ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

STUDY ON THE TRANSLATION OF THE ODA FORMATTED FORM INTO PAGE DESCRIPTION LANGUAGES

ECMA TR/48

December 1988

Free copies of this document are available from ECMA,
European Computer Manufacturers Association
114 Rue du Rhône – 1204 Geneva (Switzerland)

ECMA

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

STUDY ON THE TRANSLATION OF THE ODA FORMATTED FORM INTO PAGE DESCRIPTION LANGUAGES

ECMA TR/48

December 1988

E C M A

EUROPEAN COMPUTER MANUFACTURERS ASSOCIATION

Status of ECMA TR/48

The attached report (ECMA TR/48 - Study on the translation of the ODA Formatted Form into page description languages) was produced in 1987/88 and published in 1988.

While the principles and conclusions reached by the report are still considered by ECMA to be true, the reader should be aware that the report does not reflect the many amendments and technical corrigenda applicable to the Standard that have been agreed since the publication of this Technical Report. Therefore, some details included in this report are not fully accurate and ECMA takes no responsibility for any consequent implementation or conformance problems.

BRIEF HISTORY

This report details activities conducted under the auspices of the Print Format Task Group (TGPF) of ECMA/TC29 to study issues related to the translation of ODA documents to current commercial page description languages.

As part of its programme of work ECMA TC29-TGPF undertook to study the problems of translating documents conforming to the definition of ISO 8613 Office Document Architecture and Interchange Format (ODA), to the existing page description languages of Xerox Interpress™ and Adobe PostScript™.

The objectives behind this work item were:

- to inform the page description language companies of the capabilities of ODA for describing imageable documents;
- to obtain feedback from these companies as to the extensions required to ODA to support the requirements of high quality printing expected by the publishing industry;
- to evaluate current page description languages as prototypical implementations of the proposed SPDL and to examine their adequacy to support ODA Formatted Form.

This document is a report on the work to produce the ODA Formatted Form to PDL translators. It consists primarily of two papers by Xerox Corporation and Adobe Systems Incorporated specifying the results of their work. It also includes sections specifying the technical problems encountered in developing the translation software.

Existing page description languages are, in general, designed to satisfy the requirements of the publishing and printing industries. ODA is oriented towards the requirements of office systems. The conclusions of the report specify the implications resulting from the study for the design of a Printer Format or Standardized Page Description Language. These are to be taken into account by the ECMA Task Group in specifying the Printer Format standard.

Accepted as an ECMA Technical Report by the General Assembly of 15 December 1988.

1. INTRODUCTION

This is the final report of the work undertaken by the ECMA Task Group TC29-TGPF to study the translation of ODA-Formatted Form documents (FF) to two contemporary Page Description Languages (PDLs). The report consists primarily of two reports prepared by Xerox Corporation and Adobe Systems Incorporated. These reports are presented in Appendices A and B, respectively. These reports specify the work undertaken to develop the ODA to PDL translation software. The reports also include the conclusions each company has reached concerning the complexity of the task, the suitability of ODA for describing document images and the suitability of PDLs for describing ODA FF documents.

The two appendices contained in this report are as follows:

Appendix A - Report on the translation of ODA to Interpress

Appendix B - Report on the translation of ODA to PostScript

For preparation of this report the following documentation was also available:

- Translation test documents, (Bull)
- Translation test results, (Xerox)
- Test document application profile, (ICL)

However, due to its size and nature this documentation is of interest only to implementors. The published information contained in this Technical Report is sufficient for all but the very specialized readers.

2. REFERENCES

The following documents were used in the development of the ODA to PDL translation software.

- **ECMA-101**: Open Document Architecture (ODA) and Interchange Format - Parts 1, 2, 4-8
- **ISO 8632**: Information processing systems - Computer Graphics - Metafile for the storage and transfer of picture description information - Part 1: Functional specification - Part 3: Binary encoding.
- **ODA**: Document Application Profile H'87 (ESPRIT PODA deliverable)

- **ODA**: Document Application Profile H'88, Issue 1, October 1987 (ESPRIT PODA deliverable)
- **SPAG**: Application profile Q/113
- **ESPRIT**: Project 1024 (PODA) Technical Annex
- **STC ODA-100**: The structure of a description of Document Application Profile
- **NBS - ODA/ODIF**: Implementation Agreement, Oct. 1987

3. SUMMARY

This document is the report of a study undertaken to develop "proof of concept" level software capable of translating ODA-Formatted Form documents into the proprietary page description languages of Interpress and PostScript.

This study was proposed at the inception of the ECMA Print Format standard work as a practical mechanism by which one of the stated objectives could be attained. The objective was that ODA FF should be easily translatable into the Print Format standard.

This initial objective was refined and restated as the following threefold objectives. Firstly, to inform the market leader page description companies participating in ECMA TC29-TGPF of the capabilities of ODA for describing printable documents. Secondly, to decide, as a consequence of practical experiences, whether extensions are required to ODA to support the printing requirements of the publishing industry. Thirdly, to evaluate current PDLs as prototypical implementations of the proposed Standard Printer Format and to examine their adequacy to support ODA-Formatted Form.

The project was undertaken from July 1987 to June 1988. The project commenced by determining a source of suitable ODA test documents. Since the relevant development staff within Xerox and Adobe were not ODA experts the creation of the appropriate test documents could have been time consuming and a potential source of error. Bull, a participant in both TC29-TGPF and the ESPRIT PODA project, offered to make available a set of ODA test documents.

These tests were to be developed by Bull as part of its ESPRIT PODA sub-task to develop an ODA printer. ICL offered to develop an ODA Document Application Profile (DAP) which would describe the subset of ODA to which the test documents would conform. This DAP was based on work ICL was undertaking as a participant in the PODA project. In the PODA project ICL was developing an ODA DAP suitable for use with proprietary word processors.

The members of TGPF also decided on, and made available, a full set of the relevant standards documents to be used. Many of these were still under development within the

standards bodies, most important of these being the ODA standard itself. Consequently, availability of these documents was very limited. A full list of the standards documents used is specified in this report under References.

- The ODA test sample documents were delivered to Xerox and Adobe in three phases. The first being in November 1987 and then February 1988 and finally April 1988.

