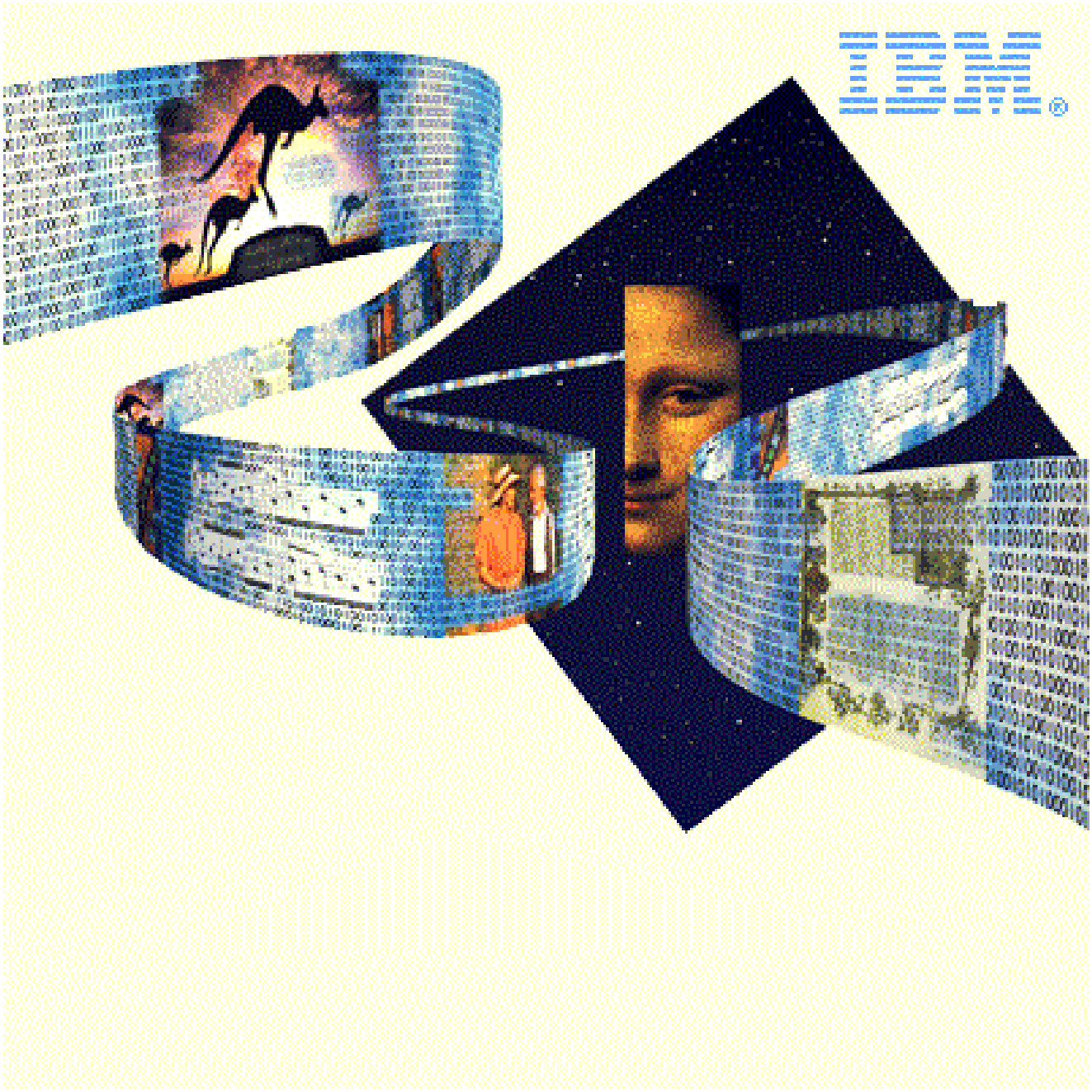

IBM Digital Library

Collection Treasury

Implementation Guide

First Edition (June 1997)

This edition applies to Version 1 of Digital Library Collection Treasury, 5799-B66 and to any subsequent releases until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

© Copyright International Business Machines Corporation 1997. All rights reserved.
Note to U.S. Government users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

The IBM Digital Library Collection Treasury Loader uses Microline View Toolkit 3.0™ from Microline Software, 1095 E. Duane Avenue, Suite 207, Sunnyvale CA 94086 and, "This application is powered by JScape ComponentWare© 1995, 1996, 1997 JScape Corporation, Scottsdale, Arizona. All rights reserved."

Trademarks and Service Marks

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

OS/2
IBM
AIX

The following terms, denoted by a double asterisk (**) when first used in this guide, are registered trademarks of other companies as follows:

Java	Sun Microsystems, Inc.
PostScript	Aldus Corporation
Netscape	Netscape Communications Corporation

Table of Contents

1.0 About This Guide 1
1.1 Purpose 1
1.2 Definition of The IBM Digital Library Collection Treasury 1
1.3 Audience 1
1.4 Conventions 1
2.0 Planning and Preparation 3
2.1 Organizational Requirements 3
2.2 System Prerequisites and Recommendations for Configuration 4
2.3 Defining The Data Model 5
2.4 Understanding the Image Preparation Process (IMGPREP) 10
2.5 Understanding Image Enhancement Tools (COMIT95) 12
2.6 Workflow Activities for DLCT 16
3.0 Installing Collection Treasury 17
3.1 Installing IMGPREP 17
3.2 Installing the Loader 19
3.3 Loading the Sample Data 21
3.4 Installing the End-User Interface 24
4.0 Producing Derivatives With COMIT95 26
4.1 Producing Reduced Images with REDUCE.EXE (Stand-alone) 26
4.2 Producing Sharpened Images with SHARPEN.EXE (stand-alone) 27
4.3 Producing Reduced and Sharpened Images with REDUCES.EXE (stand-alone) 28
4.4 Producing a Rotated Image with ROTATE.EXE (stand-alone) 29
4.5 Producing Image Color Transformation with CSTRANS.EXE (stand-alone) 30
4.6 Producing Image Watermarking with WMARK.EXE (stand-alone) 32
4.7 Image (JPEG) Compression with COMPRESS.EXE (stand-alone) 34
4.8 Image (GIF) Compression with GIFCOMPR.EXE (stand-alone) 35
4.9 Image Compression (TIFF with lossless JPEG) 36
5.0 Loading Data 38
5.1 Overview: 38
5.2 Parts of the Loader. 42
5.3 Data Model Editor Functions 43
5.4 Using the Loader. 44
5.5 Load Time Configuration 45
5.6 Using the Loader 46
6.0 The DLCT Sample Application 59
7.0 Customizing the Web Pages 68
Appendix A: Where to Find More Information 71
Appendix B: Loader Sample Input 72
B.1 Definition of IMAGES.LST file 72
B.2 IMAGES.LST File for Sample Data 73
B.3 Annotation (Tab Delimited) File for Sample Data 74
Appendix C: Sample COMIT95.INI File 75
C.1 Example of COMIT95.INI file: 75
C.2 Description of Sample COMIT95.INI 76
C.3 Valid Descriptors for COMIT95.INI 77

Appendix D: Image Preparation Command File 84
D.1 Example File IMGPREP.CMD 84
D.2 Customizing IMGPREP.CMD 86
Appendix E: Collection Treasury Application Load Time Configuration Planning Worksheets 88
E.1: Planning worksheets for the sample data using SampleArchive index class 88
E.2: Blank Worksheets 95
Appendix F: Error Codes 101
Appendix G: Microline Toolkit License Agreement and Warranty 104

