# X IMAGE EXTENSION CONCEPTS

**Revision/Update Information:** V2.4

Joseph G. Mauro
Image Processing Group
Digital Equipment Corporation
June 10, 1988

**digital equipment corporation maynard, massachusetts**

# Contents


PREFACE v

CHAPTER 1 INTRODUCTION 1–1

1.1 SCOPE AND PURPOSE 1–1

1.2 JUSTIFICATION 1–2

1.3 REQUIREMENTS 1–2

1.4 GOALS 1–2

1.4.1 Minimal Sufficiency 1–2

1.4.2 Application Classes 1–3

1.4.3 Display Systems 1–3

1.4.4 Extensibility 1–3

CHAPTER 2 IMAGING CONCEPTS 2–1

2.1 TRANSFER MODEL 2–1

2.2 VISUAL PERCEPTION 2–3

2.3 IMAGE CAPTURE 2–3

2.4 IMAGE SYNTHESIS 2–3

2.5 MATHEMATICAL MODEL 2–4

2.6 IMAGE REPRESENTATION 2–4

2.7 IMAGE COMPRESSION 2–6

2.8 IMAGE VS TEXT & GRAPHICS 2–6

2.9 IMAGE DATA OBJECT 2–6

2.10 IMAGE PROCESSING 2–7

2.11 IMAGE RENDITION 2–7

2.12 IMAGE DISPLAY 2–8

2.13 IMAGE REVISABILITY 2–8

2.14 IMAGE APPLICATIONS 2–9

2.15 IMAGE SERVER 2–9

## Contents


**CHAPTER 3 EXTENSION ARCHITECTURE CONCEPTS 3–1**

**3.1 EXTENSION GUIDELINES 3–1**

**3.2 EXTENSION LEVELS 3–2**

**3.3 GLOBAL CONCEPTS 3–3**

**3.3.1 Final vs Revisable Form 3–3**

**3.3.2 Image Data Representation 3–3**

3.3.2.1 Aternate Data Organizations and Pixels • 3–4

3.3.2.2 Discrete vs Continuous Pixels • 3–4

**3.3.3 Layout & Coordinate Systems 3–5**

3.3.3.1 Coordinate Systems • 3–5

3.3.3.1.1 Spatial Coordinate Systems • 3–5

3.3.3.1.2 Temporal Coordinate Systems • 3–6

3.3.3.1.3 Chrominance Coordinate Systems • 3–6

3.3.3.2 Layout • 3–7

3.3.3.2.1 Spatial • 3–7

3.3.3.2.2 Temporal • 3–7

**3.3.4 Image Structure – Subsets 3–8**

**3.3.5 State Information – Process History 3–8**

**3.3.6 Extension Relationships 3–8**

3.3.6.1 To Core X • 3–8

3.3.6.2 To Other Extensions • 3–8

3.3.6.3 Multiple Connections, Clients, and Resource Sharing • 3–9

3.3.6.4 To Text and Graphics • 3–9

**3.4 SERVICE CONCEPTS 3–9**

**3.4.1 Service Goals – Reasoning & Implications 3–9**

3.4.1.1 Range of Images • 3–9

3.4.1.2 Range of Applications Needs • 3–10

3.4.1.3 Range of Server Capabilities • 3–10

3.4.1.4 Display Device Independence • 3–11

3.4.1.4.1 Server Based Rendition • 3–11

3.4.1.4.2 Image Data Organization • 3–11

3.4.1.5 Image Data Transport • 3–12

3.4.1.5.1 Image Data Compression • 3–12

3.4.1.5.2 Transport Granularity • 3–12

3.4.1.6 Image Processing • 3–13

3.4.1.6.1 Enhancement Processing • 3–13

3.4.1.6.2 Rendition Processing • 3–13

3.4.1.6.3 Enhancement – Rendition Relationship • 3–14

3.4.1.6.4 Other Forms of Processing • 3–14

3.4.1.6.5 Processing Policy • 3–14

3.4.1.7 Operations Pipelining • 3–15

3.4.1.8 Server Display Hardware Control • 3–16

3.4.1.8.1 Shared Hardware Device Control • 3–16

3.4.1.8.1.1 LUTS and Bit Map • 3–16

3.4.1.8.1.2 Brightness – Contrast – Hue – Saturation • 3–16

3.4.1.8.2 Use of Image Specific Accelerators • 3–17

3.4.1.8.3 Image Capture Hardware – Live Video • 3–17

3.4.1.8.4 Extensibility • 3–18

3.4.2 **Client–Server Model** 3–18
3.4.2.1 Server Resources • 3–18
3.4.2.2 Service Classes • 3–18
3.4.2.3 Service Events • 3–19

3.5 **TRANSFER MODEL VS X WIRE** 3–19

**CHAPTER 4 EXTENSION PROTOCOL ELEMENTS** 4–1

4.1 **SERVICES** 4–1
4.1.1 **Device Independent Service Group** 4–1
4.1.1.1 Session Class Services • 4–1
4.1.1.2 Resource Class Services • 4–1
4.1.1.3 Transport Class Services • 4–2
4.1.1.4 Statistics Class Services • 4–2
4.1.1.5 Enhancement Class Services • 4–3
4.1.1.6 Process Control Class Services • 4–3
4.1.2 **Device Dependent Service Group** 4–4
4.1.2.1 Rendition Class Services • 4–4
4.1.2.2 Import/Export Class Services • 4–5
4.1.2.3 Display Class Services • 4–5
4.1.2.4 Device Control Class Services • 4–5

4.2 **RESOURCES** 4–6
4.2.1 **Image Object – IOB** 4–6
4.2.2 **Image Enhancement Context – IEC** 4–6
4.2.3 **Image Process Context – IPC** 4–7
4.2.3.1 Region of Interest – ROI • 4–7
4.2.3.2 Control Processing Plane – CPP • 4–8
4.2.3.3 Templates – TMP • 4–8
4.2.4 **Image Control Context – ICC** 4–8
4.2.5 **Image Display Context – IDC** 4–9

4.3 **SERVICE OPERATION** 4–10
4.3.1 **Flow** 4–10
4.3.2 **Requests and Replies** 4–11
4.3.3 **Errors** 4–11
4.3.4 **Events** 4–11

**CHAPTER 5 TERMINOLOGY** 5–1

5.1 **CORE PROTOCOL** 5–1

5.2 **EXTENSION PROTOCOL** 5–2
5.2.1 **Image Definitions** 5–2
5.2.2 **Image Characteristics** 5–4
5.2.3 **Visual Types** 5–5
5.2.4 **Image Classes** 5–5
5.2.5 **Image Data Organization** 5–6
5.2.6 **Image Data Encoding** 5–8
5.2.7 **Image Data Presentation** 5–9
5.2.8 **Image Server Hardware** 5–10
5.2.9 **Image Data Layout** 5–11

## Contents


| | | | |
|---|---|---|---|
| | 5.2.10 | Three Space Color | 5–11 |
| | 5.2.11 | Image Data Processing | 5–12 |
| APPENDIX A | RELATED TOPICS | | A–1 |
| A.1 | IMAGE DATA INTERCHANGE | | A–1 |


## FIGURES


| | | |
|---|---|---|
| 2–1 | IMAGE TRANSFER PROCESS | 2–2 |
| 2–2 | VISUAL TYPE – VS – IMAGE CLASS | 2–5 |
| 3–1 | IMAGE EXTENSION – VS – CORE PACKAGE | 3–2 |
| 3–2 | SERVICE CLASS GROUPS | 3–19 |
| 3–3 | TRANSFER MODEL & X WIRE LOCATION | 3–20 |
| 4–1 | OPERATIONAL MODEL | 4–10 |
| A–1 | IMAGE DATA INTERCHANGE | A–1 |


## TABLES


| | | |
|---|---|---|
| 5–1 | Core Terms | 5–1 |

# Preface

The core X architecture is a major step in the definition of open, interactive applications display services. It has, in essence, pried apart the tightly coupled, private interconnect between application and display subsystem by specifying a structured set of display services.

The core X protocol recognizes the need for minimally sufficient capabilities, and, as such, leaves the specification of robust application domain specific display services to extension packages. These extension packages relate to the core X protocol by way of an extension mechanism specified in the core X protocol.

This document contains conceptual material useful in understanding the proposed X image extension package including the X image extension protocol (XIEP), the protocol binding in the X image extension library (XIE_LIB), and the X image extension toolkit (XIE_TK). The X image extension protocol (XIEP), X image extension library (XIE_LIB), and X image extension tool kit (XIE_TK) are defined in separate documents. It is assumed that the reader is familiar with the core X architecture.

## Related Documents

- X11 Window System Protocol

  Describes the core X protocol functions and the conventions to be used in specifying extension packages.
- X Extension Package Guidelines

  Sets forth guidelines for extension packages.
- X Image Extension Protocol (XIEP)

  Describes the proposed X Image Extension Protocol. The X Image Extension Protocol Specification is the companion document to this concepts document.
- X Image Extension Lib (XIE_LIB)

  Describes the contents of the image extension library level.
- X Image Extension Toolkit (XIE_TK)

  Describes the contents of the image extension toolkit level.

## Participants


- AUTHOR:
  - Joseph Mauro
  - Digital Equipment Corporation
  - MS: MK02–2/K03
  - Continental Blvd.
  - Merrimack, New Hampshire 03057
  - 603–884–0528

- CONTRIBUTORS:
  - John Weber (Digital)
- INVITED REVIEWERS:
  - Mark Sornson (Digital)
  - Bob Milling (Digital)

## Revision History

rev v1.0 July 1, 1987 first protocol draft

rev v2.0 September 1, 1987 split concepts from protocol

rev v2.1 December 1, 1987 rework based upon internal review

rev v2.2 December 10, 1987 rework based upon internal review

rev v2.3 April 10, 1988 rework based upon protocol spec

rev v2.4 June 10, 1988 minor rework for first MIT release

## Organization

This document contains five chapters and one appendix.

- Chapter 1, Introduction, explains the purpose of the X Image Extension
- Chapter 2, Imaging Concepts, describes basic digital imaging concepts
- Chapter 3, Extension Architecture Concepts, describes the underlying concepts of the image extension
- Chapter 4, Extension Protocol Elements, defines the basic elements of the extension protocol
- Chapter 5, Extension Lexicon, defines the lexicon used in the extension
- Appendix A, Related Topics, addresses some closely associated topics

# Chapter 1

# INTRODUCTION

## 1.1 Scope and Purpose

The X image extension package is a supplement to the core X package. The X image extension package is designed to provide applications with robust support for visually interactive digital image enhancement and display operations. The purpose of this document is to provide conceptual material useful in understanding the image extension package and its wire protocol.

The core X architecture defines a client–server based interactive application display services model including components and their relationships (layering and distribution). The proposed X image extension package follows the core X architecture by specifying imaging components and their relationships within the X model.

Although the proposed X image extension package defines image specific extensions at all levels of the core X model, it focuses primarily on extensions to the wire protocol as these are fundamental to all other levels. The X Image Extension Protocol (XIEP) is intended as the format by which client and server portions of an imaging application communicate imaging service requests and replies across the X wire. As such, this concepts document focuses primarily on providing the reader with the background material and reasoning necessary to understand the XIEP protocol specification.

XIEP is intended for use by a wide range of digital imaging applications dealing with various classes of images, and requiring varying degrees of server support. The classes of images include:

- Static bitonal digital image data (scanned documents)
- Static continuous tone monochromatic digital image data (gray scale)
- Static multi spectral digital image data (color)
- Time varying versions of each of the above

XIEP is intended to support applications that simply display images, through those that involve highly interactive, human directed image processing. As a future component of a developing industry standard, XIEP will serve the needs of multiple vendors and their end users.

## 1.2 Justification

The core X protocol chooses not to provide robust support for certain application areas in order to remain minimal in extent. Instead, it provides a mechanism for adding protocol extension packages for domains such as imaging.

Analysis of imaging applications and their display (human interaction) needs indicates that there are several broad areas which X image servers and an X image extension protocol (XIEP) should provide for including:

- Device independent display interaction
- Wire image data organization and compression
- Interactive image enhancement processing
- Control of tightly coupled image capture hardware
- Use of image specific accelerators

This work is justified by the standardization of and increase in imaging display capabilities provided to applications, and by the off loading of many operations to the server.

## 1.3 Requirements

It is a requirement that XIEP comply with the architectural model set forth by X. Compliance will be measured by the similarity of architectural concepts.

It is a requirement that XIEP comply with the core protocol and protocol extension mechanisms set forth by X.

## 1.4 Goals

XIEP is designed to meet the human interaction needs of imaging applications by providing for the communication of these needs between X client and X server. XIEP has as its goals the inclusion of a minimal set of functions that support the variety of image applications classes and provide access to the full resources of the display system, and support for image extension package extensibility.

### 1.4.1 Minimal Sufficiency

XIEP should contain a minimal set of functions that support basic imaging operation classes including image object transport, interactive image enhancement processing, image display rendition, and tightly coupled image capture device control. The set should be minimal in content with as little overlap in functions as possible. The protocol must support the construction of X image extension toolkit (XIE_TK) and client specific higher level imaging functions.

### 1.4.2 Application Classes

XIEP should be sufficient to support applications that simply render images for display purposes, as well as those that provide for high visually interactive, human directed, image enhancement processing. XIEP should support rendering and processing for applications that deal with static and time varying bitonal monochrome, continuous tone monochrome (gray scale), and continuous tone multi−spectral (color) digital images.

XIEP should allow X application clients to perform image rendition themselves (prepare images in final form for display), and request X servers to display them, as is. This is essentially the current situation under X11. In addition, XIEP should allow application clients to defer rendition to the X server implying that the image extension package server will handle all device specifics of image rendition.

### 1.4.3 Display Systems

Display system architectures range from simple terminals connected through serial lines to host processors, through the modern day workstation with tightly coupled processor and display subsystem, to highly specialized interactive graphics and image processing workstations. XIEP should provide the means for using specialized display subsystem architectures and image rendition and processing hardware.

### 1.4.4 Extensibility

While it is a goal of the X image extension package to itself be extensible, unlike the core X11 protocol, the package extensibility will not be achieved via extensibility of the X image extension protocol (XIEP). Instead, it will be goal of the X image extension toolkit (XIE_TK) to provide for extensiblity. Thus, demands for additional high level services will be met over time via addition of services at the tool kit level versus additional of new protocol functions. The evidence suggests (in particular, the wide range of exiting popular enhancement functions, and the continuing development efforts in this area) that with the correct choice of low level server functions, this strategy can be made to work.

# Chapter 2

# IMAGING CONCEPTS

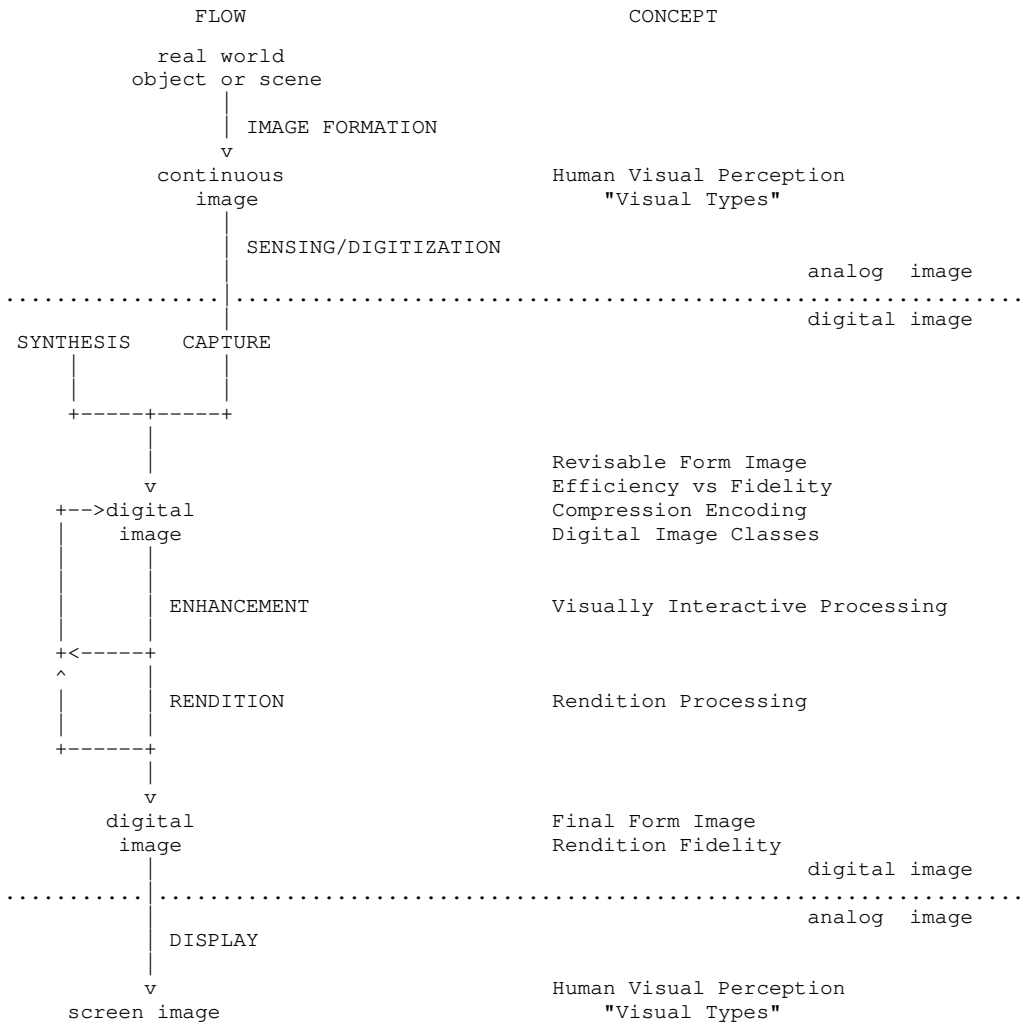
This chapter presents background material regarding the definition and sources of digital images, how they move through applications, and the states they pass through in transit. This material constitutes a simple model of digital imaging which is useful in presenting the extension concepts.