- The project to develop the ODA to PDL translators was completed at the June 1988 TC29 meeting with a presentation and demonstration by Xerox of their work. Also at that time Adobe said that, regrettably, due to a lack of resources they would be unable to complete the development of their translation software. They were however, able to provide material for the technical report describing the work done so far. This material included important conclusions from their design and development work.

4. CONCLUSIONS

The general conclusions that can be drawn from the work undertaken in this study by Xerox and Adobe can be summarized as follows:

- Xerox and Adobe have both benefited from their exposure to the ODA standard (and other associated standards) in the development of an ODA-based system prototype. This has enabled them to develop ODA expertise in the ODA standard and to make useful contributions to its future development.

- In order to satisfy the same objectives as PDLs concerning the fidelity of a printed document to the originator's intentions, ODA requires a more precise definition of certain imaging aspects. For example, the definition of underline, strikeout and use of fonts, etc in the character content architecture; the definition of hatched areas in the geometric graphics content architecture.

- ODA requires extensions in order to satisfy the same publishing-oriented objectives as PDLs concerning document imaging capabilities, e.g. arbitrary rotations and writing direction of characters; non-rectangular imaging areas.

- The development of a standard page description language should advisedly study and incorporate the capabilities of existing PDLs.

- Software to convert ODA-Formatted Form documents to existing PDLs can be developed at modest cost.

APPENDIX A

TRANSLATION OF ODA TO INTERPRESS, FINAL STUDY REPORT

Michael MacKay, Xerox Corporation

June 1988

A.1 INTRODUCTION

This report discusses the final results of the Xerox ODA-to-Interpress PDL translation activity, and both extends and amplifies the results of the interim report generated in February. The study has been very successful, the work yielding the following observations:

- Interpress provides powerful features for imaging that meet and exceed the capabilities required for printing ODA documents, as defined by the current ISO 8613 standard.

- It is possible to translate ISO 8613 (and its affiliated content architectures) to Interpress with modest effort and prior knowledge.

- Very little extra work is required to process Formatted Processable Document Architecture (FPDA) for imaging, as opposed to Formatted Document Architecture (FDA).

- It is desirable to translate directly from FPDA to Interpress, since this eliminates the intermediate step of translating FPDA to FDA.

- FDA offers no significant advantage for final processing of ODA documents for imaging.

- Xerox compressed RES raster encoding appears to reliably yield on the order of a 98% reduction in size for binary encoded rasters, as compared to the encoding used in the TG-PF test cases (e.g. the RASTERC test case was reduced from 447,888 bytes to 8401 bytes).

- Translation results in a reduction of execution complexity at imaging time. This is achieved by transforming the hierarchical object structure of ODA to a flat and linearly executable form.

- The weakest element of the ODA architecture, as currently defined, is in the area of fonts and text handling between different content architectures.

Development of the translator through this final phase of the study completes the demonstration that a PDL, in this case Interpress, can adequately represent imageable ODA documents. Additionally, the study has also demonstrated that it is both viable, and desirable, to create the PDL representation of an imageable document directly from FPDA without the use of FDA.

A.2 OBSERVATIONS AND ISSUES FROM THE TRANSLATION STUDY

A.2.1 Results of the Feasibility Study

The Xerox ODA-to-Interpress translation study has succeeded in implementing a significant subset of ISO 8624, and whose completeness is appropriate for a feasibility study.

Two primary observations have been made during the development of the translator. The first is that adding support for the creation of PDL output to an existing ODA system should represent a task of modest effort. ODA translates well, and in a reasonably straightforward fashion, to a PDL as demonstrated through the use of Interpress in this activity. Most of the difficulty in translating ODA to a PDL is embodied in handling document structure processing. It is, therefore, likely that a functional ODA document system would have little difficulty supporting a PDL, since most of the work for handling attributes and positioning would already be completed. With the availability of a common library of routines for outputting the PDL, the effort is further simplified by requiring only modest knowledge of the PDL itself; such is the case with Interpress, and the Interpress Toolkit by the Xerox Webster Research Center.

The second observation of the study is that FDA offers no significant advantage for the translation of ODA to a PDL. In fact, it is desirable to translate directly from FPDA to PDL, thus eliminating the FDA-specific processing step altogether. Although processing elements are already ordered in the FDA datastream, and there are no logical elements to recognize and ignore, extension of a system that supports FDA in order to handle FPDA does not represent a difficult task. The required extensions primarily involve division of the single-pass processing which can be used for translating FDA, into a two-pass process. The first pass over an FPDA document internalizes the structure, caches the content, and constructs relationships between objects based on their types and identifiers. The second pass uses this internalized structure to create the PDL representation. Experience developing the Xerox translator suggests that most of the existing techniques for processing FDA can be used directly with only minor modification. The amount of work involved in developing these extensions is mostly impacted by the tractability of the internalized representation chosen by the designer. Architecturally,

once an imager has implemented this additional support for handling FPDA, there is no reason to implement FDA-specific processing, since it is really just a subset of FPDA; the Xerox translator now handles both without difficulty.

These observations tend to indicate that the real question then, is not whether ODA can be translated to a PDL, but rather, whether or not a PDL is viable and desirable as the sole output data stream for imaging operations. The results of this activity suggest that use of a PDL provides superior separation of composition and imaging functions over FDA. The key distinction is that FDA actually leaves certain composition time decisions, such as precise placement and image rendition, rather loosely specified until execution of the document at imaging time. A PDL, such as Interpress, encourages composition time resolution of these decisions, thereby increasing the potential for faithful rendering of the composer's intent.

As noted in our interim report, the key difference between an FDA representation and a PDL representation of a given document is embodied in the data structures utilized and the complexity of processing required to compose an image based on those data structures. In a PDL representation, such as Interpress, the activity of printing document masters is generally amenable to linear execution which does not require maintenance of much context relative to the current page. This is desirable, since such a reduction of context decreases the amount of searching and computation that must be employed to determine the rendition and placement of an object in the image under construction. Clearly, it is possible to build an application profile that would lead to a highly simplified structure for FDA documents consisting of, perhaps, a single page containing a single block. However, the nature of the ODA architecture encourages development of hierarchical relationships between objects. These relationships can become rather heavily nested, thereby increasing dependencies between objects affecting calculation of relative placement and determination of attributes. Through use of a PDL for representation of an imageable document, we can collapse higher level calculations requiring nested contexts into direct imaging commands. Calculations such as these are better left to the composer, and through translation to a PDL result in a document description which provides for a more direct relationship with the imaging environment.