Table of Figures

Figure 1: Anatomy of an Item in DLCT 9
Figure 2: Image Preparation Process Flow 11
Figure 3: Activities for a Typical DLCT Solution 16
Figure 4: The MIME Header 24
Figure 5: Step One in Process of Loading Data 40
Figure 6: Step Two in Process of Loading Data 41
Figure 7: The Loader 51
Figure 8: The Data Model Editor with the 'General' property sheet 52
Figure 9: The 'Attribute' property sheet 53
Figure 10: The 'Content Parts' property sheet 54
Figure 11: The 'FT Setup' property sheet 55
Figure 12: The 'Mapper Setup' property sheet 56
Figure 13: The 'Mapping' property sheet, showing partial mapping for an annotation file 57
Figure 14: The 'Mapping' property sheet showing partial mapping from the images.lst file 58
Figure 15: Home Page for Sample Application 59
Figure 16: Search Results Page 60
Figure 17: Advanced Search Page 61
Figure 18 Sample Query 63
Figure 19: Results Page From a Search Using The Wild Card Character 64
Figure 20: Screen Size Image Resulting From a Thumbnail Selection 66
Figure 21: Zoom-in Image Derived From a Screen Size Image 67
Figure 22: Building the Template Title 69

© IBM Corporation

1.0 About This Guide

1.1 Purpose

The purpose of this guide is to provide information about how to install and implement the IBM Digital Library Collection Treasury (DLCT) and the planning steps that must be taken in advance.

1.2 Definition of The IBM Digital Library Collection Treasury

DLCT is a solution that allows you to load digital images into IBM Digital Library and to associate indexing information to the images. Using a standard web browser, a user can then access those images by performing a parametric search, or a free text search or a combination of the two.

1.3 Audience

This guide is intended for those who have particular responsibilities for the installation and implementation of Digital Library Collection Treasury (DLCT).

1.4 Conventions

In this guide, different text font conventions are used to display certain types of information. Any other font changes are for emphasis only. These conventions are as follows:

- Program names and filenames, unless they are part of a command or other code structure, are shown in a sans serif, non-bold type and in all capital letters unless otherwise shown.

  Example: REDUCE.EXE

- Menu selections, return/error codes, command codes, and other items of coded output and user input are shown in a bold sans serif typeface and in lower or mixed case.

  Example: **input: 24 bit color TIFF file xxxxxxxxxxxxxxxxx**

- Directory names are in normal serif typeface, lower case.
- Reference titles, such as those of books, are shown italicized in regular serif type.

  Example: *IBM Digital Library Programming Guide and Reference*

- Information preceded by the word "Note:" are in the same font as ordinary information but is in bold type.

  Example: **Note: This command must be entered before……**

2.0 Planning and Preparation

2.1 Organizational Requirements

- Familiarity with IBM Digital Library concepts on the part of everyone involved is prerequisite for the installation and implementation of DLCT.

- Three groups of skills at separate functional levels are required to install and implement DLCT:

  – Archive Manager (Domain Expert). This person is responsible for overall management of the image archive and the production of digital images and their derivatives. The Archive Manager is also primarily responsible for providing or defining attributes, which are descriptive information about particular images to be stored in DLCT, and for other key decisions related to the image archive.

  – System Administrator - This person is responsible for hardware and software installation, planning and monitoring usage of the system, and day to day operations. The person must have had experience in installing IBM Digital Library (DL), Version 1.0, including Text Search Manager and the DL Internet Connection Sample Application.

  – Programmer: Some components of this solution can be modified to suit customer requirements. To make those changes, programming skills in CGI and Perl are needed. Skill with HTML is also required for customizing the web pages that are used to access images.

2.2 System Prerequisites and Recommendations for Configuration

Hardware Prerequisites:

Collection Treasury does not impose any specific hardware other than what is required by IBM Digital Library itself.

Software Prerequisites

Software Component	Description/Comment
IBM Digital Library*, version 1.0, with the following components: 1. **Library Server** 2. **Object Server** 3. **Text Search Manager** 4. **Digital Library Internet Connection (DLIC)** 5. **Client Toolkit**	The software must have the current level of CSD applied to it. The CSD is available at the Digital Library FTP site: ftp://ftp.software.ibm.com/ps/products/digital_library/fixes/v1.0/enu
AIX* version 4.1.5 or version 4.2	The Text Search Manager, DLIC and the Client Toolkit must be on AIX.
Web Server (HTTPD)	Any NCSA compliant Web Server, including Netscape** Servers. The IBM Internet Connection* web server is available at http://www.ics.raleigh.ibm.com/
Perl	Version 5.02M version of Perl is available at the AIX Public Domain software site at http://aixpdslib.seas.ucla.edu/languages/perl4.html
JAVA 1.1.1 for AIX**	Available from the IBM Java Center at http://ncc.hursley.ibm.com/javainfo/download/index.html

Minimum System Configuration:

The following components require the AIX operating system:

1. Text Search Manager.
2. Digital Library Internet Connection (DLIC).
3. Collection Treasury Loader.

All of the above components can be combined on a single workstation, or can be on separate workstations.

[Note that a Library Server and Object Server are also required, but they can reside on any supported IBM Digital Library platform]

2.3 Defining The Data Model

A typical image archive may consist of multiple, various-sized digital images of an artifact, along with some text that describes the artifact. The reason for having multiple images is that different users may have widely diverse needs and uses.

Therefore, as a customer, your unique needs must be taken into account when you are planning and defining a data model that will fit your particular situation. DLCT provides an example of a data model that you can use to understand generally how a data model is constructed. You should first understand the sample data model thoroughly and then you can customize it to fit your needs.

The sample image archive shipped with DLCT is explained in this document. For learning purposes, we are referring to this sample archive as the "Sample Archive." The sample archive may represent images of artwork, or a collection of photo negatives or slides from special libraries, or any other group of objects that can be represented by images.

There are three major questions to be considered in defining your data model:

- How many derivative images are to be stored?
- How are the images going to be identified for quick and efficient search?
- By what process will end users access those images?

Derivatives

There are valid justifications for having multiple derivative images of an artifact. An "archival quality" image is the image directly produced by the scanning process. This image is typically the highest possible quality and usually of maximum resolution. Most users who would access IBM Digital Library over a network, however, would not want to access this image since it requires high bandwidth and a client computer with large amounts of memory. For such users, DLCT creates from this primary image various types of secondary images called "derivatives." In Sample Archive, the "archival quality" image is a TIFF file, which retains the highest possible resolution. This type of file, however, is necessarily of significant size, so DLCT contains tools that can create a derivative called a "full image."

A full image is a digitally compressed version of the archival image and is the largest and best quality of all the derivatives. It is produced using a "lossy JPEG" compression process. For browsing and quick views of images, DLCT creates a screen size representation of the full image, called a "screen image" and a derivative called a "thumbnail" image. Another derivative, called a "zoom image," is a 2X magnification of the screen image. The screen image and zoom image both have watermarks applied to them. The archival image is not stored in the IBM Digital Library. The thumbnail, screen size, full, and zoom images are stored as parts of an item in IBM Digital Library.

Indexing

To make possible the retrieval of stored images from the IBM Digital Library, some kind of indexing information must be attached to the images by which they can be identified. These items of information are called "attributes. "Index Class" is an IBM Digital Library concept that provides for a list of attributes that uniquely describe an image. A user can specify a query against the index class to access the particular image being sought. This is called a "parametric search."

The Sample Archive has two timestamp attributes, "CreationTimeStamp" and "LastUpdateTimeStamp," which have special significance. The loader checks for the existence of these attributes. When an item is created, both the CreationTimeStamp and the LastUpdateTimeStamp are generated. When an item is updated, the Last Update TimeStamp for that item is modified. When defining an index class, the System Administrator may choose to use both, either, or none of these timestamp attributes. They are provided as a convenience. The timestamp format is:

YYYY-MM-DD-HH.MM.SS.XXXXXX (where XXXXXX is a random number).