## 2.1 Transfer Model

The overall operational model for many image applications is that of a transfer process. The transfer process begins when either a real world continuous image is captured and converted into a digital image, or a digital image is synthesized by some other application. The digital image is represented and stored internally, and processed by the application (sometimes iteratively). Finally, the digital image is made suitable for display (rendering) and displayed upon a computer device. Often, the processing, rendition, and display steps are iterated.

Two threads which run through the entire process are efficiency of representation and processing, and fidelity of reproduction. In general, we would like to maximize fidelity (trueness to the original as perceived by humans – ie. psychological vs mathematical) and efficiency of representation (minimize the data storage) at the same time.

**Figure 2–1: IMAGE TRANSFER PROCESS**




Continuous images derive from real world objects and scenes, and form the basis for human visual perception of the world around us. Digital images derive either from continuous images (capture or digitization) or from two–dimensional digital data generation processes (synthesis). The X image extension is concerned only with digital images and does not care about their derivation.

Once in digital format, an application may revise an image iteratively through editing and enhancement processes. Just prior to display,the digital image must be converted into a format understood by the display device (rendition process). It is then displayed where it is converted back to analog format and perceived by the viewer.

## 2.2 Visual Perception

An image of a three−dimensional real world object or scene is a continuous (in its spatial dimensions, and in the dependence of its intensity upon wavelength and time), two−dimensional projection formed by an imaging element.

The human visual system is a combination of imaging element (lens), focal plane sensor (retina), and interpretation machinery (brain). Based upon perceived attributes of a continuous image (dependency of intensity upon wavelength and time), we often assign it to one of a number of distinct groups or visual types including black and white, grey scale, color, stills, and motion. These phrases are simply convenient labels for complex visual perception phenomena.

It is not unusual to associate the visual types (which are based upon human perception of a continuous image) with the digital representation of the continuous image. In fact, there is an optimum (efficiency and fidelity) digital image representation, known as a digital image class, for each visual type.

## 2.3 Image Capture

Image capture or digitization is the process by which a continuous image is converted into a discrete image manageable by a computer. A continuous image is an intensity or amplitude function I(x,y,w,t) which is defined on a two dimensional plane at an infinite number of locations (x and y are continuous real variables), which is defined at an infinite number of wavelengths (w is a continuous real variable), which is defined at an infinite number of times (t is a continuous real variable), and which may posess an infinite number of values (I is a continuous real variable).

Converting to a discrete representation means that I, x, y, w, and t become integers with finite range. Thus, discrete means that the continuous image is sampled at a fixed number of locations (vertices of a regular, two−dimensional sampling grid), and that the intensity is sampled at these points at a number of wavelengths or wavelength bands and times, and quantized to a fixed number of levels.

Typically, for static images (no dependence upon time), the process of conversion from continuous to discrete involves only sampling and quantization.

## 2.4 Image Synthesis

In practice, digital images are often generated by means other than sampling at the focal planes of imaging elements. Synthesized images are ones where the discrete image data at the discrete sampling grid locations is computed from physical sensory information or object models. Examples include medical Nuclear Magnetic Resonance images, Geosesmic images, and synthesized graphic images. In the last case, the digital image derives from computer models of objects and scenes which may not exist in reality. As with captured images, synthesized images are often intended for consumption by a human viewer.

## 2.5 Mathematical Model

An F valued image 'a' on X is the graph of a function a:X–>F

```
a = { (x,a(x)) : x E X, a(x) E F}

where:

 X is a (multi dimensional) coordinate set
 x is an element of the set X
 F is a value set whose values maybe
  R (real)
  Z (integer)
  Z*2k (binary integral)
  C (complex)
```

Under this definition, while the sampling grid is discrete and finite, the values of the image may be integral, real, or complex. We will see that as image processing often produces results which are non integral, it is necessary to recognize non integral valued images.

The debate over the definition of a digital image regarding is potential values (integral vs non integral) is more one of philosophy than anything else. It is clear that there are some locations in the transfer model where the point is not moot. In particular, display devices in general require discrete (integral) valued images for display purposes.

We will adopt a pragmatic approach which avoids the philosophical question which requires discrete (integral) images in client server communications and server presentation to display devices, but allows for non integral representations in the server during image processing.

## 2.6 Image Representation

Whether captured or synthesized, a primary goal is to represent what was (or will become at display time) a continuous image in discrete form with the fewest bits possible (efficiency), while being able to recreate as similar an image as possible for human viewing (fidelity). These goals result in classes of digital representation optimized for and analogous to the various visual types.

The digital image representation must store information regarding both the luminance and chrominance characteristics of the original image. We will consider static images only, or equivalently, time variations as representable as sequences of individual static images.

Luminance is often referred to as intensity or tonality. In a digital representation, the number of bits is finite, and so the intensity or tone scale has a limited number of levels. A digital image which provides for only two intensity levels (and has no chrominance information) is often referred to as bitonal, while one with several intensity levels as continuous tone.

Chrominance information may or may not be present in a digital image. When a continuous image is captured, the use of multiple sensors which integrate intensity over various wavelength bands provides some chrominance information. Images which include this information are known as poly chromatic or multi–spectral. This is typically what is done for systems like television cameras. When only a single sensor is used and the intensity is integrated over the entire spectrum that the sensor responds to, no chrominance information is available. These images are known as monochromatic.

There are many ways in which the luminance and chrominance information of an image can be digitally represented. At the extremes are representations which fully combine and fully separate the luminance and chrominance information and in between are representations with varying degrees of combination. A common example is R–G–B which has three separate luminance channels (one for each of the three chrominance bands). Broadcast television (NTSC – which is an analog format) is designed such that the luminance and chrominance signals are separate and can be recombined at the receiver (tv) allowing "black and white" (what we will now refer to as monochromatic continuous tone) tv's to receive and show the image without color.

Another aspect of image representation is the amount of information required per channel. Many schemes have unequal amounts of information per channel. The amount of information per channel is based on many things including human visual perception characteristics and efficiency (conservation of storage and communications bandwidth). Examples of this include 8 bit per pixel color (R–G–B 3–3–2) where the amount of luminance information is different for the 3 chrominance channels, and broadcast tv where the luminance and chrominance signals have significantly different bandwidths.

**Figure 2–2: VISUAL TYPE – VS – IMAGE CLASS**

| HUMAN VISUAL TYPE | OPTIMAL IMAGE CLASS | CLASS CHARACTERISTICS |
|---|---|---|
| BLACK AND WHITE | MONOCHROMATIC BITONAL | 1 BAND – 2 INTENSITIES |
| GRAY SCALE | MONOCHROMATIC CONTINUOUS TONE | 1 BAND – N INTENSITIES |
| COLOR | MULTISPECTRAL CONTINUOUS TONE | M BANDS – N INTENSITIES |

where: bands --> spectral bands

The black and white visual type is usually represented by a digital image class known as (monochromatic) bitonal. The gray scale visual type is usually represented by a digital image class known as (monochromatic) continuous tone. The color visual type is usually represented by a digital image class known as multi–spectral (polychromatic continuous tone). The stills and motion visual types are represented by individual or sets of the other digital image classes.

One can argue that digital image classes, and by analogy, the associated human visual types can be organized into a hierarchy based upon degrees of freedom or information theoretic information content. Loosely speaking, color images have an additional degree of freedom or contain more information than do grey scale and black and white, and grey scale images contain more information than do black and white. Obviously, time varying images have an additional degree of freedom over static images.

Note that although any continuous image of any visual type can be represented and communicated at some level to a human viewer using any class of digital image, the previous table shows the optimal digital image class (in terms of efficiency and fidelity) for each human visual type. Representing and rendering an image for display in an image class which is not optimal for the human visual type usually results in a loss in efficiency of representation and visual fidelity.

The fact that the different digital image class data representations are optimized for representing different visual types results in significant structural differences between them.

The visual type known as black and white is dominated by images of human communications objects such as written or drawn documents. These images are characterized by high contrast and relatively high spatial frequency, and contain symbols which are differentiated by shape rather than intensity or color. The digital image class used to represent them with greatest fidelity and efficiency (bitonal) uses relatively high

spatial resolution to capture the shape information of these symbols, and low luminance resolution (1 bit per sampling location – 2 intensity levels – black and white).

Real world scenes are represented by the gray scale and color visual types. They have far fewer sharp intensity edges than black and white visual type images, and much finer gradation and variation in intensity and hue. Images of these objects do not require the high spatial resolution that is needed for black and white written material, but instead require higher luminance and chrominance resolution. The continuous tone and multi–spectral digital image classes are used for the greyscale and color visual type images.

## 2.7 Image Compression

Image representation deals with selecting an optimal digital image class for an image of some visual type. Within these digital image classes, schemes exist to remap or encode the digital images in a fashion which reduces redundancy and results in a more compact (compressed) representation. There are many image compression schemes and to some extent, they are optimized by digital image class. Image compression schemes which result in complete fidelity (as measured by the mathematical difference between the original image and one which is reconstructed by the compression–decompression cycle) are referred to as lossless schemes, while those that lose some information are known as non–lossless schemes.

## 2.8 Image vs Text & Graphics

Although computer text, graphics, and images all represent computer information to be presented for consumption by the human visual system, they are differentiated by their internal representations, and by the ease with which computer programs can interpret their contents and semantics. Text and graphics are highly structured with limited lexicons (ASCII character set, points, lines, arcs) and possess highly evolved semantics. While digital images are a less efficient means of representing textual or graphical information, they offer a far richer range of expression (such as a photograph of a person).

## 2.9 Image Data Object

In practice, a digital image consists of two pieces: the actual bits which are a result of the sensing and digitization or synthesis processes, and the attributes that allow the bits to be interpreted meaningfully.

The raw bits, which result from the capture or synthesis processes and represent a discrete image (spatially sampled and quantized – no floating point arrays allowed!) can be organized in many different ways. The simplest organizational model is that of a three–dimensional volume where the x and y dimensions represent the sampling plane or grid locations, and the bits found in the z direction represent the value of the image at that sampling location at some instant in time. Time varying images may be thought of as a collection of such volumes.

Image attributes consist of information which does not derive from the capture or synthesis processes, but which is essential for interpreting the raw bits. Attributes include such information as the way in which the raw bits have been organized, parameters of the sensing process, and information regarding how the image data is to be displayed. Although the digital image class (bitonal, continuous tone, multi–spectral, still, time varying) is usually not declared explicitly, it is often inferable from the attributes. Assuming that the digital image class used to represent the human visual type of a particular image is chosen for purposes of achieving maximum fidelity with maximum efficiency, the visual type of the continuous image can usually be inferred from the digital image class.

## 2.10 Image Processing

Image processing is the application of image operators to image data. There are three broad classes of image processing distinguished by purpose and approach: restoration and enhancement, analysis, and coding.

Restoration processing is done to remove some well understood or predictable image distortion resulting from the imaging system or conditions. Enhancement is the processing of an image for the sole purpose of making the image data more easily interpretable by a human viewer. Enhancement processing is often done interactively under the direction of the viewer. Both restoration and enhancement processing produce an image as output and as such, maybe considered to be similiar to image synthesis.

Analysis processing seeks to infer meaning (semantics) from raw image data and involves machine interpretation of images. A major branch of analysis processing is known as pattern recognition. Analysis processing usually produces some measurment or conclusion as output rather than a new image. Often enhancement processing is performed in preparation for analysis processing (one such example would be edge detection whic produces a new image which is dimensionally simplier for an analysis process to deal with).

The process of image coding deals with mapping image data into alternate symbol sets for efficiency of storage and communications (for example compression), or for organizational purposes. Image coding produces an alternative representation of the input image as output. Depending upon the type of coding, the original image can be recovered with varying degrees of fidelity.

## 2.11 Image Rendition

Rendition is the process of making an image object suitable for display. In general, it involves converting the image into a format that the display device can deal with, and sizing and aligning it properly to fit on the screen. Rendition deals with the presentation attributes of an image.

Rendition and display are logically the final steps in the image transfer model. A continuous image of some visual type was converted into digital format of a particular image class for efficiency and fidelity reasons. The rendition process seeks to make the image object of a particular image class viewable on a display with certain hardware characteristics while achieving as high a degree of display fidelity as possible.

Display devices have hardware characteristics that determine if a particular digital image belonging to a particular image class can be displayed as is (rendition is a null operation), or if the image must be converted within its image class, or across image classes, before display.

Conversion within an image class means that rendition takes a particular revisable form image of some class (bitonal, continuous tone, multi−spectral), and produces a final form image of the same class (bitonal, continuous tone, multi−spectral). Note that although the class remains the same, attributes such as number of intensity levels may change (an example would be rendering an 8 bit per pixel continuous tone image for display in 4 bits per pixel).

Conversion across image class means that rendition takes a particular revisable form image of some class and produces a final form image of a different (usually inferior) class. An example would be rendering an 8 bit per pixel continuous tone class image for display in 1 bit per pixel (using a dithering algorithm) as a bitonal class image. Although cross−class conversion can be from any digital image class, to any digital image class, conversion more often (due to display device limitations) proceeds from digital image class with higher degree of freedom to that with lower degree of freedom (such as from multi−spectral to gray scale). Psuedo coloring is an example of a digital image class conversion which proceeds in the opposite direction (inferior to superior class). Cross−class conversions which proceed from superior to inferior

classes tend to discard information, while those which proceed in the other direction at times "invent" or provide information which was not available in the original image.

In general, if the display device will support it, fidelity is maximized if the image is rendered in a format as close to that of the image class as possible (for example, display a continuous tone image using multiple bits per pixel rather than by using 1 bit per pixel). If not, the image may be converted to a different image class (from continuous tone to bitonal through a dithering process) in order to still render and display it, and convey a facsimile of the original visual type.

Display hardware is usually such that there is a hierarchy of ability to render different image classes with fidelity. For example, if a display device can render a continuous tone image with fidelity (ie. it has N bits per pixel capabilities), in general, it will also be able to render a bitonal image with fidelity. This hierarchy is not one of image class, but rather one of the display device's ability to render image classes with fidelity.

## 2.12 Image Display

The display step takes an image which has been rendered and makes it viewable (visible on the screen). It involves transport of rendered (final form) image data (usually referred to as the bit map at this point) to the display memory and setting of display controls.

Display is an entirely separate step from rendition. There are several reasons for separating these two steps. One major reason is performance. The client may chose to render several images (a potentially time consuming task) before displaying them so that they may be displayed in a time sequence, and so that the user may return to a previously rendered image without rerendering. A second reason is that the client may wish to render an image and convert it back into revisable form without displaying it at all. In this case, the user now has a revisable form image which can be processed, and whose characteristics are aligned with those of the display device making future rendition much less expensive if not entirely null. Of course, processing of this image might make re−rendition a necessary step.

## 2.13 Image Revisability

Revisability is the ability to go back and continue to modify the contents of an image. At one end of the spectrum are revisable form images, while at the other end are final form images.

In revisable form images, the raw data and attributes represent the original image as captured or synthesized, and rendition or other non−reversible image processing operations have not been applied. The association between image attributes and raw data exists in revisable form images.

In final form, an image has been prepared for output and all presentation attributes have been accounted for (applied and stripped off). The contents of a display device display memory may be considered as final form. As rendition is an output device dependent process, it is likely that a one to many relationship exists between a revisable form image, and the various final form images that derive from it (each for a different output device).

The distinction between revisable and final form for images is not as clear as it is for text objects. With images, the final form of one image object could be used as the starting point for the revisable form of another if the presentation attributes were reassociated with the data.

## 2.14 Image Applications

An application which inputs, outputs, stores, communicates, or displays digital images is said to be an image application. Many applications will integrate image handling with text and graphics handling. The subset of image applications which must display images are candidates for use of the X image extension package. Most or all of the overall image transfer model can be found within these applications, although at times, various steps are null or compressed.

