Maximum Descender Depth Fixed Relative	M
10–14	UBIN	Reserved	X'00...00'	Reserved for future use	M
15	UBIN	Retired	X'01'	Retired Parameter	M
16	UBIN	Reserved	X'00'	Reserved for future use	M
17–18	UBIN	UscoreWd	 0–255 0–32,767	Underscore Width (units) Fixed Relative	M
19	UBIN	UscoreWdf	X'00'	Underscore Width (fraction)	M
20–21	SBIN	UscorePos	 ± 256 ± 32,767	Underscore Position Fixed Relative	M

Bytes	Description
0–1	X'0000'; Reserved for future use
2–3	Lowercase Height See "X-Height" on page 74. The Lowercase Height is the nominal height above the baseline (ignoring the ascender) of the lowercase characters in a font and is usually the height of the lowercase letter *x*. The value of lowercase height is specified by a font designer. **Notes:** 1. This value is negative if all the characters are totally below the character baseline. 2. For most fonts the lowercase height value for the 0° rotation is used for all rotations. 3. The value X'0000' can mean that this parameter is unspecified.
4–5	Cap-M Height See "Cap-M Height" on page 57. The Cap-M Height is the nominal height above the baseline for uppercase character shapes and is usually equal to the height of the uppercase letter *M*. The cap-M height value is specified by a font designer. **Notes:** 1. This value is negative if all the characters are totally below the character baseline. 2. For most fonts the cap-M height value for the 0° rotation is used for all rotations. 3. The value X'0000' can mean that this parameter is unspecified.
6–7	Maximum Ascender See "Maximum Ascender Height" on page 78. The maximum, in pels or relative units, of the character-ascender values for this character rotation. The character-ascender values are found in the corresponding FNI. **Note:** This value is zero or negative if all the characters are totally below the character baseline.
8–9	Maximum Descender See "Maximum Descender Depth" on page 81. The maximum, in pels or relative units, of the character-descender values for this character rotation. The character-descender values are found in the corresponding FNI. **Note:** This value is zero or negative if all the characters are totally above the character baseline.
10–14	X'0000000000'; Reserved for future use
15	X'01'; Constant
16	X'00'; Reserved for future use

17–19 Underscore Width

See “Underscore Width” on page 89.

Recommended thickness, in pels or relative units, of the stroke used by formatters to underscore characters in text if FND byte 64 bit 1 is B'0' (characters are not underscored). Bytes 17 and 18 represent an integer; byte 19 is set to X'00'.

If the integer value is zero, no underscore stroke is intended or specified.

When a Draw Baseline Rule (DBR) text control is used to draw the underscore, this parameter specifies the suggested width (thickness) of the underscore. If the underscore thickness is greater than 32 pels, the processor (for example, DCF) draws multiple rules. Refer to *Presentation Text Object Content Architecture Reference* for DBR information.

When a page or overlay is printed, the printing system will detect an error if the underscore thickness would cause pels to be printed beyond the boundaries of the page or overlay. Refer to *Mixed Object Document Content Architecture Reference* for Page Descriptor (PGD) information.

20–21 Underscore Position

See “Underscore Position” on page 88.

A value of zero has no meaning.

If the value of FND byte 64 bit 1 is B'0' (characters are not underscored), this value specifies the recommended distance, in pels or relative units, from the baseline to the topmost pel of the underscore stroke, excluding the topmost pel. A positive value indicates an underscore that begins below the baseline. A negative value indicates an underscore that begins above the baseline.

When a page or overlay is printed, the printing system will detect an error if the underscore position would cause pels to be printed beyond the boundaries of the page or overlay. Refer to *Mixed Object Document Content Architecture Reference* for Page Descriptor (PGD) information.

NOP – D3EEEE – No Operation

The No Operation (NOP) structured field can be used to carry comments or other types of unarchitected data.

Length (2 bytes)	**X'D3EEEE'** (3 bytes)	Flags (1 byte)	X'0000' (2 bytes)	Structured Field Data (0–32,759 bytes)

Offset	Type	Name	Range	Meaning	M/O
0–end	UNDF	NopData	Any value	Up to 32,759 bytes of data with no architectural definition	O

Triplets

Structured fields may include “triplets”, which are self-identifying extensions to the positional parameters. If a triplet is included in a structured field, all preceding positional parameters become mandatory. A triplet contains three components as shown below.

Table 18. Triplet Syntax

Offset	Type	Name	Range	Meaning	M/O
0	UBIN	TripLen	3–255	Triplet Length	M
1	CODE	TripID	1–255	Triplet Identifier	M
2–254		TripVal		Triplet Value	M

Bytes **Description**

0 Triplet Length

Specifies the length of the triplet, including this byte.

1 Triplet Identifier

Identifies this triplet.

2–254 Triplet Value

Contains the data for this triplet.

Table 19. Summary of FOCA Triplets

Triplet ID	Triplet Name	Carrying Structured Fields
X'02'	“X'02' – Fully Qualified Name Triplet” on page 177	FND
X'62'	“X'62' – Local Date and Time Stamp Triplet” on page 178	BCP, BFN
X'63'	“X'63', Type 1 – CRC Resource Management Triplet” on page 180	BCP, BFN
X'63'	“X'63', Type 2 – Font Resource Management Triplet” on page 181	BCP, BFN
X'64'	“X'64' – Object Origin Identifier Triplet” on page 183	BCP, BFN
X'65'	“X'65' – User Comment Triplet” on page 184	BCP, BFN
X'6D'	“X'6D' – Extension Font Triplet” on page 185	FNC
X'79'	“X'79' – Metric Adjustment Triplet” on page 186	CFC

X'02' – Fully Qualified Name Triplet

This triplet enables the identification and referencing of objects using Global Identifiers. The semantics of this triplet are defined in MO:DCA, but are repeated here for reader convenience.

Offset	Type	Name	Range	Meaning	M/O
0	UBIN	TripLen	5–254	Triplet Length	M
1	CODE	TripID	X'02'	Triplet Identifier	M
2	CODE	FQNType	X'07' X'08'	Specifies how the GID will be used: Family Name Typeface Name	M
3				Reserved, must be zero	M
4–253	CHAR	FQName		GID of the object. Can be up to 250 bytes long.	M

Bytes Description

0 Triplet Length

Specifies the length of the triplet, including this byte.

1 Triplet Identifier

A value of X'02' identifies this triplet as a Fully Qualified Name triplet.

2 Fully Qualified Name Type

X'07' Family name

The family name is a human-readable name for a group of fonts that are stylistic variants of a single design. This identifier corresponds to the family name of the font design, for example “Times New Roman” is the family name for the “Monotype Times New Roman Expanded” font design. The family name is a character string that normally also appears as a substring in the typeface name.

Note: Family names are not consistently identified in the industry, therefore it might be necessary for implementations to define a synonym table for mapping names. For example, the name “TimesNewRoman” might need to be mapped to “Times New Roman”.

X'08' Typeface name

The triplet contains the name of the font typeface; for example, “Times New Roman Bold Italic”. This identifier corresponds to the full name of the typeface as specified by the font supplier. This is the user interface name which, for example, might be used for specification or selection of the font design.

When the font is built from an Adobe Type 1 font, the typeface name is taken from the FullName field in the original Adobe Type 1 font.

3 Reserved

4–253 Fully Qualified Name

This parameter is EBCDIC encoded using CPGID 500 and GCSGID 103.

Structured fields using triplet X'02':

- “FND – D3A689 – Font Descriptor” on page 152

X'62' – Local Date and Time Stamp Triplet

This triplet identifies the date and time that the object was created or revised.

Offset	Type	Name	Range	Meaning	M/O
0	UBIN	TripLen	17	Triplet Length	M
1	CODE	TripID	X'62'	Identifies this triplet as a Local Date and Time Stamp triplet	M
2	CODE	TSType	X'00' X'01' X'03'	Time Stamp Type Creation (retired) Revision	M
3	CODE	Year (part 1)	X'40' X'F0'–X'F9'	Thousands and hundreds position of the year: 19*xx* 20*xx* through 29*xx*	M
4–5	CODE	Year (part 2)	X'F0F0'– X'F9F9'	Tens and units position of the year	M
6–8	CODE	Day	X'F0F0F1'– X'F3F6F6'	Day of the year	M
9–10	CODE	Hour	X'F0F0'– X'F2F3'	Hour of the day	M
11–12	CODE	Minute	X'F0F0'– X'F5F9'	Minute of the hour	M
13–14	CODE	Second	X'F0F0'– X'F5F9'	Second of the minute	M
15–16	CODE	DecSec	X'F0F0'– X'F9F9'	Hundredths of the second	M

Bytes Description

0 Triplet Length

Length of the triplet, including this byte.

1 Triplet Identifier

Identifies this triplet as the Local Date and Time Stamp triplet.

2 Time Stamp Type

Identifies the type of time stamp.

- **X'00'** Object Creation Date and Time
- **X'01'** Retired for APSRMARK product's Date and Time
- **X'03'** Object Revision Date and Time

3 Thousands and hundreds position of the year

This field identifies the first two digits of the year AD, using the Gregorian calendar. The 1900s are encoded as X'40', the 2000s are encoded as X'F0', the 2100s as X'F1', the 2200s are encoded as X'F2', and so on.

4–5 Tens and units position of the year

This field specifies the last two digits of the year AD, using the Gregorian calendar.

6–8 Day

This field specifies the day of the year, using the Gregorian calendar.

Table 20. Examples of the Date Fields in a Local Date and Time Stamp Triplet

Date	Restructured as	Encoded as
February 1, 1972	“ 72032”	X'40F7F2F0F3F2'
December 31, 1999	“ 99365”	X'40F9F9F3F6F5'
January 1, 2000	“000001”	X'F0F0F0F0F0F1'
February 3, 2072	“072034”	X'F0F7F2F0F3F4'

9–10 Hour of the Day, as a two-character string using only the EBCDIC numbers 0 (X'F0') through 9 (X'F9').

11–12 Minute of the Hour, as a two-character string using only the EBCDIC numbers 0 (X'F0') through 9 (X'F9').

13–14 Second of the Minute, as a two-character string using only the EBCDIC numbers 0 (X'F0') through 9 (X'F9').

15–16 Hundredth of the Second, as a two-character string using only the EBCDIC numbers 0 (X'F0') through 9 (X'F9').

Structured fields using triplet X'62':

- “BCP – D3A887 – Begin Code Page” on page 126
- “BFN – D3A889 – Begin Font” on page 128

X'63', Type 1 – CRC Resource Management Triplet

This triplet provides resource management information such as a Public/Private flag and a retired Cyclic Redundancy Check (CRC) value, to be used in comparing two resource objects that should be identical.

Offset	Type	Name	Range	Meaning	M/O
0	UBIN	TripLen	6	Triplet Length	M
1	CODE	TripID	X'63'	Resource Management triplet	M
2	CODE	FmtQual	X'01'	Format Qualifier	M
3–4	UBIN	RMValue		Retired Resource Management value	M
5	BITS	ResClassFlg Bit 0 Bits 1–7	 B'0' B'1' B'0000000'	 Public Private Reserved	M

Bytes **Description**

0 Triplet Length

Length of the triplet, including this byte.

1 Triplet Identifier

Identifies this triplet as a Font Resource Management triplet.

2 Format Qualifier

This value is a constant, identifying this as a type 1 resource management triplet.

3–4 Retired for Resource Management Value

This field is retired for IBM Print Services Facility™ (PSF) for Z/OS, IBM Print Services Facility (PSF) for VSE, and IBM Print Services Facility (PSF) for OS/400 private use only. A constant value of zero (0) should be used in this field for all other applications.

The CRC is a numeric value calculated by the IBM APSRMARK utility and used by the IBM Remote PrintManager to map objects to locations in its resource library.

The CRC Resource Management format can exist at the same time in the same BCP structured field with the Font Resource Management format.

5 Resource Class Flags

Bit 0 Private Use Flag

See "Private Use" on page 69.

Indicates whether or not this font resource contains data that is privately owned or protected by a license agreement:

- **B'0'** Contains no privately owned information
- **B'1'** Contains privately owned information

Bits 1–7 Reserved

Structured fields using triplet X'63', Type 1:

- "BCP – D3A887 – Begin Code Page" on page 126
- "BFN – D3A889 – Begin Font" on page 128

X'63', Type 2 – Font Resource Management Triplet

This triplet is retired for IBM Print Services Facility (PSF) for Z/OS, IBM Print Services Facility (PSF) for VSE, and IBM Print Services Facility (PSF) for OS/400 private use only.

This triplet, like the Type 1 triplet, provides additional information for comparing two resource objects that should be identical.

Offset	Type	Name	Range	Meaning	M/O
0	UBIN	TripLen	21	Triplet Length	M
1	CODE	TripID	X'63'	Resource Management triplet for fonts	M
2	CODE	FmtQual	X'02'	Resource Management triplet for fonts	M
3–6	UBIN	RMValue	X'00' to X'FFFFFFFF'	Engineering Change (EC) information	M
7	CODE	Year (part 1)	X'40' X'F0'–X'F9'	Thousands and hundreds position of the year: 19*xx* 20*xx* through 29*xx*	M
8–9	CODE	Year (part 2)	X'F0F0' – X'F9F9'	Tens and units position of the year	M
10–12	CODE	Day	X'F0F0F1' – X'F3F6F6'	Day of the year	M
13–14	CODE	Hour	X'F0F0' – X'F2F3'	Hour of the day	M
15–16	CODE	Minute	X'F0F0' – X'F5F9'	Minute of the hour	M
17–18	CODE	Second	X'F0F0' – X'F5F9'	Second of the minute	M
19–20	CODE	DecSec	X'F0F0' – X'F9F9'	Hundredths of the second	M

Bytes **Description**

0 Triplet Length

Length of the triplet, including this byte.

1 Triplet Identifier

Identifies this triplet as a Font Resource Management triplet.

2 Format Qualifier

This value is a constant, identifying this as a Type 2 resource management triplet.

3–6 Resource Management Value

The Engineering Change (EC) information is provided by the IBM APSRMARK utility and used by the IBM Remote PrintManager to manage resident fonts.

The Font Resource Management format can exist at the same time in the same BCP structured field with the CRC Resource Management format.

7 Thousands and hundreds position of the year

This field identifies the first two digits of the year AD, using the Gregorian calendar. The 1900s are encoded as X'40', the 2000s are encoded as X'F0', the 2100s as X'F1', the 2200s are encoded as X'F2', and so on.

8–9 Tens and units position of the year

This field specifies the last two digits of the year AD, using the Gregorian calendar.

10–12 Day

This field specifies the day of the year, using the Gregorian calendar.

Table 21. Examples of the Date Fields in a Font Resource Management Triplet

Date	Restructured as	Encoded as
February 1, 1972	“ 72032”	X'40F7F2F0F3F2'
December 31, 1999	“ 99365”	X'40F9F9F3F6F5'
January 1, 2000	“000001”	X'F0F0F0F0F0F1'
February 3, 2072	“072034”	X'F0F7F2F0F3F4'

13–14 Hour of the Day, as a two-character string using only the EBCDIC numbers 0 (X'F0') through 9 (X'F9').

15–16 Minute of the Hour, as a two-character string using only the EBCDIC numbers 0 (X'F0') through 9 (X'F9').

17–18 Second of the Minute, as a two-character string using only the EBCDIC numbers 0 (X'F0') through 9 (X'F9').

19–20 Hundredth of the Second, as a two-character string using only the EBCDIC numbers 0 (X'F0') through 9 (X'F9').

Structured fields using triplet X'63', Type 2:

- “BCP – D3A887 – Begin Code Page” on page 126
- “BFN – D3A889 – Begin Font” on page 128

X'64' – Object Origin Identifier Triplet

This triplet is used to identify the origin of the resource object.

Offset	Type	Name	Range	Meaning	M/O
0	UBIN	TripLen	61	Triplet Length	M
1	CODE	TripID	X'64'	Object Origin Identifier triplet	M
2	CODE	FmtQual	X'00' X'01' X'02' X'03' X'04' X'05'–X'FF'	Reserved MVS™ VM PC VSE Reserved	M
3–10	CHAR	HostID		Host/System Identifier	M
11–16	CHAR	MediaID		Media Identifier	M
17–60	CHAR	DataSID		Data Set Identifier	M

Bytes **Description**

0 Triplet Length

Length of the triplet, including this byte.

1 Triplet Identifier

Identifies this triplet as the Object Origin Identifier triplet.

2 Format Qualifier

This value is a constant, identifying this as a Type 1 resource management triplet.

3–10 Host/System Identifier

Identification of the specific system that owns the resource.

11–16 Media Identifier

Volume serial number of the data set.

17–60 Data Set Identifier

Identification of the resource file being processed.

Structured fields using triplet X'64':

- "BCP – D3A887 – Begin Code Page" on page 126
- "BFN – D3A889 – Begin Font" on page 128

X'65' – User Comment Triplet

This triplet contains user-defined data and has no effect on processing the object. Any number of comment triplets are allowed within the limits of the maximum structured field length.

Offset	Type	Name	Range	Meaning	M/O
0	UBIN	TripLen	2–255	Triplet Length	M
1	CODE	TripID	X'65'	Triplet Identifier	M
2–254	CHAR	Comment		Optional user-defined information	O

Bytes **Description**

0 Triplet Length

Specifies the length of the triplet, including this byte.

1 Triplet Identifier

A value of X'65' identifies this triplet as a User Comment triplet.

2–254 Triplet Value

See "Comment" on page 58.

Structured fields using triplet X'65':

- "BCP – D3A887 – Begin Code Page" on page 126
- "BFN – D3A889 – Begin Font" on page 128

X'6D' – Extension Font Triplet

This triplet contains the GCSGID for the base font associated with an extension font. An extension font contains user defined characters to be used with another font resource designated by the user.

Offset	Type	Name	Range	Meaning	M/O
0	UBIN	TripLen	4	Triplet Length	M
1	CODE	TripID	X'6D'	Triplet Identifier	M
2–3	CODE	GCSGID	0–65,536	Graphic Character Set GID	M

Bytes **Description**

0 Triplet Length

Specifies the length of the triplet, including this byte.

1 Triplet Identifier

A value of X'6D' identifies this triplet as an Extension Font triplet

2–3 Triplet Value

See "Graphic Character Set Global Identifier" on page 61

This value identifies the graphic character set of the base font, to which the extension font is appended.

Structured fields using triplet X'6D':

- "FNC – D3A789 – Font Control" on page 146

X'79' – Metric Adjustment Triplet

This triplet supplies metric values that can be used to adjust some of the metrics in an outline coded font. If specified for a raster coded font, the triplet is ignored.

The Metric Adjustment triplet is only to be used with the specified vertical font size and specified horizontal scale factor from the CFI structured field; if the specified vertical font size is X'0000', the triplet is ignored. If a size from another source such as a MCF structured field is used, the Metric Adjustment triplet is ignored.

This triplet is defined as follows:

Offset	Type	Name	Range	Meaning	M/O
0	UBIN	Length	X'0F'	Length of the triplet, including this field	M
1	CODE	TID	X'79'	Identifies the Metric Adjustment triplet	M
2	CODE	UnitBase	X'00'	Metric technology unit base: Fixed metrics, 10 inches	M
3–4	UBIN	XUPUB	X'0001'– X'7FFF'	Units per unit base in the X direction	M
5–6	UBIN	YUPUB	X'0001'– X'7FFF'	Units per unit base in the Y direction	M
7–8	SBIN	H-uniform increment	X'8000'– X'7FFF'	Uniform character increment value for horizontal writing	M
9–10	SBIN	V-uniform increment	X'8000'– X'7FFF'	Uniform character increment value for vertical writing	M
11–12	SBIN	H-baseline adjustment	X'8000'– X'7FFF'	Baseline offset adjustment value for horizontal writing	M
13–14	SBIN	V-baseline adjustment	X'8000'– X'7FFF'	Baseline offset adjustment value for vertical writing	M

Byte 0 Triplet length

This field contains the length of this triplet, including itself.

Byte 1 Triplet ID

A value of X'79' identifies this triplet as a Metric Adjustment triplet.

Byte 2 Metric technology unit base

Bytes 3–4 Units per unit base in the X direction

Bytes 5–6 Units per unit base in the Y direction

Bytes 3–4 and 5–6 must contain the same value.

Bytes 7–8 Uniform character increment value for horizontal writing

This value is used only with horizontal writing (character rotation = 0° or 180°), and is ignored with vertical writing (character rotation = 90° or 270°).

This field specifies a uniform character increment value using the units of measure specified in bytes 2–6.

If this value is not X'0000', the font will be treated as a uniform font and this value will be used as the uniform character increment. For each character, the A-space and B-space is not changed and the C-space is increased (or decreased) to achieve the specified character increment.

If this value is X'0000', the character increment values from the font are used.

Bytes 9–10 Uniform character increment value for vertical writing

This value is used only with vertical writing (character rotation = 90° or 270°), and is ignored with horizontal writing (character rotation = 0° or 180°).

This field specifies a uniform character increment value using the units of measure specified in bytes 2–6.

If this value is not X'0000', the font will be treated as a uniform font and this value will be used as the uniform character increment. For each character, the A-space and B-space is not changed and the C-space is increased (or decreased) to achieve the specified character increment.

If this value is X'0000', the character increment values from the font are used.

Bytes 11–12 Baseline adjustment for horizontal writing

This value is used only with horizontal writing (character rotation = 0° or 180°), and is ignored with vertical writing (character rotation = 90° or 270°).

This field specifies a baseline offset adjustment value using the units of measure specified in bytes 2–6. For a character rotation of 0°, the value will be added to the baseline offset for each character in the font. For a character rotation of 180°, the value will be subtracted from the baseline offset for each character in the font.