The results tend to indicate that there is serious reason to consider the use of a PDL as the representation of choice for final imaging of ODA documents. The evidence indicates that a properly constrained and sufficiently expressive PDL, such as Interpress, provides the ability to more than adequately represent ODA documents, while providing for a better separation of imaging and composition decisions than FDA.

A.3 GENERAL OBSERVATIONS REGARDING THE ISO 8613 STANDARD

A.3.1 Document Architecture

In general, the ODA document architecture as defined in ISO 8613, appears to provide good support for structuring of documents, particularly in the area of specifying logical relationships. The concept of well-factored entities such as layout styles and presentation styles provides a nice method of specifying and sharing information between separate objects in a document.

The layout capabilities in ISO 8613 are quite reasonable for most typical applications that would be encountered in office settings, but are over-constrained for many commercial publishing requirements. This could be considered a limitation with the ISO 8613 document architectures, mostly to the degree that it would limit acceptance of ISO 8613 in more sophisticated environments. It might be appropriate for the standard to be expanded with definitions to allow for broader domains of application. Two areas which are particularly amenable to extend flexibility are the angular positioning of frames and blocks, and the shape of frames and blocks. In the case of angular positioning of frames and blocks, enumerated limits could be relaxed, and specified only as fallback positions for systems which so choose to limit their implementation. In the case of the shape of frames and blocks, it would be valuable to consider relaxing constraints that currently limit the representation to a simple rectangular region (this is desirable for performing wraparound setting of text). It is recognized that these extensions are non-trivial. However, a degree of capability significant to sophisticated users in a variety of publishing environments could be realized through their incorporation.

Notably absent from the document architecture is any specification of colour metrics other than background, foreground, white, transparent, or colourless opaque. A significant requirement exists for extension of the standard to support definition of colour information, and colour mapping for grey-scale and monochrome devices. Consistent implementation of colour across sophisticated systems requires specification of these issues.

The ISO 8613 standard is also notably quiet on precise definition of errors and error handling. This is a significant problem in terms of ensuring that implementations provide consistent reactions to user inputs. Definition of likely errors and the appropriate action for a system to take upon encountering them is problematic, but it is reasonable to expect that the standard make an attempt at formally defining the most obvious situations.

One of the more obtuse concepts regarding ISO 8613, involves the need for a Document Application Profile (DAP). The concept appears to allow developers to make agreements regarding constraints on the level of support implemented for particular features specified in ISO 8613. This seems to thwart the goal of full and common docu-

ment interchange. It is desirable to assert through some well-defined means such as a DAP, that a document may have been created by a system capable of only a subset of ISO 8613. The fine point that is unclear, and perhaps which is not explained sufficiently in the standard, is whether conformance to any particular DAP is required for functionality conformant to ISO 8613 for a fully capable ODA system. A possible solution to this problem, is to clearly enunciate that limits imposed by conformance to any particular DAP should be handled as a dynamic set of constraints on a system, and that flexible systems should conform firstmost to the full ISO 8613 standard. If DAPs are to be handled as dynamic constraints rather than static limits, then it appears that an opportunity exists to develop a technique for electronically structuring and interchanging DAP information.

Programmatic management of object identifiers provides an interesting challenge in dealing with the ODA document architecture. The fact that identifiers are interchanged as character strings seems to add overhead to the file, since information that could otherwise be represented numerically in as little as one byte (octet), must be coded in multiple bytes corresponding to a character string. The challenge for the implementor in dealing with this representation of identifiers, is to determine the most efficient method of internalizing the identifiers and their semantics. Possible choices for internal representation include converting the identifiers to binary representations, disposing of them completely through use of structural associations that directly implement the semantics (internal pointer links between structural information which is resident in memory), or maintaining them in their character string representation. Clearly, there is no requirement on the implementor to actually internalize these identifiers as numeric character strings, although in the current Xerox implementation that is what is done (for expediency, the implementation both retains the identifiers and builds some structural links). It would be an interesting project to develop a binary representation for identifiers, since these sequences are likely to become fairly lengthy for non-trivial documents which may employ deep nesting.

A final thought regarding future directions for the document architecture, concerns its potential relationship to database systems. The current standard does not seem to provide atomic access to objects specified in a document; the external view of a document is strictly constrained to the document as a whole. This means that individual objects cannot identify association with other objects external to the document in which they are contained, nor can they be directly referenced through the standard, by other objects which are external to the document. The concern is that the attractive document structuring capabilities of ISO 8613 are inaccessible to external systems, except at the level of the document. Expanding on the previous discussion of identifiers, it may be desirable to revisit the topic of identifier semantics with regard for extensibility to enable association of interchange Data Units and external objects. This topic is particularly germane to evolving developments in hyper-media, distributed databases, and massive storage systems.

A.3.2 Raster Graphics Content Architecture

The raster graphics content architecture ISO 8613-7.2 appears to lack precision of specification and extensibility in several areas. Notably, there is no mention of how to specify colour information using the presentation attributes. Adjunct to the topic of colour, is specification of appropriate fall-back positions for grey-scale and monochrome devices.

An area where it appears there is a lack of detail in specification is regarding the topic of pel spacing and pel line spacing. The device independence implied by these attributes appears to be under defined in terms of providing for image fidelity across disparate implementations and devices. How should an implementation cope with adapting a pixel array to the dimensions specified by these attributes? What are reasonable fall-back positions? If a device chooses not to algorithmically adapt an image to these dimensions, should it image to the best of its ability, or should it refuse to print the image? This problem is all the more severe in the context of colour or grey-scale devices, and demands attention.

RGCA is otherwise straightforward in terms of simple raster imaging.

A.3.3 Geometric Graphics Document Architecture

Overall, the geometric graphics content architecture ISO 8613-8.2 is probably the best specified of the content architecture. This is in no small part due to its reliance on CGM, which is already a fairly mature standard. Only two areas of concern were identified in ISO 8632, and these concern the need for more detailed specification of potentially implementation-dependent rendering decisions.