To tailor this solution to your own needs, you will have to create an index class with a different set of attributes. One of these attributes must be designated to contain a unique value for an item. In the Sample Archive, the designated attribute is called "UniqueID."

The value in this field uniquely identifies an item in IBM Digital Library and is used during the loading process. There are additional fields, such as "Artist" and "Title," for example, that are used to associate relevant textual information with an image.

DLCT uses an index class called "SampleArchive." The contents of this index class are shown in Table 3.

Attributes	Data Type	Required?	Sample Values
Unique ID	Ext. Alpha 20	Y	123456789
Category	Ext. Alpha 50	N	Impressionist
Artist	Ext. Alpha 30	N	Thomas
Title	Ext. Alpha 200	N	Snowman
Provenance	Ext. Alpha 200	N	California
Description	Ext. Alpha 200	N	
CreationTimeStamp	TimeStamp	N	1997-06-23-14-45.32.075000
Last UpdateTimestamp	TimeStamp	N	1997-06-23-16.26.20.266000
Screen_ Image_Size	Ext. Alpha 20	N	1200
Screen_Image_Width	Ext Alpha 10	N	500
Screen_Image_Height	Ext Alpha 10	N	400
Zoom_ Image_Size	Ext Alpha 20	N	1800
Zoom_ Image_Width	Ext Alpha 10	N	800
Zoom_ Image_Height	Ext Alpha 10	N	1000
Thumbnail_Image_Size	Ext Alpha 20	N	10
Thumbnail_Image_Width	Ext Alpha 10	N	10
Thumbnail_Image_Height	Ext Alpha 10	N	20

Table 3: Sample Archive Index Class Contents

Each item in the Sample Archive belongs to the Sample Archive index class. The attribute value pairs, which are stored in the index class, are also indexed to Text Search Manager. This allows them to be searched through Search Manager.

End User Interface

Items stored in the Sample Archive can be accessed by internal users through a client program, or by external users on the web. The web setup uses IBM Digital Library Internet Connection (DLIC) to present an HTML form to the user for querying the

collection. Input to the HTML form is used to search IBM Digital Library. The input is processed within the End User Interface Services (EUIS) component of DLIC where it is constructed into a query.

The query can result in a parametric search, a free text search, or a combination of the two, depending upon which form elements are set. A parametric search scans the specified index class(es) for any attributes and values that match the search criteria as specified in the attribute operator list boxes and text areas on the advanced search page. A free text search scans the search manager indexes for either the presence or absence of particular word(s). A free text search is triggered by entries in the text areas defined on the home page or on the top half of the advanced search page.

The results of a query are displayed on a "gallery" page of thumbnail images. The user can then select a thumbnail image or the descriptive hypertext below the thumbnail. Clicking on a thumbnail results in the display of the screen size derivative image. Clicking on the hypertext results in the display of the text associated with the item and is a picture of what values have been indexed to Text Search Manager. Each item has a "note" or "table of contents" part associated with it. This part contains an item's part IDs and pseudo filenames. These values are specified at load time and are used by the Page Builder component of DL Internet Connection to construct the web pages that display the results page to the user.

The content parts and pseudo filenames used for Sample Archive are shown in Table 4 and the note part is shown in Table 5.

Content Parts	Pseudo Filenames
Thumbnail JPG	THUMB.JPG
Full JPG	FULL.JPG
Zoom Size JPG	ZOOM.JPG
Screen Size JPG	SCREEN.JPG
Table of Contents	N/A
Search Manager Text	TEXT.TXT

Table 4: Sample Archive Content Parts and Pseudo Filenames

© IBM Corporation

Anatomy of an Item in the Collection Treasury Solution

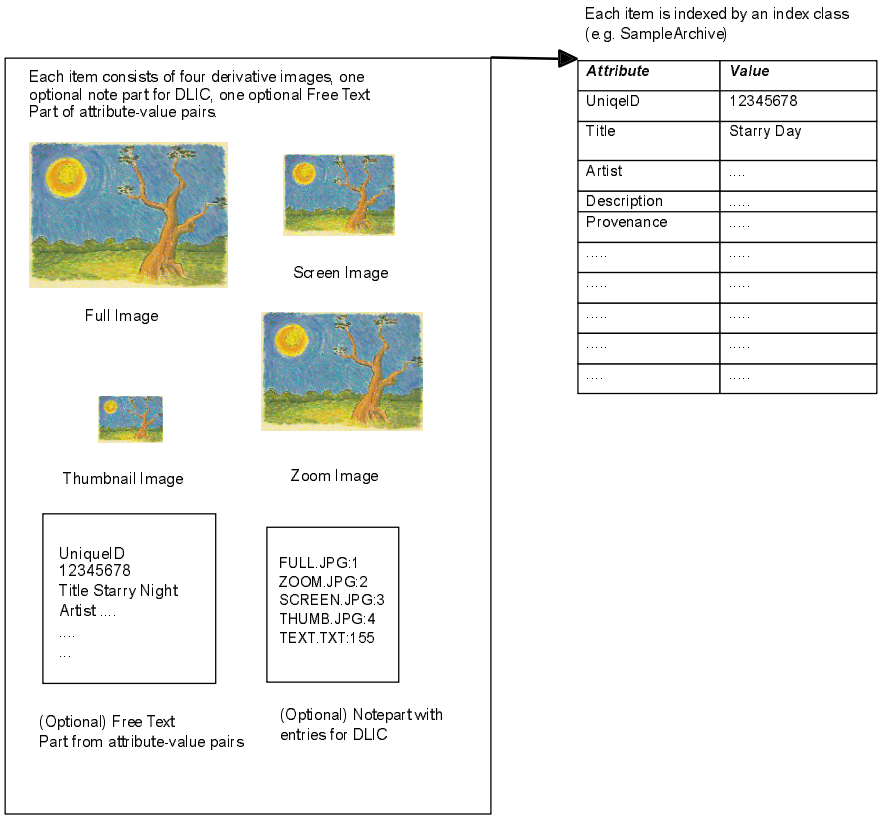

Figure 1: Anatomy of an Item in DLCT

Note Part
TEXT.TXT:155
FULL.JPG:1
ZOOM.JPG:2
SCREEN.JPG:3
THUMB.JPG:4

Table 5: Sample Archive Note Part

2.4 Understanding the Image Preparation Process (IMGPREP)

Once the images are scanned and associated TIFF files created, the 4 derivative images must be generated and loaded into the DLCT.

Image preparation processing is performed by the OS/2 REXX program, IMGPREP.CMD, for each original archival quality TIFF image stored in the scanned image subdirectory.

IMGPREP.CMD executes COMIT95.EXE which manipulates the input TIFF images and generates as output, the 4 derivative (full, zoom, screen and thumbnail) images needed for loading . COMIT95.EXE interprets information contained in the user-supplied COMIT95.INI file (see Appendix C). COMIT95 also uses information contained in the DISP.INI and DEVICE.INI files during color space transformation. These 2 files (DISP.INI and DEVICE.INI) should not be modified and should be used "as shipped."

After COMIT95 generates the derivative images, a rename function occurs and the derivative images are given new names reflecting the type of derivative.

IMGLSTY.EXE is invoked as the third and final step in the image preparation process and is referred to as the Image List Builder. IMGLSTY.EXE prepares a file IMAGES.LST, which is a list that includes for each scanned image, the names of the derivative images and their characteristics.

This process is then repeated until all original archival TIFF images are processed.