Bytes 13–14 Baseline adjustment for vertical writing

This value is used only with vertical writing (character rotation = 90° or 270°), and is ignored with horizontal writing (character rotation = 0° or 180°).

This field specifies a baseline offset adjustment value using the units of measure specified in bytes 2–6. For a character rotation of 90°, the value will be added to the baseline offset for each character in the font. For a character rotation of 270°, the value will be subtracted from the baseline offset for each character in the font.

Structured fields using triplet X'79':

- “CFC – D3A78A – Coded Font Control” on page 130

SAA CPI System Font Resource

The format for font resource data, to be passed to an application through the IBM Systems Application Architecture® (SAA®) Common Programming Interface (CPI), is defined by the Query Font Metrics, Query Font Width Table, and Query Kerning Pairs program calls. See *Systems Application Architecture: Common Programming Interface Presentation Reference*, SC26-4359.

IPDS Device Font Resource

The format for font resource data, to be downloaded is defined in the Loaded-Font Command Set chapter of the Intelligent Printer Data Stream (IPDS) architecture. See *Intelligent Printer Data Stream Reference*.

MO:DCA Presentation Document Font Reference

The format for font references, identifying the font resources used by any of the object content architectures carried by the Mixed Object Document Content Architecture (MO:DCA), is defined by the MO:DCA Map Coded Font (MCF) Structured Field. See *Mixed Object Document Content Architecture Reference*.

RFT-DCA Revisable Form Document Font Reference

The format for font references, identifying the font resources used by the Revisable Form Text Document Content Architecture (RFT-DCA), is defined by the Set CFID thru GFID (SFG) Structured Field. See *Document Content Architecture: Revisable-Form-Text Reference*, SC23-0758.

SAA CPI System Font Reference

The format for font references, defined by the IBM Systems Application Architecture (SAA) Common Programming Interface (CPI) is defined by the Create Logical Font program call. See *Systems Application Architecture: Common Programming Interface Presentation Reference*, SC26-4359.

IPDS Device Font Reference

The format for font references in the Intelligent Printer Data Stream (IPDS) architecture is the Load Font Equivalence (LFE) command. See *Intelligent Printer Data Stream Reference*.

Chapter 7. Compliance Requirements

Compliance to the font architecture is defined in terms of semantic or syntactic support of a subset of a FOCA function set. FOCA function sets are defined as a list of font parameters. The font parameters making up a FOCA Function Set are divided into three subsets, representing those parameters required for font referencing (the minimal subset), character positioning (minimal subset plus font and character positioning parameters), and character presentation or font interchange (the full or complete function set).

Semantic support means that a product supports one of the three subsets of parameters defined for a function set, the parameter definitions, and the parameter range of values. The information may be retained in any product-defined syntax, arrangement, or packaging. In addition, a product may use the formal parameter name, the acronym, a product-defined synonym, or a local identifier.

Syntactic support means that the product supports the interchange format for one of the subsets of the function set, in addition to the semantic support. The number of parameters supported under semantic support is a subset of those supported under syntactic support because additional control parameters are required to define the syntactic relationship (maps, pointers, and indices) between parameters.

Syntactic support includes the order of parameter occurrence (if order is prescribed), the encoding of parameter identities and values, and any additional parameters required for management of the structure. Syntactic support of an interchange function set of the font architecture requires support of the MO:DCA-defined resource-object wrappers (for example, BRG/ERG).

It is the responsibility of each product which uses or provides font resource information to designate, in its product specification, the FOCA function set and subset supported. If the product is a system consisting of several products or modules, the functional specification should indicate the function set and subset supported by each of the applicable products or modules.

The following items define the compliance requirements for each of four different classes of products and architectures: Document Editors and Revisable Document Data Streams, Font Services and Font Service Programming Interfaces, Document Formatters and Final-Form Document Data Streams, and Document Presentation and Presentation Service Data Streams.

- Document Editors and Revisable Document Data Streams

  This class includes any program (or module of a larger program) that processes text input from a user and generates a revisable document data stream containing references to the user identified or described font resources. Editing products support one or more document data stream architectures, which in turn might or might not support the font architecture. Editing products must designate the document data stream architectures that they support. The revisable document data stream architecture, if it supports the font architecture, must designate the function set supported and at least semantic support of the referencing subset of that function set.

- Font Services and Font Service Programming Interface

  This class includes any font utility, resource management program or font service programming interface that manages or provides font resource information to another product, including any communication service program that accesses, stores, or transforms font resource information for interchange. It also includes any font generation program or utility that generates, transforms, or modifies font resource information. A product or interface specification in this class, which supports the font architecture, must designate the function set supported and provide syntactic support of the parameters in that function set.

  Note: For any product that performs a transformation on font resources (to or from an internal format used by another product), one side of the transformation must be in the interchange format.

- Document Formatters and Final-Form Document Data Streams

This class includes any program (or module of a larger program) that uses character positioning information from a font resource to assist in the determination of the document format (for example, line breaks, column alignment, page breaks, and character string content), and the final-form document data stream used to represent that format. Any formatting product, which supports the font architecture, must designate the function set supported and at least semantic support of the font and character positioning subset. The final-form document data stream, if it supports the font architecture, must designate the function set supported and at least semantic support of the referencing subset.

Note: Formatting product support of the font architecture does not necessarily imply revision or final-form data stream support of the font architecture, although this might require complex data transformations and reference tables.

- Document Presentation and Presentation Service Data Streams

  This class includes any program (or module of a larger program) or device that uses font resource information to assist in the generation of character images on the presentation surface, and it includes the presentation service or device service data streams required to control the presentation process. Any presentation product or presentation service data stream, which supports the font architecture, must designate the function set supported and at least the semantic support of the parameters for that function set.

Using National Language Support

This font architecture fully supports all known IBM requirements for national language support, including multidirectional text and multibyte document encoding. It is, however, the responsibility of implementing products to provide the necessary collections of font information required to support the national language variations required by their product. That is, the font architecture provides for the definition of metric information for support of multiple character rotations, but the implementing product is responsible for providing the character positioning information for those rotations.

Appendix A. AFP System Font Structured-Field and Triplet Summary

The following table lists the FOCA structured fields defined for AFP System Font Resources (coded fonts, code pages, and font character sets). The table is sorted by the hexadecimal structured field identifier and provides a page number reference to the detailed description of each structured field.

Table 22. AFP System Font Structured-Field Summary

Structured Field ID	Structured Field Name	Page Number Reference
X'D38C87'	Code Page Index (CPI)	"CPI – D38C87 – Code Page Index" on page 140
X'D38C89'	Font Index (FNI)	"FNI – D38C89 – Font Index" on page 160
X'D38C8A'	Coded Font Index (CFI)	"CFI – D38C8A – Coded Font Index" on page 131
X'D3A289'	Font Patterns Map (FNM)	"FNM – D3A289 – Font Patterns Map" on page 164
X'D3A687'	Code Page Descriptor (CPD)	"CPD – D3A687 – Code Page Descriptor" on page 138
X'D3A689'	Font Descriptor (FND)	"FND – D3A689 – Font Descriptor" on page 152
X'D3A787'	Code Page Control (CPC)	"CPC – D3A787 – Code Page Control" on page 134
X'D3A789'	Font Control (FNC)	"FNC – D3A789 – Font Control" on page 146
X'D3A78A'	Coded Font Control (CFC)	"CFC – D3A78A – Coded Font Control" on page 130
X'D3A887'	Begin Code Page (BCP)	"BCP – D3A887 – Begin Code Page" on page 126
X'D3A889'	Begin Font (BFN)	"BFN – D3A889 – Begin Font" on page 128
X'D3A88A'	Begin Coded Font (BCF)	"BCF – D3A88A – Begin Coded Font" on page 125
X'D3A987'	End Code Page (ECP)	"ECP – D3A987 – End Code Page" on page 144
X'D3A989'	End Font (EFN)	"EFN – D3A989 – End Font" on page 145
X'D3A98A'	End Coded Font (ECF)	"ECF – D3A98A – End Coded Font" on page 143
X'D3AB89'	Font Name Map (FNN)	"FNN – D3AB89 – Font Name Map" on page 165
X'D3AC89'	Font Position (FNP)	"FNP – D3AC89 – Font Position" on page 172
X'D3AE89'	Font Orientation (FNO)	"FNO – D3AE89 – Font Orientation" on page 168
X'D3EE89'	Font Patterns (FNG)	"FNG – D3EE89 – Font Patterns" on page 156
X'D3EEEE'	No Operation (NOP)	"NOP – D3EEEE – No Operation" on page 175

The following table lists the FOCA triplets defined for AFP System Font Resources (coded fonts, code pages, and font character sets). The table is sorted by the hexadecimal triplet identifier and provides a page number reference to the detailed description of each triplet.

Table 23. AFP System Font Triplet Summary

Triplet ID	Triplet Name	Page Number Reference
X'02'	Fully Qualified Name triplet	"X'02' – Fully Qualified Name Triplet" on page 177
X'62'	Local Date and Time Stamp Triplet	"X'62' – Local Date and Time Stamp Triplet" on page 178

Table 23 AFP System Font Triplet Summary (cont'd.)

Triplet ID	Triplet Name	Page Number Reference
X'63'	CRC Resource Management triplet	"X'63', Type 1 – CRC Resource Management Triplet" on page 180
X'63'	Font Resource Management triplet	"X'63', Type 2 – Font Resource Management Triplet" on page 181
X'64'	Object Origin Identifier triplet	"X'64' – Object Origin Identifier Triplet" on page 183
X'65'	User Comment triplet	"X'65' – User Comment Triplet" on page 184
X'6D'	Extension Font triplet	"X'6D' – Extension Font Triplet" on page 185
X'79'	Metric Adjustment triplet	"X'79' – Metric Adjustment Triplet" on page 186

Appendix B. Mapping of ISO Parameters

This appendix provides information to aid in understanding the relation between the parameters in this architecture and the parameters defined by the ISO/IEC 9541-1 Font Information Interchange standard. Detailed information about the transformation between ISO and FOCA parameters is defined with each FOCA parameter in the body of this document.

Table 24. Mapping of ISO Font Parameters

ISO 9541 Parameters	FOCA Parameters
FONTNAME (Req.) Font Resource Name	Resource Name (optional global name prefix)
DATAVERSION (Opt.) Data Version	Function Set Code (no mapping of values required)
STANDARDVERSION (Req.) Standard Version	Function Set Code (no mapping of values required)
DATASOURCE (Opt.) Data Source	Data Source
DATACOPYRIGHT (Opt.) Data Copyright	Intellectual Property Data
DSNSOURCE (Req.) Design Source	Design Source
DSNCOPYRIGHT (Opt.) Design Copyright	Intellectual Property Data
RELUNITS (Opt.) Relative Rational Units	Measurement Units (format & coordinate system transformation)
TYPEFACE (Opt.) Typeface Name	Typeface Name
FONTFAMILY (Req.) Font Family Name	Family Name
POSTURE (Req.) Posture	Posture Class
POSTUREANGLE (Opt.) Posture Angle	Nominal Character Slope (format & rotation transformation)
WEIGHT (Req.) Weight	Weight Class
PROPWIDTH (Req.) Proportionate Width	Width Class
STRUCTURE (Req.) Structure	Structure Class
DSNGROUP (Req.) Design Group Design Sub-Group Design Specific Group	Design General Class Design Sub-Class Design Specific Group

Table 24 Mapping of ISO Font Parameters (cont'd.)

ISO 9541 Parameters	FOCA Parameters
NUMGLYPHS (Opt.) Number of Glyphs	Number of Characters
INCGLYPHCOLS (Req.) Included Glyph Collections	Graphic Character Set Global ID (requires mapping of identifiers)
EXCGLYPHCOLS (Opt.) Excluded Glyph Collections	(no corresponding parameter)
INCGLYPHS (Req.) Included Glyphs	Graphic Character Global ID (requires mapping of identifiers)
EXCGLYPHS (Opt.) Excluded Glyphs	(no corresponding parameter)
DSNSIZE (Req.) Design Size	Nominal Vertical Font Size (format & unit transformation)
MINSIZE (Req.) Recommended Minimum Size	Minimum Vertical Font Size (format & unit transformation)
MAXSIZE (Req.) Recommended Maximum Size	Maximum Vertical Font Size (format & unit transformation)
CAPHEIGHT (Opt.) Capital Height	Cap-M Height (format & unit transformation)
LCHEIGHT (Opt.) Lowercase Height	X-Height (format & unit transformation)
MINFEATSZ (Opt.) Minimum Feature Size	(no corresponding parameter)
NOMCAPSTEMWIDTH (Opt.) Nominal Capital Stem Width	(no corresponding parameter)
NOMLCSTEMWIDTH (Opt.) Nominal Lowercase Stem Width	(no corresponding parameter)
NOMWRMODE (Opt.) Nominal Writing Mode Name	(no corresponding parameter)
(no corresponding ISO parameter)	Font Class (used by non-AFP products)
(no corresponding ISO parameter)	Font Sub-Class (used by non-AFP products)
(no corresponding ISO parameter)	Font Typeface Global ID
(no corresponding ISO parameter)	Font Quality (used by IBM Displaywrite Product)
(no corresponding ISO parameter)	Font Use Code (used by IBM Presentation Manager)

Table 24 Mapping of ISO Font Parameters (cont'd.)

ISO 9541 Parameters	FOCA Parameters
(no corresponding ISO parameter)	Misc. Control Flags: Kerning Pair Data, MICR Font, Negative Image, Outline Data, Outline Font, Overstruck Font, Transformable Font, Underscored Font, Uniform Character Box Font, Kerning, Uniform A-Space, Uniform Baseline Offset, Uniform Character Increment
(no corresponding ISO parameter)	Comment
WRMODENAME (Opt.) Writing Mode Name	(derived from character rotation)
NOMESCDIR (Req.) Nominal Escapement Direction	Character Rotation (format & rotation transformation)
ESCCLASS (Req.) Escapement Class	Proportional Spaced (flag maps to both escapement codes)
AVGESCX (Req.) Average Escapement X	Average Escapement (format & coordinate system transformation)
AVGESCY (Req.) Average Escapement Y	Average Escapement (format & coordinate system transformation)
AVGLCESCX (Req.) Average Lowercase Escapement X	Average Lowercase Escapement (format & coordinate system transformation)
AVGLCESCY (Req.) Average Lowercase Escapement Y	Average Lowercase Escapement (format & coordinate system transformation)
AVGCAPESCX (Req.) Average Capital Escapement X	Average Capital Escapement (format & coordinate system transformation)
AVGCAPESCY (Req.) Average Capital Escapement Y	Average Capital Escapement (format & coordinate system transformation)
TABESCX (Req.) Tabular Escapement X	Figure Space Increment (format & coordinate system transformation)
TABESCY (Req.) Tabular Escapement Y	Figure Space Increment (format & coordinate system transformation)
MAXFONTEXT (Req.) Maximum Font Extents	Maximum Character Box Height & Width (format & coordinate system transformation)
SECTORS (Opt.) Sectors	(no corresponding parameter)
ESCADJNAME (Opt.) Escapement Adjust Name	(no corresponding parameter)
CPEA (Opt.) Class Pairwise Esc. Adjust	(no corresponding parameter)
SEC (Opt.) Scale Escapement Correction	(no corresponding parameter)

Table 24 Mapping of ISO Font Parameters (cont'd.)

ISO 9541 Parameters	FOCA Parameters
MINESCADJSZE (Opt.) Minimum Escapement Adjust Size	(no corresponding parameter)
MAXESCADJSZE (Opt.) Maximum Escapement Adjust Size	(no corresponding parameter)
SCORENAME (Opt.) Score Name	(derived from parameter name)
SCOREOFFSETX (Opt.) Score Position Offset X	Underscore Position Overscore Position Throughscore Position (format & coordinate system transformation)
SCOREOFFSETY (Opt.) Score Position Offset Y	Underscore Position Overscore Position Throughscore Position (format & coordinate system transformation)
SCORETHICK (Opt.) Score Thickness	Underscore Width Overscore Width Throughscore Width (format & coordinate system transformation)
VSNAME (Opt.) Variant Script Name	(derived from parameter name)
VSOFFSETX (Opt.) Variant Script Position Offset X	Superscript X-Axis Offset Subscript X-Axis Offset (format & coordinate system transformation)
VSOFFSETY (Opt.) Variant Script Position Offset Y	Superscript Y-Axis Offset Subscript Y-Axis Offset (format & coordinate system transformation)
VSSCALEX (Opt.) Variant Script Anamorphic Scale X	Superscript Vertical Font Size Subscript Vertical Font Size (format & coordinate system transformation)
VSSCALEY (Opt.) Variant Script Anamorphic Scale Y	Superscript Horizontal Font Size Subscript Horizontal Font Size (format & coordinate system transformation)
MINLINESP (Opt.) Recommended Minimum Line Spacing	Default Baseline Increment (format & coordinate system transformation)
MINANASCALE (Opt.) Recommended Minimum Anamorphic Scale	Minimum Horizontal Font Size (format & coordinate system transform)
MAXANASCALE (Opt.) Recommended Maximum Anamorphic Scale	Maximum Horizontal Font Size (format & coordinate system transform)
NOMALIGN (Opt.) Nominial Alignment Mode	(no corresponding parameter)
ALIGNNAME (Opt.) Alignment Mode Name	(no corresponding parameter)
ALIGNOFFSETX (Opt.) Alignment Line Position Offset X	(no corresponding parameter)

Table 24 Mapping of ISO Font Parameters (cont'd.)

ISO 9541 Parameters	FOCA Parameters
ALIGNOFFSETY (Opt.) Alignment Line Position Offset Y	(no corresponding parameter)
ALIGNSCALEX (Opt.) Alignment Mode Anamorphic Scale X	(no corresponding parameter)
ALIGNSCALEY (Opt.) Alignment Mode Anamorphic Scale Y	(no corresponding parameter)
COPYFITNAME (Opt.) Copy Fit Technique Name	(derived from parameter name)
COPYFITMEASURE (Opt.) Copy Fit Measure	Average Weighted Escapement (format & coordinate system transform)
DSNWORDADD (Opt.) Design Wordspace Addition	(no corresponding parameter)
DSNWORDAMPL (Opt.) Design Wordspace Amplification	(no corresponding parameter)
MINWORDADD (Opt.) Recommended Minimum Wordspace Addition	(no corresponding parameter)
MINWORDAMPL (Opt.) Recommended Minimum Wordspace Amplification	(no corresponding parameter)
MAXWORDADD (Opt.) Recommended Maximum Wordspace Addition	(no corresponding parameter)
MAXWORDAMPL (Opt.) Recommended Maximum Wordspace Amplification	(no corresponding parameter)
DSNLETTERADD (Opt.) Design Letterspace Addition	(no corresponding parameter)
DSNLETTERAMPL (Opt.) Design Letterspace Amplification	(no corresponding parameter)
MINLETTERADD (Opt.) Recommended Minimum Letterspace Addition	(no corresponding parameter)
MINLETTERAMPL (Opt.) Recommended Minimum Letterspace Amplification	(no corresponding parameter)
MAXLETTERADD (Opt.) Recommended Maximum Letterspace Addition	(no corresponding parameter)
MAXLETTERAMPL (Opt.) Recommended Maximum Letterspace Amplification	(no corresponding parameter)
(no corresponding ISO parameter)	Em-Space Increment
(no corresponding ISO parameter)	Maximum Character Slope
(no corresponding ISO parameter)	Minimum Character Slope
(no corresponding ISO parameter)	Nominal Horizontal Font Size

Table 24 Mapping of ISO Font Parameters (cont'd.)

ISO 9541 Parameters	FOCA Parameters
(no corresponding ISO parameter)	External Leading
(no corresponding ISO parameter)	Internal Leading
(no corresponding ISO parameter)	Maximum Ascender Height
(no corresponding ISO parameter)	Maximum Baseline Extent
(no corresponding ISO parameter)	Maximum Baseline Offset
(no corresponding ISO parameter)	Maximum Character Increment
(no corresponding ISO parameter)	Maximum Descender Depth
(no corresponding ISO parameter)	Maximum Lowercase Descender Depth
(no corresponding ISO parameter)	Minimum A-Space
(no corresponding ISO parameter)	Space Character Increment
GNAME (Req.) Glyph Name	Graphic Character Global ID (requires mapping of identifiers)
PX (Opt.) Glyph Position Point X	(coordinate system origin)
PY (Opt.) Glyph Position Point Y	(coordinate system origin)
EX (Req.) Glyph Escapement Point X	Character Increment (format & coordinate system transform)
EY (Req.) Glyph Escapement Point Y	Character Increment (format & coordinate system transform)
EXT (Req.) Glyph Extents	A-Space, B-Space, C-Space, Ascender Height, Descender Depth (format & coordinate system transform)
LGN (Opt.) Ligature Name	(no corresponding parameter)
LGSN (Opt.) Ligature Successor Name	(no corresponding parameter)
PEAN (Opt.) Pairwise Escapement Adjust Name	(derived from parameter name)
(linkage by data structure, not name)	Kerning Pair Character 1
(linkage by data structure, not name)	Kerning Pair Character 2

Table 24 Mapping of ISO Font Parameters (cont'd.)