The first problem involves rendering of hatch filled areas. If there is a precise specification of hatching other than the cursory suggestions in ISO 8632 regarding the angular fidelity of the hatched lines, it was not available. The primary concern with this problem, is that it is not only necessary to know the preferred angle of the hatched lines, but also the density relative to the scaled size of the image.

The second problem concerns rendering of stroke ends and stroke joints. ISO 8632 does not appear to provide a capability for precise control of rendering effects when strokes form a joint, or for the end of a stroke. Without specification, these non-trivial decisions default to the choice of the implementation, and this can introduce irregularities in rendering between different systems. Interpress provides a sophisticated capability for controlling rendering choices affecting stroke ends and stroke joints, and may be found to provide an interesting model for resolution of these details.

A.3.4 Character Content Architecture

The character content architecture ISO 8613-6.2 needs refinement to support typographically sophisticated systems. Undoubtedly, the completion of ISO/DIS 9541

will yield significant opportunity for definition of more sophisticated font and character handling in ISO 8613-6.2. However, at the present time problems exist in the areas of underlining, strikeout, script placement, embedded style specifications, angular positioning of text, justification, and text colour.

The underlining and strikeout operations share common problems in ISO 8613-6.2. There is no specification of how to place the lines relative to the text to which they are applied, and there is no description of what thickness to draw the lines. In the case of strikeout, there is no specification for graphic style (frequently, a single continuous line is adequate, but certain applications may require other styles). Ideally, these operations should be specified relative to the currently selected font. ISO/DIS 9541 enables this through the attribute ISO/SCORES as described in ISO/DIS 9541-5, 6.16.2.3. In the current version of the translator the offset and line width are set to values determined as percentages of the currently selected point size. The line weight of the stroke is constant regardless of the typeface style (i.e. bold or light). The values used in the translator are:

- offset below the baseline to the top edge of a single underline, or the upper line of a double underline: 14% of the current font size

- offset below the baseline to the top edge of the lower line of a double underline: 27% of the current font size

- offset above the baseline to the bottom edge of a strikeout: 31% of the current font size

- line width for all of the above: 6% of the current font size

Interpress provides two operators which are used for setting underlines and strikeout. The STARTUNDERLINE operator is called before the text sequence(s) to be imaged. After the text has been imaged through calling the SHOW operator, the MASKUNDERLINE operator is called with the thickness and offset. Double underlines are created by calling MASKUNDERLINE twice, and strikeouts are created by calling it with a negative offset.

Similar to the problems concerning underlining and strikeout, superscripting and subscripting suffer from no specification of offset calculations or rendering effect. Again, in a moderately sophisticated typographic system the ideal solution is to extract the relevant information from the currently selected font, and ISO/DIS 9541 provides for this. ISO 9541-5, 6.16.2.14 defines specification of variant scripts through the attribute ISO/VARSCRPT. In the current version of the translator, the font rendition is unmodified, and the offset is set to 30% either below the baseline for subscripting (PLD), or 30% above the baseline for superscripting (PLU).

Embedded style modifications pose significant problems, not so much because they cannot be accommodated, but because they are typographically unsophisticated and require the translator to make inline calls for new fonts while processing the text. The problems occur from the use of SGR to modify the currently selected font in terms of posture or weight. This technique is typographically unsophisticated because the change to the style specification actually results in a different typeface with different widths and other characteristics. It is arguable that fixed-pitch office printing fonts do not require this sophistication. However, the test cases utilize a variety of fonts including typographic faces, and it appears that the overall flexibility required for this mixture of typography would benefit from a consistent model rather than assuming reliance on embedded style codes.

Ideally, all of the typefaces used in the document should be specified in the document profile, thus allowing the translator to collect, construct, or substitute all the required fonts before processing the job. Instead of using SGR attributes, 1, 2, 3, 22 and 23 to control weight and posture, the formatter would simply utilize attributes 10 through 19 to select typefaces designated in the associated presentation attributes. The desirability of this technique is reinforced by observation that it closely parallels the common practice in Interpress of identifying all the fonts for a document in the Interpress preamble construction. This technique enables association of numeric frame variables with the font specifications thus alleviating inline font calls during execution of the master. Additionally, this practice enables preprocessing of the document for acquisition of resources thereby reducing computation intensive operations at print-time.

Justification and line-breaking algorithms are under-specified in ISO 8613-6.2. The standard must address in detail the topic of how to perform justification. How much inter-character and inter-word expansion/compression is acceptable? Should these values be derived from the selected font, or should they be static regardless of the font (i.e. should maximum/minimum spacing compression/expansion come from a structure specified in ISO/DIS 9541)? How should a system handle justification of short lines consisting, for example, of as little as a single word (the current wording in ISO 8613-62-5.2.2.d allows for spacing the word across the line measure by distributing the excess space in the inter-character space, which clearly, is not appropriate).

Justification is currently implemented in the translator by calculating the excess line width, and distributing it in the inter-word spacing (which is either expanded or compressed). Specific control of inter-character spacing is not implemented, and no special accommodations are made for punctuation. The translator will not justify a line if the total of the character widths is less than 75% of the specified line measure. While this does set some minimum threshold for determining the appropriateness of whether the line should be justified, it does not take into consideration fine decisions about exactly how the space should be distributed between words, characters, and punctuation. The translator does not currently limit inter-word space compression, but this needs to be addressed in the standard. A significant opportunity exists for definition of a preferred

justification algorithm in 8613-6.2. This is necessary in order to guarantee common text alignment results across different implementations.

It is valuable to note that the Xerox Interpress CORRECT operator is utilized by the translator in the justification process to overcome precision errors due to the lack of accurate widths information. Clearly, there is no substitute for having shared access to accurate widths data, but when this is unavailable, CORRECT provides the capability to produce pleasing results. End and centered alignment are less of a problem, and are implemented by either offsetting the start of the text by the excess width (which in some circumstances may be negative), or by using it to calculate an offset relative to the centre line of the display area.