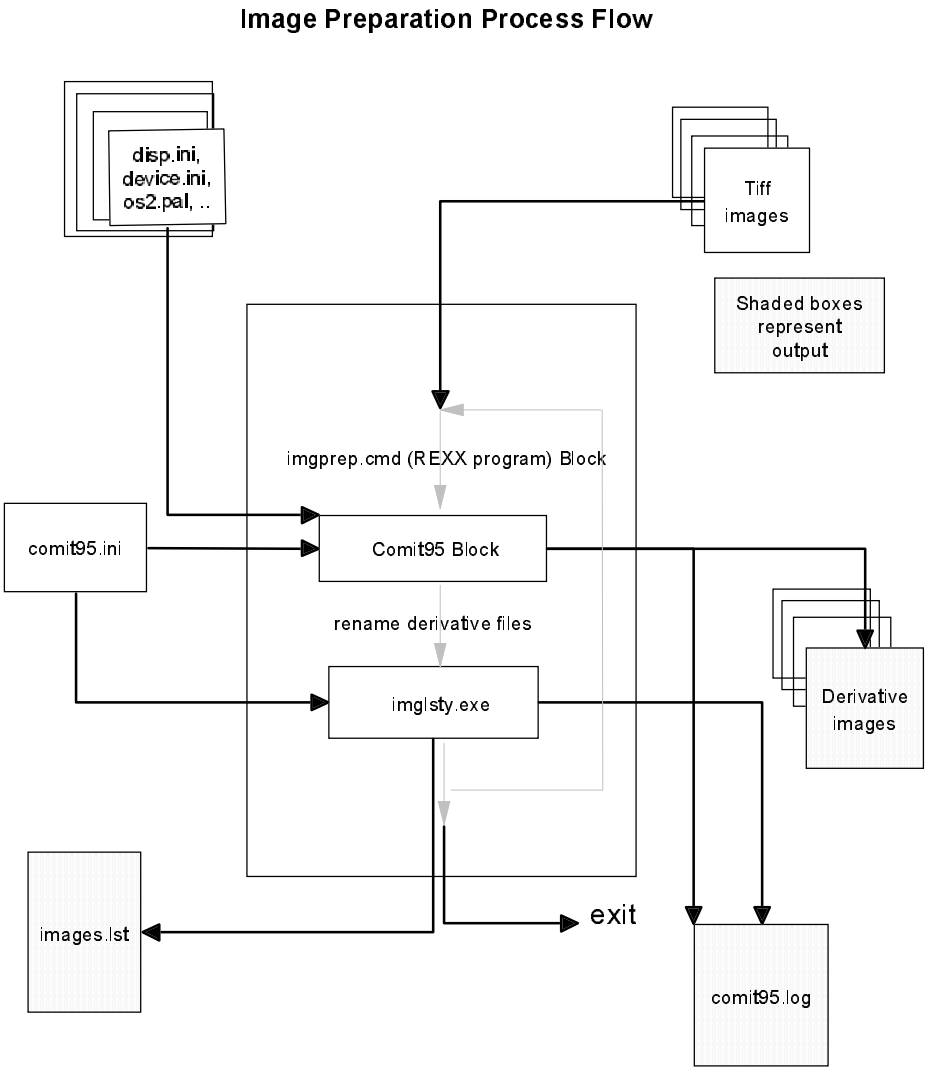

Figure 2: Image Preparation Process Flow

2.5 Understanding Image Enhancement Tools (COMIT95)

COMIT95.EXE is an image enhancing utility which is used to generate the derivative images. For DLCT, COMIT95.EXE produces the 4 derivatives from each scanned TIFF (XXXXXXXX.TIF) image. It is the batch-processing program that, for each TIFF image stored in the scanned image subdirectory

1) Reads the scanned (TIFF) image file

2) Generates the 4 derivative images

3) Stores the derivative images in an image subdirectory.

The derivative images produced by COMIT95 are specified by the parameters in the associated COMIT95.INI file. The four derivative images created for each scan (after rename):

- Full scan image, a scan resolution image with a maximum resolution of 3072 by 4000 pixels, compressed by lossly JPEG. (FXXXXXXX.JPG)
- Zoom image, a 2X reduced from the original image and compressed lossly JPEG, suggested maximum resolution of 1400 x 1400 pixels, with watermarking. (ZXXXXXXX.JPG)
- Screen image, a reduced and compressed lossy JPEG, suggested maximum resolution of 700 x 700 pixels, with watermarking. (SXXXXXXX.JPG)
- Thumbnail image, with a suggested maximum resolution of 200 by 200 pixels, compressed JPEG. (TXXXXXXX.JPG)

In naming the derivatives, the first character is:

- F for full image
- Z for zoom image
- S for screen image
- T for thumbnail image

The second through eighth characters will remain unchanged from those same characters in the scanned image (TIFF) from which the derivative image was computed. The file extension describes the format of the derivative image (e.g., JPG for JPEG.).

A sample COMIT95.INI file used to produce the four derivatives is provided in Appendix E. Based on particular needs, it may be and should be modified for your environment.

COMIT95.INI specifies one or more functions to be performed on the scanned TIFF image. Functions which COMIT95 perform include:

- Reducing the image to a desired size
- Sharpening the image
- Rotating the image clockwise via 0, 90, 180 or 270 degrees
- Transforming the image to the desired color space
- Watermarking the image
- Compressing the image using JPEG or GIF.

These functions should be performed in the order presented above. However, individual functions may be omitted from the sequence. Any image created during one sequence can be used as the input image for a subsequent sequence.

When COMIT95 operates on a scanned image (**file_name.tif**), the following intermediate files may be created based on the function(s) performed:

Reduction creates:	**file_name.rdN**
Sharpening creates:	**file_name.shN**
Rotation creates	**file_name.rtN**
Color Space Transformation creates:	**file_name.csN**
Watermarking creates:	**file_name.wmN**
Compress creates:	**file_name.jpN**
or:	**file_name.gfN**

Table 6: Intermediate Files Created by COMIT95

In the extensions of these filenames, "N" identifies the particular sequence of functions that produced the file ("1" for the first sequence; "2" for second; and so on), so that each sequence of functions produces a unique set of files.

As stated above, COMIT95.INI specifies one or more sequences of functions to be performed. For each sequence, COMIT95.INI contains information whether any step is to be performed or omitted. If a function is performed, COMIT95.INI also specifies whether the corresponding output image is to be saved , or if it is to be discarded after it has been used as input to the next step in the current sequence, or if it is to be saved temporarily to serve as input to a subsequent sequence. If the third option is specified, the image is saved but will be discarded after ***all*** operation sequences have been completed.

(See Appendix C for details.)

Example: *To reduce and save a TIFF image, and produce a compressed JPEG image from the reduced TIFF image,* one must provide the following basic information:

- The original TIFF as input,
- Where to save the reduced TIFF as output,
- The initial function to perform (reduce)
- The maximum width for reduction
- The maximum height for reduction
- The second function to perform (compress), and finally
- The format of the compressed file as output.

Converting this for use in COMIT95.INI parameters:

- **source_dir=d:\imagein**
- **source_ext=tif**
- **reduce_usage=perform&save,d:\imgout**
- **output_size=700,500**
- **sharpen_usage=omit**
- **rotate_usage=omit**
- **ctransform_usage=omit**
- **watermark_usage=omit**
- **compress_usage=perform&save,d:\imgout**
- **output_format=jpeg**

The **source_dir** and **source_ext** are used to specify the drive/directory and extension for the image that is to be used as input to this sequence of operations. The filename will be taken from the input file specified on the command line for COMIT95. (Note: since IMGPREP.CMD calls COMIT95.EXE, IMGPREP.CMD must be modified to ensure that the correct drive/directory is pointed to for the original TIFF images.)