ISO 9541 Parameters	FOCA Parameters
PEAX (Opt.) Pairwise Escapement Adjust X	Kerning Pair X Adjust (format & coordinate system transform)
PEAY (Opt.) Pairwise Escapement Adjust Y	(no corresponding parameter)
SPEAFORWDX (Opt.) Sector Pairwise Forward Adjust X	(no corresponding parameter)
SPEAFORWDY (Opt.) Sector Pairwise Forward Adjust Y	(no corresponding parameter)
SPEABACKWDX (Opt.) Sector Pairwise Backward Adjust X	(no corresponding parameter)
SPEABACKWDY (Opt.) Sector Pairwise Backward Adjust Y	(no corresponding parameter)
CPEAI (Opt.) Class Pairwise Escapement Adjustment Indicator	(no corresponding parameter)
EAI (Opt.) Escapement Adjustment Indicator	(no corresponding parameter)
MINEX (Opt.) Minimum Adjusted Escapement X	(no corresponding parameter)
MINEY (Opt.) Minimum Adjusted Escapement Y	(no corresponding parameter)
MAXEX (Opt.) Maximum Adjusted Escapement X	(no corresponding parameter)
MAXEY (Opt.) Maximum Adjusted Escapement Y	(no corresponding parameter)
(no corresponding ISO parameter)	Baseline Offset
(no corresponding ISO parameter)	Character Box Height
(no corresponding ISO parameter)	Character Box Width
(no corresponding ISO parameter)	Frequency of Use (Implemented by IBM Presentation Manager)

Appendix C. Pattern Technology Information

The following information defines the technologies which have been implemented for the definition of character shapes. Each technology is associated with a value of the Technology parameter in the Font Control Structured Field. This information is provided for information only and is not considered part of the font architecture, nor is it under control mode.

CID Keyed Outline Font Technology

The data format for this outline technology is documented in *Adobe Type 1 Font Format*, Adobe Systems Incorporated, 1990, and *Composite Font Extensions*, Adobe Systems Incorporated, 1989.

The following notes apply to the FOCA implementation of the CID Keyed outline font technology:

- The CID Keyed font files are totally contained within the FOCA Pattern Data parameter.
- All character string data within the CID Keyed font file uses the ASCII character encoding technique.
- The glyph procedures will contain the Adobe glyph names, while the FOCA metrics file may use IBM Graphic Character Global Identifiers (or other names as defined by the Graphic Character GID Encoding parameter). Implementations must consider the possibility of different graphic character GID encodings and perform any necessary mapping.

Type 1 PFB Outline Font Technology

The data format for this outline technology is documented in *Adobe Type 1 Font Format*, Adobe Systems Incorporated, 1990.

The following notes apply to the FOCA implementation of the Adobe Type 1 PFB outline font technology:

- The Type 1 PFB file is totally contained within the FOCA Pattern Data parameter.
- All character string data within the Type 1 PFB file uses the ASCII character encoding technique.
- The PFB glyph procedures will contain the Adobe glyph names, while the FOCA metrics file may use IBM Graphic Character Global Identifiers (or other names as defined by the Graphic Character GID Encoding parameter). Implementations must consider the possibility of different graphic character GID encodings and perform any necessary mapping.

TrueType/OpenType Outline Font Technology

The data format for the TrueType outline technology is documented in the *TrueType Font Files Technical Specification* from Microsoft® Corporation and the *TrueType Reference Manual* from Apple Computer, Inc.

The OpenType font format is an extension of the TrueType font format that allows better support for international character sets and broader multi-platform support. The OpenType font format, which was developed jointly by the Adobe and Microsoft Corporations, is further described in the following document available from the Microsoft web site:

OpenType Specification - Microsoft Corporation

Note: These technologies are not supported within FOCA font objects, but are supported within AFP data streams as data object resources.

Laser Matrix N-Bit Wide Horizontal Sections

An image is formatted as a single rectangle in the binary element sequence of a unidirectional raster scan with no interlaced fields and with parallel raster lines, from left to right (plus x-direction), from top to bottom (plus y-direction). each binary element representing a pel, after decompression, without grayscale, is 0 for no dot, 1 for dot. More than one binary element can represent a pel, after decompression, corresponding to a grayscale algorithm. A pel is nominally centered about the point at which it appears.

Scan lines may range from 1 bit wide to the device limit.

Notices

The AFP Consortium or consortium member companies might have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents.

The following statement does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: AFP Consortium PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. The AFP Consortium might make improvements and/or changes in the architecture described in this publication at any time without notice.

Any references in this publication to Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this architecture and use of those Web sites is at your own risk.

The AFP Consortium may use or distribute any information you supply in any way it believes appropriate without incurring any obligation to you.

This information contains examples of data and reports used in daily business operations. To illustrate them in a complete manner, some examples include the names of individuals, companies, brands, or products. These names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

These terms are trademarks or registered trademarks of Ricoh Co., Ltd., in the United States, other countries, or both:

ACMA
Advanced Function Presentation
AFP
AFPCC
AFP Color Consortium
AFP Color Management Architecture
Bar Code Object Content Architecture
BCOCA
CMOCA
Color Management Object Content Architecture
InfoPrint®
Intelligent Printer Data Stream
IPDS
Mixed Object Document Content Architecture
MO:DCA
Ricoh®

Adobe®, the Adobe logo, PostScript®, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

AFPC and AFP Consortium are trademarks of the AFP Consortium.

IBM® is a registered trademark of the International Business Machines Corporation in the United States, other countries, or both. MVS, PrintManager, Print Services Facility, SAA®, and Systems Application Architecture® are trademarks or registered trademarks of the International Business Machines Corporation in the United States, other countries, or both.

ISO® is a registered trademark of the International Organization for Standardization.

Microsoft® is a registered trademark of the Microsoft Corporation.

UP³I is a trademark of UP³I Limited.

Other company, product, or service names might be trademarks or service marks of others.

Glossary

This glossary contains terms that apply to the FOCA Architecture and also terms that apply to other related presentation architectures.

If you do not find the term that you are looking for, please refer to the *IBM Dictionary of Computing*, document number ZC20-1699 or the *InfoPrint Dictionary of Printing*.

The following definitions are provided as supporting information only, and are not intended to be used as a substitute for the semantics described in the body of this reference.

A

absolute coordinate. One of the coordinates that identify the location of an addressable point with respect to the origin of a specified coordinate system. Contrast with *relative coordinate*.

absolute move. A method used to designate a new presentation position by specifying the distance from the designated axes to the new presentation position. The reference for locating the new presentation position is a fixed position as opposed to the current presentation position.

absolute positioning. The establishment of a position within a coordinate system as an offset from the coordinate system origin. Contrast with *relative positioning*.

Abstract Syntax Notation One (ASN.1). A notation for defining data structures and data types. The notation is defined in international standard ISO/IEC 8824(E). See also *object identifier*.

ACK. See *Positive Acknowledge Reply*.

Acknowledge Reply. A printer-to-host reply that returns printer information or reports exceptions. An Acknowledge Reply can be positive or negative. See also *Positive Acknowledge Reply* and *Negative Acknowledge Reply*.

Acknowledgment Request. A request from the host for information from the printer. An example of an Acknowledgment Request is the use of the acknowledgment-required flag by a host system to request an Acknowledge Reply from an attached printer.

acknowledgment-required flag (ARQ). A flag that requests a printer to return an Acknowledge Reply. The acknowledgment-required flag is bit zero of an IPDS command's flag byte.

active coded font. The coded font that is currently being used by a product to process text.

additive primary colors. Red, green, and blue light, transmitted in video monitors and televisions. When used in various degrees of intensity and variation, they create all other colors of light; when superimposed equally, they create white. Contrast with *subtractive primary colors*.

addressable position. A position in a presentation space or on a physical medium that can be identified by a coordinate from the coordinate system of the presentation space or physical medium. See also *picture element*. Synonymous with *position*.

Advanced Function Presentation (AFP). An open architecture for the management of presentable information that is developed by the AFP Consortium (AFPC). AFP comprises a number of data stream and data object architectures:

- Mixed Object Document Content (MO:DCA) Architecture; formerly referred to as AFPDS
- Intelligent Printer Data Stream (IPDS) Architecture
- AFP Line Data Architecture
- Bar Code Object Content Architecture (BCOCA)
- Color Management Object Content Architecture (CMOCA)
- Font Object Content Architecture (FOCA)
- Graphics Object Content Architecture for AFP (AFP GOCA)
- Image Object Content Architecture (IOCA)
- Metadata Object Content Architecture (MOCA)
- Presentation Text Object Content Architecture (PTOCA)

AEA. See *alternate exception action*.

AFM file. A file containing the metric information required for positioning the characters of a font. The metric information contained in this file was extracted from a PFB file, in an ASCII file format defined by Adobe Systems Inc., and used for character positioning and page formatting.

AFP. See *Advanced Function Presentation*.

AFP Consortium (AFPC). A formal open standards body that develops and maintains AFP architecture. Information about the consortium can be found at `www.afpcinc.org`.

AFP data stream. A presentation data stream that is processed in AFP environments. The MO:DCA architecture defines the strategic AFP interchange data stream. The IPDS architecture defines the strategic AFP printer data stream.

AFPDS. A term formerly used to identify the composed-page MO:DCA-based data stream interchanged in AFP environments. See also *MO:DCA* and *AFP data stream*.

AIAG. See *Automotive Industry Action Group*.

AIM. See *Automatic Identification Manufacturers, Inc*.

all points addressable (APA). The capability to address, reference, and position data elements at any addressable

position in a presentation space or on a physical medium. Contrast with character cell addressable, in which the presentation space is divided into a fixed number of character-size rectangles in which characters can appear. Only the cells are addressable. An example of all points addressable is the positioning of text, graphics, and images at any addressable point on the physical medium. See also *picture element*.

alternate exception action (AEA). In the IPDS architecture, a defined action that a printer can take when a clearly defined, but unsupported, request is received. Control over alternate exception actions is specified by an Execute Order Anystate Exception-Handling Control command.

American National Standards Institute (ANSI). An organization consisting of producers, consumers, and general interest groups. ANSI establishes the procedures by which accredited organizations create and maintain voluntary industry standards in the United States. It is the United States constituent body of the International Organization for Standardization (ISO).

anamorphic scaling. Scaling an object differently in the vertical and horizontal directions. See also *scaling*, *horizontal font size*, and *vertical font size*.

annotation. A comment or explanation associated with the contents of a document component. An example of an annotation is a string of text that represents a comment on an image object on a page.

annotation link. In MO:DCA, a link type that specifies the linkage from a source document component to a target document component that contains an annotation.

annotation object. In MO:DCA, an object that contains an annotation. Objects that are targets of annotation links are annotation objects.

ANSI. See *American National Standards Institute*.

APA. See *all points addressable*.

append. In MO:DCA, an addition to or continuation of the contents of a document component. An example of an append is a string of text that is an addition to an existing string of text on a page.

append link. In MO:DCA, a link type that specifies the linkage from the end of a source document component to a target document component that contains an append.

append object. In MO:DCA, an object that contains an append. Objects that are targets of append links are append objects.

application. (1) The use to which an information system is put. (2) A collection of software components used to perform specific types of work on a computer.

application program. A program written for or by a user that applies to the user's work.

arc. A continuous portion of the curved line of a circle or ellipse. See also *full arc*.

architected. Identifies data that is defined and controlled by an architecture. Contrast with *unarchitected*.

arc parameters. Variables that specify the curvature of an arc.

area. In GOCA, a set of closed figures that can be filled with a pattern or a color.

area filling. A method used to fill an area with a pattern or a color.

ARQ. See *acknowledgment-required flag*.

article. The physical item that a bar code identifies.

ascender. The parts of certain lowercase letters, such as *b*, *d*, or *f*, that at zero-degree character rotation rise above the top edge of other lowercase letters such as *a*, *c*, and *e*. Contrast with *descender*.

ascender height. The character shape's most positive character coordinate system Y-axis value.

ASN.1. See *Abstract Syntax Notation One*.

A-space. The distance from the character reference point to the least positive character coordinate system X-axis value of the character shape. A-space can be positive, zero, or negative. See also *B-space* and *C-space*.

aspect ratio. (1) The ratio of the horizontal size of a picture to the vertical size of the picture. (2) In a bar code symbol, the ratio of bar height to symbol length.

asynchronous exception. Any exception other than those used to report a synchronous data-stream defect (action code X'01' or X'1F'), function no longer achievable (action code X'06'), or synchronous resource-storage problem (action code X'0C'). Asynchronous exceptions occur after the received page station. An example of an asynchronous exception is a paper jam. See also *data-stream exception*. Contrast with *synchronous exception*.

attribute. A property or characteristic of one or more constructs. See also *character attribute*, *color attribute*, *current drawing attributes*, *default drawing attributes*, *line attributes*, *marker attributes*, and *pattern attributes*.

Automatic Identification Manufacturers, Inc. (AIM). A trade organization consisting of manufacturers, suppliers, and users of bar codes.

Automotive Industry Action Group (AIAG). The coalition of automobile manufacturers and suppliers

working to standardize electronic communications within the auto industry.

B

b_c. See *current baseline print coordinate*.

b_i. See *initial baseline print coordinate*.

B. See *baseline direction*.

+B. Positive baseline direction.

B_c. See *current baseline presentation coordinate*.

B_o. See *baseline presentation origin*.

background. (1) The part of a presentation space that is not occupied with object data. (2) In GOCA, that portion of a graphics primitive that is mixed into the presentation space under the control of the current values of the background mix and background color attributes. Contrast with *foreground*. (3) In GOCA, that portion of a character cell that does not represent a character. (4) In bar codes, the spaces, quiet zones, and area surrounding a printed bar code symbol.

background color. The color of a background. Contrast with *foreground color*.

background mix. (1) An attribute that determines how the color of the background of a graphics primitive is combined with the existing color of the graphics presentation space. (2) An attribute that determines how the points in overlapping presentation space backgrounds are combined. Contrast with *foreground mix*.

band. An arbitrary layer of an image. An image can consist of one or more bands of data.

bar. In bar codes, the darker element of a printed bar code symbol. See also *element*. Contrast with *space*.

bar code. An array of elements, such as bars, spaces, and two-dimensional modules that together represent data elements or characters in a particular symbology. The elements are arranged in a predetermined pattern following unambiguous rules defined by the symbology. See also *bar code symbol*.

Bar Code command set. In the IPDS architecture, a collection of commands used to present bar code symbols in a page, page segment, or overlay.

bar code density. The number of characters per inch (cpi) in a bar code symbology. In most cases, the range is three to ten cpi. See also *character density*, *density*, and *information density*.

bar code object area. The rectangular area on a logical page into which a bar code presentation space is mapped.

Bar Code Object Content Architecture (BCOCA). An architected collection of constructs used to interchange and present bar code data.

bar code presentation space. A two-dimensional conceptual space in which bar code symbols are generated.

bar code symbol. A combination of characters including start and stop characters, quiet zones, data characters, and check characters required by a particular symbology, that form a complete, scannable entity. See also *bar code*.

bar code symbology. A bar code language. Bar code symbologies are defined and controlled by various industry groups and standards organizations. Bar code symbologies are described in public domain bar code specification documents. Synonymous with *symbology*. See also *Canadian Grocery Product Code (CGPC)*, *European Article Numbering (EAN)*, *Japanese Article Numbering (JAN)*, and *Universal Product Code (UPC)*.

bar height. In bar codes, the bar dimension perpendicular to the bar width. Synonymous with *bar length* and *height*.

bar length. In bar codes, the bar dimension perpendicular to the bar width. Synonymous with *bar height* and *height*.

bar width. In bar codes, the thickness of a bar measured from the edge closest to the symbol start character to the trailing edge of the same bar.

bar width reduction. In bar codes, the reduction of the nominal bar width dimension on film masters or printing plates to compensate for systematic errors in some printing processes.

base-and-towers concept. A conceptual illustration of an architecture that shows the architecture as a base with optional towers. The base and the towers represent different degrees of function achieved by the architecture.

base support level. Within the base-and-towers concept, the smallest portion of architected function that is allowed to be implemented. This is represented by a base with no towers. Synonymous with *mandatory support level*.

baseline. A conceptual line with respect to which successive characters are aligned. See also *character baseline*. Synonymous with *printing baseline* and *sequential baseline*.

baseline coordinate. One of a pair of values that identify the position of an addressable position with respect to the origin of a specified I,B coordinate system. This value is specified as a distance in addressable positions from the I axis of an I,B coordinate system. Synonymous with *B-coordinate*.

baseline direction (B). The direction in which successive lines of text appear on a logical page. Synonymous with *baseline progression* and *B-direction*.

baseline extent. A rectangular space oriented around the character baseline and having one dimension parallel to the character baseline. The space is measured along the Y axis of the character coordinate system. For bounded character boxes, the baseline extent at any rotation is its character coordinate system Y-axis extent. Baseline extent varies with character rotation. See also *maximum baseline extent*.

baseline increment. The distance between successive baselines.

baseline offset. The perpendicular distance from the character baseline to the character box edge that is parallel to the baseline and has the more positive character coordinate system Y-axis value. For characters entirely within the negative Y-axis region, the baseline offset can be zero or negative. An example is a subscript character. Baseline offset can vary with character rotation.

baseline presentation origin (B_o). The point on the B axis where the value of the baseline coordinate is zero.

baseline progression (B). The direction in which successive lines of text appear on a logical page. Synonymous with *baseline direction* and *B-direction*.

B axis. The axis of the I,B coordinate system that extends in the baseline or B-direction. The B axis does not have to be parallel to the Y_p axis of its bounding X_p,Y_p coordinate space.

BCOCA. See *Bar Code Object Content Architecture*.

B-coordinate. One of a pair of values that identify the position of an addressable position with respect to the origin of a specified I,B coordinate system. This value is specified as a distance in addressable positions from the I axis of an I,B coordinate system. Synonymous with *baseline coordinate*.

B-direction (B). The direction in which successive lines of text appear on a logical page. Synonymous with *baseline direction* and *baseline progression*.

Bearer Bars. Bars that surround an Interleaved 2-of-5 bar code to prevent misreads and short scans that might occur when a skewed scanning beam enters or exits the bar code symbol through its top or bottom edge. When plates are used in the printing process, Bearer Bars help equalize the pressure exerted by the printing plate over the entire surface of the symbol to improve print quality. There are two styles: 1) four bars that completely surround the bar/space pattern and 2) two bars that are placed at the top and the bottom of the bar/space pattern.

Begin Segment Introducer (BSI). An IPDS graphics self-defining field that precedes all of the drawing orders in a graphics segment.

between-the-pels. The concept of pel positioning that establishes the location of a pel's reference point at the edge of the pel nearest to the preceding pel rather than through the center of the pel.

B-extent. The extent in the B-axis direction of an I,B coordinate system. The B-extent must be parallel to one of the axes of the coordinate system that contains the I,B coordinate system. The B-extent is parallel to the Y_p-extent when the B axis is parallel to the Y_p axis or to the X_p-extent when the B axis is parallel to the X_p axis.

bilevel custom pattern. In GOCA, a custom pattern that is defined as uncolored, then has a single color assigned to it when it is used to fill an area. Contrast with *full-color custom pattern*.

BITS. A data type for architecture syntax, indicating one or more bytes to be interpreted as bit string information.

blend. A mixing rule in which the intersection of part of a new presentation space P_{new} with part of an existing presentation space $P_{existing}$ changes to a new color attribute that represents a color-mixing of the color attributes of P_{new} with the color attributes of $P_{existing}$. For example, if P_{new} has foreground color-attribute blue and $P_{existing}$ has foreground color-attribute yellow, the area where the two foregrounds intersect changes to a color attribute of green. See also *mixing rule*. Contrast with *overpaint* and *underpaint*.

body. (1) On a printed page, the area between the top and bottom margins that can contain data. (2) In a book, the portion between the front matter and the back matter.

boldface. A heavy-faced type. Printing in a heavy-faced type.

boundary alignment. A method used to align image data elements by adding padding bits to each image data element.

bounded character box. A conceptual rectangular box, with two sides parallel to the character baseline, that circumscribes a character and is just large enough to contain the character, that is, just touching the shape on all four sides.

BSI. See *Begin Segment Introducer*.

B-space. The distance between the character coordinate system X-axis values of the two extremities of a character shape. See also *A-space* and *C-space*.

buffered pages. Pages and copies of pages that have been received but not yet reflected in committed page counters and copy counters.

C

Canadian Grocery Product Code (CGPC). The bar code symbology used to code grocery items in Canada.

cap-M height. The average height of the uppercase characters in a font. This value is specified by the designer of a font and is usually the height of the uppercase *M*.

CCSID. See *Coded Character Set Identifier*.

CGCSGID. See *Coded Graphic Character Set Global Identifier*.

CGPC. See *Canadian Grocery Product Code*.