Text, graphics, and images are simply different data protocols and sets of services for visual information managed on a computer. While many applications will need to wield all three sets (text, graphics, and images) to visually display what they wish to communicate, the integration of text, graphics, and image services under X are outside of the scope of this work (the X image extension protocol). Minimally, it is up to the application to accomplish the integration of use of these services. Compound document display and edit packages are perfect examples of applications which face this challenge.

In addition to not solving the integrated approach to the use of X based text, graphics, and image services, the X image extension package will not provide ANY text or graphics services.

The processing of images for display purposes and the display of images are intimately entwined. The X architecture has segregated processing for display and display from other aspects of applications. Within the X model, an application has a component which requires the display processing (and human interaction) services called the client component, and a component which provides these services called the server component. Naturally, the server component resides at or near the equipment which provides for display and human interaction. The X architecture also provides for logical/physical separation of these two components by a "wire".

## 2.15 Image Server

An X server for imaging applications minimally includes a general purpose host processor on which the X server runs and a display subsystem. There is potential for great diversity regarding the server host configuration, the complexity and capability of the display subsystem, and the relationship between host processor and display subsystem.

Regarding the relationship between the host processor and the display subsystem, at one end of the server spectrum are terminals where host processor and display subsystem are separated by a relatively low bandwidth connection and often a character oriented protocol. The separation between these components maybe relatively small (local serial line), or large (modems and packet switch networks). At the other end are workstations where the host processor and display memory are connected by a high bandwidth bus and often overlapping address spaces. While the X image extension package does not preclude the terminals cases, the large size of image objects implies the need for a relatively high interconnect bandwidth between server host processor and display subsystem (the workstation configuration).

Regarding display subsystem capabilities, at the low end are devices which do not allow direct access to the display memory and have no user accessible display processing capabilities. At the high end are display subsystems with specialized processor–memory architectures (including parrallel–pipelined processors, and shared memory multi processor configurations) which provide on–board support for enhancement and rendition processing.

As image capable workstations and display subsystems develop, there is a trend to closely couple image capture devices to image display subsystems. Close coupling involves sharing the display memory between the capture device and the display hardware. With these configurations, it is usually possible for the viewer to see the image as it is being digitized. In these cases, the capture–>represent–>enhance–>render–>display transfer process has no loops and is very tightly packed and highly optimized.

# Chapter 3

# EXTENSION ARCHITECTURE CONCEPTS

There are a number of concepts which are fundamental to the image extension architectural model. These concepts are introduced in this chapter, and those which result in elements of the X Image Extension Protocol (XIEP) are explored in depth in the following chapter. In addition, the relationship between the core X architecture and protocol, and the proposed extension architecture and protocol is addressed briefly throughout the chapter.

## 3.1 Extension Guidelines

In formulating an image extension protocol, the two most important questions are:

1. What services do client portions of image applications require of an image server?
2. What services is an image server well positioned to provide?

From the answer to these questions, a high level service set can be defined. The next step is to proceed to identify the primitives that the server should implement to support the high level service set. Finally, the primitives must be mapped into a minimally sufficient protocol for communicating both the service primitive requests and responses.

In order to answer the first question, one must analyze the human interaction components of many imaging applications. If performed by a single individual, in a serial fashion, this is potentially a long task. A more practical approach to this problem is to draw upon the experience of many diverse image applications developers (which to some extent is the approach taken to date).

Regarding the second question, the core X model provides some guidelines on what should live on the server side of the wire. Clearly, all interaction with the display device must go through the server. Likewise, all interaction with human controlled interaction devices, such as keyboards and pointing devices, must go through the server.

In general, due perhaps more to the maturity and standardization of the spaces, the core X package provides some degree of device independence for text and graphics. Although this is not an explicit goal of core protocol, it seems like a reasonable direction for the extension protocol: the image extension package should provide device independence to the client. This means that the XIE_TOOLKIT, XIE_LIB, and XIE_SERVER will handle all image display device dependencies. Minimally, this implies that rendition is an extension package issue. Server extensions are encouraged to provide for usage of specialized display hardware primarily for display purposes.

The XIE_SERVER may be home to operations which, although not specifically device dependent, would derive benefit (performance efficiency) from being in close proximity to the display subsystem and special–ized display subsystem hardware. This is the basis for postulating that interactive enhancement processing must be provided for (allowed) in the server.

## 3.2 Extension Levels

The following diagram shows the relationship between client and image extension server, and how the full image extension package parallels the existing X components. It also positions the specification of XIEP, the image extension protocol for the X wire (denoted by the ∗).

**Figure 3–1: IMAGE EXTENSION – VS – CORE PACKAGE**

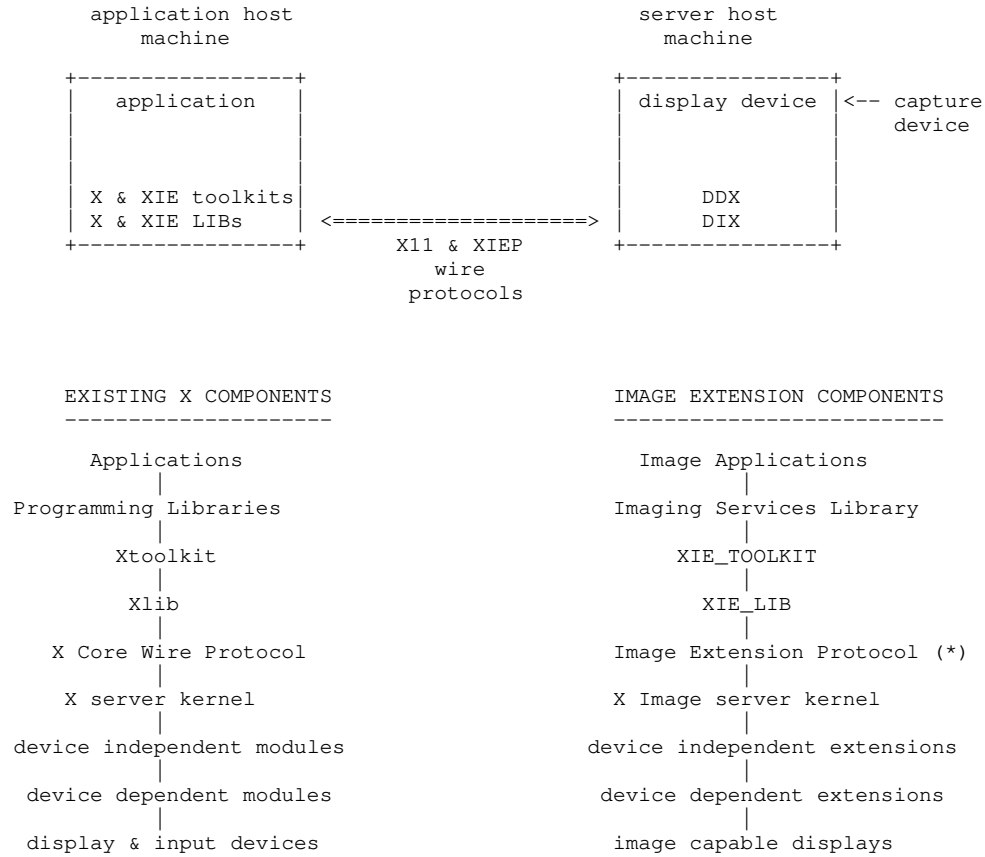



Image specific components and issues can be positioned against the core X levels. Image accelerators (either integral to the display subsystem, or simply ancillary server host processors) exist and are accessed at the device dependent server level. The device independent image extension protocol functions (XIEP) and image specific resources exist at the device independent server level.

Extensions to X lib implement the X image extension protocol. XIE_LIB is likely to follow the precedent set by the core protocol and simply represent language bindings of the XIE Protocol.

The constraint of minimality for the X Image Extension Protocol forces many useful X Image Extension Package functions into the the software between the client and the wire: ie. into the X Image Extension Toolkit. In essence, the protocol offers the lowest level functions upon which higher level services can be built. These higher level services include things such as rendition models and control, and image specific display constructs (widgets, etc), and will be packaged in the X Image Extension Toolkit. (see appendix for a more detailed discussion of rendition distribution in the X Image Extension Package.

## 3.3 Global Concepts

The following concepts are significant, if not highly visible or localized, components of the extension architecture. In general, these concepts are important to understanding XIEP and bear upon its overall structure, content, and use. They have been segregated into the global concepts section because they do not translate literally to any one or set of protocol elements.

### 3.3.1 Final vs Revisable Form

Central to the extension proposal is the relationship between revisable and final form images. The contents of the server's image display represents a final form image. Thus, the viewer always sees and visually interacts with a final form image. Behind every final form image which the user views is a revisable form image from which the final form image derives by way of the rendition process.

The relationship between the revisable form image behind the final form image that the viewer sees may range from being nearly identical (whereupon the rendition process is simply a logical step or null) to one where the revisable form image contains much more information than the final form image on the display. This additional information might be useful in zoom, locate, or other display operations.

It is the responsibility of the application to initiate rendition and display operations which create a final form image from a revisable form image and cause it to become viewable. It is the responsibility of the image extension package (XIE toolkit, lib, and server) to provide rendition services which insulate the application from device particulars. It is also the responsibility of the image extension package (XIE toolkit, lib, and server) to provide the means to associate rendered image locations (final form) with revisable form locations (whenever possible). This amounts to providing an inverse to any geometric transform applied during rendition. Obviously, it is required in order to make screen image pointing operations meaningful.

The server will also provide a means by which a final form image may be converted back into a revisable form image (ie. attributes reassociated with bitmap). Note that the resulting revisable form image will in general not be identical to the revisable form image from which the final form image was derived.

### 3.3.2 Image Data Representation

Image data representation is a generic issue which is found across the entire transfer model. Image data representation during wire transport is chosen for application convenience and transport efficiency. Image data representation during server storage is chosen for server convenience. Image data representation during screen display is chosen for compliance with display hardware requirements.

Two particularly thorny conceptual issues involve the concept of the "pixel". The first one has to do with what is this logical thing known as a pixel especially as it relates to various data organizations. The second addresses discrete versus continuous pixel values.

#### 3.3.2.1 Aternate Data Organizations and Pixels

The concept of a regular, two−dimensional grid is central to both the image capture and image display processes as both image data sampling and image display are done on a regular, two dimensional grid. In general, the sampling and display grids will not be of the same form or size. We will use the term pixel to mean a capture or display grid location. This section addresses the association between image information and pixels.

The relationship between image data representation and pixels is potentially meaningful only at the capture and display points in the transfer model. The relationship is potentially meaningless at other points in the model (especially places where image data is kept in compressed format). Most displays prescribe the relationship between the underlying data that goes into making up a display's pixel at display time (i.e. the structure of a pixel). Although it is a convenient concept, it is a mistake to assume that behind a display screen, the information that goes into lighting up a pixel can be found as a stack of bits. While some displays do require that all the information which makes up a pixel be gathered together into a bit string (ie. multiple chrominance channels are gathered together on a pixel basis), others require it to be separated by chrominance channel. The situation grows more complex when we consider displays which allow different bandwitdhs per channel between luminance and chrominance channels or just between different chrominance channels. Thus, the relationship between a pixel (as a display−grid element) and the underlying image data and its representation is potentially a complex one and not necessarily a one−to−one correspondence between pixel value, luminance, and chrominance.

For polychromatic or multispectral images, the various representations which provide for either separate or integrated luminance and chrominance information, and for either equal or different bandwidth (information content) between luminance and chrominance channels or between multiple chrominance channels creates a potential mismatch between the front end of the transfer model (before display and "pixels"), and the backend (at display and "pixels"). For the monochromatic digital image classes, this problem does not exist.

The X image extension package will deal with both matches and mismatches between image data repre−sentation structure and content at the display boundary in the transfer model. As such, it will deal with matches and mismatches in representation (RGB vs YIQ vs ...) and content (equal vs unequal bandwidth between luminance and chrominance channels, and between multiple chrominance channels). It is the server's responsibility to convert from the application representation into a representation and pixel struc−ture amenable to server based enhancement processing. It is the responsibility of the X Image Extension Package (XIE toolkit, XIE lib, and XIE server) to provide for the conversion of image data into a format compatible with that recognized by the display subsystem (rendition process).

#### 3.3.2.2 Discrete vs Continuous Pixels

This section focuses on the representation of the intensity function for a digital image in the X Image Extension. In particular, it will examine where in the Image extension the intensity must be discrete (quantized to a fixed number of levels − integer), and where it may be continuous (infinite number of levels − real).

The image concepts sections on capture, synthesis, and representation stressed that digital images are discrete entities (spatially sampled and quantized). As capture and synthesis are client issues (outside of the domain of the image extension), the client may do as it wishes, but the image extension encourages clients to consider only fully quantized 2d arrays as digital images.

This policy becomes binding at wire transport time where transport will only include support for fully quantized (positive integral valued) digital images in uncompressed formats (the bits represent integer values of intensity for components), and compressed forms where the bits represent code sequences for compression schemes. In other words, the wire transport machinery will NOT provide support for real valued (floating point) image intensities.

This same policy applies within the server at post rendition and display time. Final form image objects on the server will be integral valued entities. Obviously, rendition implies this as all workstation display machinery requires this.

It is during revisable form representation of image objects on the server that full quantization and integral value only may be relaxed. The purpose for its relaxation is the nature of image processing (both enhancement and rendition). The result of applying an image operator to image data may in some instances be considered by the user to be a new image, while in others it may simply be considered as a two dimensional array of computational data results. The choice is often but not always more a matter of user perspective and intended usage than of image operator semantics. In particular, many image processing operations may produce non integral or negative output values. Examples include the arithmetic operators "divide", and "subtract".

There are two possible solutions to this problem: the first is to require that image objects remain fully quantized at all times, and therefore expressly define the semantics of each and every image operator such that this is preserved. In the division case, one such policy is to implement only an integer divide which truncates. A second strategy is to allow real values in server based revisable form image objects and delay the final quantization until rendition time. Obviously, the display device can not deal with these non integral values, and if these data arrays are to be treated as images for display purposes, at rendition time the situation must be corrected.

The policy will be to allow image enhancement and rendition operators to produce image objects with real values (rather than integer). Protocol functions (image operators) which must produce one or the other will note it in their definition. Note that this has no bearing on the integral valued only position at wire transport and display other than perhaps the potentail need for an explicit quantizing protocol function.

## 3.3.3 Layout & Coordinate Systems

Implicit in all display packages are the coordinate systems used by the package and their relationship to coordinate systems used within the application model. At a somewhat less abstract level are the display layout problems that an application must deal with and the machinery required to accomplish these layouts. The next few subsections provide an overview of the extension package coordinate systems and layout concepts.

### 3.3.3.1 Coordinate Systems

By our definition, continuous, two–dimensional images (at the focal plane) have dependencies upon two spatial coordinates (planar), time, and wavelength. This section examines some of the coordinate system issues involved in the extension package regarding these coordinates.

#### 3.3.3.1.1 Spatial Coordinate Systems

The image extension package does not provide for any world coordinate system. Therefore, it is not possible through the image extension protocol to maintain relationships back to real world objects, scenes, and relationships within them. It is left to those imaging applications that require such capabilities to maintain and conduct their own conversion from a private world coordinate system to the system of device and image object coordinates supported by the extension package, and to relate graphic and image display service requests to a shared world coordinate system.

The image extension package does not support three–dimensional images (depth information) and as such perspective transforms and all three– dimensional imaging operations must be managed within the application (above the client). A three–dimensional image application can still use the image extension by feeding it two–dimensional slices or projections.

The image extension package uses the device coordinates (screen and window relative) as defined by the core X11 protocol in all display requests.

In addition to this device coordinate system, the extension protocol defines an image object (IOB) relative coordinate system. Regardless of the data organization at transport time, or the internal data organization of an image object, all image objects have their own integral, two−dimensional, rectilinear, attached coordinate system. For monochromatic images, there is no confusion, but for situations where the bandwidth between luminance and chrominance channels differs, or that between multiple chrominance channels differs, the coordinate systems of the image object applies to the highest bandwidth (biggest) component. All regions of interest are defined relative to the image object coordinate system.

Rendition is the point of alignment or conversion between image object and device coordinates. The extension protocol provides the means by which a location within a rendered image (expressed in device coordinates) can be converted back into a location within the revisable form image (expressed in IOB coordinates) from which the final form rendered image derives. This capability will be provided for all invertable rendition transforms (at least linear geometric transforms). It should be clear that this capability is essential if an application is to associate what a user sees and points to (final form image) with a location in the revisable form image.

Despite the availability of this coordinate conversion service, application clients may chose to perform their own mapping back from device coordinates (mouse interrupts) to revisable form image object coordinates.

#### 3.3.3.1.2 Temporal Coordinate Systems

The extension package has a concept of time sequences of frames (see layout), but does not provide any temporal coordinate system. As such, an application which maintains a temporal coordinate system (such as a high−speed photography application), must convert from a time coordinate to an individual frame id or frame sequence index when dealing with the extension package.