The **function_usage** parameter is used to specify whether the corresponding function is to be performed (***perform***) or omitted (***omit***). If the function is to be performed, it must then be specified whether the output is to be saved (***save***), temporarily saved (***tsave***), or discarded (***discard***). If the output is to be saved, the correct ***drive/directory*** must also be specified.

The **output_size** specifies the ***width*** and ***height*** of the reduced image in pixels. For example, **output_size=700,500** specifies that a reduction factor is chosen so that the width of the output image will be *at most* 700 pixels and the height of the output image will be *at most* 500 pixels. The chosen reduction will be proportional to the original image.

The **output_format=jpeg** specifies the format of the output from the image compression process. In this case, the output is JPEG.

A detailed layout and explanation of COMIT95.INI, its parameters and values can be found in Appendix C.

COMIT95 normally displays only the version and copyright information for the modules which are called and any error messages generated. However, the OS/2 REXX image preparation program (IMGPREP.CMD) invokes COMIT95.EXE with the "/n" flag. This "/n" flag provides additional information including input file size, information on which of the intermediate files are to be kept, the exact commands used to run the functions, etc.

If COMIT95 or any of its submodules is called with either no arguments or the wrong number of arguments, information describing the correct calling sequence is displayed. The operation then terminates with a return code 3. This may occur if the COMIT95.INI does not contain the necessary parameters for the given function.

To help users "fine tune" the images using COMIT95, each of the functions (reduce, sharpen, rotate, etc.) may be run as a stand-alone program operating on an input TIFF image file and producing an output TIFF image file. (There are 2 exceptions: 1) the reduction and sharpening of certain images may be combined into a single step and 2) the output of the compression routine is a JPEG, GIF or TIFFJPEG). For further details, see Chapter 4.0, Producing Derivatives With COMIT95.

2.6 Workflow Activities for DLCT

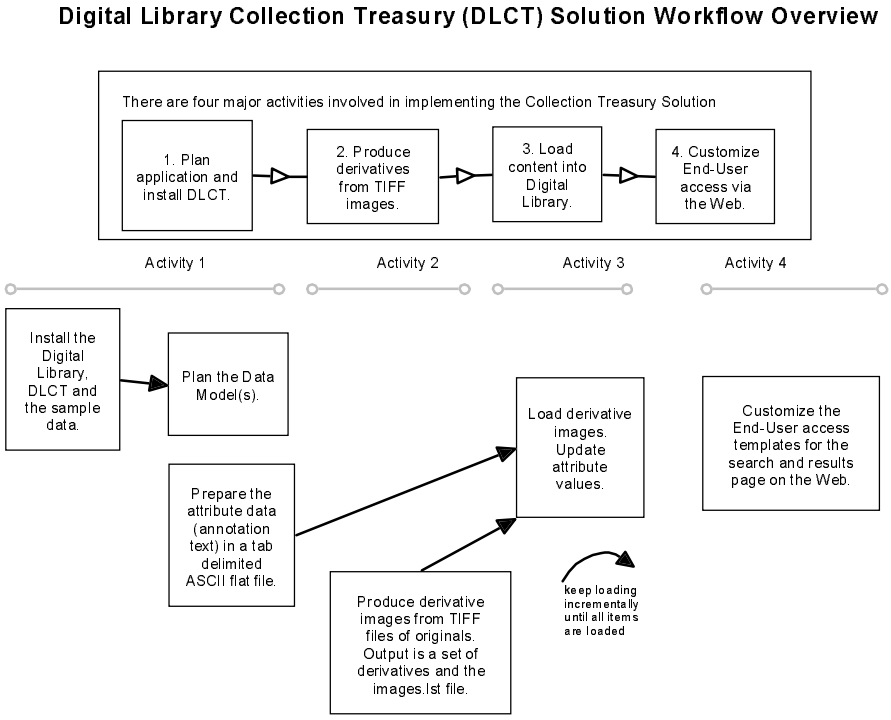

Figure 3: Activities for a Typical DLCT Solution

3.0 Installing Collection Treasury

3.1 Installing IMGPREP

IMGPREP is the OS/2 REXX program which is supplied with DLCT to produce the derivative images for loading into IBM Digital Library. IMGPREP consists of:

- IMGPREP.CMD - REXX file which calls COMIT95 to enhance the TIFF images and create the 4 derivative images.
- COMIT95
- IMGLSTY.EXE - Image List Builder - builds the image list and is called from IMGPREP.CMD.

To install IMGPREP,

1. Insert the CD-rom into the CD-rom drive on the OS/2 workstation where you plan to run the image preparation process.
2. Open an OS/2 window.
3. Ensure you are at the prompt for the CD-rom drive.
4. At the command prompt, type **Install** and press **Enter** to start the installation process. (Online help is available during the installation process by clicking on the **Help** button.)
5. After a few seconds the installation window appears.
6. Click on the **Continue** button.
7. The install program queries your system for previously installed versions of IMGPREP components.
8. After the install program checks for previous version, a product information window appears.
9. Click on **OK** to continue the installation. (Note that the install procedure updates your CONFIG.SYS file if the option is checked. If you did not check the **Update CONFIG.SYS** you will need to manually update your CONFIG.SYS file's **LIBPATH** and **SET PATH** to point to the **drive\directory** where IMGPREP is installed.
10. The **Install - directories** window describes your installation options. Currently, IMGPREP is the only component and you **must** select it. To

select IMGPREP component for installation, click on the **Select all** button or highlight IMGPREP.

11. To view a description of the component, click on **Descriptions**.
12. Indicate the *drive\directory* where you want to install the IMGPREP components. The first directory contains the executables, DLLs and any other files need by IMGPREP (You will execute IMGPREP from this *drive\directory*). The second directory contains the SAMPLE files which you may modify as part of your image preparation customization (e.g COMIT95.INI). **Note: to run IMGPREP correctly, you will need to copy these three customized files to the first drive/directory.**
13. Once you are back to the **Install - directories** window, click on **Install** to continue the installation.
14. An **Install - progress** window appears showing the progress of the installation.
15. When the **Installation and Maintenance** window appears, you have successfully installed the IMGPREP component.
16. Click on **OK** to end the installation process.
17. Finally, click on **Exit** to quit the installation program.

List of Executables, DLL's, and Header files: (*drive n*:\IMGPREP)

ANY2ANY.EXE
COMIT95.EXE
COMPRESS.EXE
COMPRTIF.EXE
CSTRANS.EXE
GIFCOMPR.EXE
IMGLSTY.EXE
REDUCE.EXE
REDUCES.EXE
ROTATE.EXE
SHARPEN.EXE
WMARK.EXE
OS21.PAL
DDE4MBS.DLL
JPGENDC.DLL
DEVICE.INI
DISP.INI
GENY.HDR
GENYUV.HDR

List of Customizable Files (*drive n*:\IMGPREP\SAMPLES)

COMIT95.INI
IMGPREP.CMD
LABELX

COMIT95.INI - contains one or more "option sets" that describe a single sequence of operations to be performed. The sample COMIT95.INI contains COMIT95 control statements for producing the 4 basic image derivatives used by DLCT. See Appendix C for details. After the COMIT95.INI has been updated to reflect your particular needs, save it in the IMGPREP main directory (e.g.: C:\IMGPREP).

IMGPREP.CMD - REXX utility command which process all TIFF files in a given directory path (**inpath**) and stores the derivative files in an output directory path (**imgpath**) in preparation for loading into the Digital Library Collection Treasury system.

Customization consists of modifying 1) **inpath = 'c:\imgprep\in'** statement to reflect the drive/directory of the input TIFF images and 2) **imgpath = 'c:\out'** statement to provide the output drive/directory for the image derivatives. After these changes are made, the file is placed in the main directory for IMGPREP (e.g.: C:\IMGPREP)

LABELX - contains a "label parameter" used by the image list building process. This label parameter can be 1-10 characters in length and may be used to describe where the TIFF data originated for the particular run of IMGPREP.CMD. (e.g.: *d* DRIVE).