CHAR. A data type for architecture syntax, indicating one or more bytes to be interpreted as character information.

character. (1) A member of a set of elements used for the organization, control, or representation of data. A character can be either a graphic character or a control character. See also *graphic character* and *control character*. (2) In bar codes, a single group of bar code elements that represent an individual number, letter, punctuation mark, or other symbol.

character angle. The angle that is between the baseline of a character string and the horizontal axis of a presentation space or physical medium.

character attribute. A characteristic that controls the appearance of a character or character string.

character baseline. A conceptual reference line that is coincident with the X axis of the character coordinate system.

character box. A conceptual rectangular box with two sides parallel to the character baseline. A character's shape is formed within a character box by a presentation process, and the character box is then positioned in a presentation space or on a physical medium. The character box can be rotated before it is positioned.

character-box reference edges. The four edges of a character box.

character cell addressable. Allowing characters to be addressed, referenced, and positioned only in a fixed number of character-size rectangles into which a presentation space is divided. Contrast with *all points addressable*.

character cell size. The size of a rectangle in a drawing space used to scale font symbols into the drawing space.

character code. An element of a code page or a cell in a code table to which a character can be assigned. The element is associated with a binary value. The assignment of a character to an element of a code page determines the binary value that will be used to represent each occurrence of the character in a character string.

character coordinate system. An orthogonal coordinate system that defines font and character measurement distances. The origin is the character reference point. The X axis coincides with the character baseline.

character density. The number of characters per inch (cpi) in a bar code symbology. In most cases, the range is three to ten cpi. See also *bar code density*, *density*, and *information density*.

character direction. In GOCA, an attribute controlling the direction in which a character string grows relative to the inline direction. Values are: left-to-right, right-to-left, top-to-bottom, and bottom-to-top. Synonymous with *direction*.

character escapement point. The point where the next character reference point is usually positioned. See also *character increment* and *presentation position*.

character identifier. The unique name for a graphic character.

character increment. The distance from a character reference point to a character escapement point. For each character, the increment is the sum of a character's A-space, B-space, and C-space. A character's character increment is the distance the inline coordinate is incremented when that character is placed in a presentation space or on a physical medium. Character increment is a property of each graphic character in a font and of the font's character rotation.

character increment adjustment. In a scaled font, an adjustment to character increment values. The adjustment value is derived from the kerning track values for the font used to present the characters.

character metrics. Measurement information that defines individual character values such as height, width, and space. Character metrics can be expressed in specific fixed units, such as pels, or in relative units that are independent of both the resolution and the size of the font. Often included as part of the more general term *font metrics*. See also *character set metrics* and *font metrics*.

character pattern. The scan pattern for a graphic character of a particular size, style, and weight.

character-pattern descriptor. Information that the printer needs to separate font raster patterns. Each character pattern descriptor is eight bytes long and specifies both the character box size and an offset value; the offset value permits the printer to find the beginning of the character raster pattern within the character raster pattern data for the complete coded font.

character positioning. A method used to determine where a character is to appear in a presentation space or on a physical medium.

character precision. The acceptable amount of variation in the appearance of a character on a physical medium from a specified ideal appearance, including no acceptable variation. Examples of appearance characteristics that can

vary for a character are character shape and character position.

character reference point. The origin of a character coordinate system. The X axis is the character baseline.

character rotation. The alignment of a character with respect to its character baseline, measured in degrees in a clockwise direction. Examples are 0°, 90°, 180°, and 270°. Zero-degree character rotation exists when a character is in its customary alignment with the baseline. Character rotation and font inline sequence are related in that character rotation is a clockwise rotation; font inline sequence is a counter-clockwise rotation. Contrast with *rotation*.

character set. A finite set of different graphic characters or control characters that is complete for a given purpose. For example, the character set in ISO Standard 646, *7-Bit Coded Character Set for Information Processing Interchange*.

character set attribute. An attribute used to specify a coded font.

character set metrics. The measurements used in a font. Examples are height, width, and character increment for each character of the font. See also *character metrics* and *font metrics*.

character shape. The visual representation of a graphic character.

character shape presentation. A method used to form a character shape on a physical medium at an addressable position.

character shear. The angle of slant of a character cell that is not perpendicular to a baseline. Synonymous with *shear*.

character string. A sequence of characters.

check character. In bar codes, a character included within a bar code message whose value is used to perform a mathematical check to ensure the accuracy of that message. Synonymous with *check digit*.

check digit. In bar codes, a character included within a bar code message whose value is used to perform a mathematical check to ensure the accuracy of that message. Synonymous with *check character*.

CID file. A file containing the font information required for presenting the characters of a font. The shape information (glyph procedures) contained in this file is in a binary encoded format defined by Adobe Systems Inc., optimized for large character set fonts (for example, Japanese ideographic fonts having several thousand characters).

CIE. See Commission Internationale d'Éclairage.

CIELAB color space. Internationally accepted color space model used as a standard to define color within the graphic arts industry, as well as other industries. L*, a*, and b* are plotted at right angles to one another. Equal distances in the space represent approximately equal color difference.

CIEXYZ color space. The fundamental CIE-based color space that allows colors to be expressed as a mixture of the three tristimulus values X, Y, and Z.

CJK fonts. Fonts that contain a set of unified ideographic characters used in the written Chinese, Japanese, and Korean languages. The character encoding is the same for each language, but there might be glyph variants between languages.

clear area. A clear space that contains no machine-readable marks preceding the start character of a bar code symbol or following the stop character. Synonymous with *quiet zone*. Contrast with *intercharacter gap* and *space*.

clipping. Eliminating those parts of a picture that are outside of a clipping boundary such as a viewing window or presentation space. See also *viewing window*. Synonymous with *trimming*.

CMAP file. A file containing the mapping of code points to the character index values used in a CID file. The code points conform to a particular character coding system which is used to identify the characters in a document data stream. The character index values are assigned in a CID file for identification of the glyph procedure used to define the character shape. The mapping information in this file is in an ASCII file format defined by Adobe Systems Inc.

CMOCA. See *Color Management Object Content Architecture*.

CMR. See Color management resource.

CMY. Cyan, magenta, and yellow, the subtractive primary colors.

CMYK color space. The color model used in four-color printing. Cyan, magenta, and yellow, the subtractive primary colors, are used with black to effectively create a multitude of other colors.

Codabar. A bar code symbology characterized by a discrete, self-checking, numeric code with each character represented by a standalone group of four bars and the three spaces between them.

CODE. A data type for architecture syntax that indicates an architected constant to be interpreted as defined by the architecture.

Code 39. A bar code symbology characterized by a variable-length, bidirectional, discrete, self-checking, alphanumeric code. Three of the nine elements are wide

and six are narrow. It is the standard for LOGMARS (the Department of Defense) and the AIAG.

Code 128. A bar code symbology characterized by a variable-length, alphanumeric code with 128 characters.

Coded Character Set Identifier (CCSID). A 16-bit number identifying a specific set consisting of an encoding scheme identifier, character set identifiers, code page identifiers, and other relevant information that uniquely identifies the coded graphic character representation used.

coded font. (1) A resource containing elements of a code page and a font character set, used for presenting text, graphics character strings, and bar code HRI. See also *code page* and *font character set*. (2) In FOCA, a resource containing the resource names of a valid pair of font character set and code page resources. The graphic character set of the font character set must match the graphic character set of the code page for the coded font resource pair to be valid. (3) In the IPDS architecture, a raster font resource containing code points that are directly paired to font metrics and the raster representation of character shapes, for a specific graphic character set. (4) In the IPDS architecture, a font resource containing descriptive information, a code page, font metrics, and a digital-technology representation of character shapes for a specific graphic character set.

coded font local identifier. A binary identifier that is mapped by the controlling environment to a named resource to identify a coded font. See also *local identifier*.

coded graphic character. A graphic character that has been assigned one or more code points within a code page.

coded graphic character set. A set of graphic characters with their assigned code points.

Coded Graphic Character Set Global Identifier (CGCSGID). A four-byte binary or a ten-digit decimal identifier consisting of the concatenation of a GCSGID and a CPGID. The CGCSGID identifies the code point assignments in the code page for a specific graphic character set, from among all the graphic characters that are assigned in the code page.

code page. (1) A resource object containing descriptive information, graphic character identifiers, and code points corresponding to a coded graphic character set. Graphic characters can be added over time; therefore, to specifically identify a code page, both a GCSGID and a CPGID should be used. See also *coded graphic character set*. (2) A set of assignments, each of which assigns a code point to a character. Each code page has a unique name or identifier. Within a given code page, a code point is assigned to one character. More than one character set can be assigned code points from the same code page. See also *code point* and *section*.

Code Page Global Identifier (CPGID). A unique code page identifier that can be expressed as either a two-byte binary or a five-digit decimal value.

code point. A unique bit pattern that can serve as an element of a code page or a site in a code table, to which a character can be assigned. The element is associated with a binary value. The assignment of a character to an element of a code page determines the binary value that will be used to represent each occurrence of the character in a character string. Code points are one or more bytes long. See also *code table* and *section*.

code table. A table showing the character allocated to each code point in a code. See also *code page* and *code point*.

color. A visual attribute of things that results from the light they emit, transmit, or reflect.

color attribute. An attribute that affects the color values provided in a graphics primitive, a text control sequence, or an IPDS command. Examples of color attributes are foreground color and background color.

color component. A dimension of a color value expressed as a numeric value. For example, a color value might consist of one, two, three, four, or eight components, also referred to as channels.

color conversion. The process of converting colors from one color space to another.

color image. Images whose image data elements are represented by multiple bits or whose image data element values are mapped to color values. Constructs that map image-data-element values to color values are look-up tables and image-data-element structure parameters. Examples of color values are screen color values for displays and color toner values for printers.

color management. The technology to calibrate the color of input devices (such as scanners or digital cameras), display devices, and output devices (such as printers or offset presses).

Color Management Object Content Architecture (CMOCA). An architected collection of constructs used for the interchange and presentation of the color management information required to render a print file, document, group of pages or sheets, page, overlay, or data object with color fidelity.

color management resource. An object that provides color management in presentation environments.

color management system. A set of software designed to increase the accuracy and consistency of color between color devices like a scanner, display, and printer.

color model. The method by which a color is specified. For example, the RGB color space specifies color in terms

of three intensities for red (R), green (G), and blue (B). Also referred to as *color space*.

color of medium. The color of a presentation space before any data is added to it. Synonymous with *reset color*.

color space. The method by which a color is specified. For example, the RGB color space specifies color in terms of three intensities for red (R), green (G), and blue (B). Also referred to as *color model*.

color table. A collection of color element sets. The table can also specify the method used to combine the intensity levels of each element in an element set to produce a specific color. Examples of methods used to combine intensity levels are the additive method and the subtractive method. See also *color model*.

command. (1) In the IPDS architecture, a structured field sent from a host to a printer. (2) In GOCA, a data-stream construct used to communicate from the controlling environment to the drawing process. The command introducer is environment dependent. (3) A request for system action.

command set. A collection of IPDS commands.

command-set vector. Information that identifies an IPDS command set and data level supported by a printer. Command-set vectors are returned with an Acknowledge Reply to an IPDS Sense Type and Model command.

Commission Internationale d'Éclairage (CIE). An association of international color scientists who produced the standards that are used as the basis of the description of color.

compression algorithm. An algorithm used to compress image data. Compression of image data can decrease the volume of data required to represent an image.

construct. An architected set of data such as a structured field or a triplet.

continuous code. A bar code symbology characterized by designating all spaces within the symbol as parts of characters, for example, Interleaved 2 of 5. There is no intercharacter gap in a continuous code. Contrast with *discrete code*.

continuous-form media. Connected sheets. An example of connected sheets is sheets of paper connected by a perforated tear strip. Contrast with *cut-sheet media*.

control character. (1) A character that denotes the start, modification, or end of a control function. A control character can be recorded for use in a subsequent action, and it can have a graphic representation. See also *character*. (2) A control function the coded representation of which consists of a single code point.

control instruction. A data construct transmitted from the controlling environment and interpreted by the environment interface to control the operation of the graphics processor.

controlled white space. White space caused by execution of a control sequence. See also *white space*.

controlling environment. The environment in which an object is embedded, for example, the IPDS and MO:DCA data streams.

control sequence. A sequence of bytes that specifies a control function. A control sequence consists of a control sequence introducer and zero or more parameters.

control sequence chaining. A method used to identify a sequential string of control sequences so they can be processed efficiently.

control sequence class. An assigned coded character that identifies a control sequence's syntax and how that syntax is to be interpreted. An example of a control sequence class is X'D3', that identifies presentation text object control sequences.

control sequence function type. The coded character occupying the fourth byte of an unchained control sequence introducer. This code defines the function whose semantics can be prescribed by succeeding control sequence parameters.

control sequence introducer. The information at the beginning of a control sequence. An unchained control sequence introducer consists of a control sequence prefix, a class, a length, and a function type. A chained control sequence introducer consists of a length and a function type.

control sequence length. The number of bytes used to encode a control sequence excluding the control sequence prefix and class.

control sequence prefix. The escape character used to identify a control sequence. The control sequence prefix is the first byte of a control sequence. An example of a control sequence prefix is X'2B'.

coordinate system. A Cartesian coordinate system. An example is the image coordinate system that uses the fourth quadrant with positive values for the Y axis. The origin is the upper left-hand corner of the fourth quadrant. A pair of (x,y) values corresponds to one image point. Each image point is described by an image data element. See also *character coordinate system*.

coordinates. A pair of values that specify a position in a coordinate space. See also *absolute coordinate* and *relative coordinate*.

copy control. A method used to specify the number of copies for a presentation space and the modifications to be made to each copy.

copy counter. Bytes in an Acknowledge Reply that identify the number of copies of a page that have passed a particular point in the logical paper path.

copy group. A set of copy subgroups that specify all copies of a sheet. In the IPDS architecture, a copy group is specified by a Load Copy Control command. In MO:DCA, a copy group is specified within a Medium Map. See also *copy subgroup*.

copy modification. The process of adding, deleting, or replacing data on selected copies of a presentation space.

copy set. A collection of pages intended to be printed multiple times. For example, when multiple copies of a book or booklet is printed, each copy of the book or booklet is a copy set. This term was originally used with copy machines to identify collections of copies that are delivered as sets or stapled as sets. The term was also used when printing multiple copies of an MVS data set.

copy subgroup. A part of a copy group that specifies a number of identical copies of a sheet and all modifications to those copies. Modifications include the media source, the media destination, medium overlays to be presented on the sheet, text suppressions, the number of pages on the sheet, and either simplex or duplex presentation. In the IPDS architecture, copy subgroups are specified by Load Copy Control command entries. In MO:DCA, copy subgroups are specified by repeating groups in the Medium Copy Count structured field in a Medium Map. See also *copy group*.

correlation. A method used in the IPDS architecture to match exceptions with commands.

correlation ID. A two-byte value that specifies an identifier of an IPDS command. The correlation ID is optional and is present only if bit one of the command's flag byte is B'1'.

CPGID. See *Code Page Global Identifier*.

cpi. Characters per inch.

C-space. The distance from the most positive character coordinate system X-axis value of a character shape to the character escapement point. C-space can be positive, zero, or negative. See also *A-space* and *B-space*.

current baseline coordinate. The baseline presentation position at the present time. The baseline presentation position is the summation of the increments of all baseline controls since the baseline was established in the presentation space. The baseline presentation position is established in a presentation space either as part of the initialization procedures for processing an object or by an Absolute Move Baseline control sequence. Synonymous with *current baseline presentation coordinate*.

current baseline presentation coordinate (B_c). The baseline presentation position at the present time. The baseline presentation position is the summation of the increments of all baseline controls since the baseline was established in the presentation space. The baseline presentation position is established in a presentation space either as part of the initialization procedures for processing an object or by an Absolute Move Baseline control sequence. Synonymous with *current baseline coordinate*.

current baseline print coordinate (b_c). In the IPDS architecture, the baseline coordinate corresponding to the current print position on a logical page. The current baseline print coordinate is a coordinate in an I,B coordinate system. See also *I,B coordinate system*.

current drawing attributes. The set of attributes used at the present time to direct a drawing process. Contrast with *default drawing attributes*.

current drawing controls. The set of drawing controls used at the present time to direct a drawing process. Contrast with *default drawing controls*.

current inline coordinate. The inline presentation position at the present time. This inline presentation position is the summation of the increments of all inline controls since the inline coordinate was established in the presentation space. An inline presentation position is established in a presentation space either as part of the initialization procedures for processing an object or by an Absolute Move Inline control sequence. Synonymous with *current inline presentation coordinate*.

current inline presentation coordinate (I_c). The inline presentation position at the present time. This inline presentation position is the summation of the increments of all inline controls since the inline coordinate was established in the presentation space. An inline presentation position is established in a presentation space either as part of the initialization procedures for processing an object or by an Absolute Move Inline control sequence. Synonymous with *current inline coordinate*.

current inline print coordinate (i_c). In the IPDS architecture, the inline coordinate corresponding to the current print position on a logical page. The current inline print coordinate is a coordinate in an I,B coordinate system. See also *I,B coordinate system*.

current logical page. The logical page presentation space that is currently being used to process the data within a page object or an overlay object.

current position. The position identified by the current presentation space coordinates. For example, the coordinate position reached after the execution of a drawing order. See also *current baseline presentation*

coordinate and *current inline presentation coordinate*. Contrast with *given position*.

custom line type value. A user-defined line type, defined by a series of pairs of a dash/dot length followed by a move length. Contrast with *standard line type value*.

custom pattern. In GOCA, a user-defined pattern, defined by the picture drawn by a series of drawing orders between a Begin Custom Pattern drawing order and an End Custom Pattern drawing order. Custom patterns can be either bilevel custom patterns or full-color custom patterns. Contrast with patterns in the *default pattern set*.

custom pattern mode. In GOCA, a mode that is entered when a Begin Custom Pattern drawing order is executed and exited when an End Custom Pattern drawing order is executed. While in this mode, drawing is done in a separate, temporary graphics presentation space rather than in the graphics presentation space of the current GOCA object.

cut-sheet media. Unconnected sheets. Contrast with *continuous-form media*.

D

data block. A deprecated term for object area.

data element. A unit of data that is considered indivisible.

data frame. A rectangular division of computer output on microfilm.

data mask. A sequence of bits that can be used to identify boundary alignment bits in image data.

data object. In the IPDS architecture, a presentation-form object that is either specified within a page or overlay or is activated as a resource and later included in a page or overlay via the IDO command. Examples include: PDF single-page objects, Encapsulated PostScript objects, and IO Images. See also *resource* and *data object resource*.

data-object font. (1) In the IPDS architecture, a complete-font resource that is a combination of font components at a particular size, character rotation, and encoding. A data-object font can be used in a manner analogous to a coded font. The following useful combinations can be activated into a data-object font:

- A TrueType/OpenType font, an optional code page, and optional linked TrueType/OpenType objects; activated at a particular size, character rotation, and encoding
- A TrueType/OpenType collection, either an index value or a full font name to identify the desired font within the collection, an optional code page, and optional linked TrueType/OpenType objects; activated at a particular size, character rotation, and encoding

See also *data-object-font component*. (2) In the MO:DCA architecture, a complete non-FOCA font resource object that is analogous to a coded font. Examples of data-object fonts are TrueType fonts and OpenType fonts.

data-object-font component. In the IPDS architecture, a font resource that is either printer resident or is downloaded using object container commands. Data-object-font components are used as components of a data-object font. Examples of data-object-font components include TrueType/OpenType fonts and TrueType/OpenType collections. See also *data-object font*.

data object resource. In the IPDS architecture, an object-container resource or IO-Image resource that is either printer resident or downloaded. Data object resources can be:

- Used to prepare for the presentation of a data object; such as with a Color Management Resource or Resident Color Profile Resource
- Included in a page or overlay via the Include Data Object command; examples include: PDF single-page objects, Encapsulated PostScript objects, and IO Images
- Invoked from within a data object; examples include: PDF Resource objects

See also *data object* and *resource*.

data stream. A continuous stream of data that has a defined format. An example of a defined format is a structured field.

data-stream exception. In the IPDS architecture, a condition that exists when the printer detects an invalid or unsupported command, order, control, or parameter value from the host. Data-stream exceptions are those whose action code is X'01', X'19', or X'1F'. See also *asynchronous exception* and *synchronous exception*.

DBCS. See *double-byte character set*.

decoder. In bar codes, the component of a bar code reading system that receives the signals from the scanner, performs the algorithm to interpret the signals into meaningful data, and provides the interface to other devices. See also *reader* and *scanner*.

default. A value, attribute, or option that is assumed when none has been specified and one is needed to continue processing. See also *default drawing attributes* and *default drawing controls*.

default drawing attributes. The set of drawing attributes adopted at the beginning of a drawing process and usually at the beginning of each root segment that is processed. See also *root segment*. Contrast with *current drawing attributes*.

default drawing controls. The set of drawing controls adopted at the start of a drawing process and usually at the start of each root segment that is processed. See also *root segment*. Contrast with *current drawing controls*.

default indicator. A field whose bits are all B'1' indicating that a hierarchical default value is to be used. The value can be specified by an external parameter. See also *external parameter*.

default pattern set. In GOCA, a set of predefined patterns, like solid, dots, or horizontal lines. Contrast with *custom pattern*.

density. The number of characters per inch (cpi) in a bar code symbology. In most cases, the range is three to ten cpi. See also *bar code density*, *character density*, and *information density*.

deprecated. An architected construct is marked as "deprecated" to indicate that it should no longer be used because it has been superseded by a newer construct. Use or support of a deprecated construct is permitted but no longer recommended. Constructs are deprecated rather than immediately removed to provide backward compatibility.

descender. The part of the character that extends into the character coordinate system negative Y-axis region. Examples of letters with descenders at zero-degree character rotation are *g*, *j*, *p*, *q*, *y*, and *Q*. Contrast with *ascender*.

descender depth. The character shape's most negative character coordinate system Y-axis value.

design metrics. A set of quantitative values, recommended by a font designer, to describe the characters in a font.

design size. The size of the unit Em for a font. All relative font measurement values are expressed as a proportion of the design size. For example, the width of the letter *I* can be specified as one-fourth of the design size.

Device-Control command set. In the IPDS architecture, a collection of commands used to set up a page, communicate device controls, and manage printer acknowledgment protocol.

device dependent. Dependent upon one or more device characteristics. An example of device dependency is a font whose characteristics are specified in terms of addressable positions of specific devices. See also *system-level font resource*.

device independent. Not dependent upon device characteristics.

device-independent color space. A CIE-based color space that allows color to be expressed in a device-independent way. It ensures colors to be predictably and accurately matched among various color devices.

device level font resource. A device-specific font object from which a presentation device can obtain the font information required to present character images.

device profile. A structure that provides a means of defining the color characteristics of a given device in a particular state.

device resolution. The number of pels that can be printed in an inch, both horizontally and vertically. This is the resolution that the printer uses when printing. Some printers can be configured to print with a variety of resolutions that can be selected by the operator. The device resolution can be different in the two directions (for example, a resolution of 360 by 720).

device-version code page. In the IPDS architecture, a device version of a code page contains all of the characters that were registered for the CPGID at the time the printer was developed; since then, more characters might have been added to the registry for that CPGID. A device-version code page is identified by a CPGID. See also *code page*.

digital half-toning. A method used to simulate gray levels on a bi-level device.

digital image. An image whose image data was sampled at regular intervals to produce a digital representation of the image. The digital representation is usually restricted to a specified set of values.

direction. In GOCA, an attribute that controls the direction in which a character string grows relative to the inline direction. Values are: left-to-right, right-to-left, top-to-bottom, and bottom-to-top. Synonymous with *character direction*.

discrete code. A bar code symbology characterized by placing spaces that are not a part of the code between characters, that is, intercharacter gaps.