A.4 PROBLEMS AND ISSUES WITH TG-PF TEST CASES

A.4.1 Specific Issues Regarding Processing of the Geometric Graphics Test Cases

datamod1:
One error was discovered wherein the word "graphics" was misplaced in the first pie slice. The coordinates specified in the text were 700,550 whereas the actual encoded coordinates were 620,550. This error in the position value causes the word to overlap the adjoining pie slice.

datamod2:
Four errors were uncovered in datamod2, one of which was corrected. The most significant error involved the sequence length for element "1 1 2 0", which was incorrectly specified as 0x01F4. Upon encountering the error induced by the apparently mis-specified sequence length, the Xerox ASN.1 parser recovered gracefully by translating everything up to the error. However, since the length error made accurate location of objects in the datastream impossible, it prevented processing of the GGCA content in "1 1 2 0", and the subsequent character content. The author was able to correct the test case by editing the sequence length to 0x01FA in the actual binary file.

The less severe errors in the test case specification involved the actual GGCA content for element "1 1 2 0". The second problem involved the first bar of the graph which represents the inflation rate for Italy, wherein the size of the graphic example did not correspond to the coordinates specified in the text of the test case. The problem is that the coordinates encoded in the test case for the bar were coincident with the coordinates for the bounding box of the graph, which were specified using a polyline. In the encoded master, the result of execution causes these lines to overlap, and the bar is therefore indistinguishable from the bounding box. Because of this problem, and that hatching would obscure the rest of the graphic, hatching was not implemented for this test case. The third error concerned the y axis text data tags, wherein the graphic example showed a string for " + 5.0" and the test case contained no correspondent string. The fourth error

was in the text representation of the metafile which was supplied with the test case, wherein the third pair of VDC coordinates for the point (120, 80) on the polyline representing the bounding box were not shown.

datamod3:
One error was discovered, wherein the text strings for the years "1986" and "1987" overlap because they are positioned at the same X coordinate.

A.4.2 Specific Issues Regarding Processing of the Raster Graphics Test Cases

In general the raster graphics test cases did not provide a realistic test of ODA embedded raster content. Given that the raster test cases as supplied can be printed in an ODA document only through a private agreement by PODA participants, this does not represent a true test of the ISO 8613 standard itself. In fact, it did not appear that the rasters conform to the description of a bitmap encoded raster image as specified in ISO 8613-7.2.

Despite these discrepancies, the rasters were successfully decoded, compressed and printed as separate Interpress files. The processing mostly used existing tools which are written in Mesa, and which run in the Xerox XDE environment on a D-series processor.

The processing steps each yielded interesting results in terms of the compression achieved from the original files. The rasters were first processed through a "C" program written by the author to pack the bytes into bits. These files were then edited into Xerox AIS (Array of Intensity Samples) files for display on a Xerox D-series machine. The AIS files were then processed using an existing Mesa program into Xerox RES (Raster Encoding Standard) compressed bitmaps. The overall result of this conversion yielded up to a 98% reduction in size based on the original pixel-per-byte encoded test cases. The detailed results are displayed in Table 1 below.

Test Case Name	PODA Test Case (octets)	AIS (octets)	RES compressed (octets)	Net Reduction (PODA/RES)
RASTER C Company Names	447'888	57'010	8'401	98,12%
RASTER C3 ODA Logo	657'152	83'168	18'903	97,12%

Table 1

A.5 PROJECT MILESTONES, RESOURCES, AND METRICS

The translator described in this paper has been under development by the author since late July 1987, and is largely a clean-sheet design due to having no previous experience with ODA. Much of the first two months was spent reading both the 1986 and 1987 drafts of ISO 8613. Additionally, related standards were acquired, and TGPF correspondence was assimilated. Design of the basic data structures and data flow continued through October, and the ISO Development Environment (ISODE) was acquired from Northrop Research and Technology Centre. ISODE is a toolkit of OSI development code including an ASN.1 scanner/parser. The ASN.1 scanner was extracted with slight modification, and coding was started in November with Microsoft C under MS-DOS.

The first intelligent file processing including a comprehensive logging facility was in place by mid-November. Extraction of text corresponding to page objects and creation of a simple Interpress master was operational by early December. By mid-January, structure processing was reliable and simple unformatted text could be positioned correctly on the page. Focus was then directed exclusively to character content processing. By early February the translator could sense all embedded commands and was approximately 60% capable of correctly processing the supplied test cases. (Lack of formal definition for font handling contributed significantly to the lack of completeness).

The interim segment of the study was completed at the end of February, and work continued on adding support for GGCA, RGCA, and FPDA. The original program was heavily reworked from a structural perspective through the month of March, and support was added for typographic fonts widths, and interactive as well as batch operation (the previous version supported only batch processing). The raster graphics files were reviewed, but due to a misunderstanding regarding the structure, were not correctly decoded until May.

Through April, work was directed at designing and implementing support for FPDA, including separate functions for internalizing the document and creating output. Support mechanisms were developed and handling content caching (structure is retained in memory, and content is cached to a temporary file), and essential object identifier operations (such as procedures for deriving the identity of a superior object, generating the identity of a subordinate object, and internal search routines for matching objects in a list). Work was also begun on implementing GGCA/CGM support, and by the end of April, FPDA was stable.

At the time of this report, the translator is stable for essential subsets for FPDA, FDA, GGCA and CCA. Raster graphics were decoded, compressed, and printed, but RGCA itself is formally untested (this is probably not a significant issue, given that the rasters were correctly decoded despite the fact that they were not contained in an actual ODA structure). The program consists of approximately 14,000 lines of code, not including libraries or other linked object files. Of the 14,000 lines of code, approximately 1,200 were derived

from the ISODE toolkit. The translator utilizes the Xerox WRC Interpress Toolkit for Interpress output; the identical libraries are available publicity through the Xerox Systems Institute.

A.6 OVERVIEW OF THE TRANSLATOR STRUCTURE

The ODA to Interpress translator is written in the C language and is structured in four layers. The top layer is responsible for program control, and provides a simple interactive or batch user interface (operational mode is under command line control). The second layer contains the internalizer (ODA parser) and the Output Generator (Interpress creator). The internalizer's ODA parser provides processing for capturing the ASN.1 encoded datastream into an efficient internal representation. The Output Generator works from this internal representation to create the Interpress master. The third layer handles content portions through modular components that implement the appropriate content architecture. The fourth and lowest layer implements low level functions, and includes the ASN.1 stream scanner, the ODA identifier mechanism, the cache control mechanism, and the font widths mechanisms.