3.2 Installing the Loader

The Loader is an AIX component. Use **smit** or **smitty** to install the loader package from the DLCT CD-ROM. If you are installing a new version of this product, then you must remove the old version first before installing the new one.

Prerequisites: The Loader requires the following components

1. frn.frntoolkit v1.0, The IBM Digital Library Folder Manager Client API
2. Java 1.1.1 for AIX

Before installing the Loader make sure that you have the toolkit and Java installed. Note that you MUST have Version 1.1.1 of Java, and must make sure that you can run the demonstration Java applications that come with the Java 1.1.1 package.

The Loader will be installed in the **/usr/lpp/dlct** directory. The directory structure is as follows:

/usr/lpp/dlct	The product directory
/usr/lpp/dlct/ias	The Loader component directory
/usr/lpp/dlct/ias bin	The executable components.
/usr/lpp/dlct/ias/view	The templates and other files needed for the End User Interface.
/usr/lpp/dlct/ias/misc	Miscellaneous utilities.
/usr/lpp/dlct/ias/images	Images used by the Loader application for its user interface.
/usr/lpp/dlct/ias/nativelib	Library files for use by the Loader application
/usr/lpp/dlct/ias/com	Java Class files for use by the Loader application.
/usr/lpp/dlct/ias/COM	Java Class files for use by the Loader application.
/usr/lpp/dlct/ias/mlsoft	Java Class files for use by the Loader application.

The loader runs as a client of the Digital Library, and hence its environment must be setup to be that of a client. If you installed the Sample Application that comes with Digital Library, then the environment from which you loaded the sample application is the client environment, and the user id running the Loader must also have this enviromnet. In addition, the following additional modifications are needed for the Loader.

Modify your environment based on the sample **setup.profile** file in the /usr/lpp/dlct/ias/misc directory. You must have the /usr/lpp/dlct/ias directory in your CLASSPATH environment variable and the /usr/lpp/dlct/ias/nativelib directory in your LD_LIBRARY_PATH environment variable, and the /usr/lpp/dlct/ias/bin directory in your PATH environment variable.

It is recommended that you copy these entries from the setup.profile file to your .PROFILE, modify them if needed. This way your environment will be automatically setup each time you log in to your account.

Once your environment is set, you can execute a simple test to see if you can log on to the Library Server.

Execute '**testlogon**' (which resides in /usr/lpp/dlct/ias/bin, and which in turn executes '**java com.ibm.dl.iaload.TestLogon**). This tests whether you can log on to the Library Server, and hence is a good test to check whether your environment is setup. This script requires three arguments, viz., the Library Server name, the userid and password. If the logon was successful, you will get a 'Logon successful' message. Otherwise, you will get a return code from Digital Library. You will have to look up the return code in the "Digital Library Messages and Codes" book to diagnose the problem.

If your logon to the Library Server was successful, then you can try launching the loader. The script '**loader**' in **/usr/lpp/dlct/ias/bin** will invoke the loader. This script, invokes the Java Virtual Machine™ on the com.ibm.dl.iaload.IASuper class .

The Loader should come up, and prompt you for a login to the Library Server. Enter the userid, password and the Library Server name and click the 'OK' button. The status bar on the loader should change from "Not Connected" to "Connected".

3.3 Loading the Sample Data

Before the Sample Data can be loaded, the Index Class **SampleArchive** must be created. The script 'createSampleArchive' in ***/usr/lpp/dlct/ias/bin*** can be used to create it. This script requires as input the Library Server name, userid, password and Object Server Name. The Sample Data reside in ***/samples*** on the DLCT CD-ROM. There are two categories of data: Aspiring Artists and Product Team in two separate directories under ***samples.*** You can copy the entire ***/samples*** directory to your Working directory or you can run the Loader from the CD-ROM. You can copy the samples by using the following command from your working directory:

cp -r /cdrom/samples .
where **/cdrom** must correspond to your cd-rom directory.

From the working directory or cd-rom, **cd** to the sample directory you want to load. For instance:

cd samples/aspiring

to load the Aspiring Artists data.

Invoke the Loader from this directory. A Login dialog box should appear. Enter your Library Server name, userid and password and click on the OK button.

Upon a successful login, the Status Bar will say '**Connected**!' and the Message Console will contain the following messages:

Library logon succeeded.
Got Index Class Names
Got Content Class Info

From the Menu Bar, select Actions. This brings up a Menu, select **Data Model Editor.**

From the Menu Bar, select File. This brings up a Menu, select **Load**. This brings up a Load Data Model dialog box.

You can modify the filter to be ***.ltc,** and click on Update. This should result in two files being displayed in the Files list. Select **as_creat.ltc**, and click on the OK button.
You should see a set of tabbed property sheets. These sheets describe the data model for creating the Aspiring Artists images.

The General sheet identifies which index class we will be using (**SampleArchive)**, the number of explicit Content Parts (**4),** and specifies that we will be using Search Manager and creating an implicit free text part, and that we will be creating an implicit note part of the form required by DLIC.

The Attributes sheet shows which attributes from the SampleArchive index class will be used in creating the implicit free text part.

The Content Parts sheet shows the explicit part IDs and what type of Content they are (**JPG**). Each part is a 'BASE' type. The TOC entries are the values which get loaded into the note part and must match the values expected by the DLIC templates.

The FT Setup sheet describes the Search Engine (**SM)** and its indexing information and what type of part the implicit text part will be (**ASCII)**.

The Mapper setup sheet specifies the input file of **aspiring.lst** which describes the derivative images in the Comit95 image list format.

The Mapping sheet graphically depicts which fields of a Comit95 image list record maps to what attribute or content part file name of an item.

From File on the Menu Bar, select Verify. There will likely be Warning Messages. Messages similar to the following should appear in the Message Console of the Loader Main Window:

Checking details of index class SampleArchive in loaded Data Model against those in the library server...
Warning:Changing Index Class ID from xx to yy
Warning:Changed Attriubte ID xx in Data Model at Index yy to zz from library server.
Checked Index Class Details
Success:Verification successful

From the Loader Main Window, click on the Add Item button. You should see several messages scrolling by on the Message Console. There are a total of 26 Aspiring Artists items and there should be a message associated with each one of them. If errors occur, items could be left in an invalid state. Click on the Delete Item button to remove them. (Doing a Delete Item when items do not exist is okay.) An example of a successful Add Item message follows:

```
10 : Success
Success:
Added Part with ID 1 class 207 type 128 to item R#T@7ADZEYGUZVEB
Added Part with ID 2 class 207 type 128 to item R#T@7ADZEYGUZVEB
Added Part with ID 3 class 207 type 128 to item R#T@7ADZEYGUZVEB
Added Part with ID 4 class 207 type 128 to item R#T@7ADZEYGUZVEB
Added Item R#T@7ADZEYGUZVEB with Primary Key sl000060
Added Note Part
Added Attributes Free Text Part with PartID 155
Added SM Index for part id = 155 in item id = R#T@7ADZEYGUZVEB
```

The above step has loaded the items with a few of their attributes and all of their content parts. The next step is to update the rest of the attributes, using a tab delimited text file (for Aspiring Artists this would be "aspiring.txt") and to update the implicit Search Manager part with all of the attriubtes.

From File on the Data Model Editor Menu Bar, select Load. Go through the same process as before except this time select **as_updat.ltc** from the Files list and click the OK button**.** From the File Menu Bar, select Verify. If the verification was successful (there will be some Warning messages this time but it is okay), go back to the Loader Main Window and click on the Update Item button.