#### 3.3.3.1.3 Chrominance Coordinate Systems

For reasons which will not be explained here (see ref: CORNSWEET) color images intended for human consumption are, in general, represented with three independent components. The theory of color spaces is particularly convoluted and requires a good understanding of the human visual system. There are many sets of basis vectors for a color three space, or if one prefers, many different color three spaces.

Unlike the two−dimensional spatial coordinates, where standardization upon rectilinear (vs polar) coordinates is a fairly reasonable thing to do, standardization upon a single color three space coordinate system may not be as reasonable. Although the computer industry display devices are primarily R−G−B, there are advantages to representing images in other color three spaces (such as Y−I−Q or H−L−S). One important distinction to note is that some color spaces maintain a one−to−one relationship between component and spectral band (for example, R−G−B), while others do not (Y−I−Q). In these latter cases, each of the dimensions of the color three space is not necessarily a different spectral band. In some cases, these color systems are designed to separate chrominance and luminance information for representation efficiency.

Higher band multi−spectral images (such as n band Landsat images) tend to be of the former nature: each component maps directly to a single spectral band.

The extension protocol will recognize more than one three space color representation at key interfaces and support linear (invertable) color three space transforms The extension protocol will rely upon these three space systems (or protocols) for the relationships between component and wavelength band.

The extension protocol will not recognize images of more than 3 spectral bands, and as such, applications with N spectral band data (such as Landsat) will be responsible for spectral dimensionality reduction (potentially accomplished by an application applied transform/mapping or by application band selection).

#### 3.3.3.2 Layout

Applications which deal with the display of digital images often face a variety of applications specific screen layout issues. Although it would seem that layout is something which is entirely in the realm of a window manager, there are layout issues which perhaps transcend the bounds of core window management. The next two subsections present examples of such image layout issues and the policy that the extension package will adopt.

##### 3.3.3.2.1 Spatial

Applications often need to display either two images, or an image and a graphics scene, overlaid and spatially registered. The image extension package looks to the core X package for window overlay capabilities. Barring availability of these capabilities, the extension package will provide the means to spatially register and overlay two images from their revisable form through their rendition and display steps. This is likely to be done through image composition operators (logical point functions). In the case of graphics and image overlays, barring help from the core X package, the approach will be to convert the graphics information to an image and then proceed as above.

The approach described above should provide for maintenance of spatial registration under operation of zoom, pan, and scroll.

The extension package will not provide any specific mechanisms in support of "N up" spatial layout or "tiling" problems. The extension package looks to the core protocol and toolkit levels built upon it for support of tiling.

##### 3.3.3.2.2 Temporal

Some image applications deal with time sequences of images. In a time sequence, the image subject state is changing with time, but the basic subject is the same (ie. a time lapse sequence of a flower as it blooms, or a sequence of heart radiological images as it pumps blood). While the extension package does not provide a temporal coordinate system, it will support the concept of a time sequence of spatially registered images, and provide for control over the time delta between time sequence frame displays.

The slide carousel is the basic mechanism for organizing a time sequence. The slide carousel will contain final form images (X displayables or pixmaps) produced by rendering revisable form images. Optional references back to the source revisable form images may be kept. The image extension package will provide mechanisms for playing the time sequence with uniform time spacing, or single stepping under application control. Direction of play is also under application control.

A very significant subset of time sequence problems is the display of live video. Live video consists of the display of 30 still frames per second. Considering the size of individual images, a few seconds of live video would require more memory than most workstations posesses. Typically, in live video applications, the source images will be supplied from a digital image disk, or digitized from an analog source such as a tv camera or vcr by a frame grabber in the server. The temporal layout machinery will give consideration to the support of this special class of time sequence image applications.

### 3.3.4 Image Structure – Subsets

Computer displays present users with screens full of information. Within any screen or window, there are established rules for identifying smaller elements when the contents derive from underlying text or graphics. Text and graphics information (visual information which is represented on computer systems via text and graphics protocols) have syntactic objects (characters, words, paragraphs, lines, arcs, ...) which the user is familiar with and which can be used to identify a component of a screen.

When the screen contents represent an underlying digital image, no such user familiar, underlying syntactic structure exists. Instead, it is replaced by the concept of identifying portions or proper subsets of the complete image. These subsets are used to identify areas that will be operated on. There are two proper subsets: regions of interest, and control planes. A region of interest is a fully connected subset of an image such as the contents of a rectangular area within the image, while a control plane determines membership in the subset on a pixel−by−pixel basis. A control plane maybe heavy or sparsely populated.

The extension protocol supports the definition and use of both regions of interest and control planes for operations on connected and disconnected subsets of a complete image.

### 3.3.5 State Information – Process History

Other than the information required to associate display locations (coordinates) with image object coordi−nates (for pointing problems), the server will not maintain any processing history information. It is left to the client to maintain this information if so required.

### 3.3.6 Extension Relationships

#### 3.3.6.1 To Core X

The X image extension will be accessed through the core X server as prescribed by the core X protocol extension mechanism. The extension will make use of core x resources where ever possible (such as drawables, pixmaps, and clipping regions), but the policy is not to extend existing (core) resources with regards to either content or semantics.

Where extension would be required, the image extension will define private resources necessary to support the specified functionality. For example, rather than attempt to extend an already complicated structure such as the graphic context (GC) with new fields, or new meanings for existing fields, the extension package defines its own resources for rendition purposes. While this entails some duplication of information, percentage wise, the amount of duplication is very very small when compared with a 1 MB revisable form image, and no control confusion is likely to arise.

#### 3.3.6.2 To Other Extensions

It is more than likely that an application may require access to multiple extension packages at the same time, such as a graphics and image extension. Control over concurrent residency of multiple extensions to the server is under core X. While exchange of information between multiple concurrently resident extensions is always possible via the client, this approach is potentially inefficient due to wire transport. The image extension package includes import/export services which provide a limited means of exchanging information (primarily rendered image data) between the image extension and another extension through core X resources.

#### 3.3.6.3 Multiple Connections, Clients, and Resource Sharing

Following the guidelines established by the core X protocol, the image extension will support sharing of resources between multiple connections from the same or different client. Multiple connections (from the same or different application clients) may reference the same resource (ie. resource handles will be server globals rather than connection globals). Managing the sharing of server resources and all ramifications is a client problem. See the appendix for a more detailed discussion of applications interchange of image data.

#### 3.3.6.4 To Text and Graphics

As was stated earlier, text, graphics, and images are simply different data protocols and sets of services for visual information managed on a computer. While many applications will need to wield all three sets (text, graphics, and images) to visually display what they wish to communicate, the integration of text, graphics, and image services under X are outside of the scope of this work (the X image extension protocol).

In addition to not solving the integrated approach to the use of X based text, graphics, and image services, the X image extension package will not provide ANY text or graphics services. Applications should lokk to core X11 or graphics extension packages for such services.

The author recognizes that at the display memory level, what has been referred to as text, graphics, and image all looks the same: simply display memory bits in various states. The intension of the X image extension package is not to subsume text and graphics because of this apparent unification of representation at the display memory, but rather to deal only with image data protocol elements in the transfer model before the display memory which the traditional text and graphics data protocol services have ignore.

## 3.4 Service Concepts

This section explores in more depth the reasoning behind and implications of the service goals and the highest level organization of the services.

### 3.4.1 Service Goals – Reasoning & Implications

The introduction touched briefly upon the functional goals of the proposed image extension. The next few subsections present these goals and their implications for the extension package in more detail.

#### 3.4.1.1 Range of Images

The proposal calls for integral support for applications requiring display services for the following classes of images:

- Static bitonal digital image data (black and white)
- Static continuous tone monochromatic digital image data (gray scale)
- Static multi spectral digital image data (color)
- Time varying versions of the above

The digital representation of these images (digital image classes vs human visual types) will vary widely regarding the amount of luminance, chrominance, spatial, and temporal information.

The image domain boundaries implied above are somewhat artifical in that they have been established partially by human visual perception attributes and by the applications that they are an integral part of (such as facsimile or television). In reality, there is a continuum of images and image attributes out of which applications have concentrated on a few important subspaces. It is also a fallacy to think that all applications are confined to only one of these classes as many significant applications (such as publishing) cross the domain boundaries.

Because the domain boundaries are artifical, and because applications must be able to work across these domains, the extension package must unify the representation and handling of the images in these various domains. The implication is that the extension must support a uniform and robust image data representation at least at key interface points, and that there will not be separate service packages for each of the domains.

#### 3.4.1.2 Range of Applications Needs

The X client component of image applications will demonstrate a wide range of image needs to the X servers. The range will likely span simple image display of bitonal images through highly interactive processing and analysis of continuous tone and multi−spectral images.

The extension package must not force an application to operate at any particular service level, but rather (like other layered and open systems) allow the application access to both low level basic, and higher level layered services. The extension must not offer separate packages for the various degrees of need.

#### 3.4.1.3 Range of Server Capabilities

X servers are likely to vary widely in their ability to support imaging client needs. These variations may be due to varying server hardware platforms (virtual and physical address space, computational horsepower). With limited virtual address space, processing power, and display hardware on some hosts for imaging servers, it may be impractical to attempt to support certain imaging client needs.

As the image extension architecture is intended to unify image extensions to X across all X servers, and as it is clear that there are underlying reasons why all servers will not be able to offer all image capabilities, the extension proposal introduces the concept of service sets. Service set is a compliance mechanism whereby the architecture specifies what imaging capabilities must be available through the server for a particular set.

Service sets are not functionality hierarchies. Instead, they represent different functional areas that a server may or may not be able to provide for. A server must support at least one service set (set #1) to be considered an image extension server. A server may support other services sets including any combination of the other service sets. The following is a summary of the service sets:

- Set 0: transport services, rendition services
- Set 1: enhancement services
- Set 2: processing pipelines
- Set 3: real time image capture

#### 3.4.1.4 Display Device Independence

The overall transfer model set forth in Chapter 2 is not an invention of the extension package, but rather an observation of the process which underlies the capture and display of digital images in all systems. The model is sometimes obscured in actual practice by steps being combined (tightly packed) or even missing (null).

In the past, image applications have been intimate with display devices and the display device characteristics propagated or permeated back into the actual application. Examples of this are applications which keep basic image data with characteristics (pixels, scan lines, spectral bands) which match the display device capabilities rather than those deriving from the source device or synthesis process.

While display device independence for the client is not an explicit goal of core X, it is deemed a meaningful goal for the image extension package. Display device independence implies that application client image data need not be in a format related to display device characteristics and the application client need not understand or implement the rendition process. This is not to say that rendition is no longer a device dependent process, but rather to say that it is now the headache of the image extension package rather than the client.

##### 3.4.1.4.1 Server Based Rendition

A major implication of display device independence is that the X Image Extension Package (XIE Toolkit, XIE Lib, and XIE Server) will need to understand and implement the device dependent rendition process. This does not mean that the video subsystem (actual display hardware) must understand and perform rendition, nor does it mean that the complete understanding and implementation at a high level of rendition is on the server side of the wire. It does mean that the X image extension package is responsible for rendition and that server based primitives will exist which can be used under the direction of the X image extension toolkit to implement high level rendition models. This includes rendition which preserves and which crosses digital image classes. Rendition will likely include scaling (up and down), x−y and z resolution conversion (up and down), device luminance corrections (gamma − tone scale adjustment), and mono−poly chromatic conversions.

Relief from understanding and implementing the rendition process can result in significant complexity reduction benefits to the application client. It will also allow image applications access to a much broader set of display devices with different characteristics with no additional application knowledge or cost.

Figure 4−3 graphically depicts the basic shift in the wire location and the fact that rendition processing has moved from the client to the server side of the wire. (see appendix for more on rendition in the X Image Extension).

##### 3.4.1.4.2 Image Data Organization

The organization of image data depends both upon the digital image class and the application. The goal is to allow the application to use an optimal image data format rather than project the device format back into the application. The burden should be on the server to convert from application selected data organization to server preferred format.

The implication is that the client chooses one of several image data organizations (for the raw data component of a revisable form image − attribute format is common), and the server complies at the wire with that format. The supported PCM or raw image data formats are touched upon in the chapter on extension protocol elements (transport class services).

#### 3.4.1.5 Image Data Transport

Image data transport is the movement of image data across the X wire between client and server. The transport may involve the movement of uncompressed (PCM) image data in the application client chosen image data organization, or compressed forms. The transport may also be monolithic or segmented at the application client level.

##### 3.4.1.5.1 Image Data Compression

Because the extension package pushes image rendition processing across the X wire into the server, revisable form images often will move across the X wire. These revisable form images will often be much larger than the final form images which are displayed. An example is sending a digital image of an E size drawing which was sampled at 200 dpi across the wire. This revisable form image is approximately 7.5 MB in size. The rendered image which will fit on a 1k by 1k by 1b screen is approximately 150 KB in size. Clearly, when the X wire is physical, there is a need for image compression in at least the transport services. There may also be value for compression of revisable form images in image storage within the server.

Support for image compression in wire transport is central to the X image extension package. Standard image compression schemes will be implemented on both sides of the wire and used to optimize transport when the wire is physical. There are compression standards in place (or under development) which should be complied with, and that image compression algorithms or approaches are digital image class dependent.

While transport will make use of compression for efficiency, the PCM data organization selected by the application client is recovered on both sides of the wire via decompression. In other words, compression is applied to the application client selected PCM format.

##### 3.4.1.5.2 Transport Granularity

At the application client level, the image extension will support transport ranging from monolithic to segmented.

Monolithic transport implies that the entire image (PCM or compressed version of the application client chosen format) will be transported across the wire as an atomic (un−interruptable) event.

Segmented transport implies that the application client may send the image across in a piecemeal fashion where the segments are regular pieces such as scan lines or scan line blocks (tiles), chromatic components, or luminance planes. Each segment is in either PCM or compressed version of the application client chosen format, and each segment will be transported across the wire as an atomic (un−interruptable) event. The image extension server will understand how to reassemble the image segments into a complete image.

Segmented transport allows other X operations to be interleaved with the transport of an image (including event interrupts and servicing). A particularly useful instance of this is an application which uses a transport, render, display pipe to fill a display screen in a continuous fashion as it sends image segments across the wire. The screen begins to show image display as the first segment emerges from the pipeline versus waiting until the entire image has been completely transported and the entire image has been completely rendered before displaying anything.

Both monolithic and segmented transports are asynchronous (application client starts and is notified of completion).

#### 3.4.1.6 Image Processing

As stated in the image concepts section, image processing is the application of image operators to image data. Image processing is divided into three broad classes including restoration and enhancement, analysis, and coding.

In the extension package operational model, image processing will potentially be applied at a number of points including transport across the wire (where compression or coding processing may be used), enhancement processing, and rendition where preparation for display occurs.

Support in the extension package for coding or compression processing has been touched upon in the prior section on compression.

Support in the extension package for image processing used in the rendition process has been touched upon in the prior section on rendition.

The following subsections deal with support in the extension package for enhancement and other forms of image processing, and with the relationship between rendition and enhancement processing.

##### 3.4.1.6.1 Enhancement Processing

Image enhancement processing is technically the application of image operators to image data solely for the purpose of increasing the viewability of an image. The metric for increased viewability is determined by the viewing application and the user. Examples of basic image enhancement processes include contrast enhancement and edge sharpening.

Because the metric for enhancement is in the eye of the viewer, enhancement is often a trial and error, or interactive and iterative process. Interactivity depends upon speed of iteration, and this speed is often a function of process distance from the display end of the transfer model pipeline (especially when the X wire is physical). This is a key reason for including some enhancement processing in the server.

Enhancement processing is a very large field with a potentially infinite number of recipes for visually enhancing an image. As such, attempting to incorporate robust support for enhancement processing in the server is a difficult problem. Once again, the server will implement a set of primitives which support many enhancement techniques, thus pushing the extensibility issue for enhancement into the tool kit level.

##### 3.4.1.6.2 Rendition Processing

Rendition services provide default schemes or models for client image rendition which are compatible with the display situation. The models seek to maximize visual fidelity between the original captured or synthesized image and its display on the target display device. Some control over fidelity is provided through the alternative rendition models. These schemes are atomic (indivisable) from the client's perspective, yet typically involve multiple low level image rendition processing steps such as scaling (x–y, and z), rotation, cropping, and image class conversion. While the rendition models are independent of display device, the rendition process is highly device dependent.

Rendition services explicitly recognize rendition within and across image classes (monochromatic bitonal, monochromatic continuous tone, and poly chromatic continuous tone or multi spectral). While the intent of rendition services is to relieve the client of all concerns regarding mismatch between revisable form image and display device capabilities, it is felt that the protocol must afforded the client some control over cross class image rendition as it has a significant bearing upon visual fidelity, and upon complexity of the server rendition machinery.

Clients which find the default schemes insufficient or unacceptable are free to assemble enhancement processing routines into rendition pipelines of their own definition.