Upon program startup, a file of user commands is scanned for information regarding fonts and dynamic program capabilities. The program initializes itself relative to these specifications by internalizing the widths for the requested fonts, and by building a temporary cache on the local disk. If the program is in interactive mode (the default), a variety of commands are provided which allow the user to internalize a named ODA document, analyze its internalized structure, create Interpress output from the internalized representation, reset the environment to allow processing of subsequent files, and to quit the program. If batch processing is selected by a command line switch, a file name for an ODA document must be supplied, and processing results in the creation of an Interpress master.

The program performs translation in two passes. In the first pass, the internalizer extracts objects from the ODA datastream in the form of ISODE Presentation Elements using the ISODE ASN.1 PSAP scanner. Each of Presentation Elements is analyzed semantically, and internalized into a linked tree structure based on its identity as an ODIF Interchange Data Unit. Structural information is maintained in resident memory in the form of a context. Sequenced data associated with content portions is cached to a buffered temporary file by the Cache Manager. The Cache Manager returns a handle which is stored with the content information that can later be used to retrieve the sequenced content data during output processing (or for editing). This design allows low-overhead access to structural information, and large linear storage for potentially long sequences of content information (such as GGCA metafiles and RGCA bitmaps).

A context is an abstraction used by the program to associate a collection of attributes with an object. Context include information about the type of the object, its identity, all of the rendition information and positioning information associated with it, and the object's

relationship to other objects, such as subordinates. Contexts are maintained by the Context Manager. The Context Manager accommodates requests for allocating and deleting contexts, inheriting information from superior contexts, and managing the relationship between contexts.

The Context Manager provides a call-back mechanism to iterate a path on contexts conforming to the associated identifiers. A client can ask the Context Manager to iterate a path of contexts either from a leaf towards the root (as might be the case when searching for an attribute), or from the root towards a leaf (as might be the case when determining the current position at the start of a block). The Context Manager relies on the ID Manager to handle the task of matching and constructing identifiers, and on other routines for handling memory allocation and deallocation. The ID Manager is a separate process that provides simple operations for constructing and interpreting ODA identifiers expressed in the form of numeric character strings.

The Output Generator performs the second pass processing which results in creation of an Interpress master. The Output Generator utilizes the internalized structure by starting at the document layout root, and processing layout objects in their sequential order based on the associated identifier. As the processing encounters layout objects whose subordinates are content portions, the structure for the associated content portion is located, its sequence data is retrieved from the cache, and it is processed by routines for the appropriate content.

Separate processes are implemented for CCA and GGCA content. Character content processing is implemented for a significant portion of ISO 8613-6.2; only embedded commands that dynamically affect the style of the current font are unimplemented. Geometric Graphics content processing is currently capable of parsing all of CGM, but only the minimum set of capabilities required for processing the supplied test cases is actually implemented for output creation.

A.7 CONCLUSIONS

The translation of ODA to a PDL, in this case Interpress , is a tractable problem for which the study has yielded valuable results. Firstmost, it has been demonstrated that a sufficiently powerful and expressive PDL can effectively render imageable ODA objects for printing. Secondly, it has been demonstrated that creation of pdl output directly from FPDA is not only possible, but actually desirable. Thirdly, it has been demonstrated that FDA is not requisite for processing imageable content. An finally, the study has demonstrated the general strength of ISO 8613 and its affiliated standards, particularly in terms of its ability to describe structured documents that contain information which can be imaged.

APPENDIX B

TRANSLATION OF ODA TO POSTSCRIPT, INTERIM REPORT

Mr. J. Foley, Adobe System Incorporated

June 1988

B.1 INTRODUCTION

This is an interim report from the Adobe Systems contributor to ECMA TC29-TGPF, regarding the current status of the ODA-to-PostScript translator, and the information acquired to date regarding the goals of this translation study.

The three goals of this translation study are:

- inform the current PDL vendors of the capabilities of ODA for describing printable documents;
- evaluate the current PDLs for adequacy in supporting ODA Formatted Form;
- obtain feedback from the current PDL vendors regarding extensions needed by ODA for the support of publishing-quality printing.

Information pertinent to each of these three goals, in this order, is provided after the next section.

B.2 CURRENT STATUS OF THE ADOBE PROJECT

Working implementation of the ODA FF-to-PostScript translator and imager has been delayed both by the difficulty of implementing the Layout Structure analyzer, and by other project demands on this contributor's time. A recursive-descent parser for ODA Layout Structure has been implemented, but is still not working reliably. Thus, completion of the content architecture translators has been delayed.

However, work on the content imagers has yielded information that is of consequence in meeting the above three goals. The greatest amount of information and suggestions is available regarding the Character Content imager; the least complete content imager is for Geometric Graphics.

The following interim report should be of use to TC29.

B.3 OBSERVED CAPABILITIES OF ODA

The first objective of the study was to inform the current PDL vendors regarding ODA and its features.

This contributor greatly increased his familiarity with ODA during the course of this study. The following section takes note of important features of ODA, and in some cases contrasts them with the PDL approach.

B.3.1 Overall Layout Model

The possibility of overlapping frames seems to provide adequate flexibility for most "office document" layout tasks.

One obvious limitation is the orthonormality required of frames. This restricts the domain of application of the overall layout model.

Another limitation on flexibility is the approach of enumerating finite lists of choices, e.g., 0, 90, 180 and 270 degree angles only. This means that such descriptions can only be adequate for specific domains of application, i.e., those which use only the specified choices. The language approach provided in PDLs seeks to assure that a broad range of application domains may be served, particularly including new application domains that may be invented AFTER the PDL is established.

However, it should be noted that implementing the finite-choice form of document description may be easier, up to a point. This may be suitable for descriptions that are meant to be specific to particular application domain(s). It may be acceptable to the ODA community to accept the use of ODA with only "office documents".

B.3.2 Character Content Architecture

The ODA Character Content Architecture seems capable of achieving most typographic effects, in environments with a suitable choice of fonts.