The last step is to go to the System Administration machine and do the Text Search Indexing. Assuming the document queue was empty before starting the Aspiring Artists load, it should contain 52 entries. Indexing should cause the secondary index queue to be incremented by 26.

To load the other set of data, the same steps would be followed from the prodteam directory.

Error codes of 10000 and above are from this loader and are defined in the Appendix F. Other error codes are from the Digital Library and are defined in ***IBM Digital Library Messages and Codes, Version 1.***

3.4 Installing the End-User Interface

Upon installation of the DLCT package on AIX from the CD-ROM, the End-User Interface code will reside in **/usr/lpp/dlct/ias/view**. It must be copied into

/usr/lpp/frn/samples/cliette. Before doing this, make backup copies of the following files:

FRNCLIET.INI
FRNCLIET.CPP
FRNCLIET
FRNEUIS.TBL

The file FRNCLIET.INI needs to be modified. It should already contain your hostname and be working for the DL Sample Application with its magazine and journal index classes.

Modify the THUMBNAIL=THUMB.GIF in the **[VisualInfo]** section to be THUMBNAIL=THUMB.JPG.

Update the Catalog section to look as follows:

```
[Catalog]
;Catalog1=DLSEARCH_JOURNALS_X DLSEARCH_Title
;Catalog2=DLSEARCH_MAGAZINES_X DLSEARCH_Title
Catalog3=SampleArchive Title
```

Update the **TEMPLATES** line in the **[EUIS_CONSTANTS]** section to include **jpg** as follows:

```
TEMPLATES=html:notemplate::htm:notemplate::text:document::document:
document::videostream:notemplate::videostream2:notemplate::acrobat:notem
plate::pdf:notemplate::gif:notemplate::jpg:notemplate:
```

The **MIME Header** section is illustrated in Figure 4:

```
[MIME_HEADER]
NUMBER_OF_HEADERS=8
HEADER1=TEXT_Content-type: text/plain
HEADER2=HTML_Content-type: text/html
HEADER3=HTM_Content-type: text/html
HEADER4=PDF_Content-type: application/pdf
HEADER5=GIF_Content-type: image/gif
HEADER6=BMP_Content-type: image/bmp
HEADER7=AVI_Content-type: application/avi
HEADER8=JPG_Content-type: image/jpg
```

Figure 4: The MIME Header

Copy the templates **(*.WPB)** and the ARCHIVE.HTML, FRNCLIET and FRNEUIS.TBL files from the **view** directory to **/usr/lpp/frn/samples/cliette**. Modify the ARCHIVE.HTML file to contain your hostname.

Next, you will have to restart DLIC. We provide a script for this called **dlic.restart** which is available in the /usr/lpp/dlct/ias/**misc** directory.

4.0 Producing Derivatives With COMIT95

4.1 Producing Reduced Images with REDUCE.EXE (Stand-alone)

The program REDUCE.EXE, allows the user to take a 8-bit grayscale or 24-bit color TIFF image (as input) and create a "reduced" output TIFF image of the same type.

To execute REDUCE.EXE, 1) the user must be in an OS/2 window and 2) be in the drive/directory where COMIT95 has been installed. Next the user enters the following command:

```
reduce infile outfile hsize vsize
```

where:

infile	input TIFF filename (can include drive and directory),
outfile	output TIFF filename (can include drive and directory),
hsize	maximum horizontal output dimension (pixels per inch),
vsize	maximum vertical output dimension (pixels per inch)

Note: If the output dimensions are not proportional to the input dimensions, the program (reduce.exe) will perform a "best fit". This implies that the output dimensions may not always be what you provided. Also, **a dimension of 0 indicates no reduction in that dimension.**

Examples:

1) Reduce a 24-bit color TIFF image (d:\tiff\color.tif) which is 800 x 800 pixels per inch, to a 24-bit color TIFF image (d:\imageout\colorr1.tif) 400 x 400 pixels per inch.

infile:	24-bit color TIFF file (800 x 800) d:\tiff\color1.tif
outfile:	24-bit color TIFF file (400 x 400) d:\imageout\colorr1.tif
hsize:	400
vsize:	400

```
reduce d:\tiff\color1.tif d:\imageout\colorr1.tif  400 400
```

2). Reduce an 8-bit gray scale TIFF image (d:\tiff\gray1.tif) which is 1000 x 1200 pixels per inch to an 8-bit gray scale TIFF image (d:\imageout\grayr1.tif) 500 x 600 pixels per inch.

infile: 24-bit color TIFF file (1000 x 1200) d:\tiff\gray1.tif
outfile: 24-bit color TIFF file (500 x 600) d:\imageout\grayr1.tif
hsize: 500
vsize: 600

reduce d:\tiff\gray1.tif d:\imageout\grayr1.tif 500 600

4.2 Producing Sharpened Images with SHARPEN.EXE (stand-alone)

The program SHARPEN.EXE, allows the user to sharpen 24-bit color TIFF images. SHARPEN.EXE requires 3 parameters: input TIFF filename, output TIFF filename and a sharpening specification. This sharpening specification must be a numeric value in the range 0.5 to 4.5 inclusive and has a default value of 1.5.

To execute SHARPEN.EXE, 1) the user must be in an OS/2 window and 2) be in the drive/directory where COMIT95 has been installed. Next the user enters the following command:

sharpen infile outfile sharp

where:
infile input TIFF filename (can include drive and directory),
outfile output TIFF filename (can include drive and directory),
sharp sharpness specification (0.5 thru 4.5, default = 1.5)

Examples:

1) Sharpen a 24-bit color TIFF image (d:\tiff\color.tif) creating a 24-bit color TIFF image (d:\imageout\colors1.tif)

infile: 24-bit color TIFF file (800 x 800) d:\tiff\color1.tif
outfile: 24-bit color TIFF file (400 x 400)
sharp: 1.5

sharpen d:\tiff\color1.tif d:\imageout\colosr1.tif 1.5

4.3 Producing Reduced and Sharpened Images with REDUCES.EXE (stand-alone)

The program REDUCES.EXE, allows the user to take a 8-bit grayscale or 24-bit color TIFF image (as input) and create a "reduced" and "sharpened" output TIFF image of the same type. REDUCES.EXE should be used **in place** of REDUCE.EXE and SHARPEN.EXE when **both** are required.

To execute REDUCES.EXE, 1) the user must be in an OS/2 window and 2) be in the drive/directory where COMIT95 has been installed. Next the user enters the following command:

reduces infile outfile hsize vsize

where:

infile	input TIFF filename (can include drive and directory),
outfile	output TIFF filename (can include drive and directory),
hsize	maximum horizontal output dimension (pixels per inch),
vsize	maximum vertical output dimension (pixels per inch),
sharp	sharpness specification (0.5 thru 4.5, default = 1.5),
filter lambda	3.0 (normally a constant - should not need to change)

Note: If the output dimensions are not proportional to the input dimensions, the program (reduces.exe) will perform a "best fit". Note: The **hsize** and **vsize** parameters are *maximums.*

Examples:

1) Reduces a 24-bit color TIFF image (d:\tiff\color.tif) which is 800 x 800 pixels per inch, to a 24-bit color TIFF image (d:\imageout\colorr1.tif) 400 x 400 pixels per inch.

infile: 24-bit color TIFF file (800 x 800) d:\tiff\color1.tif
outfile: 24-bit color TIFF file (400 x 400)
d:\imageout\colorr1.tif
hsize: 400
vsize: 400
sharp 1.5
filter lambda 3.0

reduces d:\tiff\color1.tif d:\imageout\colorr1.tif 400 400 1.5 3.0

2) Reduce an 8-bit gray scale TIFF image (d:\tiff\gray1.tif) which is 800 x 800 pixels per inch to an 8-bit gray scale TIFF image (d:\imageout\grayr1.tif) 650 x 650 pixels per inch.

infile: 8-bit color TIFF file (800 x 800) d:\tiff\gray1.tif
output: 8-bit color TIFF file (650 x 650) d:\imageout\grayr1.tif
hsize: 650
vsize: 650
sharp 1.5
filter-lambda 3.0

reduce d:\tiff\gray1.tif d:\imageout\grayr1.tif xxx xxx

4.4 Producing a Rotated Image with ROTATE.EXE (stand-alone)

Rotation is done to change the value of the TIFF "Orientation" flag to 1 (0^{th} row = top, 0^{th} column = left). Rotation is performed in 90 degree clockwise increments with valid degree values of: 0, 90, 180, 270.