#### 3.4.1.6.3 Enhancement – Rendition Relationship

The relationship between enhancement and rendition processing is somewhat indistinct especially when one recognizes that enhancement and rendition processing may share low level operators, and when one allows (as the extension package does) the client to generate a revisable form object from a prerendered image via import services. As such, the distinction is based more upon purpose (rendition is device specific processing that creates a final form image which is compliant with the output device, while enhancement is relatively device independent) than anything else.

#### 3.4.1.6.4 Other Forms of Processing

The labels enhancement, coding, and rendition do not cover all the uses of image processing in the extension server. There are instances where an application will need to apply an image operator to image data to change its state for purposes other than coding, enhancement, or rendition.

As an example, consider a progressive transmission application (such as low bandwidth teleconferencing) which chooses to optimize the use of interconnect bandwidth by sending image data to the destination (receiving) extension server one plane at a time (most significant first) so that a progressively better image develops. In this case, the application will need to combine the planes as they become available within the extension server (server side of the wire) if the bandwidth savings are to be realized. In this case, combinatorial operators will be required.

Fortunately, most of the operators required in this "other" category are usually deemed essential for image enhancement purposes. They are pointed out here more as additional justification for their presence.

#### 3.4.1.6.5 Processing Policy

In addition to the coding (compression) and rendition processing, the X image extension protocol will support a closed, minimal and primitive set of image processing (enhancement) functions. The X image extension package will provide for higher level extensibility through the X image extension tool kit.

Closed means that there will be no extension mechanism which allows a user to either add new image processing algorithms to the package or downline load them (other than the pipeline macro facility). Closure ensures that a population explosion does not occur and that incompatibilities do not develop.

Minimal and primitive means that the protocol will only implement low level functions which do not overlap in purpose, and upon which client side components can build higher level services. As a simple example, convolution filtering will be implemented in the protocol, but commonly recognized imaging services which can be built upon filtering capabilities such as median and sharpening filters will be relegated to higher level client side components such as the X Image Extension Tool kit. As such, readers who are looking for defacto imaging industry standard services should look to the X Image Tool Kit. The protocol must be sufficient to be able to build these higher level services upon server side processing.

Enhancement processing is often presented in terms of achieving a desired effect and then in terms of approaches to achieving the effect. Approaches to achieving the desired effect is often broken down into techniques and implementations. As those familiar with signal processing know, signal processing is often accomplished in any of several data domains. These domains are roughly categorized into two classes: spatial and spatial frequency. In the spatial domain is we operate on PCM data which is representative of the luminance and chrominance of an image, while in the spatial frequency domain, we operate on ampli–tude coefficients of bases vectors such as Fourier bases vectors (Fourier Transform). Many enhancement processes can be implemented in either of these two domain classes.

While spatial frequency or transform domain processing is an extremely important approach in digital signal processing, and some workstations (servers) may have hardware which support it, it is felt to be outside the realm of reasonability to expect such support of main stream servers at this point in time. As such, no explicit protocol functions will exist which require or imply transform domain processing support. Examples of the implications of this policy include that the protocol at this time will not include machinery which supports the exchange of discrete spatial frequency domain filter coefficients or analytic spatial frequency domain filter descriptions, or coefficient rooting techniques for histogram modification.

This position on spatial frequency domain processing is flexible in that many important image enhancement processes are defined in terms of spatial frequency domain processing. If this position is reversed, the following classes of protocol functions would likely be added:

- perform various transforms (Fourier, DCT, Hadamard, ...)
- perform arithmetic operations on transform data
- generate statistical information about transform data
- apply template (area) functions in transform domain (filters)

It is expected that analytic transform domain filters would be implemented at the X image extension tool kit level.

Note that the current position (no spatial frequency domain support in the protocol) does not imply that servers may not use transform domain approaches. For example, a server with 2d–FFT capability may chose to implement convolution with large kernels in the transform domain.

All server based image processing functions (enhancement and rendition other than the final step in a rendition process) will produce image objects. These image objects may posess integral or non–integral (real) values. Protocol functions (image operators) which must produce one or the other will note it in their definition. These resulting image objects may be considered as true images which the user will eventual view, or simply computational data arrays which store temporary computation information and are discarded without viewing by an end user. The extension package does not distinguish them. The extension package will provide a function to integerize and quantize (force integral values).

#### 3.4.1.7 Operations Pipelining

Image extension server operations are categorized into transport, enhancement, rendition, and display. To some extent, these represent steps that an image must pass through to get from the application client to the display. While these operational classes can be cleanly separated, and an image moved through them by completing each step on the entire image before beginning the next step, it is often useful to move a single image through them in a pipelined fashion.

In pipelines, part of the same image may be in the first step while another part may be completing the last step. A very useful pipeline is the transport–render–display pipe where a segmented image is moved across the wire and to the display in a continuous fashion.

Pipelining has a number of advantages. First, it allows the client application to assemble multiple syn–chronous steps which are likely to be used over and over (macros). Second, in cases where display is part of the pipe, it feeds out visual information as it becomes available rather than waiting for step wise completion. Third, as multiple processors become available in workstations, it allows for some degree of parallel processing as different pipe elements can be executed on different processors.

The image extension package will provide for the definition and execution of process control pipes.

#### 3.4.1.8 Server Display Hardware Control

The X server is of necessity intimate with the display device. In the past, applications have been intimate with the display device (through a driver) and have often had sole access to it (non–shared resource).

Display devices (video subsystems) include scarce resources (such as physical lookup tables, display bitmap memory, display controls, specialized processors) which must now be managed as shared rather than an application private resource.

Whenever possible, these resources should be accessed indirectly through protocol functions which involve their use rather than direct access protocol functions designed to provide remote control register bit diddling. But, when there are no protocol functions which serve this purpose, and access to these resources is still essential to applications, direct access protocol functions should be provided. It is still important in these cases that these direct access protocol functions provide access to logical resources which are mapped by the server to the physical device resources so that the physical device resources remain sharable.

The following subsections examine some specific instances and assert extension package policy regarding these video subsystem resources.

##### 3.4.1.8.1 Shared Hardware Device Control

There are a set of display hardware resources and controls which are important to imaging applications. These include look up tables (luts), display memory (bit map), pan and scroll registers, and brightness, contrast, hue, and saturation registers.

On todays workstations these consist of a single set of resources and controls for the entire display (ie. one physical set which must be shared by all concurrent screen users). In future workstations, there may be multiple sets perhaps allowing one per window. Access to these device controls cannot be hidden behind or within more general protocol functions.

The core protocol has addressed access to and control over some of these while others have been left unexplored. As all affect the display of anything on the screen (text, graphics, images), they should all be dealt with by the core protocol.

The image extension will use core protocol mechanisms where they exist, and will follow the core protocol in defining a logical to physical relationship for the others. It is hoped that anything that the X image extension must do in this area would migrate into the core X11 protocol over time.

###### 3.4.1.8.1.1 LUTS and Bit Map

The core protocol defines a scheme for maintaining multiple logical sets of lookup tables, and the means for associating (loading) the physical tables from the logical tables. This scheme, and potential extensions to it for displays with multiple LUTs, will be used by the extension package.

Likewise, the extension package will rely upon the core protocol machinery for access to the display bit map.

###### 3.4.1.8.1.2 Brightness – Contrast – Hue – Saturation

Many of the current crop of imaging frame buffers provide software access to hardware which controls the overall settings for brightness, contrast, hue, and saturation. In essence, this is digital access to the knobs on a television.

The extension package will again provide a logical to physical mapping and access to these controls.

#### 3.4.1.8.2 Use of Image Specific Accelerators

Server platforms and display subsystems may both have special silicon accelerators useful in various image operations such as compression / decompression and rendition. It is left to the discretion of the extension server implementor if and how to make best use of these accelerators. But, the extension protocol calls for notification of the client (upon request) by the server of accelerator assist for various protocol functions.

Regarding access by the client to these accelerators, the policy is that hardware accelerators used by protocol supported operations should not be made explicitly available through control requests. They are simply used locally by the server on the client's behalf and without the client's knowledge.

#### 3.4.1.8.3 Image Capture Hardware – Live Video

The core X protocol provides for human interface and interaction mechanisms. This includes the means to pass information from input devices such as keyboards and mice on to the client side of the wire where the application decision is made regarding its impact on operations or display.

In certain instances, there may be very close proximity between image capture and display hardware. In these cases, the capture device may be controlled through the same hardware interface as the display subsystem and may actually digitize an image and deposit it directly into the display memory (RS170 frame grabber and frame buffer display memory typically share board space and bus interface mechanisms).

As the capture of an image is not strictly a human interaction or signaling mechanism such as keyboard input or mouse pointing, a hard line position is that image capture devices should not logically be on the server side of the X wire. But, the tight coupling or intimacy of such capture devices with the display makes it at least convenient to provide control mechanisms in the protocol. Support for control of time sequences of digitized images which derive from analog video sources (cameras, vcr's, broad band broadcast...) and pass through the digital domain on the server via frame digitizers, make it a requirement that the control mechanisms be in the protocol.

As such, the policy is that the extension package will provide protocol mechanisms (control requests) for image input devices (such as frame grabbers) which are tightly coupled to the display hardware (accessed through common hardware interface and sharing display memory). These mechanisms will support the windowing of the video information into X windows. This is in line with the concept of a tightly packed "capture–>render–>display" pipeline and time sequence support.

The policy will allow control over tightly coupled image capture devices (frame grabbers) and live video windowing (video rate time sequencing) to occur under X. Video from digital or analog sources which can be presented to the display subsystem at video frame rates in DIGITAL (vs analog) format and can be converted back to analog format by the display subsystem in a way which is compatible with the display refresh mechanisms, can be displayed in application specified X windows. The policy will NOT support analog video windowing (switching / mixing analog video in the display subsystem, control over analog broadcast video equipment etc.).

Applications which should benefit from this capability are those which use video information (time sequences of individual frames) which derives from a camera or perhaps a laser disk, and which is in digital format already, or is converted by devices such as frame grabber/digitizers into digital format within the server display subsystem.

Note that this position still leaves loosely coupled image capture devices which happen to be plugged into an open bus on the server node (such as document scanners) on the client side of the wire (they will not be supported in the image extension protocol). It also leaves the computer interfaces (typically rs232 serial line ports) of video source devices such as "video disc players", vcr's, video tuner's, etc. on the client side of the wire.

#### 3.4.1.8.4 Extensibility

The core X system wisely provides the ability to extend itself into various application domains. The image extension package is one such extension into the imaging application domain.

In order to be useful, the imaging extension package must also provide for growth and extension. Perhaps the most troublesome area for the extension package is image enhancement processing where the number of algorithms, approaches, and combinations is limitless. Providing an open extension mechanism at the protocol level for this area is inviting disaster.

Instead, the policy will be to attempt to achieve consensus regarding what is the basic processing machinery necessary to support multiple enhancement approaches. This machinery will reside on the server side of the wire and be accessed through the wire protocol. The server side processing machinery will be kept to a very basic level and no extension mechnicanism will be provided.

Flexibility is achieved through the use of parameters for this machinery, through the ability to build higher level enhancement processes with various combinations of the machinery, and through low level support for these macros on the server side in terms of "process pipelines".

The extension capability will be provided at the tool kit level where the problem of a potentially explosive number of "my favorite image enhancement process" can better be dealt with.

### 3.4.2 Client–Server Model

The image extension follows the core protocol in the use of a client–server model. Functions which the server performs on behalf of the application client are known as services. Although most services accomplish things which an application cares directly about, some services involve the overhead of the client server model and potential physical separation.

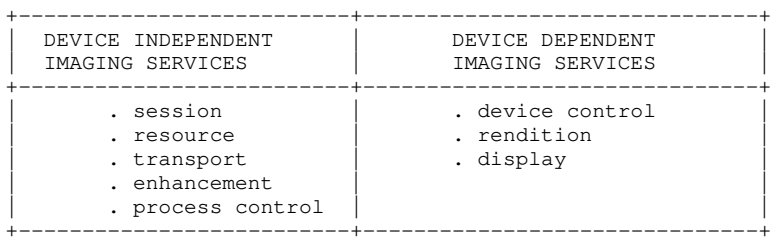
#### 3.4.2.1 Server Resources

Within the client portion of the application, the application maintains private data structures for storage of image data, and applications state and control information. Within the server portion of the application, objects which are maintained and used by the server in providing the services, and are known about by the client, are referred to as resources. The extension package resource concepts are presented in the chapter on extension protocol elements.

#### 3.4.2.2 Service Classes

A taxonomy of imaging services may be organized at the lowest level into CLASSES based upon com–monalty of purpose, and organized at the highest level into GROUPS based upon device dependence. This section simply sets down the aggregation. The next chapter explores the contents in more detail.

**Figure 3–2: SERVICE CLASS GROUPS**




Within the X image extension server, imaging service classes may be grouped into two major categories: those which are device independent, and those which are device dependent. Although all service implementations may in some way depend upon the device they reside upon, some services require intimate knowledge of the display and display processing subsystem to perform their job, while others do not. As such, it is service dependence upon intimate knowledge of the display subsystem that is the metric for group categorization of the services.

Although there is a logical relationship (in terms of position within the X model levels diagram) between the two service groups, the physical relationship is not quite as simple as all device independent services being built upon the device dependent services. In fact, to a large extent, the device independent services are not built upon the device dependent services.

#### 3.4.2.3 Service Events

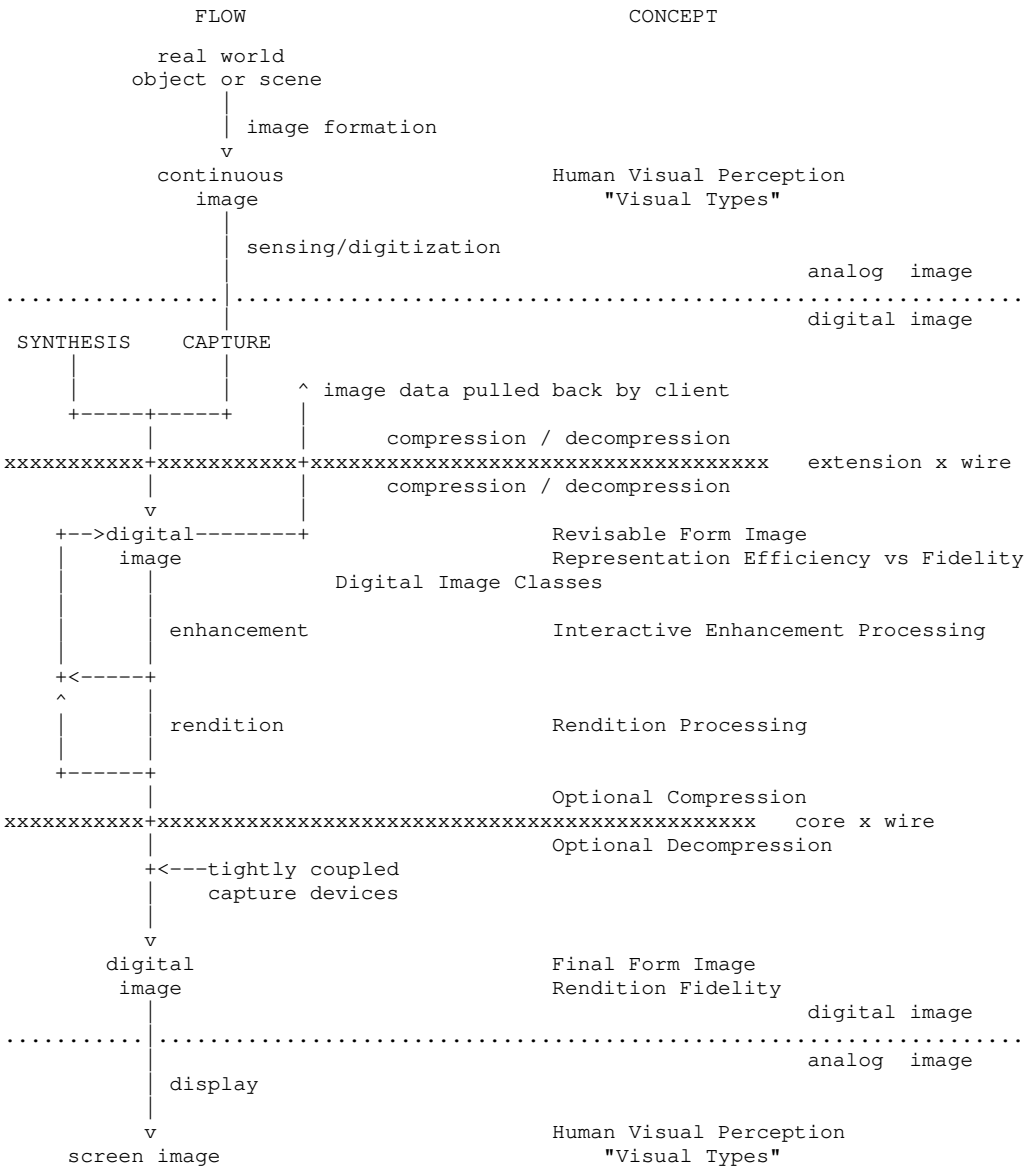
All core protocol event mechanisms will be available to image extension package users. In addition, a set of image extension package specific events will be available. There is no relationship between these two sets of events other than that established by an application. Image specific events are discussed briefly in the chapter on extension protocol elements.