However, it does not seem to provide means to GUARANTEE particular effects unambiguously and in an implementation-independent way. Greater detail on this issue is presented in section 5.2 below.

One particular feature that the Character Content Architecture does not provide, that can be extremely useful in typical office documents, is the ability to provide angled column headings and labels on charts and tables. One should not have to use Geometric Graphics to achieve this extremely useful typographic feature.

The concept of font substitution remains problematical from Adobe's viewpoint. Clearly, it would be wonderful to find a good way to do it, and the efforts of both ODA and the 9541 Font Interchange Standard are worthwhile. But it remains to be seen if it can be made typographically acceptable.

B.3.3 Raster Graphics Content Architecture

Each basic component may contain only a single content portion. Therefore, there can be no "stitching" problems, at least within a single block.

Only rectangular arrays of rectangular pels are allowed. Non-square pels are allowed. This seems adequate for most purposes.

Pels extending beyond the basic layout object are clipped, and rectangular clipping is provided.

Multiple "codings" are allowed in the processable form (bitmap, Group3, and Group4).

B.3.4 Geometric Graphics Content Architecture

A geometric graphics picture consists of a single CGM file containing a single CGM Picture. There is essentially 100% compatibility with CGM, an established graphic standard. Conversely, a full CGM interpreter (for CGM binary encoding only) must be built into any ODA imager.

CGM is sufficiently rich to support most "office" or business graphics requirements. It leans more toward stereotypical "computer" graphics (hatch patterns, vectors) than toward "graphic arts" graphics typically used in publishing (continuous-tone/half-tone colour variations, smooth high-order curves).

B.4 ADEQUACY OF POSTSCRIPT FOR IMAGING ODA FORMATTED FORM DOCUMENTS

The second objective of the study was to create a proof-of-principle demonstration that the current PDL's could adequately transcribe ODA Formatted Form documents. The following provides some observations relevant to this goal.

B.4.1 Overall Imaging Model

Clearly the biggest challenge in writing this translator is the creation of the ODA-parsing front end. My experience indicates that one would be much better off using general ODA toolkits rather than attempting to build the ODA system from scratch.

It is equally clear, in my view, that adding a Formatted Form-to-PostScript translator to a pre-existing ODA system would be very straightforward. That project would involve

only as much work as adding a PostScript driver to an ordinary application. For instance, writing the translator for character content involves approximately the same effort as writing a PostScript printer driver for a program like "WordStar".

Adding a PostScript translator/imager in this way would be tremendously simpler than implementing a full RIP (raster image processor) and imaging technology. That task is of a completely different order of magnitude.

The imaging capability of the PostScript language and printers is definitely a superset of the functions needed by existing ODA content architectures. Therefore, given the above observations it would seem that creating an ODA-to-PostScript driver is a practical way to implement ODA Formatted Form imaging.

B.4.2 Character Content Architecture

The Character Content imager in the ODA to PostScript translator is very straightforward:

- A certain amount of formatting functionality has been left by ODA for the imaging process (the "implementation-dependent" features). The resolution of the implementation decision has been left flexible. The translator reads a "profile" file before starting, so that the user may configure the translator to his taste on these issues (e.g., how to do crossed-out graphic rendition).

- The remainder of the task is completely equivalent to writing an emulator for some specific printer. I have been involved with PostScript emulation of HP PCL, Diablo 630, and APPS-5 imagers. Translation of ODA Character Content character strings is not particularly different.

- There is a small challenge in writing emulators efficiently, since it is a programming task that can be done many different ways. For the purpose of this study project, I am using a fairly simple approach that will probably be less efficient than if I had spent more time on it. However, optimization can be added later.

B.4.3 Raster Graphic Content Architecture

The Raster Graphic Content imager in the ODA to PostScript translator must perform two tasks:

- The image data must be provided to the PostScript interpreter in an appropriate form. The document application profile for this study requires only the raster graphics bitmap format to be supported in the test translator. In the general case, facsimile format must also be supported. Adobe Systems has most of Group3 and Group4 implemented in software, although it is not included in any of our current products. (Please note that the Study Translator has not progressed to the stage of

actually testing this capability in the context of ODA raster graphics content translation).

- The image data must be preceded by a very small amount of PostScript code, which then causes the PostScript interpreter to read the image data directly out of the data stream. No further "translation" is needed.

The current raster graphics content architecture allows only for bi-valued pels. If such images happen to be transmitted at the resolution of the imaging device, the PostScript interpreter will automatically take advantage of this fact to print the images very efficiently. If the device resolution does not match, the PostScript interpreter is able to do a very high quality rendering by using its built-in halftoning abilities.

When colour and gray-scale raster graphics are added to ODA's capabilities, PostScript interpreters will be able to print them with equal ease, as PostScript printers with gray-scale, halftone, and colour capabilities are already on the market.

B.4.4 Geometric Graphic Content Architecture

The Geometric Graphics Content is the largest job to translate, because CGM has a complete graphic and text imaging model.

Most elements of the CGM translate very readily and efficiently into PostScript language. For instance, rotated and anamorphically scaled text is allowed in CGM, and the PostScript interpreter does this easily. Polylines may also be imaged very efficiently, thanks to the PostScript language's procedure capability.

A few features of CGM require the translator to be more clever. For instance, Polygon-Sets (not required in the Study Translator) allow some edges to be visible, while others are invisible. This result can be described several different ways in PostScript, and care must be taken to use an efficient representation. However, I have not encountered any features that do not have a straightforward and efficient mapping into PostScript language commands or command sequences.

As with the Character Content architecture, Geometric Graphics leaves many aspects of the page image under-specified from the perspective of a PDL. For instance, the issue of line joins and end caps is ignored. These attributes, needed by the PostScript interpreter, must be supplied either in the translator itself or in a profile file.

B.5 SUGGESTED EXTENSIONS OF ODA CAPABILITIES

The following is simply a list of observations and suggestions for possible extensions to ODA and its content architectures, which might improve ODA's usefulness in electronic publishing and other applications more sophisticated than those usually categorized as

"office applications". Some of these comments are preceded by a sub-clause number in parentheses, from the July 1987 DIS 8613.

B.5.1 Overall Layout Model (Part 2)

The inability to provide rotated frames is a serious lack. In particular, one should be able to specify angled column headings and labels on textual charts and tables without having to resort to Geometric Graphics.