DOCS. See *drawing order coordinate space*.

document. (1) A machine-readable collection of one or more objects that represents a composition, a work, or a collection of data. (2) A publication or other written material.

document component. An architected part of a document data stream. Examples of document components are documents, pages, page groups, indexes, resource groups, objects, and process elements.

document content architecture. A family of architectures that define the syntax and semantics of the document component. See also *document component* and *structured field*.

document editing. A method used to create or modify a document.

document element. A self-identifying, variable-length, bounded record, that can have a content portion that provides control information, data, or both. An application or device does not have to understand control information

or data to parse a data stream when all the records in the data stream are document elements. See also *structured field*.

document fidelity. The degree to which a document presentation preserves the creator's intent.

document formatting. A method used to determine where information is positioned in presentation spaces or on physical media.

document presentation. A method used to produce a visible copy of formatted information on physical media.

double-byte character set (DBCS). A character set that can contain up to 65536 characters.

double-byte coded font. A coded font in which the code points are two bytes long.

downloaded resource. In the IPDS architecture, a resource in a printer that is installed and removed under control of a host presentation services program. A downloaded resource is referenced by a host-assigned name that is valid for the duration of the session between the presentation services program and the printer. Contrast with *resident resource*.

drag. To use a pointing device to move an object. For example, clicking on a window border, and dragging it to make the window larger.

draw functions. Functions that can be done during the drawing of a picture. Examples of draw functions are displaying a picture, boundary computation, and erasing a graphics presentation space.

draw rule. A method used to construct a line, called a rule, between two specified presentation positions. The line that is constructed is either parallel to the inline I axis or baseline B axis.

drawing control. A control that determines how a picture is drawn. Examples of drawing controls are arc parameters, transforms, and the viewing window.

drawing order. In GOCA, a graphics construct that the controlling environment builds to instruct a drawing processor about what to draw and how to draw it. The order can specify, for example, that a graphics primitive be drawn, a change to drawing attributes or drawing controls be effected, or a segment be called. One or more graphics primitives can be used to draw a picture. Drawing orders can be included in a structured field. See also *order*.

drawing order coordinate space (DOCS). A two-dimensional conceptual space in which graphics primitives are drawn, using drawing orders, to create pictures.

drawing processor. A graphics processor component that executes segments to draw a picture in a presentation space. See also *segment*, *graphics presentation space*, and *image presentation space*.

drawing units. Units of measurement used within a graphics presentation space to specify absolute and relative positions.

duplex. A method used to print data on both sides of a sheet. Normal-duplex printing occurs when the sheet is turned over the Y_m axis. Tumble-duplex printing occurs when the sheet is turned over the X_m axis.

duplex printing. A method used to print data on both sides of a sheet. Contrast with *simplex printing*.

dynamic segment. A segment whose graphics primitives can be redrawn in different positions by dragging them from one position to the next across a picture without destroying the traversed parts of the picture.

E

EAN. See *European Article Numbering*.

EBCDIC. See *Extended Binary-Coded Decimal Interchange Code*.

Efficient XML Interchange (EXI). A format that allows XML documents to be encoded as binary data, rather than as plain text.

element. (1) A bar or space in a bar code character or a bar code symbol. (2) A structured field in a document content architecture data stream. (3) In GOCA, a portion of a segment consisting of either a single order or a group of orders enclosed in an element bracket, in other words, between a *begin* element and an *end* element. (4) A basic member of a mathematical or logical class or set.

Em. In printing, a unit of linear measure referring to the baseline-to-baseline distance of a font, in the absence of any external leading.

Em square. A square layout space used for designing each of the characters of a font.

encoding scheme. A set of specific definitions that describe the philosophy used to represent character data. The number of bits, the number of bytes, the allowable ranges of bytes, the maximum number of characters, and the meanings assigned to some generic and specific bit patterns, are some examples of specifications to be found in such a definition.

Encoding Scheme Identifier (ESID). A 16-bit number assigned to uniquely identify a particular encoding scheme specification. See also *encoding scheme*.

environment interface. The part of the graphics processor that interprets commands and instructions from the controlling environment.

escape sequence. (1) In the IPDS architecture, the first two bytes of a control sequence. An example of an escape sequence is X'2BD3'. (2) A string of bit combinations that is used for control in code extension procedures. The first of these bit combinations represents the control function Escape.

escapement direction. In FOCA, the direction from a character reference point to the character escapement point, that is, the font designer's intended direction for successive character shapes. See also *character direction* and *inline direction*.

ESID. See *Encoding Scheme Identifier*.

established baseline coordinate. The current baseline presentation coordinate when no temporary baseline exists or the last current baseline presentation coordinate that existed before the first active temporary baseline was created. If temporary baselines are created, the current baseline presentation coordinate coincides with the presentation coordinate of the most recently created temporary baseline.

European Article Numbering (EAN). The bar code symbology used to code grocery items in Europe.

exception. (1) An invalid or unsupported data-stream construct. (2) In the IPDS architecture, a condition requiring host notification. (3) In the IPDS architecture, a condition that requires the host to resend data. See also *data-stream exception*, *asynchronous exception*, and *synchronous exception*.

exception action. Action taken when an exception is detected.

exception condition. The condition that exists when a product finds an invalid or unsupported construct.

exchange. The predictable interpretation of shared information by a family of system processes in an environment where the characteristics of each process must be known to all other processes. Contrast with *interchange*.

EXI. See *Efficient XML Interchange*.

expanded. A type width that widens all characters of a typeface.

Extended Binary-Coded Decimal Interchange Code (EBCDIC). A coded character set that consists of eight-bit coded characters.

Extensible Markup Language (XML). A set of rules for encoding documents in a format that is both human-readable and machine-readable.

Extensible Metadata Platform (XMP). An ISO standard, originally created by Adobe Systems Incorporated, for the creation, processing, and interchange of standardized and custom metadata for all kinds of resources.

external leading. The amount of white space, in addition to the internal leading, that can be added to interline spacing without degrading the aesthetic appearance of a font. This value is usually specified by a font designer. Contrast with *internal leading*.

external parameter. A parameter for which the current value can be provided by the controlling environment, for example, the data stream, or by the application itself. Contrast with *internal parameter*.

F

factoring. The movement of a parameter value from one state to a higher-level state. This permits the parameter value to apply to all of the lower-level states unless specifically overridden at the lower level.

FGID. See *Font Typeface Global Identifier*.

Filename Map File. A file containing the mapping of object names to file names for use in establishing a font file system. Object names and file names do not conform to the same naming requirements, so it is necessary to provide a mapping between them. The mapping information in this file is in an ASCII file format defined by Adobe Systems Inc.

fillet. A curved line drawn tangential to a specified set of straight lines. An example of a fillet is the concave junction formed where two lines meet.

final form data. Data that has been formatted for presentation.

first read rate. In bar codes, the ratio of the number of successful reads on the first attempt to the total number of attempts made to obtain a successful read. Synonymous with *read rate*.

fixed medium information. Information that can be applied to a sheet by a printer or printer-attached device that is independent of data provided through the data stream. Fixed medium information does not mix with the data provided by the data stream and is presented on a sheet either before or after the text, image, graphics, or bar code data provided within the data stream. Fixed medium information can be used to create preprinted forms, or other types of printing, such as colored logos or letterheads, that cannot be created conveniently within the data stream.

fixed metrics. Graphic character measurements in physical units such as pels, inches, or centimeters.

FNN Linked. In FOCA, the FNN (Font Name map) structured field permits the mapping of a set of IBM GCGIDs to the character index values which occur in either

a CMAP file or a Rearranged file. Because the set of GCGIDs and the set of character index values must correspond to the same set of characters, it is necessary to identify which CMAP or Rearranged file (among the many that could be located in a font file system) is associated (linked) with the FNN structured field. Note that the Font Name Map is known as the Character ID Map in IPDS.

FOCA. See *Font Object Content Architecture*.

font. A set of graphic characters that have a characteristic design, or a font designer's concept of how the graphic characters should appear. The characteristic design specifies the characteristics of its graphic characters. Examples of characteristics are character shape, graphic pattern, style, size, weight class, and increment. Examples of fonts are fully described fonts, symbol sets, and their internal printer representations. See also *coded font* and *symbol set*.

font baseline extent. In the IPDS architecture, the sum of the uniform or maximum baseline offset and the maximum baseline descender of all characters in the font.

font character set. A FOCA resource containing descriptive information, font metrics, and the digital representation of character shapes for a specified graphic character set.

font control record. The record sent in an IPDS Load Font Control command to specify a font ID and other font parameters that apply to the complete font.

font height (FH). (1) A characteristic value, perpendicular to the character baseline, that represents the size of all graphic characters in a font. Synonymous with *vertical font size*. (2) In a font character set, nominal font height is a font-designer defined value corresponding to the nominal distance between adjacent baselines when character rotation is zero degrees and no external leading is used. This distance represents the baseline-to-baseline increment that includes the font's maximum baseline extent and the designer's recommendation for internal leading. The font designer can also define a minimum and a maximum vertical font size to represent the limits of scaling. (3) In font referencing, the specified font height is the desired size of the font when the characters are presented. If this size is different from the nominal vertical font size specified in a font character set, the character shapes and character metrics might need to be scaled prior to presentation.

font index. (1) The mapping of a descriptive font name to a font member name in a font library. An example of a font member in a font library is a font resource object. Examples of attributes used to form a descriptive font name are typeface, family name, point size, style, weight class, and width class. (2) In the IPDS architecture, an LF1-type raster-font resource containing character metrics for each code point of a raster font or raster-font section for a particular font inline sequence. There can be a font index for 0 degree, 90 degree, 180 degree, and 270 degree font inline sequences. A font index can be downloaded to a printer using the Load Font Index command. An LF1-type coded font or coded-font section is the combination of one fully described font and one font index. See also *fully described font*.

font inline sequence. The clockwise rotation of the inline direction relative to a character pattern. Character rotation and font inline sequence are related in that character rotation is a clockwise rotation; font inline sequence is a counter-clockwise rotation.

font local identifier. A binary identifier that is mapped by the controlling environment to a named resource to identify a font. See also *local identifier*.

font metrics. Measurement information that defines individual character values such as height, width, and space, as well as overall font values such as averages and maximums. Font metrics can be expressed in specific fixed units, such as pels, or in relative units that are independent of both the resolution and the size of the font. See also *character metrics* and *character set metrics*.

font modification parameters. Parameters that alter the appearance of a typeface.

font object. A resource object that contains some or all of the description of a font.

Font Object Content Architecture (FOCA). An architected collection of constructs used to describe fonts and to interchange those font descriptions.

font production. A method used to create a font. This method includes designing each character image, converting the character images to a digital-technology format, defining parameter values for each character, assigning appropriate descriptive and identifying information, and creating a font resource that contains the required information in a format that can be used by a text processing system. Digital-technology formats include bit image, vector drawing orders, and outline algorithms. Parameter values include such attributes as height, width, and escapement.

font referencing. A method used to identify or characterize a font. Examples of processes that use font referencing are document editing, document formatting, and document presentation.

Font Typeface Global Identifier (FGID). A unique font identifier that can be expressed as either a two-byte binary or a five-digit decimal value. The FGID is used to identify a type style and the following characteristics: posture, weight class, and width class.

font width (FW). (1) A characteristic value, parallel to the character baseline, that represents the size of all graphic characters in a font. Synonymous with *horizontal font size*. (2) In a font character set, nominal font width is a font-designer defined value corresponding to the nominal

character increment for a font character set. The value is generally the width of the space character and is defined differently for fonts with different spacing characteristics.

- For fixed-pitch, uniform character increment fonts: the fixed character increment, that is also the space character increment
- For PSM fonts: the width of the space character
- For typographic, proportionally spaced fonts: one-third of the vertical font size, that is also the default size of the space character.

The font designer can also define a minimum and a maximum horizontal font size to represent the limits of scaling. (3) In font referencing, the specified font width is the desired size of the font when the characters are presented. If this size is different from the nominal horizontal font size specified in a font character set, the character shapes and character metrics might need to be scaled prior to presentation.

foreground. (1) The part of a presentation space that is occupied by object data. (2) In GOCA, the portion of a graphics primitive that is mixed into the presentation space under the control of the current value of the mix and color attributes. See also *pel*. Contrast with *background*.

foreground color. A color attribute used to specify the color of the foreground of a primitive. Contrast with *background color*.

foreground mix. An attribute used to determine how the foreground color of data is combined with the existing color of a graphics presentation space. An example of data is a graphics primitive. Contrast with *background mix*.

form. A division of the physical medium; multiple forms can exist on a physical medium. For example, a roll of paper might be divided by a printer into rectangular pieces of paper, each representing a form. Envelopes are an example of a physical medium that comprises only one form. The IPDS architecture defines four types of forms: cut-sheet media, continuous-form media, envelopes, and computer output on microfilm. Each type of form has a top edge. A form has two sides, a front side and a back side. Synonymous with *sheet*.

format. The arrangement or layout of data on a physical medium or in a presentation space.

formatter. A process used to prepare a document for presentation.

formblend. This mixing rule is only used when a preprinted form overlay (PFO) is merged as presentation space P_{PFO} with other presentation data (presentation space P_{data}). The intersection of P_{PFO} and P_{data} is assigned the following color attribute:

- Wherever the color attribute of P_{PFO} is either color of medium, or "white" (CMYK = X'00000000' for a printer, RGB = X'FFFFFF' for an RGB display), the intersection is assigned the color attribute of P_{data}. Likewise, wherever the color attribute of P_{data} is either color of medium, or "white" (CMYK = X'00000000' for a printer, RGB = X'FFFFFF' for an RGB display), the intersection is assigned the color attribute of P_{PFO}.
- With other overlapping color values, the intersection assumes a new color attribute that is generated in a device-specific manner to simulate how the P_{data} color attribute would mix onto a preprinted form that has the color attribute of P_{PFO}. In general, this mixing is a blending of the color attributes of P_{data} and P_{PFO} that is determined by the two color attributes and by the print media and the print technology.

See also *mixing rule*.

Formdef. See *Form Definition*.

Form Definition (Formdef). A print control object that contains an environment definition and one or more Medium Maps. Synonymous with *Form map*.

Form Map. A print control object that contains an environment definition and one or more Medium Maps. Synonymous with *Form Definition*. See also *Medium Map*.

full arc. A complete circle or ellipse. See also *arc*.

full-color custom pattern. In GOCA, a custom pattern that has its colors completely assigned during its definition, and can therefore contain any number of colors. Contrast with *bilevel custom pattern*.

fully described font. In the IPDS architecture, an LF1-type raster-font resource containing font metrics, descriptive information, and the raster representation of character shapes, for a specific graphic character set. A fully described font can be downloaded to a printer using the Load Font Control and Load Font commands. An LF1-type coded font or coded-font section is the combination of one fully described font and one font index. See also *font index*.

function set. A collection of architecture constructs and associated values. Function sets can be defined across or within subsets.

FW. See *font width*.

G

GCGID. See *Graphic Character Global Identifier*.

GCSGID. See *Graphic Character Set Global Identifier*.

GCUID. See *Graphic Character UCS Identifier*.

GID. See *global identifier*.

given position. The coordinate position at which drawing is to begin. A given position is specified in a drawing order. Contrast with *current position*.

Global Identifier (GID). Any of the following:

- Code Page Global ID (CPGID)
- Graphic Character Global Identifier (GCGID)
- Graphic Character UCS Identifier (GCUID)
- Font Typeface Global Identifier (FGID)
- Graphic Character Set Global Identifier (GCSGID)
- Coded Graphic Character Set Global Identifier (CGCSGID)
- In MO:DCA, an encoded graphic character string that provides a reference name for a document element.
- Global Resource Identifier (GRID)
- Object identifier (OID)
- Coded Character Set Identifier (CCSID).

global resource identifier (GRID). An eight-byte identifier that identifies a coded font resource. A GRID contains the following fields in the order shown:

1. GCSGID of a minimum set of graphic characters required for presentation. It can be a character set that is associated with the code page, or with the font character set, or with both.
2. CPGID of the associated code page
3. FGID of the associated font character set
4. Font width in 1440ths of an inch.

glyph. A member of a set of symbols that represent data. Glyphs can be letters, digits, punctuation marks, or other symbols. Synonymous with *graphic character*. See also *character*.

GOCA. See *Graphics Object Content Architecture*.

grapheme. (1) A minimally distinctive unit of writing in the context of a particular writing system. For example, å ("a + Combining Ring Above" or "Latin Small Letter A with Ring Above") is a grapheme in the Danish writing system. (2) What an end-user thinks of as a character.

graphic character. A member of a set of symbols that represent data. Graphic characters can be letters, digits, punctuation marks, or other symbols. Synonymous with *glyph*. See also *character*.

Graphic Character Global Identifier (GCGID). An alphanumeric character string used to identify a specific graphic character. A GCGID can be from four-bytes to eight-bytes long.

graphic character identifier. The unique name for a graphic character in a font or in a graphic character set. See also *character identifier*.

Graphic Character Set Global Identifier (GCSGID). A unique graphic character set identifier that can be expressed as either a two-byte binary or a five-digit decimal value.

Graphic Character UCS Identifier (GCUID). An alphanumeric character string used to identify a specific graphic character. The GCUID naming scheme is used for additional characters and sets of characters that exist in UNICODE; each GCUID begins with the letter *U* and ends with a UNICODE code point. The Unicode Standard is fully compatible with the earlier Universal Character Set (UCS) Standard.

Graphics command set. In the IPDS architecture, a collection of commands used to present GOCA data in a page, page segment, or overlay.

graphics data. Data containing lines, arcs, markers, and other constructs that describe a picture.

graphics model space. A two-dimensional conceptual space in which a picture is constructed. All model transforms are completed before a picture is constructed in a graphics model space. Contrast with *graphics presentation space*. Synonymous with *model space*.

graphics object. An object that contains graphics data. See also *object*.

graphics object area. A rectangular area on a logical page into which a graphics presentation space window is mapped.

Graphics Object Content Architecture (GOCA). An architected collection of constructs used to interchange and present graphics data.

graphics presentation space. A two-dimensional conceptual space in which a picture is constructed. In this space graphics drawing orders are defined. The picture can then be mapped onto an output medium. All viewing transforms are completed before the picture is generated for presentation on an output medium. An example of a graphics presentation space is the abstract space containing graphics pictures defined in an IPDS Write Graphics Control command. Contrast with *graphics model space*.

graphics presentation space window. The portion of a graphics presentation space that can be mapped to a graphics object area on a logical page.

graphics primitive. A basic construct used by an output device to draw a picture. Examples of graphics primitives are arc, line, fillet, character string, and marker.

graphics processor. The processing capability required to interpret a GOCA object, that is, to present the picture represented by the object. It includes the environment interface, that interprets commands and instructions, and the drawing processor, that interprets the drawing orders.

graphics segment. A set of graphics drawing orders contained within a Begin Segment command. See also *segment*.

grayscale image. Images whose image data elements are represented by multiple bits and whose image data element values are mapped to more than one level of brightness through an image data element structure parameter or a look-up table.

GRID. See *global resource identifier*.

guard bars. The bars at both ends and the center of an EAN, JAN, or UPC symbol, that provide reference points for scanning.

GZIP. A widely-used, free software compression algorithm.

H

HAID. See *Host-Assigned ID*.

height. In bar codes, the bar dimension perpendicular to the bar width. Synonymous with *bar height* and *bar length*.

hexadecimal. A number system with a base of sixteen. The decimal digits 0 through 9 and characters A through F are used to represent hexadecimal digits. The hexadecimal digits A through F correspond to the decimal numbers 10 through 15, respectively. An example of a hexadecimal number is X'1B', that is equal to the decimal number 27.

highlight color. A spot color that is used to accentuate or contrast monochromatic areas. See also *spot color*.

highlighting. The emphasis of displayed or printed information. Examples are increased intensity of selected characters on a display screen and exception highlighting on an IPDS printer.

hollow font. A font design in which the graphic character shapes include only the outer edges of the strokes.

home state. An initial IPDS operating state. A printer returns to home state at the end of each page, and after downloading a font, overlay, or page segment.

horizontal bar code. A bar code pattern presenting the axis of the symbol in its length dimension parallel to the X_{bc} axis of the bar code presentation space. Synonymous with *picket fence bar code*.

horizontal font size. (1) A characteristic value, parallel to the character baseline, that represents the size of all graphic characters in a font. Synonymous with *font width*. (2) In a font character set, nominal horizontal font size is a font-designer defined value corresponding to the nominal character increment for a font character set. The value is generally the width of the space character and is defined differently for fonts with different spacing characteristics.

- For fixed-pitch, uniform character increment fonts: the fixed character increment, that is also the space character increment
- For PSM fonts: the width of the space character
- For typographic fonts and proportionally spaced fonts: one-third of the vertical font size, that is also the default size of the space character.