## 3.5 Transfer Model vs X Wire

The image transfer process depicted in Figure 2–1 shows how continuous images are captured, and how captured and synthesized digital images move through enhancement, and rendition processes towards display. Figure 4–3 shows the location of the core X protocol wire and the extension X protocol wire with relationship to stages of the transfer model.

There is a significant shift of the X protocol wire between the core and the extension protocol with regards to the transfer model. Based upon the guidelines set forth in the previous section, the extension protocol moves rendition processing (which is a display device dependent function) across the wire into the server. The extension also lets those elements of interactive enhancement move across the wire.

**Figure 3–3: TRANSFER MODEL & X WIRE LOCATION**



# Chapter 4

# EXTENSION PROTOCOL ELEMENTS

This chapter presents an overview of the extension protocol including service classes, server resources, and an operational model.

## 4.1 Services

Services are the functions performed by the server for the client. In the protocol, they are implemented as client requests. Some of the services listed below are potentially provided by or through the use of existing core protocol services (such as resource class services). They are listed here for completeness.

### 4.1.1 Device Independent Service Group

#### 4.1.1.1 Session Class Services

Session class services establish the bounds and semantics of an image extension client–server session. Key activities such as server initialization and termination, as well as issues such as server capabilities and temporal scope of service requests are involved in session class services.

Examples (non–exhaustive list) of session class services include:

- Connect to extension server
- Disconnect from extension server

#### 4.1.1.2 Resource Class Services

Resources are a means by which important operational information may be organized by the server and accessed by the client. To some extent, they represent both housekeeping and an organizational strategy.

Examples of resource class services include:

- Create resource
- Delete resource
- Modify resource
- Query resource
- Copy resource

#### 4.1.1.3 Transport Class Services

Transport involves the movement of image objects across the X wire. This movement is both incoming (client–>server) and outgoing (server–>client) image data.

Efficiency and convenience are key concepts in transport services. Efficiency involves the support of encoded (compressed) image data, while convenience involves the ability of the client to specify image data formats, and to break complete image transport into multiple steps.

Transport processing includes the movement across the wire of both PCM (uncompressed) and compression encoded image data, and the actual encoding and decoding of this data by both client and server.

Supported PCM formats include:

- FULL_PIXEL_COMPATION
- PARTIAL_PIXEL_EXPANSION
- FULL_PIXEL_EXPANSION

Supported Compressed formats include:

- CCITT Modified Huffman and Modified Modified Read
  for bitonal data and continuous tone monochromatic/polychromatic
  in FULL_PIXEL_EXPANSION mode
- Discrete Cosine Transform
  for continuous tone monochromatic/polychromatic
  in PARTIAL_PIXEL_EXPANSION mode

Refer to the chapter on lexicon for more details of these formats.

Examples of transport class services include:

- Send image object to server
- Receive image object from server

#### 4.1.1.4 Statistics Class Services

Statistical information about image data is often used in enhancement processing strategies and operators. Statistical services do not modify image data, but rather generate statistical information about image data. In certain instances, the statistical information maybe calculated on a pixel by pixel basis and may itself be stored in a server–based destination image object. The resulting "statistical image" maybe used in enhancement processing operations.

Examples of statistics class services include:

- Calculate Statistical Moments
  - Area Minimum
  - Area Maximum
  - Area Mean
  - Area Median
  - Standard Deviation
  - Variance
- Generate Histogram

Due to the potentially unbounded nature of enhancement processing, enhancement class services will be restricted to include only a minimal set of basic enhancement processing machinery that will support multiple enhancement approaches. Parameters and application of this machinery will be left to the direction of the toolkit and application client levels. An example of this approach is to include filtering capabilities (spatial convolver) but not the filters themselves.

#### 4.1.1.5 Enhancement Class Services

In the enhancement processing of an image, the server uses local server hardware to apply image processes to the image data in the server–based source image object. The result of the operation is stored in a server–based destination image object which may be the same as the source image object.

An important enhancement concept is that both the input and output of the enhancement services are revisable form image objects. Thus, the results of an image enhancement operation may be fed back into other enhancement operations, made available for transport back to the client, or made available for rendition processing.

Enhancement processing is a particularly rich and popular area. While the X image extension package tool kit will be the home for many high level image enhancement processing algorithms and processing macros, the wire protocol for enhancement services will be kept minimal. Examples of enhancement class services include:

- Point Processing
  - Algebraic
  - Logical
  - Luminance
- Area Operators (Filtering)
- Geometric Transforms

Due to the potentially unbounded nature of enhancement processing, enhancement class services will be restricted to include only a minimal set of basic enhancement processing machinery that will support multiple enhancement approaches. Parameters and application of this machinery will be left to the direction of the toolkit and application client levels. An example of this approach is to include filtering capabilities (spatial convolver) but not the filters themselves.

#### 4.1.1.6 Process Control Class Services

The server will support pipelining and interleaving of atomic server operations. This provides a home for the very useful and everyday concept of tightly coupled transport–>render–>display, as well as more elaborate concepts such as interleaving enhancement, rendition, and display of large image objects, and enhancement processing macros.

The image extension will suport both primitive pipelines (ones which contain only references to atomic protocol functions), and nested pipelines (ones which reference other pipelines).

With process control, the interleaving of many mundane operations can be efficiently represented and performed on the server side. The inclusion of pipeline control operators will provide for unattended and client intervention (interactive) pipeline execution.

Examples of process control services include:

- Create pipeline
- Delete pipeline

- Modify pipeline
- Query pipeline content
- Query pipeline state
- Execute pipeline
- Terminate pipeline

## 4.1.2 Device Dependent Service Group

### 4.1.2.1 Rendition Class Services

In the rendition of an image, the server uses a combination of local server software and hardware to apply rendition operations under the direction of XIE toolkit based rendition models to the image data in the server–based source image object. The particulars of these operations depend upon the state of the image object, the display characteristics, client requests, and the rendition models.

Rendition services are designed to offer the client substantial flexibility. Rendition services are divided into two classes: rendition functions which prepackage commonly used rendition models and offer the highest possible fidelity (fill screen, fill window, preserve aspect ratio); and rendition functions under complete control of the client. The former services are packed in XIE toolkit functions. The latter case allows the client to control (override) default operation in order to prevent the "too smart by half" syndrome (server keeps client from rendering the image the way the client wants it, because server thinks it knows better). A null rendition function is also included.

Low level (server based) rendition processing functions take a revisable form image and produce another revisable form image. High level (XIE toolkit based) rendition processing functions use the low level functions in processing pipelines. The last step in these high level pipelines is exporting the revisable form image into a PIXMAP which is considered to be a final form image. The final form image is suitable only for display purposes. It is possible to convert a final form image back to a revisable form image via the import/export class services (see next section).

Examples of high level (XIE toolkit based) rendition services include:

- Render best (rendition fidelity)
- Render this way
- Render to fit object in window
- Render at real size (whatever fits)
- Render preserving aspect ratio

Examples of low level (server based) rendition services include:

- scale
- rotate
- crop
- dither

#### 4.1.2.2 Import/Export Class Services

It is expected that at least the core X protocol and services will be coresident with the image extension protocol and services, and that other extension packages may be resident as well. Many applications, such as compound document editors, may have need of multiple domain extension packages at the same time and it is quite reasonable to expect an application to manage sessions with multiple resident server packages.

Each of these packages may define their own resources and the organizational and operational semantics of these resources. For efficiency reasons , XIEP provides import/export services which are designed to allow the data portion of an image object to be taken from or placed into core protocol structures, thus saving the application two potentially large wire transmissions and allowing an application to use the power of an image extension on data that may derive fromthe core package or perhaps a graphics extension.

Included in the import/export services is the ability to take an image which has been rendered (converted to final form) and make a new revisable form image from it (see previous section).

#### 4.1.2.3 Display Class Services

Although closely associated, display is a distinct step from rendition. Once the rendition operations are complete, the client may request display of the rendered image. The server is responsible for loading display device hardware with the rendered image object to make the image viewable on request of the client.

Display class services will support single frame and time sequences of frames. Time sequence support is deemed necessary to ensure uniform time spacing. Multi−image display (2−N up etc.) will be supported at the XIE toolkit level through multiple display class service requests.

#### 4.1.2.4 Device Control Class Services

In general, the extension protocol does not offer direct access to specialized server hardware. Instead, this hardware (accelerators in particular) is accessed indirectly through protocol functions which benefit by use of the hardware.

Device control functions are protocol functions which offer direct access to display device hardware. They are included primarily to take care of the few cases where a protocol function degenerates to direct hardware access (as in frame capture operations), and secondarily, as an escape mechanism.

There are a number of dangers associated with this service class. The foremost is that unwise and excessive inclusion of device control protocol functions is bound to lead to incompatibilities and defeat the purpose of a common protocol. Perhaps a more specific danger is conflict between window control provided by the core protocol and overall device control made possible here.

The X Image Extension will provide device control access services for a limited set of device controls. These services will affect device control attributes associated with individual windows (ie. logical device controls). The extension will use a mechanism similar to that used with lookup tables (colormap) to map logical to physical device control settings as windows are selected.

Examples of display device control services include:

- Set/query intensity
- Set/query contrast
- Set/query hue
- Set/query brightness
- Set/query saturation

- Image capture controls
  - live video windowing
  - frame or page grabbing

## 4.2 Resources

The purpose of this section is to provide an overview of the server resources of which the client has knowledge and ability to reference. The existence and maintenance of these structures represents an object oriented organizational strategy towards managing the control and state information involved in providing the services. These structure are referenced through the resource id in image service requests. The structures are not bundled into a mega structure so that mix and match flexibility is maintained at service request time. Mega resource structures may be supported in the XIEtoolkit if it makes sense.

The overview describes the purpose of the resource, general content, and usage. Where relationships between server resources exist, they are pointed out.

### 4.2.1 Image Object – IOB

An image object (IOB) is a structure which contains the image data and closely associated attributes within a server. IOBs are always revisable form entities. As such, an image object can be a source or destination in a transport, enhancement, or rendition process. IOBs are not used in display operations. XIEP provides the means by which the output of a render operation which produces a final form image, or input to a display operation (final form images) may be converted back into an IOB (revisable form images).

Clients know IOBs as resources. A client can create and delete an IOB, and modify as well as query the attributes section of an IOB. The only way a client can directly (ie. other than by a server based processing operation) modify or query the image data section of an IOB is through a transport operation.

In general, an IOBs is an operand to most transfer, enhancement, and rendition requests. In all cases, an image context of one form or another is also involved in these operations, and except where explicitly noted, the same IOB may be used as the source and destination for an operation. In general, IOBs are to be considered as very scarce resources by the client, and server space considerations may cause multiple IOB creation requests to fail.

### 4.2.2 Image Enhancement Context – IEC

An Image Enhancement Context (IEC) is an association of image processing operands and operators used in the enhancement processing of an image.

An IEC resource is referenced every time an image object or region thereof is to be processed for enhance–ment purposes.

Its primary purpose is to allow enhancement steps to be represented in processing macros (see Image Control Context).

Clients know IECs as resources. A client can create and delete an IEC, and modify as well as query the contents of an IEC. The same IEC can be used for many enhancement processing requests, and the relationship between an IEC and IOB is established and maintained only during an enhancement operation. A client can maintain multiple distinct IECs which can be used to render the same (or different IOB's) in different ways.

### 4.2.3 Image Process Context – IPC

An Image Process Context (IPC) is a structure which is used for one of three purposes:

1. define the subset of an image to process
2. act as an area operand in area image operations
3. both of the above (concurrently)

When used for subsetting purposes, an IPC provides control over the sub areas of an image object (objects) to which enhancement processing or rendition operations should be applied. In essence the IPC selects the subset, or specific points in an image to which the operation is applied (acts as a boolean control mask). Two forms of IPC exist for this purpose: region of interest (ROI), and control processing plane (CPP).

When used as an area operand, an IPC specifies a set of operands and their spatial relationship. The spatial relationship determines how the operands in the set are applied under area operations. Two possibilities exist: fully populated and sparsely populated. An example of a fully populated IPC area operand is a 3x3 or 5x5 convolution kernel mask where all elements in the mask have a value and are used in the area operation – the spatial relationship between the operands is unambiguous. In the sparsely populated case, the values of the individual operands are specified for a subset of a square region. As such, the locations of these operand values must be specified explicitly along with the values. In this case, the elements that constitute an area operation only occur at the specified locations (an example would be a cross shaped filter). IPC's used for the purposes described in this paragraph will be referred to as templates.

Note that a major difference between the first two classes is that in the first, the IPC is simply a "selector", while in the second, the IPC is used both for area operation selection and operand value specification.

#### 4.2.3.1 Region of Interest – ROI

A region of interest (ROI) is a closed form (analytic) definition of a single, fully connected, subregion of an image. ROI's define the geometric bounds on an image object within which the operation is to be performed and outside of which it is not to be performed.

Regions of interest can be any two–dimensional closed shape defined with reference to an image object's coordinate system, such that the question of within can be answered for any pixel in the image object. It should be possible to define closed form analytic regions such as circles and rectangles, as well as more general closed polygons defined by a list of coordinate vertices.

Regions of interest are optional operands in image enhancement processing and rendition requests. When not specified, the system presumes that the user specified a region of interest of the same geometric structure and equal to the full extent of the image object upon which the operation is being performed.

A region of interest can only be used to control the subregion to which processing applies. It has no other affect on the semantics of the operation (no control mechanisms).

Clients know ROIs as IPC resources. A client can create and delete an ROI, and modify as well as query the contents of an ROI. The same ROI can be used for many transport, enhancement, and rendition requests, and the relationship between an ROI and IOB is established and maintained only during an the operation. A client can maintain multiple ROI's reflecting different regions of image objects which may be of interest.

An ROI is defined with relation to the coordinates of an IOB.

If at all possible, the image extension will use the core X graphics clipping structures as the region of interest resource. In this case, the image extension will simply define the semantics of the core X clipping structures when used in conjunction with imaging services.

#### 4.2.3.2 Control Processing Plane – CPP

A Control Processing Plane (CPP) is a mechanism used only in enhancement processing. Its purpose is to allow control on a pixel by pixel basis over the application of an enhancement process. It can be used both to select subregions, and to control the particulars of certain operators.

CPPs are only used in enhancement processing and cannot be used in transport or rendition processing. Unlike ROIs, CPPs do not have an analytic form of representation.

Clients know CPPs as IPC resources. A client can create and delete a CPP, and modify as well as query the contents of a CPP. The same CPP can be used for many enhancement requests, and the relationship between a CPP and IOB is established and maintained only during an the operation. A client may maintain multiple CPPs.

CPPs are in essence the data component of a revisable form image object (IOB), and may be implemented as IOB's (one of two IOB operands to an image enhancement operation). It may also be possible to use the core X control plane structures as the control processing plane resource. In this case, the image extension will simply define the semantics of the core X control structures when used in conjunction with imaging services.

Some examples of how CPP's maybe used may help explain the breadth of it's potential applications. In the case where the user needs to specify the pixels (on a point by point basis rather than the ROI bounded area case) to which pointwise algebraic operations are applied, a CPP which is a simple bitonal image (contro map) may be used. In the case where areawise algebraic operations are to be performed (such as filtering), weighted area masks including spatial convolution filter kernels may be specified as a CPP with zero and non zero integral values.

#### 4.2.3.3 Templates – TMP

A template (TMP) is a mechanism used only in enhancement area processing. Its purpose is to allow control over which local area source image pixels (pixels near the current target pixel) participate in area operations, and values of the second area operand on a pixel by pixel basis.

TMPs are only used in enhancement processing and cannot be used in transport or rendition processing. Unlike ROIs, TMPs do not have an analytic form of representation.

Clients know TMPs as IPC resources. A client can create and delete a TMP, and modify as well as query the contents of a TMP. The same TMP can be used for many enhancement requests, and the relationship between a TMP and IOB is established and maintained only during an the operation. A client may maintain multiple TMPs.

### 4.2.4 Image Control Context – ICC

An Image Control Context (ICC) is a structure which describes the relationship between the various transfer, enhancement, and rendition processes that make up a image processing pipeline. It is possible with an ICC to describe enhancement processing macros, as well as various forms of transfer, enhancement, rendition and display pipelines.