The designers of ODA document architecture might also consider issues of labels on graphics (e.g., pie charts). If the labels are provided as part of the CGM graphic, they are merely part of a unitary Picture, with no discrete semantics. On the other hand, if they are blocks of Character Content overlapped on the Geometric Graphic image, then they might be recognized as a special type of content. Rules might be provided for "floating" them on the graphic, so that a layout process could optimize their positioning. For instance, changing the width of a layout column might move the labels on a pie chart rather than causing the graphic to actually change size.

The text capability from CGM has many desirable features that Character Content lacks, for instance the ability to scale and rotate. On the other hand, it is burdened with a non-typographic model including character boxes and foreground and background colours. It would be advantageous to provide a single enhanced text architecture, and discourage use of CGM text in geometric content.

- (3.3.5) "Borders are under-specified, especially with regard to the placement of thick lines, and the specification and behavior of non-solid line types.

- (7.3.2) The page position may move around on the nominal page depending on various factors. In some circumstances, it is to be "positioned such that the possibility of information loss is minimized". This is ambiguous.

- (3.3.2.2) Handling or errors in the document architecture is unspecified. In particular, if frames are NOT entirely contained within superior frames, what is the desirable action?

B.5.2 Character Content Architecture (Part 6)

- (5.2.2 and later) The "justification process" is under-specified. The layout process should decide exactly how to distribute excess whitespace among interword and intercharacter spacing. Since ODA allows this process to be left for the imager to perform, it means the same "formatted form" document may be sent to two different printers of EQUIVALENT functionality and yet result in quite different page images. This is not acceptable in publishing applications. It should also be noted that "excess whitespace" may be negative in some line-justifying circumstances.

- (VARIOUS) There is a set of features which should be interpreted quite differently for constant-spacing and variable-spacing fonts. In many cases, control codes/presentation attributes are given which would be suitable for each type of font. However, it is usually not observed that one control code or presentation attribute is suitable for constant-spacing fonts and unsuitable for variable-spacing fonts (or vice-versa). The following two comments describe some of these situations.

  - (5.2.4 subscript/superscript) The simple forms that move up or down one half line, and do not recognize a change in character size or width, are quite suitable for fixed-pitch fonts and daisy-wheel printing. They are very under-specified if they are to be used with typographic (variable-pitch) fonts. Attributes which must be specified in such a case include: how much to move up or down; what size character to use if the font does not include special sub/superscript graphic characters; and whether the line-justification process takes the potential change in character width into account.
    Conversely, there are control codes that provide (somewhat) more specificity. The would be difficult to implement on most daisy-wheel style printers, yet the Character Content Architecture specification does not seem to make normative statements about whether and how they should be implemented across a range of system capabilities. The result may be inability to depend on the specific imaging results.

  - (11.1.8 character spacing) The different controls available for controlling the character spacing behavior of each of constant-spacing and variable-spacing fonts should be discussed and contrasted.

- (6.1.2) In the case where "posture" is changed without changing font, it should be possible to specify what obliquing angle to use.

- (6.1.3) Underlining and double underlining are under-specified, with regard to: line width, position, and variation of these attributes for different fonts within a document and on a single line.

- (6.1.5) Textual image inversion is not a typical typographic feature. In most typographic fonts, the line box is a figment of the formatter's imagination, not a characteristic of the font. Therefore, in order for the imager to perform image inversion, the back and forward extents of the line box must be specified.

- (6.1.6) Cross-out is a useful feature which is quite under-specified. Cross-out may be done with slashes, dashes, equal signs, x's, or continuous lines, among other ways. If done with continuous line, width and position must be specified. If done with character overtype on variable-spacing fonts, one must specify character positioning rule, e.g., centered on each character or start-aligned with each character. Also note the lawyers frequently require a double crossout, and use " = " signs to achieve it.

- (6.2.3) It is good that invocation of typographic font overrides attributes of "weight" and "posture". But it SHOULD be possible to invoke a constant-spacing typographic font and still specify weight and posture, since this is exactly the category of font upon which it makes sense to perform algorithmic weight and posture modifications.

- (11.1.5) It seems unspecified whether graphic renditions must nest or are independent. For example, when a graphic rendition is turned on AFTER a subscript or superscript control, does it remain effective after the sub/super is returned to normal?

- (12.1.1.3.3 and 12.2.1.1) Alignment really should not be left for the imaging process under any circumstances. It is formatting, plain and simple.

Finally, the methods of dealing with errors in Character Content seem to be unspecified. This should be improved, because the method of dealing with errors will affect imaging. For example, if font substitution is a formatted form character content portion causes a string to extend beyond the end edge and/or beyond the edge of the block, what is the appropriate result?

B.5.3 Raster Graphics Content Architecture (Part 7)

The ODA designers have already taken note of the need for support of colour and grayscale raster images.

- (5.4) The section on positioning does not define the reference point for positioning of an individual pel. It is "assumed to be within the image of the pel". Greater precision is useful, and will be absolutely necessary for the Tiled Raster Graphics architecture. We propose the following model: Each pel is considered to be a half-closed region, "bounded" on the left and bottom edges including the lower left corner, and "open" on the top and right edges including the three remaining corners. Thus, every geometric point in the raster graphic is unambiguously contained within exactly one pel. Further, let there be a precise positioning point for the pel. This should logically be either the center of the pel or the lower left corner (which is included in the pel). There are implementation advantages to using the corner rather than the center, and this is what we recommend.

- (5.4) Rotated images (beyond just 90-degree orientations) should be allowed. This is absolutely required for advertising and newsletter applications. Skewed images (mapped to non-rectangular parallelograms) are less frequently used but should also be considered

B.5.4 Geometric Graphic Content Architecture (Part 8)

The appearance of the various marker types is under-specified.

CGM Text uses a variety of non-typographic attributes, including character boxes and foreground and background colors, but provides some attractive features like rotation and scaling.

It is useful to be able to clip to non-rectangular areas. This is especially necessary for advertising applications in publishing.

CGM leaves a number of attributes unspecified or under-specified from the perspective of a PDL. This is properly the topic of a separate paper. Examples include line joins and end caps.