The font designer can also define a minimum and a maximum horizontal font size to represent the limits of scaling. (3) In font referencing, the specified horizontal font size is the desired size of the font when the characters are presented. If this size is different from the nominal horizontal font size specified in a font character set, the character shapes and character metrics might need to be scaled prior to presentation.

horizontal scale factor. (1) In outline-font referencing, the specified horizontal adjustment of the Em square. The horizontal scale factor is specified in 1440ths of an inch. When the horizontal and vertical scale factors are different, anamorphic scaling occurs. See also *vertical scale factor*. (2) In FOCA, the numerator of a scaling ratio, determined by dividing the horizontal scale factor by the vertical font size. If the value specified is greater or less than the specified vertical font size, the graphic characters and their corresponding metric values are stretched or compressed in the horizontal direction relative to the vertical direction by the scaling ratio indicated.

host. (1) In the IPDS architecture, a computer that drives a printer. (2) In IOCA, the host is the controlling environment.

Host-Assigned ID (HAID). A two-byte ID in the range X'0001'–X'7EFF' that is assigned to an IPDS resource by a presentation-services program in the host. This ID uniquely identifies a resource until that resource is deactivated, in which case the HAID can be reused. HAIDs are used in IPDS resource management commands.

Host-Assigned Resource ID. The combination of a Host-Assigned ID with a section identifier, or a font inline sequence, or both. The section identifier and font inline sequence values are ignored for both page segments and overlays. See also *section identifier* and *font inline sequence*.

HRI. See *human-readable interpretation*.

human-readable interpretation (HRI). The printed translation of bar code characters into equivalent Latin alphabetic characters, Arabic numeral decimal digits, and common special characters normally used for printed human communication.

hypermedia. Interlinked pieces of information consisting of a variety of data types such as text, graphics data, image, audio, and video.

hypertext. Interlinked pieces of information consisting primarily of text.

I

i_c. See *current inline print coordinate*.

i_i. See *initial inline print coordinate*.

I. See *inline direction*.

+I. Positive inline direction.

I_c. See *current inline presentation coordinate*.

I_o. See *inline presentation origin*.

I axis. The axis of an I,B coordinate system that extends in the inline direction. The I axis does not have to be parallel to the X_p axis of its bounding X_p,Y_p coordinate space.

I,B coordinate system. The coordinate system used to present graphic characters. This coordinate system is used to establish the inline direction and baseline direction for the placement of successive graphic characters within a presentation space. See also *X_p, Y_p coordinate system*.

ICC. See International Color Consortium.

ICC profile. A file in the International Color Consortium profile format, containing information about the color reproduction capabilities of a device such as a scanner, a digital camera, a monitor, or a printer. An ICC profile includes three elements: 128-byte file header, tag table, and tagged element data. The intent of this format is to provide a cross-platform device profile format. Such device profiles can be used to translate color data created on one device into another device's native color space.

ID. Identifier. See also *Host-Assigned ID (HAID)*, *correlation ID*, *font control record*, and *overlay ID*.

IDE. See *image data element*.

I-direction. (1) The direction in which successive characters appear in a line of text. (2) In GOCA, the direction specified by the character angle attribute. Synonymous with *inline direction*.

IDP. See *image data parameter*.

IEEE. Institute of Electrical and Electronics Engineers.

I-extent. The X_p-extent when the I axis is parallel to the X_p axis or the Y_p-extent when the I axis is parallel to the Y_p axis. The definition of the I-extent depends on the X_p- or Y_p-extent because the I,B coordinate system is contained within an X_p,Y_p coordinate system.

image. An electronic representation of a picture produced by means of sensing light, sound, electron radiation, or other emanations coming from the picture or reflected by the picture. An image can also be generated directly by software without reference to an existing picture.

image content. Image data and its associated image data parameters.

image coordinate system. An X,Y Cartesian coordinate system using only the fourth quadrant with positive values for the Y axis. The origin of an image coordinate system is its upper left hand corner. An X,Y coordinate specifies a presentation position that corresponds to one and only one image data element in the image content.

image data. Rectangular arrays of raster information that define an image.

image data element (IDE). A basic unit of image information. An image data element expresses the intensity of a signal at a corresponding image point. An image data element can use a look-up table to introduce a level of indirection into the expression of grayscale image or color image.

image data parameter (IDP). A parameter that describes characteristics of image data.

image distortion. Deformation of an image such that the original proportions of the image are changed and the original balance and symmetry of the image are lost.

image object. An object that contains image data. See also *object*.

image object area. A rectangular area on a logical page into which an image presentation space is mapped.

Image Object Content Architecture (IOCA). An architected collection of constructs used to interchange and present images.

image point. A discrete X,Y coordinate in the image presentation space. See also *addressable position*.

image presentation space (IPS). A two-dimensional conceptual space in which an image is generated.

image segment. Image content bracketed by Begin Segment and End Segment self-defining fields. See also *segment*.

IM Image. A migration image object that is resolution dependent, bi-level, and cannot be compressed or scaled. Contrast with *IO Image*.

IM-Image command set. In the IPDS architecture, a collection of commands used to present IM-Image data in a page, page segment, or overlay.

immediate mode. The mode in which segments are executed as they are received and then discarded. Contrast with *store mode*.

indexed object. An object in a MO:DCA document that is referenced by an Index Element structured field in a MO:DCA index. Examples of indexed objects are pages and page groups.

information density. The number of characters per inch (cpi) in a bar code symbology. In most cases, the range is three to ten cpi. See also *bar code density*, *character density*, and *density*.

initial addressable position. The values assigned to I_c and B_c by the data stream at the start of object state. The standard action values are I_o and B_o.

initial baseline print coordinate (b_i). The baseline coordinate of the first print position on a logical page. See also *initial inline print coordinate*.

initial inline print coordinate (i_i). The inline coordinate of the first print position on a logical page. See also *initial baseline print coordinate*.

inline-baseline coordinate system. See *I,B coordinate system*.

inline coordinate. The first of a pair of values that identifies the position of an addressable position with respect to the origin of a specified I,B coordinate system. This value is specified as a distance in addressable positions from the B axis of an I,B coordinate system.

inline direction (I). (1) The direction in which successive characters appear in a line of text. (2) In GOCA, the direction specified by the character angle attribute. Synonymous with *I-direction*.

inline margin. The inline coordinate that identifies the initial addressable position for a line of text.

inline presentation origin (I_o). The point on the I axis where the value of the inline coordinate is zero.

input profile. An ICC profile that is associated with the image and describes the characteristics of the device on which the image was created.

Intelligent Printer Data Stream (IPDS). An architected host-to-printer data stream that contains both data and controls defining how the data is to be presented.

interchange. The predictable interpretation of shared information in an environment where the characteristics of each process need not be known to all other processes. Contrast with *exchange*.

intercharacter adjustment. Additional distance applied to a character increment that increases or decreases the distance between presentation positions, effectively modifying the amount of white space between graphic characters. The amount of white space between graphic characters is changed to spread the characters of a word for emphasis, distribute excess white space on a line among the words of that line to achieve right justification, or move the characters on the line closer together as in kerning. Examples of intercharacter adjustment are intercharacter increment and intercharacter decrement.

intercharacter decrement. Intercharacter adjustment applied in the negative I-direction from the current presentation position. See also *intercharacter adjustment*.

intercharacter gap. In bar codes, the space between two adjacent bar code characters in a discrete code, for example, the space between two characters in Code 39. Synonymous with *intercharacter space*. Contrast with *clear area*, *element*, and *space*.

intercharacter increment. Intercharacter adjustment applied in the positive I-direction from the current presentation position. See also *intercharacter adjustment*.

intercharacter space. In bar codes, the space between two adjacent bar code characters in a discrete code, for example, the space between two characters in Code 39. Synonymous with *intercharacter gap*. Contrast with *element* and *space*.

interleaved bar code. A bar code symbology in which characters are paired, using bars to represent the first character and spaces to represent the second. An example is Interleaved 2 of 5.

intermediate device. In the IPDS architecture, a device that operates on the data stream and is situated between a printer and a presentation services program in the host. Examples include devices that capture and cache resources and devices that spool the data stream.

internal leading. A font design parameter referring to the space provided between lines of type to keep ascenders separated from descenders and to provide an aesthetically pleasing interline spacing. The value of this parameter usually equals the difference between the vertical font size and the font baseline extent. Contrast with *external leading*.

internal parameter. In PTOCA, a parameter whose current value is contained within the object. Contrast with *external parameter*.

International Color Consortium (ICC). A group of companies chartered to develop, use, and promote cross-platform standards so that applications and devices can exchange color data without ambiguity.

International Organization for Standardization (ISO). An organization of national standards bodies from various countries established to promote development of standards to facilitate international exchange of goods and

services, and develop cooperation in intellectual, scientific, technological, and economic activity.

interoperability. The capability to communicate, execute programs, or transfer data among various functional units in a way that requires the user to have little or no knowledge of the unique characteristics of those units.

introducer. In GOCA, that part of the data stream passed from a controlling environment to a communication processor that indicates whether entities are to be processed in immediate mode or store mode. See also *immediate mode* and *store mode*.

IOCA. See *Image Object Content Architecture*.

IO Image. An image object containing IOCA constructs. Contrast with *IM Image*.

IO-Image command set. In the IPDS architecture, a collection of commands used to present IOCA data in a page, page segment, or overlay.

IPDS. See *Intelligent Printer Data Stream*.

IPDS dialog. A series of IPDS commands and IPDS acknowledge replies. An IPDS dialog begins with the first IPDS command that an IPDS device receives and ends either when an IPDS command explicitly ends the dialog or when the carrying-protocol session ends. There can be multiple independent sessions each with an IPDS dialog. See also *session*.

IPS. See *image presentation space*.

ISO. See *International Organization for Standardization*.

italics. A typeface with characters that slant upward to the right. In FOCA, italics is the common name for the defined inclined typeface posture attribute or parameter.

J

JAN. See *Japanese Article Numbering*.

Japanese Article Numbering (JAN). The bar code symbology used to code grocery items in Japan.

jog. To cause printed sheets to be stacked in an output stacker offset from previously stacked sheets. Jogging is requested by using an IPDS Execute Order Anystate Alternate Offset Stacker command.

K

Kanji. A graphic character set for symbols used in Japanese ideographic alphabets.

kerning. The design of graphic characters so that their character boxes overlap, resulting in the reduction of space between characters. This allows characters to be designed for cursive languages, ligatures, and proportionally spaced fonts. An example of kerning is the printing of adjacent graphic characters so they overlap on the left or right side.

kerning track. A straight-line graph that associates vertical font size with white space adjustment. The result of this association is used to scale fonts.

kerning track intercept. The X-intercept of a kerning track for a given vertical font size or white space adjustment value.

kerning track slope. The slope of a kerning track.

keyword. A two-part self-defining parameter consisting of a one-byte identifier and a one-byte value.

L

ladder bar code. A bar code pattern presenting the axis of the symbol in its length dimension parallel to the Y_{bc} axis of the bar code presentation space. Synonymous with *vertical bar code*.

LAN. See *local area network*.

landscape. A presentation orientation in which the X_m axis is parallel to the long sides of a rectangular physical medium. Contrast with *portrait*.

language. A set of symbols, conventions, and rules that is used for conveying information. See also *pragmatics*, *semantics*, and *syntax*.

LCID. See *Local Character Set Identifier*.

leading. A printer's term for the amount of space between lines of a printed page. Leading refers to the lead slug placed between lines of type in traditional typesetting. See also *internal leading* and *external leading*.

leading edge. (1) The edge of a character box that in the inline direction precedes the graphic character. (2) The front edge of a sheet as it moves through a printer.

legibility. Characteristics of presented characters that affect how rapidly, easily, and accurately one character can be distinguished from another. The greater the speed, ease, and accuracy of perception, the more legible the presented characters. Examples of characteristics that affect legibility are character shape, spacing, and composition.

LID. See *local identifier*.

ligature. A single glyph representing two or more characters. Examples of characters that can be presented as ligatures are *ff* and *ffi*.

line attributes. Those attributes that pertain to straight and curved lines. Examples of line attributes are line type and line width.

line type. A line attribute that controls the appearance of a line. The line type can either be a standard line type value or a custom line type value. Contrast with *line width*.

line width. A line attribute that controls the appearance of a line. Examples of line width are normal and thick. Contrast with *line type*.

link. A logical connection from a source document component to a target document component.

Loaded-Font command set. In the IPDS architecture, a collection of commands used to load font information into a printer and to deactivate font resources.

local area network (LAN). A data network located on a user's premises in which serial transmission is used for direct data communication among data stations.

Local Character Set Identifier (LCID). A local identifier used as a character, marker, or pattern set attribute.

local identifier (LID). An identifier that is mapped by the controlling environment to a named resource.

location. A site within a data stream. A location is specified in terms of an offset in the number of structured fields from the beginning of a data stream, or in the number of bytes from another location within the data stream.

logical page. A presentation space. One or more object areas can be mapped to a logical page. A logical page has specifiable characteristics, such as size, shape, orientation, and offset. The shape of a logical page is the shape of a rectangle. Orientation and offset are specified relative to a medium coordinate system.

logical unit. A unit of linear measurement expressed with a unit base and units per unit-base value. For example, in MO:DCA and IPDS architectures, the following logical units are used:

- 1 logical unit = 1/1440 inch
  (unit base = 10 inches,
  units per unit base = 14,400)
- 1 logical unit = 1/240 inch
  (unit base = 10 inches,
  units per unit base = 2400)

Synonymous with *L-unit*.

look-up table (LUT). A logical list of colors or intensities. The list has a name and can be referenced to select a color or intensity. See also *color table*.

lossless. A form of image transformation in which all of the data is retained. Contrast with *lossy*.

lossy. A form of image transformation in which some of the data is lost. Contrast with *lossless*.

lowercase. Pertaining to small letters as distinguished from capital letters. Examples of small letters are *a*, *b*, and *g*. Contrast with *uppercase*.

L-unit. A unit of linear measurement expressed with a unit base and units per unit-base value. For example, in MO:DCA and IPDS architectures, the following L-units are used:

- 1 L-unit = 1/1440 inch
  (unit base = 10 inches,
  units per unit base = 14,400)
- 1 L-unit = 1/240 inch
  (unit base = 10 inches,
  units per unit base = 2400)

Synonymous with *logical unit*.

LUT. See *look-up table*.

M

M/O. A table heading for architecture syntax. The entries under this heading indicate whether the construct is mandatory (M) or optional (O).

magnetic ink character recognition (MICR). Recognition of characters printed with ink that contains particles of a magnetic material.

mainframe interactive (MFI). Pertaining to systems in which nonprogrammable terminals are connected to a mainframe.

mandatory support level. Within the base-and-towers concept, the smallest portion of architected function that is allowed to be implemented. This is represented by a base with no towers. Synonymous with *base support level*.

marker. A symbol with a recognizable appearance that is used to identify a particular location. An example of a marker is a symbol that is positioned by the center point of its cell.

marker attributes. The characteristics that control the appearance of a marker. Examples of marker attributes are size and color.

marker cell. A conceptual rectangular box that can include a marker symbol and the space surrounding that symbol.

marker precision. A method used to specify the degree of influence that marker attributes have on the appearance of a marker.

marker set. In GOCA, an attribute used to access a coded font.

marker symbol. A symbol that is used for a marker.

maximum ascender height. The maximum of the individual character ascender heights. A value for maximum ascender height is specified for each supported character rotation. Contrast with *maximum descender depth*.

maximum baseline extent. In FOCA, the sum of the maximum of the individual character baseline offsets and the maximum of the individual character descender depths, for a given font.

maximum descender depth. The maximum of the individual character descender depths. A value for maximum descender depth is specified for each supported character rotation. Contrast with *maximum ascender height*.

meaning. A table heading for architecture syntax. The entries under this heading convey the meaning or purpose of a construct. A meaning entry can be a long name, a description, or a brief statement of function.

measurement base. A base unit of measure from which other units of measure are derived.

media. Plural of medium. See also *medium*.

media destination. The destination to which sheets are sent as the last step in the print process. Some printers support several media destinations to allow options such as print job distribution to one or more specific destinations, collated copies without having to resend the document to the printer multiple times, and routing output to a specific destination for security reasons. Contrast with *media source*.

media source. The source from which sheets are obtained for printing. Some printers support several media sources so that media with different characteristics (such as size, color, and type) can be selected when desired. Contrast with *media destination*.

medium. A two-dimensional conceptual space with a base coordinate system from which all other coordinate systems are either directly or indirectly derived. A medium is mapped onto a physical medium in a device-dependent manner. Synonymous with *medium presentation space*. See also *logical page*, *physical medium*, and *presentation space*.

Medium Map. A print control object in a Form Map that defines resource mappings and controls modifications to a form, page placement on a form, and form copy generation. See also *Form Map*.

medium presentation space. A two-dimensional conceptual space with a base coordinate system from which all other coordinate systems are either directly or indirectly derived. A medium presentation space is mapped onto a physical medium in a device-dependent manner. Synonymous with *medium*. See also *logical page*, *physical medium*, and *presentation space*.

metadata. Descriptive information that is associated with and augments other data.

metadata object. In AFP, the resource object that carries metadata.

Metadata Object Content Architecture (MOCA). A resource object architecture to carry metadata that serves to provide context or additional information about an AFP object or other AFP data.

MFI. See *mainframe interactive*.

MICR. See *magnetic ink character recognition*.

Microfilm frame. A rectangular area on the microfilm bounded by imaginary, intersecting grid lines within which a data frame may be recorded. The grid lines are part of gauges used for checking microfilm, but they do not actually appear on the microfilm.

mil. 1/1000 inch.

mix. A method used to determine how the color of a graphics primitive is combined with the existing color of a graphics presentation space. See also *foreground mix* and *background mix*.

Mixed Object Document Content Architecture (MO:DCA). An architected, device-independent data stream for interchanging documents.

mixing. (1) Combining foreground and background of one presentation space with foreground and background of another presentation space in areas where the presentation spaces intersect. (2) Combining foreground and background of multiple intersecting object data elements in the object presentation space.

mixing rule. A method for specifying the color attributes of the resulting foreground and background in areas where two presentation spaces intersect.

MO:DCA. See *Mixed Object Document Content Architecture*.

MO:DCA IS/1. MO:DCA Interchange Set 1. A subset of MO:DCA that defines an interchange format for presentation documents.

MO:DCA IS/2. MO:DCA Interchange Set 2. A retired subset of MO:DCA that defines an interchange format for presentation documents.

MO:DCA IS/3. MO:DCA Interchange Set 3. A subset of MO:DCA that defines an interchange format for print files that supersedes MO:DCA IS/1.

MO:DCA-L. A MO:DCA subset that defines the OS/2 Presentation Manager (PM) metafile. This format is also known as a .met file. The definition of this MO:DCA subset is stabilized and is no longer being developed as part of the MO:DCA architecture. It is defined in the document *MO:DCA-L: The OS/2 Presentation Manager Metafile (.met) Format*, available at www.afpcinc.org.

MO:DCA-P. A subset of the MO:DCA architecture that defines presentation documents. This term is now synonymous with the term MO:DCA.

MOCA. See *Metadata Object Content Architecture*.

model space. A two-dimensional conceptual space in which a picture is constructed. All model transforms are completed before a picture is constructed in a graphics model space. Contrast with *graphics presentation space*. Synonymous with *graphics model space*.

model transform. A transform that is applied to drawing-order coordinates. Contrast with *viewing transform*.

module. In a bar code symbology, the nominal width of the smallest element of a bar or space. Actual bar code symbology bars and spaces can be a single module wide or some multiple of the module width. The multiple need not be an integer.

modulo-N check. A check in which an operand is divided by a modulus to generate a remainder that is retained and later used for checking. An example of an operand is the sum of a set of digits. See also *modulus*.

modulus. In a modulo check, the number by which an operand is divided. An example of an operand is the sum of a set of digits. See also *modulo-N check*.

monospaced font. A font with graphic characters having a uniform character increment. The distance between reference points of adjacent graphic characters is constant in the escapement direction. The blank space between the graphic characters can vary. Synonymous with *uniformly spaced font*. Contrast with *proportionally spaced font* and *typographic font*.

move order. A drawing order that specifies or implies movement from the current position to a given position. See also *current position* and *given position*.

N

NACK. See *Negative Acknowledge Reply*.

name. A table heading for architecture syntax. The entries under this heading are short names that give a general indication of the contents of the construct.

named color. A color that is specified with a descriptive name. An example of a named color is “green”.

navigation. The traversing of a document based on links between contextually related document components.

navigation link. A link type that specifies the linkage from a source document component to a contextually related target document component. Navigation links can be used to support applications such as hypertext and hypermedia.

Negative Acknowledge Reply (NACK). In the IPDS architecture, a reply from a printer to a host, indicating that an exception has occurred. Contrast with *Positive Acknowledge Reply*.

nested resource. A resource that is invoked within another resource using either an Include command or a local ID. See also *nesting resource*.

nesting coordinate space. A coordinate space that contains another coordinate space. Examples of coordinate spaces are medium, overlay, page, and object area.

nesting resource. A resource that invokes nested resources. See also *nested resource*.

neutral white. A color attribute that gives a device-dependent default color, typically white on a screen and black on a printer. Note that neutral white and color of medium are two different colors.

nonprocess runout (NPRO). An operation that moves sheets of physical media through the printer without printing on them. This operation is used to stack the last printed sheet.

no operation (NOP). A construct whose execution causes a product to proceed to the next instruction to be processed without taking any other action.

NOP. See *no operation*.

normal-duplex printing. Duplex printing that simulates the effect of physically turning the sheet around the Y_m axis.

NPRO. See *nonprocess runout*.

N-up. The partitioning of a side of a sheet into a fixed number of equal size partitions. For example, 4-up divides each side of a sheet into four equal partitions.