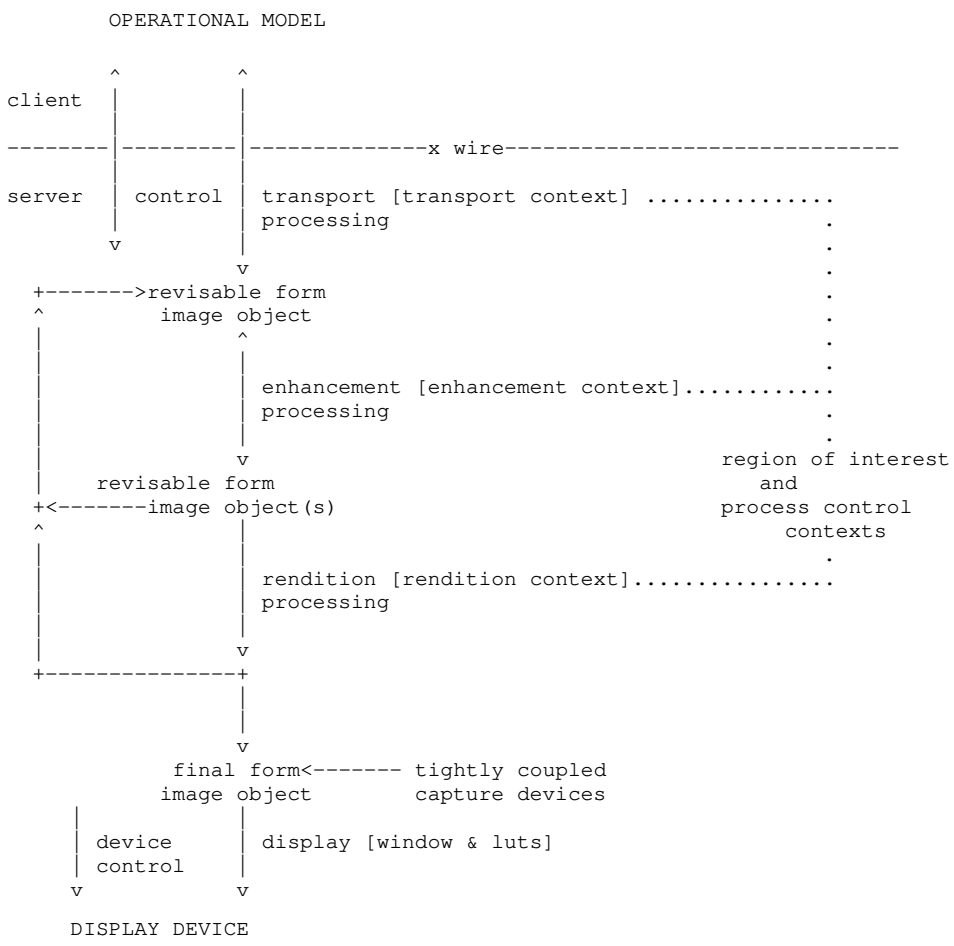
As an ICC is a higher level structure which may imply transfer, processing, and rendition, it is not referred to in those atomic requests, but rather in pipeline (process control) requests which explicitly signal the existence and use of the ICC.

### 4.2.5 Image Display Context – IDC

An Image Display Context (IDC) is a structure which controls the display of a final form image or sequence of images. It contains the connection between revisable form image, final form image, display window, display sequence, and sequence timers. IDC's used to control the display of image sequences are also known as slide carousels.

## 4.3 Service Operation

**Figure 4–1: OPERATIONAL MODEL**




### 4.3.1 Flow

The image extension protocol follows the same basic operational model as the core protocol. In particular, clients communicate with servers through requests, and servers communicate with clients through replies to requests, errors, and events.

An image application begins by establishing a connection using the core X11 protocol to an X server. The application then queries the server in order to determine if support exists for the image extension protocol, and what level of support exists. If the image extension is supported, and at an appropriate level, the application establishes a connection to the extension package (session services) through (within) the core X1 protocol connection.

Once the connection has been established, the client and toolkit levels use resource requests to allocate and load critical server resources. With these resources allocated, the application can make image data transport, enhancement, rendition, display, and display control requests of the server.

Upon completion, the image application disconnects first from the image extension package, and then from the X core package.

Using this model, an image application can continue to run as it has in the past. In this case, the component of the application on the client side of the wire takes full responsibility for maintaining all of its private image data structures, for applying all processing operations to these structures, for understanding the display device characteristics, and for rendering an image suitable for display in a window. In short, the client side of the application continues to bear responsibility for generating final form images. XIEP simply provides the means to transport the final form image to the display.

At the other extreme, an image application may cause most or all of its image objects to reside on the server side of the wire. In this case, all enhancement processing and rendering requests are made through the extension package (XIE Toolkit, XIE Lib, XIE Server), and the extension package bears responsibility for generating final form images.

### 4.3.2 Requests and Replies

Requests and replies are the primary means by which the client and server communicate. The image extension requests will be aggregated by the service class to which they belong (i.e. the service classes will be broken down into requests and replies).

### 4.3.3 Errors

Errors are notifications to the client of server problems with requests.

### 4.3.4 Events

Events are situations or occurrences which are detectable by the server and of potential interest to the client. Bear in mind that multiple clients may share a single server, and as such,a client may be notified of an event caused by a different client. Image extension events might include changes in overall display characteristics (brightness, contrast,...), transport complete, processing complete, rendition complete, and time sequence display complete.

# Chapter 5

# TERMINOLOGY

The terminology section is segmented in a topical fashion in order to keep closely associated terms together. Although terms are capitalized within this section, they are not capitalized or visually differentiated from surrounding text in the body of the protocol.

## 5.1 Core Protocol

The following terms are defined in the core protocol and used at times in the extension protocol.

**Table 5–1: Core Terms**

| | | | |
|---|---|---|---|
| Access ctl list | Active grab | Ancestors | Atom |
| Backing store | Bit gravity | Bitmap | Button grabbing |
| Byte order | Children | Client | Clipping regions |
| Color cell | Colormap | Connection | Containment |
| Coordinate system | Cursor | Depth | Device |
| Drawable | Event | Event mask | Event sychronization |
| Event propagation | Event source | Exposure event | Extension |
| Grab | Graphics context | Hotspot | Identifier |
| Inferiors | Input focus | Input manager | InputOnly window |
| InputOutput window | Key grabbing | Keyboard grabbing | Mapping |
| Modifier keys | Obscures | Occludes | Padding |
| Parent window | Passive grab | Pixel value | Pixmap |
| Plane mask | Pointer | Pointer grabbing | Pointing device |
| Property | Property list | Redirecting control | Reply |
| Request | Resource | Root | Root window |
| Save set | Screen | Server | Server grabbing |
| Sibling | Stacking order | Stipple | Tile |
| Timestamp | Type | Unviewable | Viewable |

**Table 5–1 (Cont.): Core Terms**

| | | | |
|---|---|---|---|
| Visible | Window gravity | Window manager | XYFormat |
| ZFormat | | | |

## 5.2 Extension Protocol

The extension protocol descriptive terminology section is intended to be sufficient for clearly communicating concepts. It is not intended to be exhaustive.

### 5.2.1 Image Definitions

**CONTINUOUS_IMAGE**

An optically formed counterpart of a real world object or scene. An example would be what is found on the shutter or focal plane of a 35 mm camera.

**DIGITAL_IMAGE**

The discrete version of a CONTINUOUS_IMAGE. The continuous image is sampled at the vertices of a regular two dimensional SAMPLING_GRID, and then quantized to a fixed number of levels. In practice, a digital image maybe computed, synthesized, or randomly generated, as well as deriving from optical processes.

**IMAGE_OBJECT**

An image object is the representation of a digital image which a computer application manipulates. It consists of IMAGE_DATA plus IMAGE_ATTRIBUTES. Unless explicitly stated otherwise, any use of the term "image" in the specification implies the term IMAGE_OBJECT.

**IMAGE_DATA**

IMAGE_DATA consists of the bits which represent the result of the digitizing or synthesis process.

**IMAGE_ATTRIBUTES**

IMAGE_ATTRIBUTES are the ancillary information required to interpret, process, and present for viewing IMAGE_DATA. There are three classes of independent IMAGE_ATTRIBUTES:

- IMAGE_PRESENTATION_ATTRIBUTES
- IMAGE_ORGANIZATION_ATTRIBUTES
- IMAGE_ENCODING_ATTRIBUTES

**IMAGE_PRESENTATION_ATTRIBUTES**

The attributes of an image which describe how it is to be presented to a user for viewing. IMAGE_PRESENTATION_ATTRIBUTES have no bearing upon the organization and amount of image data. Often, it is desirable to make the values of these attributes identical to those established by the capture or synthesis process so that the image will appear identical to what it looked like at capture or synthesis time. (see Image Data Presentation Definitions).

**IMAGE_ORGANIZATION_ATTRIBUTES**

IMAGE_ORGANIZATION_ATTRIBUTES provide information about the structure and association of the bits which make up the two dimensional image. (see Image Data Organization Definitions).

**IMAGE_ENCODING_ATTRIBUTES**

IMAGE_ENCODING_ATTRIBUTES define the mapping between the unencoded SYMBOL_SET and the encoded SYMBOL_SET where encoding is done primarily for compression or bandwidth reduction purposes. (see Image Data Encoding Definitions).

**IMAGE_APPLICATION**

A computer application that deals in the input, output, storage, or processing of IMAGE_OBJECTS. This protocol specification is primarily concerned with the processing and output of IMAGE_OBJECTS to displays.

**FIDELITY**

The degree to which the computer system can reproduce a likeness of the original (source) continuous image without distortion. Issues affecting fidelity including the sampling and display SPATIAL_ RESOLUTION, and the LUMINANCE_RESOLUTION or CHROMINANCE_RESOLUTION.

**REPRESENTATION_EFFICIENCY**

A measure of how few bits can be used to represent a particular DIGITAL_IMAGE with some specified degree of FIDELITY.

**FINAL_FORM_IMAGE**

A FINAL_FORM_IMAGE is one that has been rendered suitable for display on a particular device. All internal structural entities have been removed, and all rendition processing completed. The generation of a FINAL_FORM_IMAGE from a REVISABLE_FORM_IMAGE may involve irreversible TRANSFORMs such as resolution reduction in the x,y, and z directions. It is always possible to take a FINAL_FORM_ IMAGE and redefine it as the starting point for a new and entirely different REVISABLE_FORM_IMAGE.

**REVISABLE_FORM_IMAGE**

A REVISABLE_FORM_IMAGE is the source image used in reversible and irreversible image rendering operations. A REVISABLE_FORM_IMAGE may (by chance) be suitable for display as is, on a particular device. If this is the case, the REVISABLE_FORM_IMAGE is converted to a FINAL_FORM_IMAGE by performing a null rendering operation. REVISABLE_FORM_IMAGEs may maintain some internal organization or structure and continue to carry rendition attributes along with them

**DISPLAY_GRID**

The regular, two dimensional grid at whose vertices the original image is reconstructed during display. A display grid location is often referred to as a PIXEL.

**SAMPLING_GRID**

The regular, two dimensional grid at whose vertices the original image is sampled during digitization. A sampling grid location is often referred to as a PIXEL.

### 5.2.2 Image Characteristics

**SPATIAL**

The aspects of an image which relate to space (typically x & y dimensions, and resolution in these dimensions) including how the image varies with respect to the spatial dimensions.

**TEMPORAL**

The aspects of an image which relate to time (t dimension, and resolution in this dimension) (typified by the difference between a still frame, and a motion sequence) including how the image varies at any SPATIAL location with respect to the time dimension.

**RESOLUTION**

The fineness or granularity of an image measured either at the capture or rendition point. The granularity is usually expressed in both spatial frequency (dots per inch) and temporal frequency (frames per second). Resolution is a meaningless term away from the capture and rendition points. Instead, total pixels per scan line, total scan lines, and total frames in a sequence are more useful.

**SPATIAL_RESOLUTION**

A measure of the separation between adjacent points (in both SPATIAL dimensions) or granularity of the SAMPLING_GRID and / or the DISPLAY_GRID. SPATIAL_RESOLUTION has nothing to do with image size or total data content. It has to do with fineness at capture and output times.

**TEMPORAL_RESOLUTION**

A measure of the separation in time between the capture or display of two adjacent image object frames.

**SPATIAL_FREQUENCY**

A measure of the rapidity of image change with respect to space.

**TEMPORAL_FREQUENCY**

A measure of the rapidity of image change with respect to time.

**INTENSITY**

The square of the amplitude of the electro_magnetic field as measured at the focal plane of the imaging device. Intensity is what we perceive as brightness.

**INTENSITY_LEVELS**

The number of distinct brightness levels represented in the image.

**CONTRAST**

The difference between minimum and maximum INTENSITY levels in an image.

**LUMINANCE**

The component of an image which represents the INTENSITY at either a single wavelength or averaged over a wavelength band. INTENSITY, LUMINANCE, and brightness are psynonomous.

**CHROMINANCE**

The component of an image which describes the dependence of INTENSITY on wavelength or wavelength band.

### 5.2.3 Visual Types

**VISUAL_TYPE**

A label applied to the human perception of a continuous image based upon perceived characteristics of intensity distribution, and dependency upon wavelength and time. The continuous image of an object (and occasionally the object itself) is said to be of one of the following VISUAL_TYPEs include BLACK_ AND_WHITE, GREY_SCALE, COLOR, STILL, and MOTION.

**BLACK_AND_WHITE**

A VISUAL_TYPE where the continuous images have very high contrast, and no perceived temporal or chrominance dependency (i.e. what we perceive when reading this page).

**GREY_SCALE**

A VISUAL_TYPE where the continuous image has a relatively wide range of intensities which vary smoothly, and no perceived temporal or chrominance dependencies. GREY_SCALE is perhaps more an artifact of early television and photography than of nature.

**COLOR**

A VISUAL_TYPE where the continuous image has a wide range of luminances and chrominances which vary smoothly, and no temporal dependencies.

**STILL**

A VISUAL_TYPE which demonstrates no temporal variation (e.g. this page).

**MOTION**

A VISUAL_TYPE which demonstrates temporal variations (e.g. television).

### 5.2.4 Image Classes

**IMAGE_CLASS**

The grouping of a digital image based upon which visual type it represents with highest efficiency and fi– delity of reproduction. IMAGE_CLASSes include MONOCHROMATIC–BITONAL, MONOCHROMATIC– CONTINUOUS TONE, and POLYCHROMATIC–CONTINUOUS TONE or MULTISPECTRAL. All three classes have either the STATIC or TIME_VARYING modifier.

**MONOCHROMATIC–BITONAL**

An IMAGE_CLASS in which two INTENSITY_LEVELS are represented. At sampling and display times, a PIXEL for the MONOCHROMATIC–BITONAL IMAGE_CLASS is one bit deep. The display of an image of this class either does not use a COLORMAP, or uses the BLACK and WHITE entries of the COLORMAP. The MONOCHROMATIC–BITONAL IMAGE_CLASS is optimal (REPRESENTATION_ EFFICIENCY and RENDITION_FIDELITY) for the BLACK_AND_WHITE VISUAL_TYPE.

**MONOCHROMATIC–CONTINUOUS_TONE**

An IMAGE_CLASS in which 2**N –1 INTENSITY_LEVELS are represented. At sampling and dis– play times, a PIXEL for the CONTINUOUS_TONE IMAGE_CLASS is N bits deep where N > 1. CONTINUOUS_TONE is accomplished on color displays with the PIXEL value being used as an in– dex into a COLORMAP, and the same value used to drive the RED, GREEN, and BLUE display guns. The MONOCHROMATIC–CONTINUOUS_TONE IMAGE_CLASS is optimal (REPRESENTATION_ EFFICIENCY and RENDITION_FIDELITY) for the GREY_SCALE VISUAL_TYPE.

**POLYCHROMATIC−CONTINUOUS_TONE or MULTI_SPECTRAL**

An IMAGE_CLASS in which multiple (>2) INTENSITY_LEVELS are represented for multiple (2 or more) SPECTRAL_COMPONENTs. MULTI_SPECTRAL is typically accomplished on color displays via the use of the COLORMAP. The POLYCHROMATIC_CONTINUOUS_TONE or MULTI_SPECTRAL IMAGE_CLASS is optimal (REPRESENTATION_EFFICIENCY and RENDITION_FIDELITY) for the COLOR VISUAL_TYPE.

**STATIC**

A modifier of the other IMAGE_CLASSes indicating no TEMPORAL dependency, and implying only one unchanging IMAGE_OBJECT.

**TIME_VARYING**

A modifier of the other IMAGE_CLASSes indicating TEMPORAL dependency, and implying multiple IMAGE_OBJECTs which represent the continuous image at different times (usually a time sequence).

### 5.2.5 Image Data Organization

**IMAGE_DATA_ORGANIZATION**

The way in which a logical PCM model of IMAGE_DATA is mapped onto a physical storage model. The overall purpose is to enable convenient association of and access to all of the bits which represent the information found at each sampling grid point. Three significant IMAGE_DATA_ORGANIZATIONS exist, each possessing different access characteristics. The core X protocol recognizes only two of these formats (see names in parenthesis).