O

object. (1) A collection of structured fields. The first structured field provides a begin-object function, and the last structured field provides an end-object function. The object can contain one or more other structured fields whose content consists of one or more data elements of a particular data type. An object can be assigned a name, that can be used to reference the object. Examples of objects are presentation text, font, graphics, and image

objects. (2) Something that a user works with to perform a task.

object area. A rectangular area in a presentation space into which a data object is mapped. The presentation space can be for a page or an overlay. Examples are a graphics object area, an image object area, and a bar code object area.

object data. A collection of related data elements bundled together. Examples of object data include graphic characters, image data elements, and drawing orders.

object identifier (OID). (1) A notation that assigns a globally unambiguous name to an object or a document component. The notation is defined in international standard ISO/IEC 8824(E). (2) A variable length (2-bytes long to 129-bytes long) binary ID that uniquely identifies an object. OIDs use the ASN.1 definite-short-form object identifier format defined in the ISO/IEC 8824:1990(E) international standard and described in the MO:DCA Registry Appendix of the *Mixed Object Document Content Architecture Reference*. An OID consists of a one-byte identifier (X'06'), followed by a one-byte length (between X'00' and X'7F'), followed by 0–127 content bytes.

OCR-A. See *Optical Character Recognition-A*.

OCR-B. See *Optical Character Recognition-B*.

offline. A device state in which the device is not under the direct control of a host. Contrast with *online*.

offset. A table heading for architecture syntax. The entries under this heading indicate the numeric displacement into a construct. The offset is measured in bytes and starts with byte zero. Individual bits can be expressed as displacements within bytes.

OID. See *object identifier*.

online. A device state in which the device is under the direct control of a host. Contrast with *offline*.

opacity. In bar codes, the optical property of a substrate material that minimizes showing through from the back side or the next sheet.

Optical Character Recognition-A (OCR-A). A font containing the character set in ANSI standard X3.17-1981, that contains characters that are both human-readable and machine-readable.

Optical Character Recognition-B (OCR-B). A font containing the character set in ANSI standard X3.49-1975, that contains characters that are both human-readable and machine-readable.

order. (1) In GOCA, a graphics construct that the controlling environment builds to instruct a drawing processor about what to draw and how to draw it. The order can specify, for example, that a graphics primitive be drawn, a change to drawing attributes or drawing controls be effected, or a segment be called. One or more graphics primitives can be used to draw a picture. Orders can be included in a structured field. Synonymous with *drawing order*. (2) In the IPDS architecture, a construct within an execute-order command. (3) In IOCA, a functional operation that is performed on the image content.

ordered page. In the IPDS architecture, a logical page that does not contain any page segments or overlays, and in which all text data and all image, graphics, and bar code objects are ordered. The order of the data objects is such that physical pel locations on the physical medium are accessed by the printer in a sequential left-to-right and top-to-bottom manner, where these directions are relative to the top edge of the physical medium. Once a physical pel location has been accessed by the printer, the page data does not require the printer to access that same physical pel location again.

orientation. The angular distance a presentation space or object area is rotated in a specified coordinate system, expressed in degrees and minutes. For example, the orientation of printing on a physical medium, relative to the X_m axis of the X_m,Y_m coordinate system. See also *presentation space orientation* and *text orientation*.

origin. The point in a coordinate system where the axes intersect. Examples of origins are the addressable position in an X_m,Y_m coordinate system where both coordinate values are zero and the character reference point in a character coordinate system.

orthogonal. Intersecting at right angles. An example of orthogonal is the positional relationship between the axes of a Cartesian coordinate system.

outline font. A shape technology in which the graphic character shapes are represented in digital form by a series of mathematical expressions that define the outer edges of the strokes. The resultant graphic character shapes can be either solid or hollow.

output profile. An ICC profile that describes the characteristics of the output device for which the image is destined. The profile is used to color match the image to the device's gamut.

overhead. In a bar code symbology, the fixed number of characters required for starting, stopping, and checking a bar code symbol.

overlay. (1) A resource object that contains presentation data such as, text, image, graphics, and bar code data. Overlays define their own environment and are often used as pre-defined pages or electronic forms. Overlays are classified according to how they are presented with other presentation data: a medium overlay is positioned at the origin of the medium presentation space before any pages are presented, and a page overlay is positioned at a specified point in a page's logical page. A Page Modification Control (PMC) overlay is a special type of

page overlay used in MO:DCA environments. (2) The final representation of such an object on a physical medium. Contrast with *page segment*.

Overlay command set. In the IPDS architecture, a collection of commands used to load, deactivate, and include overlays.

overlay ID. A one-byte ID assigned by a host to an overlay. Overlay IDs are used in IPDS Begin Overlay, Deactivate Overlay, Include Overlay, and Load Copy Control commands.

overlay state. An operating state that allows overlay data to be downloaded to a product. For example, a printer enters overlay state from home state when the printer receives an IPDS Begin Overlay command.

overpaint. A mixing rule in which the intersection of part of a new presentation space P_{new} with an existing presentation space $P_{existing}$ keeps the color attribute of P_{new}. This is also referred to as "opaque" mixing. See also *mixing rule*. Contrast with *blend* and *underpaint*.

overscore. A line parallel to the baseline and placed above the character.

overstrike. In PTOCA, the presentation of a designated character as a string of characters in a specified text field. The intended effect is to make the resulting presentation appear as though the text field, whether filled with characters or blanks, has been marked out with the overstriking character.

overstriking. The method used to merge two or more graphic characters at the same addressable position in a presentation space or on a physical medium.

P

page. (1) A data stream object delimited by a Begin Page structured field and an End Page structured field. A page can contain presentation data such as text, image, graphics, and bar code data. (2) The final representation of a page object on a physical medium.

page counter. Bytes in an IPDS Acknowledge Reply that specify the number of pages that have passed a particular point in a logical paper path.

page group. A named group of sequential pages. A page group is delimited by a Begin Named Page Group structured field and an End Named Page Group structured field. A page group can contain nested page groups. All pages in the page group inherit the attributes and processing characteristics that are assigned to the page group.

page segment. (1) In the IPDS architecture, a resource object that can contain text, image, graphics, and bar code data. Page segments do not define their own environment, but are processed in the existing environment. (2) In MO:DCA, a resource object that can contain any mixture of bar code objects, graphics objects, and IOCA image objects. A page segment does not contain an active environment group. The environment for a page segment is defined by the active environment group of the including page or overlay. (3) The final representation of such an object on a physical medium. Contrast with *overlay*.

Page-Segment command set. In the IPDS architecture, a collection of commands used to load, deactivate, and include page segments.

page-segment state. An operating state that makes page-segment data available to a product. For example, a printer enters page-segment state from home state when it receives an IPDS Begin Page Segment command.

page state. In the IPDS architecture, an operating state that makes page data available to a product. For example, a printer enters page state from home state when it receives an IPDS Begin Page command.

paginated object. A data object that can be rendered on a single page or overlay. An example of a paginated object is a single image in a multi-image TIFF file.

parameter. (1) A variable that is given a constant value for a specified application. (2) A variable used in conjunction with a command to affect its result.

partition. Dividing the medium presentation space into a specified number of equal-sized areas in a manner determined by the current physical media.

partitioning. A method used to place parts of a control into two or more segments or structured fields. Partitioning can cause difficulties for a receiver if one of the segments or structured fields is not received or is received out of order.

pattern. An array of symbols used to fill an area.

pattern attributes. The characteristics that specify the appearance of a pattern.

pattern reference point. In GOCA, a position in the graphics presentation space to be used as the origin of a custom pattern; the pattern is tiled in all directions from this position.

pattern set. An attribute in GOCA used to access a symbol set or coded font.

pattern symbol. The geometric construct that is used repetitively to generate a pattern. Examples of pattern symbols are dots, squares, and triangles.

PCS. (1) See *Print Contrast Signal*. (2) See Profile Connection Space.

pel. The smallest printable or displayable unit on a physical medium. In computer graphics, the smallest element of a physical medium that can be independently assigned color and intensity. Pels per inch is often used as a measurement of presentation granularity. Synonymous with *picture element* and *pixel*.

PFB file. A file containing the font information required for presenting the characters of a font. The shape information (glyph procedures) contained in this file is in a binary encoded format defined by Adobe Systems Inc., optimized for small character set fonts having one to two hundred characters (for example, English, Greek, and Cyrillic).

PFO. See *preprinted form overlay*.

physical medium. A physical entity on which information is presented. Examples of a physical medium are a sheet of paper, a roll of paper, an envelope, and a display screen. See also *medium presentation space* and *sheet*.

physical printable area. A bounded area defined on a side of a sheet within which printing can take place. The physical printable area is an attribute of sheet size and printer capabilities, and cannot be altered by the host. The physical printable area is mapped to the medium presentation space, and is used in user printable area and valid printable area calculations. Contrast with *user printable area* and *valid printable area*.

picket fence bar code. A bar code pattern presenting the axis of the symbol in its length dimension parallel to the X_{bc} axis of the bar code presentation space. Synonymous with *horizontal bar code*.

picture chain. A string of segments that defines a picture. Synonymous with *segment chain*.

picture element. The smallest printable or displayable unit on a physical medium. In computer graphics, the smallest element of a physical medium that can be independently assigned color and intensity. Picture elements per inch is often used as a measurement of presentation granularity. Synonymous with *pel* and *pixel*.

pixel. The smallest printable or displayable unit on a physical medium. In computer graphics, the smallest element of a physical medium that can be independently assigned color and intensity. Picture elements per inch is often used as a measurement of presentation granularity. Synonymous with *pel* and *picture element*.

point. (1) A unit of measure used mainly for measuring typographical material. There are seventy-two points to an inch. (2) In GOCA, a parameter that specifies the position within the drawing order coordinate space. See also *drawing order coordinate space*.

polyline. A sequence of connected lines.

portrait. A presentation orientation in which the X_m axis is parallel to the short sides of a rectangular physical medium. Contrast with *landscape*.

position. A position in a presentation space or on a physical medium that can be identified by a coordinate from the coordinate system of the presentation space or physical medium. See also *picture element*. Synonymous with *addressable position*.

Positive Acknowledge Reply (ACK). In the IPDS architecture, a reply to an IPDS command that has its acknowledgment-required flag on and in which no exception is reported. Contrast with *Negative Acknowledge Reply*.

posture. Inclination of a letter with respect to a vertical axis. Examples of inclination are upright and inclined. An example of upright is Roman. An example of inclined is italics.

pragmatics. Information related to the usage of a construct. See also *semantics* and *syntax*.

preprinted form. A form or sheet that is not blank when it is selected as input media for presentation.

preprinted form overlay (PFO). An overlay and associated processing designed to simulate a preprinted form.

presentation device. A device that produces character shapes, graphics pictures, images, or bar code symbols on a physical medium. Examples of a physical medium are a display screen and a sheet of paper.

presentation position. An addressable position that is coincident with a character reference point. See also *addressable position* and *character reference point*.

presentation services. In printing, a software component that communicates with a printer using a printer data stream, such as the IPDS data stream, to print pages, download and manage print resources, and handle exceptions.

presentation space. A conceptual address space with a specified coordinate system and a set of addressable positions. The coordinate system and addressable positions can coincide with those of a physical medium. Examples of presentation spaces are medium, logical page, and object area. See also *graphics presentation space*, *image presentation space*, *logical page*, *medium presentation space*, and *text presentation space*.

presentation space orientation. The number of degrees and minutes a presentation space is rotated in a specified coordinate system. For example, the orientation of printing on a physical medium, relative to the X_m axis of the X_m,Y_m coordinate system. See also *orientation* and *text orientation*.

presentation text object. An object that contains presentation text data. See also *object*.

Presentation Text Object Content Architecture (PTOCA). An architected collection of constructs used to interchange and present presentation text data.

print contrast. A measurement of the ratio of the reflectivities between the bars and spaces of a bar code symbol, commonly expressed as a percent. Synonymous with *Print Contrast Signal*.

Print Contrast Signal (PCS). A measurement of the ratio of the reflectivities between the bars and spaces of a bar code symbol, commonly expressed as a percent. Synonymous with *print contrast*.

print control object. A resource object that contains layout, finishing, and resource mapping information used to present a document on physical media. Examples of print control objects are Form Maps and Medium Maps.

print direction. In FOCA, the direction in which successive characters appear in a line of text.

printing baseline. A conceptual line with respect to which successive characters are aligned. See also *character baseline*. Synonymous with *baseline* and *sequential baseline*.

print quality. In bar codes, the measure of compliance of a bar code symbol to the requirements of dimensional tolerance, edge roughness, spots, voids, reflectance, PCS, and quiet zones defined within a bar code symbology.

print unit. In the IPDS architecture, a group of pages bounded by XOH-DGB commands and subject to the group operation *keep group together as a print unit*. A print unit is commonly referred to as a *print job*.

process color. A color that is specified as a combination of the components, or primaries, of a color space. A process color is rendered by mixing the specified amounts of the primaries. An example of a process color is C=0.1, M=0.8, Y=0.2, K=0.1 in the cyan/magenta/yellow/black (CMYK) color space. Contrast with *spot color*.

process element. In MO:DCA, a document component that is defined by a structured field and that facilitates a form of document processing that does not affect the presentation of the document. Examples of process elements are Tag Logical Elements (TLEs) that specify document attributes and Link Logical Elements (LLEs) that specify linkages between document components.

Profile Connection Space (PCS). The reference color space defined by ICC, in which colors are encoded in order to provide an interface for connecting source and destination transforms. The PCS is based on the CIE 1931 standard colorimetric observer.

prolog. The first portion of a segment's data. Prologs are optional. They contain attribute settings and drawing controls. Synonymous with *segment prolog*.

propagation. A method used to retain a segment's properties through other segments that it calls.

proper subset. A set whose members are also members of a larger set.

proportion. Relationship of the width of a letter to its height.

proportional spacing. The spacing of characters in a printed line so that each character is allotted a space based on the character's width.

Proportional Spacing Machine font (PSM font). A font originating with the electric typewriter and having character increment values that are integer multiples of the narrowest character width.

proportionally spaced font. A font with graphic characters that have varying character increments. Proportional spacing can be used to provide the appearance of even spacing between presented characters and to eliminate excess blank space around narrow characters. An example of a narrow character is the letter *i*. Synonymous with *typographic font*. Contrast with *monospaced font* and *uniformly spaced font*.

PSM font. See *Proportional Spacing Machine font*.

PTOCA. See *Presentation Text Object Content Architecture*.

Q

quiet zone. A clear space that contains no machine-readable marks preceding the start character of a bar code symbol or following the stop character. Synonymous with *clear area*. Contrast with *intercharacter gap* and *space*.

R

range. A table heading for architecture syntax. The entries under this heading give numeric ranges applicable to a construct. The ranges can be expressed in binary, decimal, or hexadecimal. The range can consist of a single value.

rasterize. To convert presentation data into raster (bitmap) form for display or printing.

raster pattern. A rectangular array of pels arranged in rows called scan lines.

readability. The characteristics of visual material that determine the degree of comfort with which it can be read over a sustained period of time. Examples of

characteristics that influence readability are type quality, spacing, and composition.

reader. In bar code systems, the scanner or combination of scanner and decoder. See also *decoder* and *scanner*.

read rate. In bar codes, the ratio of the number of successful reads on the first attempt to the total number of attempts made to obtain a successful read. Synonymous with *first read rate*.

Rearranged File. A file containing the mapping of code points to the character index values used in a CID file and to the character names used in one or more PFB files. This is a special case of the CMAP file which permits linking of multiple font files and formats together. The code points conform to a particular character coding system which is used to identify the characters in a document data stream. The mapping information in this file is in an ASCII file format defined by Adobe Systems Inc.

recording algorithm. An algorithm that determines the relationship between the physical location and logical location of image points in image data.

recovery-unit group. In the IPDS architecture, a group of pages identified by the XOH Define Group Boundary command and controlled by the Keep-Group-Together-as-a-Recovery-Unit group operation specified by the XOH Specify Group Operation command. The recovery-unit group also includes all copies specified by the Load Copy Control command.

redaction. The process of applying an opaque mask over a page so that a selected portion of the page is visible. Since this function is typically used to prevent unauthorized viewing of data, an associated security level is also provided.

reflectance. In bar codes, the ratio of the amount of light of a specified wavelength or series of wavelengths reflected from a test surface to the amount of light reflected from a barium oxide or magnesium oxide standard under similar illumination conditions.

relative coordinate. One of the coordinates that identify the location of an addressable point by means of a displacement from some other addressable point. Contrast with *absolute coordinate*.

relative line. A straight line developed from a specified point by a given displacement.

relative metrics. Graphic character measurements expressed as fractions of a square, called the *Em square*, whose sides correspond to the vertical size of the font. Because the measurements are relative to the size of the Em square, the same metrics can be used for different point sizes and different raster pattern resolutions. Relative metrics require defining the unit of measure for the Em square, the point size of the font, and, if applicable, the resolution of the raster pattern.

relative move. A method used to establish a new current position. Distance and direction from the current position are used to establish the new current position. The direction of displacement is inline along the I axis in the I-direction, or baseline along the B axis in the B-direction, or both.

relative positioning. The establishment of a position within a coordinate system as an offset from the current position. Contrast with *absolute positioning*.

repeat string. A method used to repeat the character content of text data until a given number of characters has been processed. Any control sequences in the text data are ignored. This method provides the functional equivalence of a Transparent Data control sequence when the given number of repeated characters is equal to the number of characters in the text data.

repeating group. A group of parameter specifications that can be repeated.

reserved. Having no assigned meaning and put aside for future use. The content of reserved fields is not used by receivers, and should be set by generators to a specified value, if given, or to binary zeros. A reserved field or value can be assigned a meaning by an architecture at any time.

reset color. The color of a presentation space before any data is added to it. Synonymous with *color of medium*.

resident resource. In the IPDS architecture, a resource in a printer or in a resource-caching intermediate device. A resident resource can be installed manually or can be captured by the device if it is intended for public use. A resident resource is referenced by a global ID that is valid for the duration of the resource's presence in the device. Contrast with *downloaded resource*.

resolution. (1) A measure of the sharpness of an input or output device capability, as given by some measure relative to the distance between two points or lines that can just be distinguished. (2) The number of addressable pels per unit of length.

resolution correction. A method used to present an image on a printer without changing the physical size or proportions of the image when the resolutions of the printer and the image are different.

resolution-correction ratio. The ratio of a device resolution to an image presentation space resolution.

resolution modification. A method used to write an image on an image presentation space without changing the physical size of the image when the resolutions of the presentation space and the image are different.

resource. An object that is referenced by a data stream or by another object to provide data or information. Resource objects can be stored in libraries. In MO:DCA, resource objects can be contained within a resource group.

Examples of resources are fonts, overlays, and page segments. See also *downloaded resource* and *resident resource*.

resource caching. In the IPDS architecture, a function in a printer or intermediate device whereby downloaded resources are captured and made resident in the printer or intermediate device.

retired. Set aside for a particular purpose, and not available for any other purpose. Retired fields and values are specified for compatibility with existing products and identify one of the following:

- Fields or values that have been used by a product in a manner not compliant with the architected definition
- Fields or values that have been removed from an architecture

RGB. Red, green and blue, the additive primary colors.

RGB color space. The basic additive color model used for color video display, as on a computer monitor.

RM4SCC. See *Royal Mail 4 State Customer Code*.

Roman. Relating to a type style with upright letters.

root segment. A segment in the picture chain that is not called by any other segment. If a single segment that is not in a segment chain is drawn, it is treated as a root segment for the duration of the drawing process.

rotating. In computer graphics, turning all or part of a picture about an axis perpendicular to the presentation space.

rotation. The orientation of a presentation space with respect to the coordinate system of a containing presentation space. Rotation is measured in degrees in a clockwise direction. Zero-degree rotation exists when the angle between a presentation space's positive X axis and the containing presentation space's positive X axis is zero degrees. Contrast with *character rotation*.

row. A subarray that consists of all elements that have an identical position within the high dimension of a regular two-dimensional array.

Royal Mail 4 State Customer Code (RM4SCC). A two-dimensional bar code symbology developed by the United Kingdom's Royal Mail postal service for use in automated mail-sorting processes.

rule. A solid line of any line width.

S

sans serif. A type style characterized by strokes that end with no flaring or crossing of lines at the stroke-ends. Contrast with *serif*.

SBCS. See *single-byte character set*.

SBIN. A data type for architecture syntax, that indicates that one or more bytes be interpreted as a signed binary number, with the sign bit in the high-order position of the leftmost byte. Positive numbers are represented in true binary notation with the sign bit set to B'0'. Negative numbers are represented in twos-complement binary notation with a B'1' in the sign-bit position.

scaling. Making all or part of a picture smaller or larger by multiplying the coordinate values of the picture by a constant amount. If the same multiplier is applied along both dimensions, the scaling is uniform, and the proportions of the picture are unaffected. Otherwise, the scaling is anamorphic, and the proportions of the picture are changed. See also *anamorphic scaling*.

scaling ratio. (1) The ratio of an image-object-area size to its image-presentation-space size. (2) In FOCA, the ratio of horizontal to vertical scaling of the graphic characters. See also *horizontal scale factor*.

scan line. A series of picture elements. Scan lines in raster patterns form images. See also *picture element* and *raster pattern*.

scanner. In bar codes, an electronic device that converts optical information into electrical signals. See also *reader*.

scrolling. A method used to move a displayed image vertically or horizontally so that new data appears at one edge as old data disappears at the opposite edge. Data disappears at the edge toward which an image is moved and appears at the edge away from which the data is moved.

SDA. See *special data area*.

section. A portion of a double-byte code page that consists of 256 consecutive entries. The first byte of a two-byte code point is the section identifier. A code-page section is also called a code-page ward in some environments. See also *code page* and *code point*.

section identifier. A value that identifies a section. Synonymous with *section number*.

section number. A value that identifies a section. Synonymous with *section identifier*.

secure overlay. An overlay that can be printed anywhere within the physical printable area. A secure overlay is not affected by an IPDS Define User Area command.

segment. (1) In GOCA, a set of graphics drawing orders contained within a Begin Segment command. See also *graphics segment*. (2) In IOCA, image content bracketed by Begin Segment and End Segment self-defining fields. See also *image segment*.

segment chain. A string of segments that defines a picture. Synonymous with *picture chain*.

segment exception condition. An architecture-provided classification of the errors that can occur in a segment. Segment exception conditions are raised when a segment error is detected. Examples of segment errors are segment format, parameter content, and sequence errors.

segment offset. A position within a segment, measured in bytes from the beginning of the segment. The beginning of a segment is always at offset zero.

segment prolog. The first portion of a segment's data. Prologs are optional. They contain attribute settings and drawing controls. Synonymous with *prolog*.

segment properties. The segment characteristics used by a drawing process. Examples of segment properties are segment name, segment length, chained, dynamic, highlighted, propagated, and visible.

segment transform. A model transform that is applied to a whole segment.

self-checking. In bar codes, using a checking algorithm that can be applied to each character independently to guard against undetected errors.

semantics. The meaning of the parameters of a construct. See also *pragmatics* and *syntax*.

sequential baseline. A conceptual line with respect to which successive characters are aligned. See also *character baseline*. Synonymous with *baseline* and *printing baseline*.

sequential baseline position. The current addressable position for a baseline in a presentation space or on a physical medium. See also *baseline coordinate* and *current baseline presentation coordinate*.

serif. A short line angling from or crossing the free end of a stroke. Examples are horizontal lines at the tops and bottoms of vertical strokes on capital letters, for example, *I* and *H*, and the decorative strokes at the ends of the horizontal members of a capital *E*. Contrast with *sans serif*.

session. In the IPDS architecture, the period of time during which a presentation services program has a two-way communication with an IPDS device. The session consists of a physical attachment and a communications protocol; the communications protocol carries an IPDS dialog by transparently transmitting IPDS commands and acknowledge replies. See also *IPDS dialog*.

setup file. In the IPDS architecture, an object container that provides setup information for a printer. Setup files are downloaded in home state and take effect immediately. Setup files are not managed as resources.

shade. Variation of a color produced by mixing it with black.

shape compression. A method used to compress digitally encoded character shapes using a specified algorithm.

shape technology. A method used to encode character shapes digitally using a specified algorithm.

shear. The angle of slant of a character cell that is not perpendicular to a baseline. Synonymous with *character shear*.

shearline direction. In GOCA, the direction specified by the character shear and character angle attributes.

sheet. A division of the physical medium; multiple sheets can exist on a physical medium. For example, a roll of paper might be divided by a printer into rectangular pieces of paper, each representing a sheet. Envelopes are an example of a physical medium that comprises only one sheet. The IPDS architecture defines four types of sheets: cut-sheet media, continuous-form media, envelopes, and computer output on microfilm. Each type of sheet has a top edge. A sheet has two sides, a front side and a back side. Synonymous with *form*.

show-through. In bar codes, the generally undesirable property of a substrate that permits underlying markings to be seen.

side. A physical surface of a sheet. A sheet has a front side and a back side. See also *sheet*.

simplex printing. A method used to print data on one side of a sheet; the other side is left blank. Contrast with *duplex printing*.

single-byte character set (SBCS). A character set that can contain up to 256 characters.

single-byte coded font. A coded font in which the code points are one byte long.

slope. The posture, or incline, of the main strokes in the graphic characters of a font. Slope is specified in degrees by a font designer.

space. In bar codes, the lighter element of a printed bar code symbol, usually formed by the background between bars. See also *element*. Contrast with *bar*, *clear area*, *intercharacter gap*, and *quiet zone*.

space width. In bar codes, the thickness of a bar code symbol space measured from the edge closest to the symbol start character to the trailing edge of the same space.

spanning. In the IPDS architecture, a method in which one command is used to start a sequence of constructs.

Subsequent commands continue and terminate that sequence. See also *control sequence chaining*.

special data area (SDA). The data area in an IPDS Acknowledge Reply that contains data requested by the host or generated by a printer as a result of an exception.

Specifications for Web Offset Publications (SWOP). A standard set of specifications for color separations, proofs, and printing to ensure consistency of color printing.

spot. In bar codes, the undesirable presence of ink or dirt in a bar code symbol space.

spot color. A color that is specified with a unique identifier such as a number. A spot color is normally rendered with a custom colorant instead of with a combination of process color primaries. See also *highlight color*. Contrast with *process color*.

stack. A list that is constructed and maintained so that the next item to be retrieved and removed is the most recently stored item still in the list. This is sometimes called last-in-first-out (LIFO).

standard action. The architecture-defined action to be taken on detecting an exception condition, when the controlling environment specifies that processing should continue.

standard line type value. A predefined line type, like solid, invisible, or dash-dot. Contrast with *custom line type value*.

start-stop character or pattern. In bar codes, a special bar code character that provides the scanner with start and stop reading instructions as well as a scanning direction indicator. The start character is normally at the left end and the stop character at the right end of a horizontally oriented symbol.

store mode. A mode in which segments are stored for later execution. Contrast with *immediate mode*.

stroke. A straight or curved line used to create the shape of a letter.

structured field. A self-identifying, variable-length, bounded record, that can have a content portion that provides control information, data, or both. See also *document element*.

structured field introducer. In MO:DCA, the header component of a structured field that provides information that is common for all structured fields. Examples of information that is common for all structured fields are length, function type, and category type. Examples of structured field function types are begin, end, data, and descriptor. Examples of structured field category types are presentation text, image, graphics, and page.

subset. Within the base-and-towers concept, a portion of architecture represented by a particular level in a tower or by a base. See also *subsetting tower*.

subsetting tower. Within the base-and-towers concept, a tower representing an aspect of function achieved by an architecture. A tower is independent of any other towers. A tower can be subdivided into subsets. A subset contains all the function of any subsets below it in the tower. See also *subset*.

substrate. In bar codes, the surface on which a bar code symbol is printed.

subtractive primary colors. Cyan, magenta, and yellow colorants used to subtract a portion of the white light that is illuminating an object. Subtractive colors are reflective on paper and printed media. When used together with various degrees of coverage and variation, they have the ability to create billions of other colors. Contrast with *additive primary colors*.

suppression. A method used to prevent presentation of specified data. Examples of suppression are the processing of text data without placing characters on a physical medium and the electronic equivalent of the “spot carbon,” that prevents selected data from being presented on certain copies of a presentation space or a physical medium.

surrogate pair. A sequence of two Unicode code points that allow for the encoding of as many as 1 million additional characters without any use of escape codes.

SWOP. See Specifications for Web Offset Publications.

symbol. (1) A visual representation of something by reason of relationship, association, or convention. (2) In GOCA, the subpicture referenced as a character definition within a font character set and used as a character, marker, or fill pattern. A bitmap can also be referenced as a symbol for use as a fill pattern. See also *bar code symbol*.

symbol length. In bar codes, the distance between the outside edges of the quiet zones of a bar code symbol.

symbol set. A coded font that is usually simpler in structure than a fully described font. Symbol sets are used where typographic quality is not required. Examples of devices that might not provide typographic quality are dot-matrix printers and displays. See also *character set*, *marker set*, and *pattern set*.

symbology. A bar code language. Bar code symbologies are defined and controlled by various industry groups and standards organizations. Bar code symbologies are described in public domain bar code specification documents. Synonymous with *bar code symbology*. See also *Canadian Grocery Product Code (CGPC)*, *European Article Numbering (EAN)*, *Japanese Article Numbering (JAN)*, and *Universal Product Code (UPC)*.

synchronous exception. In the IPDS architecture, a data-stream, function no longer achievable, or resource-storage exception that must be reported to the host before a printer can return a Positive Acknowledge Reply or can increment the received-page counter for a page containing the exception. Synchronous exceptions are those with action code X'01', X'06', X'0C', or X'1F'. See also *data-stream exception*. Contrast with *asynchronous exception*.

syntax. The rules governing the structure of a construct. See also *pragmatics* and *semantics*.

system-level font resource. A common-source font from which:

- Document-processing applications can obtain resolution-independent formatting information.
- Device-service applications can obtain device-specific presentation information.

T

temporary baseline. The shifted baseline used for subscript and superscript.

temporary baseline coordinate. The B-value of the I,B coordinate pair of an addressable position on the temporary baseline.

temporary baseline increment. A positive or negative value that is added to the current baseline presentation coordinate to specify the position of a temporary baseline in a presentation space or on a physical medium. Several increments might have been used to place a temporary baseline at the current baseline presentation coordinate.

text. A graphic representation of information. Text can consist of alphanumeric characters and symbols arranged in paragraphs, tables, columns, and other shapes. An example of text is the data sent in an IPDS Write Text command.

Text command set. In the IPDS architecture, a collection of commands used to present PTOCA text data in a page, page segment, or overlay.

text orientation. A description of the appearance of text as a combination of inline direction and baseline direction. See also *baseline direction*, *inline direction*, *orientation*, and *presentation space orientation*.

text presentation. The transformation of document graphic character content and its associated font information into a visible form. An example of a visible form of text is character shapes on a physical medium.

text presentation space. A two-dimensional conceptual space in which text is generated for presentation on an output medium.

throughscore. A line parallel to the baseline and placed through the character.

tint. Variation of a color produced by mixing it with white.

toned. Containing marking agents such as toner or ink. Contrast with *untoned*.

transform. A modification of one or more characteristics of a picture. Examples of picture characteristics that can be transformed are position, orientation, and size. See also *model transform*, *segment transform*, and *viewing transform*.

transform matrix. A matrix that is applied to a set of coordinates to produce a transform.

translating. In computer graphics, moving all or part of a picture in the presentation space from one location to another without rotating.

transparent data. A method used to indicate that any control sequences occurring in a specified portion of data can be ignored.

trimming. Eliminating those parts of a picture that are outside of a clipping boundary such as a viewing window or presentation space. See also *viewing window*. Synonymous with *clipping*.

triplet. A three-part self-defining variable-length parameter consisting of a length byte, an identifier byte, and parameter-value bytes.

triplet identifier. A one-byte type identifier for a triplet.

truncation. Planned or unplanned end of a presentation space or data presentation. This can occur when the presentation space extends beyond one or more boundaries of its containing presentation space or when there is more data than can be contained in the presentation space.

tumble-duplex printing. A method used to simulate the effect of physically turning a sheet around the X_m axis.

twip. A unit of measure equal to 1/20 of a point. There are 1440 twips in one inch.

type. A table heading for architecture syntax. The entries under this heading indicate the types of data present in a construct. Examples include: BITS, CHAR, CODE, SBIN, UBIN, UNDF.

typeface. All characters of a single type family or style, weight class, width class, and posture, regardless of size. For example, Helvetica Bold Condensed Italics, in any point size.

type family. All characters of a single design, regardless of attributes such as width, weight, posture, and size. Examples are Courier and Gothic.

type structure. Attributes of characters other than type family or typeface. Examples are solid shape, hollow shape, and overstruck.

type style. The form of characters within the same font, for example, Courier or Gothic.

type weight. A parameter indicating the degree of boldness of a typeface. A character's stroke thickness determines its type weight. Examples are light, medium, and bold. Synonymous with *weight class*.

type width. A parameter indicating a relative change from the font's normal width-to-height ratio. Examples are normal, condensed, and expanded. Synonymous with *width class*.

typographic font. A font with graphic characters that have varying character increments. Proportional spacing can be used to provide the appearance of even spacing between presented characters and to eliminate excess blank space around narrow characters. An example of a narrow character is the letter *i*. Synonymous with *proportionally spaced font*. Contrast with *monospaced font* and *uniformly spaced font*.

U

UBIN. A data type for architecture syntax, indicating one or more bytes to be interpreted as an unsigned binary number.

unarchitected. Identifies data that is neither defined nor controlled by an architecture. Contrast with *architected*.

unbounded character box. A character box that can have blank space on any sides of the character shape.

underpaint. A mixing rule in which the intersection of part of a new presentation space P_{new} with part of an existing presentation space $P_{existing}$ keeps the color attribute of $P_{existing}$. This is also referred to as "transparent" or "leave alone" mixing. See also *mixing rule*. Contrast with *blend* and *overpaint*.

underscore. A method used to create an underline beneath the characters in a specified text field. An example of underscore is the line presented under one or more characters. Also a special graphic character used to implement the underscoring function.

UNDF. A data type for architecture syntax, indicating one or more bytes that are undefined by the architecture.

Unicode. A character encoding standard for information processing that includes all major scripts of the world. Unicode defines a consistent way of encoding multilingual text. Unicode specifies a numeric value, a name, and other attributes, such as directionality, for each of its characters; for example, the name for $ is "dollar sign" and its numeric value is X'0024'. This Unicode value is called a *Unicode code point* and is represented as U+nnnn. Unicode provides for three encoding forms (UTF-8, UTF-16, and UTF-32), described as follows:

UTF-8 A byte-oriented form that is designed for ease of use in traditional ASCII environments. Each UTF-8 code point contains from one to four bytes. All Unicode code points can be encoded in UTF-8 and all 7-bit ASCII characters can be encoded in one byte.

UTF-16 The default Unicode encoding. A fixed, two-byte Unicode encoding form that can contain surrogates and identifies the byte order of each UTF-16 code point via a Byte Order Mark in the first 2 bytes of the data. Surrogates are pairs of Unicode code points that allow for the encoding of as many as 1 million additional characters without any use of escape codes.

UTF-16BE UTF-16 that uses big endian byte order; this is the byte order for all multi-byte data within AFP data streams. The Byte Order Mark is not necessary when the data is externally identified as UTF-16BE (or UTF-16LE).

UTF-16LE UTF-16 that uses little endian byte order.

UTF-32 A fixed, four-byte Unicode encoding form in which each UTF-32 code point is precisely identical to the Unicode code point.

UTF-32BE UTF-32 serialized as bytes in most-significant-byte-first order (big endian). UTF-32BE is structurally the same as UCS-4.

UTF-32LE UTF-32 serialized as bytes in least-significant-byte-first order (little endian).

Uniform Symbol Specification (USS). A series of bar code symbology specifications published by AIM; currently included are USS-Interleaved 2 of 5, USS-39, USS-93, USS-Codabar, and USS-128.

uniformly spaced font. A font with graphic characters having a uniform character increment. The distance between reference points of adjacent graphic characters is constant in the escapement direction. The blank space between the graphic characters can vary. Synonymous with *monospaced font*. Contrast with *proportionally spaced font* and *typographic font*.

unit base. A one-byte code that represents the length of the measurement base. For example, X'00' might specify that the measurement base is ten inches.

Universal Product Code (UPC). A standard bar code symbology, commonly used to mark the price of items in stores, that can be read and interpreted by a computer.

untoned. Unmarked portion of a physical medium. Contrast with *toned*.

UP³I. Universal Printer Pre- and Post-Processing Interface; an industry standard interface designed for use in complex printing systems. A specification for this interface can be obtained at www.afpcinc.org.

UPA. See *user printable area*.

UPC. See *Universal Product Code*.

uppercase. Pertaining to capital letters. Examples of capital letters are *A*, *B*, and *C*. Contrast with *lowercase*.

upstream data. IPDS commands that exist in a logical path from a specific point in a printer back to, but not including, host presentation services.

usable area. An area on a physical medium that can be used to present data. See also *viewport*.

user printable area (UPA). The portion of the physical printable area to which user-generated data is restricted. See also *logical page*, *physical printable area*, and *valid printable area*.

USS. See *Uniform Symbol Specification*.

V

valid printable area (VPA). The intersection of a logical page with the area of the medium presentation space in which printing is allowed. If the logical page is a secure overlay, the area in which printing is allowed is the physical printable area. If the logical page is not a secure overlay and if a user printable area is defined, the area in which printing is allowed is the intersection of the physical printable area with the user printable area. If a user printable area is not defined, the area in which printing is allowed is the physical printable area. See also *logical page*, *physical printable area*, *secure overlay*, and *user printable area*.

variable space. A method used to assign a character increment dimension of varying size to space characters. The space characters are used to distribute white space within a text line. The white space is distributed by expanding or contracting the dimension of the variable space character's increment dependent upon the amount of white space to be distributed. See also *variable space character* and *variable space character increment*.

variable space character. The code point assigned by the data stream for which the character increment varies according to the semantics and pragmatics of the variable space function. This code point is not presented, but its character increment parameter is used to provide spacing. See also *variable space character increment*.

variable space character increment. The variable value associated with a variable space character. The variable space character increment is used to calculate the dimension from the current presentation position to a new presentation position when a variable space character is found. See also *variable space character*.

verifier. In bar code systems, a device that measures the bars, spaces, quiet zones, and optical characteristics of a bar code symbol to determine if the symbol meets the requirements of a bar code symbology, specification, or standard.

vertical bar code. A bar code pattern that presents the axis of the symbol in its length dimension parallel to the Y_{bc} axis of the bar code presentation space. Synonymous with *ladder bar code*.

vertical font size. (1) A characteristic value, perpendicular to the character baseline, that represents the size of all graphic characters in a font. Synonymous with *font height*. (2) In a font character set, nominal vertical font size is a font-designer defined value corresponding to the nominal distance between adjacent baselines when character rotation is zero degrees and no external leading is used. This distance represents the baseline-to-baseline increment that includes the font's maximum baseline extent and the designer's recommendation for internal leading. The font designer can also define a minimum and a maximum vertical font size to represent the limits of scaling. (3) In font referencing, the specified vertical font size is the desired size of the font when the characters are presented. If this size is different from the nominal vertical font size specified in a font character set, the character shapes and character metrics might need to be scaled prior to presentation.

vertical scale factor. In outline-font referencing, the specified vertical adjustment of the Em square. The vertical scale factor is specified in 1440ths of an inch. When the horizontal and vertical scale factors are different, anamorphic scaling occurs. See also *horizontal scale factor*.

viewing transform. A transform that is applied to model-space coordinates. Contrast with *model transform*.

viewing window. That part of a model space that is transformed, clipped, and moved into a graphics presentation space.

viewport. The portion of a usable area that is mapped to the graphics presentation space window. See also *graphics model space* and *graphics presentation space*.

visibility. The property of a segment that declares whether the part of a picture defined by the segment is to be displayed or not displayed during the drawing process.

void. In bar codes, the undesirable absence of ink in a bar code symbol bar element.

VPA. See *valid printable area*.

W

ward. A deprecated term for section.

weight class. A parameter indicating the degree of boldness of a typeface. A character's stroke thickness determines its weight class. Examples are light, medium, and bold. Synonymous with *type weight*.

white space. The portion of a line that is not occupied by characters when the characters of all the words that can be placed on a line and the spaces between those words are assembled or formatted on a line. When a line is justified, the white space is distributed among the words, characters, or both on the line in some specified manner. See also *controlled white space*.

width class. A parameter indicating a relative change from the font's normal width-to-height ratio. Examples are normal, condensed, and expanded. Synonymous with *type width*.

window. A predefined part of a graphics presentation space. See also *graphics presentation space window*.

writing mode. An identified mode for the setting of text in a writing system, usually corresponding to a nominal escapement direction of the graphic characters in that mode; for example, left-to-right, right-to-left, top-to-bottom.

X

X_{bc} extent. The size of a bar code presentation space in the X_{bc} dimension. See also *bar code presentation space*.

X_{bc},Y_{bc} coordinate system. The bar code presentation space coordinate system.

X-dimension. In bar codes, the nominal dimension of the narrow bars and spaces in a bar code symbol.

X_g,Y_g coordinate system. In the IPDS architecture, the graphics presentation space coordinate system.

X-height. The nominal height above the baseline, ignoring the ascender, of the lowercase characters in a font. X-height is usually the height of the lowercase letter *x*. See also *lowercase* and *ascender*.

X_{io},Y_{io} coordinate system. The IO-Image presentation space coordinate system.

XML. See *Extensible Markup Language*.

XMP. See *Extensible Metadata Platform*.

X_m,Y_m coordinate system. (1) In the IPDS architecture, the medium presentation space coordinate system. (2) In MO:DCA, the medium coordinate system.

X_{oa},Y_{oa} coordinate system. The object area coordinate system.

X_{ol},Y_{ol} coordinate system. The overlay coordinate system.

X_p extent. The size of a presentation space or logical page in the X_p dimension. See also *presentation space* and *logical page*.

X_p,Y_p coordinate system. The coordinate system of a presentation space or a logical page. This coordinate system describes the size, position, and orientation of a presentation space or a logical page. Orientation of an X_p, Y_p coordinate system is relative to an environment-specified coordinate system. An example of an environment-specified coordinate system is the X_m,Y_m coordinate system. The X_p,Y_p coordinate system origin is specified by an IPDS Logical Page Position command. See also *logical page*, *medium presentation space*, and *presentation space*.

X_{pg},Y_{pg} coordinate system. The coordinate system of a page presentation space. This coordinate system describes the size, position, and orientation of a page presentation space. Orientation of an X_{pg},Y_{pg} coordinate system is relative to an environment specified coordinate system, for example, an X_m,Y_m coordinate system.

Y

Y_{bc} extent. The size of a bar code presentation space in the Y_{bc} dimension. See also *bar code presentation space*.

Y_p extent. The size of a presentation space or logical page in the Y_p dimension. See also *presentation space* and *logical page*.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Advanced Function Presentation Consortium

Font Object Content Architecture Reference

AFPC-0007-06