- FULL_PIXEL_COMPACTION (Z format in core X)
- FULL_PIXEL_EXPANSION (X−Y format in core X)
- PARTIAL_PIXEL_EXPANSION (unrecognized in core X)

**PIXEL**

A DISPLAY_GRID or SAMPLING_GRID location. Often used to mean the collection of bits representing the image information at the sampling and display grid locations. This second usage is somewhat dangerous as in modern displays there may be no simple stack of bits behind the display at each DISPLAY_GRID location representing the image information for that PIXEL.

**BITS_PER_PIXEL**

In DISPLAY_SUBSYSTEMs which do maintain a FRAME_BUFFER with the image information for each PIXEL collected into a stack of N bits, the number N. Unless otherwise qualified, the term BITS_PER_ PIXEL implies the total number of bits which when aggregated form a PIXEL. Occasionally, the qualified phrase BITS_PER_PIXEL_PER_SPECTRAL_COMPONENT, or BITS_PER_PIXEL_PER_PLANE_ GROUP will be used. Either of these values must by definition be less than or equal to BITS_PER_PIXEL.

**BIT_PLANE**

A two dimensional collection of bits such that there is one and only one bit at each of the points in the two dimensional grid. The stacking of all BIT_PLANES which form an image results in a PIXEL at each grid point.

**PLANE_GROUP**

An organized collection or stack of PLANEs. A PLANE_GROUP may be from 1 to N PLANES deep where N is the number of BITS_PER_PIXEL.

**PCM**

Pulse Code Modulation (or raw bits) is the reference form for uncompressed images. In PCM the image symbol set consists of PIXELS, where PIXELS are the quantized representation of image luminance or CHROMINANCE channel as sampled at regular two dimensional grid points.

**FULL_PIXEL_COMPACTION**

An IMAGE_DATA organization in which all bits which make up all SPECTRAL_COMPONENTs of a PIXEL are logically and physically adjacent in transmission and storage. All bits of an N bit PIXEL are aggregated to form an element of a single, two dimensional array. PIXEL bit aggregates may be separated by pad bits for alignment purposes.

**FULL_PIXEL_EXPANSION**

An IMAGE_DATA organization in which all bits which make up all SPECTRAL_COMPONENTs of a PIXEL are physically separated such that every bit of an N bit PIXEL is an element of a separate two dimensional array (PLANE), there being N such arrays (PLANEs).

**PARTIAL_PIXEL_EXPANSION**

An IMAGE_DATA organization in which all bits which make up each SPECTRAL_COMPONENT are physically adjacent by SPECTRAL_COMPONENT. The BITS_PER_PIXEL for each individual SPECTRAL_COMPONENT are compacted into individual, two dimensional arrays, there being one such array for each SPECTRAL_COMPONENT.

**SCAN_LINE**

A logical collection of PIXELs organized in a linear fashion which represents an array row in the two dimensional display grid.

**PIXELS_PER_LINE**

The total number of PIXELS per SCAN_LINE. Note that this is not necessarily the number of bits per SCAN_LINE.

**NUMBER_OF_LINES**

The total number of SCAN_LINEs in the image.

**PIXEL_STRIDE**

The difference in bit addresses between the first bit of successive PIXELS. PIXEL_STRIDE will typically be equal to the number of bits per pixel stored in a particular IMAGE_PLANE_GROUP. IF PIXEL alignment requires fill bits between PIXELs, the difference between PIXEL_STRIDE and the number of BITS_PER_PIXEL per IMAGE_PLANE_GROUP will equal the fill value.

**SCANLINE_STRIDE**

The difference in bit addresses between the first bit of successive SCAN_LINEs. SCANLINE_STRIDE will typically be equal to the number of bits (not PIXELs) per SCAN LINE stored in a particular IMAGE_ PLANE_ GROUP. IF SCAN_LINE alignment requires fill bits between SCAN_LINEs, the difference between SCANLINE_STRIDE and the number of bits per SCAN_LINE will equal the fill value.

**SPECTRAL_COMPONENT**

The INTENSITY dependence of an image at either a particular wavelength or averaged over a wavelength band. A MULTI_SPECTRAL image is made up of two or more SPECTRAL_COMPONENTS.

**SPECTRAL_COMPONENT_MAPPING**

The correlation between the physical image data and the various SPECTRAL_COMPONENTS of an image.

**COMPONENT_SPACE_ORGANIZATION**

**DATA_PLANES_PER_PIXEL**

**DATA_PLANE_SIGNIFICANCE**

**NUMBER_OF_COMPONENTS**

The number of SPECTRAL_COMPONENTs in a MULTI_SPECTRAL image.

**BITS_PER_COMPONENT**

The number of bits used for each image component in a data plane. The sum of the BITS_PER_ COMPONENT over all components equals the BITS_PER_ PIXEL value. In that some three color image data representations do not have a one to one correspondence between representation component and SPECTRAL_COMPONENT, BITS_PER_COMPONENT should not be interperted as bits per spec− tral component. In addition, the BITS_PER_COMPONENT maybe be three different numbers (one per component).

### 5.2.6 Image Data Encoding

**ENCODING**

The process by which data is mapped into a new SYMBOL_SET for some specific purpose such as COMPRESSION.

**COMPRESSION**

The process by which an image is mapped from its PCM representation (uncompressed SYMBOL_ SET) into a new representation (compressed SYMBOL_ SET) for purposes of bandwidth reduction. The application of COMPRESSION techniques is a function of the IMAGE_DATA_ORGANIZATION, and is often done on a SPECTRAL_COMPONENT by SPECTRAL_COMPONENT basis.

**COMPRESSION_SCHEME**

The scheme used to accomplish COMPRESSION. Well known COMPRESSION_SCHEMES include run length compression, linear predictor compressors, and transform schemes.

**COMPRESSION_PARAMETERS**

The parameters required by a particular COMPRESSION_SCHEME.

**DECOMPRESSION**

The process by which an image is returned from its compressed representation to its PCM representation.

**DECOMPRESSION_SCHEME**

The scheme used to accomplish DECOMPRESSION. The DECOMPRESSION_SCHEME is chosen as the complement of the COMPRESSION_SCHEME.

**DECOMPRESSION_PARAMETERS**

The parameters required by a particular DECOMPRESSION_SCHEME.

**ENCODING_FIDELITY**

The degree of similarity between an original uncompressed image, and one which has been compressed, then decompressed. COMPRESSION_SCHEME DECOMPRESSION_SCHEME pairs which result in absolute ENCODING_FIDELITY are known as lossless, while those which have reduced FIDELITY are known as lossey. ENCODING_FIDELITY is measured with a mathematical metric.

**SYMBOL_SET**

The atomic elements which from which an image is composed. There is no uniques SYMBOL_SET for an image. Usually, the definition of the SYMBOL_SET is based upon the compression scheme. Typical symbol primitives include bits, bytes, and the entire image as a state space. The number of unique symbols which these primitives can represent (i.e. the SYMBOL_SET), and the number of symbols used to compose an image both depend upon the choice of symbol primitives.

### 5.2.7 Image Data Presentation

**RENDITION**

The process by which an image object is made suitable for display. It involves converting an image into a format that the display hardware recognizes.

**RENDITION_FIDELITY**

The degree to which the display subsystem can render an image belonging to a particular IMAGE_CLASS. RENDITION_FIDELITY is a function of the similarity between display hardware (display memory organization), and IMAGE_CLASS.

**DISPLAY**

The physical hardware (video logic and tube) which allows a user to view an image. The process of taking an image which has been rendered displayable and putting it on the screen (making it viewable).

**PRESENTATION_ATTRIBUTES**

The attributes of an IMAGE_OBJECT that define how it should be displayed for viewing. These attributes determine the particulars of the RENDITION process. Note that unlike DATA_ORGANIZATION_ATTRIBUTES, PRESENTATION_ATTRIBUTES are not essential for interperting the IMAGE_DATA, but are essential for displaying it.

**PIXEL_PATH**

The direction of the PIXEL capture path along an individual SCAN LINE. PIXEL_PATH is defined against the ISO and CCITT standards, and establishes the angle of a SCAN_LINE with relationship to a reference coordinate system.

**LINE_PROGRESSION**

The direction of SCAN_LINE capture across the image plane. LINE_PROGRESSION is defined against the ISO and CCITT standards, and establishes the angle of a normal to a SCAN_LINE with relationship to a reference coordinate system.

**PIXEL_ASPECT_RATIO**

The ratio of the distance between PIXEL centers along the PIXEL_PATH, and along the LINE_PROGRESSION on the SAMPLING_GRID and/or DISPLAY_GRID.

**BRIGHTNESS_POLARITY**

The direction of the mapping from PIXEL magnitude to visual brightness. (i.e., whether a PIXEL value of 0 is displayed as light or dark)

**GRID_TYPE**

The physical format of of the regular PIXEL sampling grid.

**TIMING_UNIT**

The number of atomic units of time between display of IMAGE_OBJECTS in a time sequence.

### 5.2.8 Image Server Hardware

**SERVER_HARDWARE**

The equipment which constitutes the computer system on which the XIEP server software executes including server HOST_PROCESSOR, HOST_MEMORY, HOST_BUS, and DISPLAY_SUBSYSTEM.

**DISPLAY_SUBSYSTEM**

The collection of DISPLAY_MEMORY, DISPLAY_PROCESSOR(s), and video logic used to drive a DISPLAY_MONITOR. There is an explicit communications connection between the DISPLAY_ SUBSYSTEM, and the HOST_PROCESSOR through serial or parrallel lines, or a backplane or bus.

**DISPLAY_MEMORY**

The memory in which the states of the screen pixels are kept and from which these screen pixels are refreshed. DISPLAY_MEMORY is often referred to as the bit map.

**DISPLAY_PROCESSOR**

Any general or special purpose processor dedicated to display processing and usually closely coupled to the DISPLAY_MEMORY.

**DISPLAY_MONITOR**

The tube or bottle (screen) upon which a viewer sees an image.

**DISPLAY_BUS**

The interconnect between DISPLAY_PROCESSOR(s), and DISPLAY_MEMORY. (often, this is a private interconnect).

**LOOK_UP_TABLE**

A table mechanism by which an input index (stored as a value in the DISPLAY_MEMORY) is used to find the intensity to be output on the various channels (guns) of the DISPLAY_MONITOR.

**HOST_PROCESSOR**

The cpu on which the X server executes. The X server may access other general or special purpose processors which are closely associated with the display subsystem in order to accomplish its job.

**HOST_MEMORY**

The memory used by the HOST_PROCESSOR to run the X server. This memory may or may not be directly accessible by the display subsystem and is usually not the same as the DISPLAY_MEMORY.

**HOST_BUS**

The interconnect between HOST_PROCESSOR(s) and HOST_MEMORY. (often, this is an open interconnect).

**FRAME_GRABBER**

A device which samples and digitizes an image or an analog representation of an image. Typically, the term FRAME implies a broadcast television video standard concept (RS170 – 30 frames per second refresh). As such, FRAME_GRABBERs are typically devices which digitize RS170 standard analog baseband video.

**FRAME_BUFFER**

The memory into which a FRAME_GRABBER deposits the image signal it digitizes. (often, this is the DISPLAY_MEMORY).

**TIGHTLY_COUPLED**

Used in this document to refer to the connection between image capture devices such as frame grabbers and scanners, and the DISPLAY_MEMORY. A capture device which deposits an image directly into the DISPLAY_MEMORY without passing through applications buffers (HOST_MEMORY) will be referred to as TIGHTLY_COUPLED.

### 5.2.9 Image Data Layout

**TILING**

A screen layout where adjacent rectangular images adjoin each other seamlessly and with no overlap. The adjacent images represent parts of a larger common scene.

**N_UP**

A screen layout where two or more different images are viewed adjacent to each other at the same time.

**OVERLAY**

A screen layout where two or more images are layed out on top of each other in SPATIAL_ REGISTRATION to observe differences. The images usually represent the same scene at different times or states.

**SPATIAL_REGISTRATION**

The state in which all common points in two images of the same scene would be found at the same PIXELs if displayed in an OVERLAY fashion.

**TIME_SEQUENCE**

The display of individual images of the same sceen taken at uniformly spaced time intervals.

### 5.2.10 Three Space Color

**SPECTRAL_SPACE**

A metrified way by which to describe locations in the space of human color perception and relate these back to physical principles (wavelength and intensity).

**RED_GREEN_BLUE or RGB**

A color three space widely used in the computer industry. The information in each of the SPECTRAL_ COMPONENTS (red green blue) represents the luminance integrated over some wavelength band centered around that color.

**CMY**

TBS

**YUV**

TBS

**HSV**

TBS

**HLS**

TBS

**YIQ**

TBS

### 5.2.11 Image Data Processing

**IMAGE_PROCESSING**

The process of applying an IMAGE_OPERATOR to an IMAGE_DATA operand or operands.

**TRANSFORM**

To modify IMAGE_DATA via the application of an IMAGE_OPERATOR.

**RESTORATION_PROCESSING**

IMAGE_PROCESSING whose purpose is to remove well understood distorations generated by the imaging system or conditions.

**ENHANCEMENT_PROCESSING**

IMAGE_PROCESSING whose purpose is to increase the subjective quality of an image thus making it easier to view.

**ANALYSIS_PROCESSING**

IMAGE_PROCESSING whose purpose is interpret the semantics or content and meaning of an image.

**CODING_PROCESSING**

IMAGE_PROCESSING whose purpose is to remap images between alternative representations primarily for purposes of compression.

**ITERATIVE_PROCESSING**

IMAGE_PROCESSING where the output of an IMAGE_PROCESSING operation is fed back into the same operator (i.e. the same operator is applied over and over again to the image).

**PIPELINE_PROCESSING**

IMAGE_PROCESSING where the output of an IMAGE_PROCESSING operation is fed into another operator (i.e. the image is fed through a serial stream of operators).

# Appendix A

# RELATED TOPICS

This appendix describes some background material on topics which are not central to the image extension package but are closely associated.

## A.1 Image Data Interchange

Image data formats are a topic of much interest lately. These formats are instances of image data interchange protocols. In the X environment, there are three locations where such protocols exist: application client to application client direct (IP1), application client to/from server (IP2), and application client to application client by way of a shared server (IP3) (see fig. A–1).

The interchange of image data between the application client and server remains within the application, yet spans the X wire boundary. Protocols for the interchange of image data across the X wire (IP2) are considered to be part of the X extension package (in fact, IP2 is XIEP the X image extension protocol) and must comply with the X core protocol. Of necessity, the extension proposal also addresses IP3 (sharing of server resources).

**Figure A–1: IMAGE DATA INTERCHANGE**

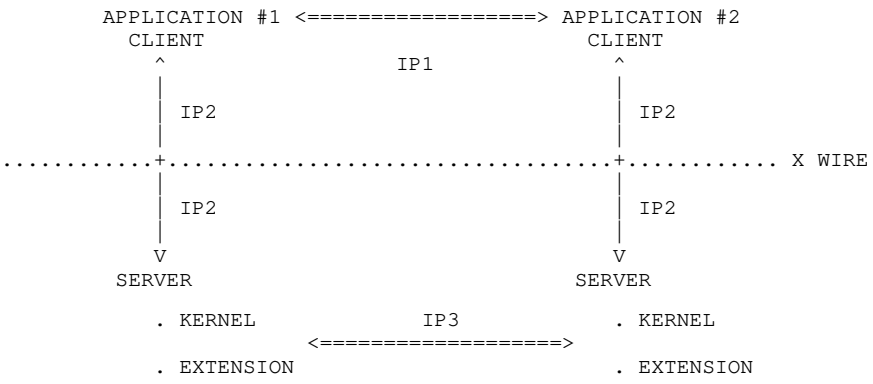



**Figure A–1 Cont'd. on next page**

**Figure A–1 (Cont.): IMAGE DATA INTERCHANGE**

It is the purpose, and nature of the IP1 image data protocol, and its relationship to IP2 and IP3 that are the subject of this section.

Application level image data interchange formats are the means by which various applications recognize and exchange common image data. Applications level interchange of data often has nothing to do with display processes, and, therefore, they (IP1) exist outside of the domain of X.

Such IP1 interchange protocols are likely to be standardized within image data classes and applications domains. This is seen in the emergence of image application domain industry standards. Thus, we are seeing medical radiological image data interchange standardized and engineering drawing image data interchange standardized in different ways. Often, applications level interchange protocols are packaged in or as disk file formats although they are equally applicable to all channels (both storage and communications).

Protocols for application to application interchange of image data are NOT part of the X Image Extension Package or the X Image Extension Protocol. However, in that the IP1 and IP2+IP3 paths should allow for the same interchange, it does make sense to propose an IP1 which is aligned with IP2 and IP3. As such, the extension protocol specification will include a specification for IP1 which can be implemented as a communications stream, or packaged in a disk file. It is expected that this specification will be aligned at least in content with the emerging compound document architecture interchange format (ISO 8613–7 ODA/ODIF).