To execute ROTATE.EXE, 1) the user must be in an OS/2 window and 2) be in the drive/directory where COMIT95 has been installed. Next the user enters the following command:

rotate N infile outfile [mem [L2 [L1]]]
where:

N number of clockwise degrees (0 ,90, 180, 270)
infile input TIFF filename (can include drive and directory)
outfile output TIFF filename (can include drive and directory)

optional parameters**:**

mem working memory size, in megabytes (default = 8)
L2 Level 2 cache size, in kilobytes (default=256)
L1 Level 1 cache size, in kilobytes (default=8)

Examples:

1) Rotate a 24-bit color TIFF image (d:\tiff\color.tif) 90 degrees clockwise.

N 90 degrees
infile 24-bit color TIFF file (800 x 800) d:\tiff\color1.tif
outfile 24-bit color TIFF file (400 x 800) d:\imageout\colorr2.tif

rotate 90 d:\tiff\color1.tif d:\imageout\colorrt1.tif

2) Rotate a 24-bit color TIFF image (d:\tiff\color.tif) 270 degrees clockwise.

N 90 degrees
infile: 24-bit color TIFF file (800 x 800) d:\tiff\color1.tif
outfile: 24-bit color TIFF file (400 x 800) d:\imageout\colorr1.tif

rotate 270 d:\tiff\color1.tif d:\imageout\colorrt2.tif

4.5 Producing Image Color Transformation with CSTRANS.EXE (stand-alone)

The CSTRANS.EXE transforms an image to a different color space (for example, from the color space used by the scanner to SMPTE). The parameters are the input and output filenames and optionally, the name of the color space ti be used for the output file (as specified in the DISP.INI file). If no color space is specified, "smpte65" is used.

To execute CSTRANS.EXE, 1) the user must be in an OS/2 window and 2) be in the drive/directory where COMIT95 has been installed. Next, the user enters the following command:

cstrans infile outfile colorspace

where:

infile inputTIFF filename (can include drive and directory)
outfile output TIFF filename (can include drive and directory)
colorspace name of colorspace defined in DISP.INI

Example:

1) Ensure color space "smpte65" is used for input file d:\tiff\color1.tif

infile: 24-bit color TIFF file (800 x 800) d:\tiff\color1.tif
outfile: 24-bit color TIFF file (800 x 800) d:\imageout\colorcs1.tif
colorspace: smpte65

4.6 Producing Image Watermarking with WMARK.EXE (stand-alone)

The WMARK.EXE watermarks an image. It takes the following parameters: an input filename, a watermark image filename (also referred to as a watermark mask), and an output file name.

The filenames may be folowed by one or more of the options listed below, in any order. If an option is specified twice (or if both **/d** and **/s** are specified), then the last value specified is taken. Options which specify **X,Y** may be given a single value and that single value applies to both X and Y.

- **/d X,Y** dimensions used to scale watermark image (default = no scaling)
- **/s X,Y** fraction of input dimensions to scale watermark
- **/v N** fraction of input dimensions by which the scaled dimensions may vary (default = 0)
- **/p X,Y** watermark position as a fraction of input dimensions (default = 0.5, 0.5)
- **/w** fraction of input image dimensions that the watermark image can "wander" (default = 0,0)
- /i watermark intensity, values: 0 - 100 (default = 5)
- **/g** watermark granularity, values: 0 - 1 (default = 0.4)
- **/r** random number generator seed (default is to seed from timer)

The size of the watermark is controlled by the **/d, /s,** and **/v** parameters. The size may be specified either as a dimension in pixels or as a fraction of the input dimensions. In either case, the specified size used as a "frame" and the watermark image is scaled to fit completely within it. The **/v** parameter may be used to modify the scaling slightly under the control of the random number generator, making the watermark more difficult to remove. A small value for **/v** (e.g. 0.02) is suggested if all watermarks are intended to be about the same (absolute or relative) size.

The watermark is positioned in the image using the **/p** and **/w** parameters. The **/p** parameter specifies where the center of the watermark should be placed relative to the size of the input image. Thus the default of 0.5,0.5 centers the watermark. The **/w** parameter is similar in concept to the **/v** parameter. It is used to allow the watermark center to "wander" around the specified position, again making the

watermark more difficult to remove. As with the **/v** option, a small value of the **/w** parameter (e.g. 0.02) is suggested if the watermarks for a group of images are intended to appear similar.

The appearance of the watermark is controlled by the **/i** and **/g** parameters, which may vary the intensity and the granularity (smoothness) of the watermark. A value of 5 to 10 seems to work well for intensity. The granularity is the intensity of the noise field by which the watermark pixels are scaled before combining them with the original image. A granularity of 0 means that no noise will be added and the watermark will be smooth. A granularity of 0.4 seems to work well.

The **/r** parameter may be used to seed the random number generator with a known value if it is desirable to be able to reproduce the placement (with **/v** and/or **/w**) and noise field associated with a particular watermark.

Note: The watermark mask TIFF image must be an 8-bit grayscale TIFF image and must be smaller (pixels x pixels) than the image to be watermarked.

To execute WMARK.EXE, 1) the user at in an OS/2 window and 2) be in the drive/directory where COMIT95 has been installed. Next the user enters the command:

wmark infile wmarkin outfile [/d X,Y] [/s X,Y] [/v N] [/p X,Y] [/w X,Y] [i N] [/g N [/r N]

where:

infile	input TIFF filename (can include drive and directory
wmarkin	watermark mask TIFF image (can include drive and directory)
outfile	output TIFF filename (can include drive and directory)
/d X,Y	dimensions used to scale watermark (default=no scaling)
/s X,Y	fraction of input dimensions to scale watermark
/v N	fraction of input dimensions by which the scaled dimensions may vary (default = 0)
/p X,Y	watermark position as a fraction of input dimensions (default = 0.5, 0.5)
/w X,Y	fraction of input image dimensions that the watermark image can "wander" (default = 0,0)
/i N	watermark intensity, values 0 - 100 (default = 5
/g N	watermark granularity, values 0 - 1 (default = 0.4)
/r N	random number generator seed (default = seed from timer)

Example:

1) Add watermark mask (d:\wmmask\wmlogo.tif) to a 24-bit color TIFF image (d:\tiff\color1.tif) creating a 24-bit color TIFF output image (d:\imageout\colorwm1.tif). Take all defaults.

infile	24-bit color TIFF file (800 x 800) d:\tiff\color1.tif
wmarkin	8-bit grayscale TIFF file (700 x 700) d:\wmask\wmlogo.tif
outfile	24-bit color TIFF file (800 x 800) d:\imageout\colorwm1.tif

wmark d:\tiff\color1.tif d\wmmask\wmlogo.tif d:\imageout\colorwm1.tif

4.7 Image (JPEG) Compression with COMPRESS.EXE (stand-alone)

The COMPRESS.EXE file performs JPEG image compression. The parameters are: The input and output (compressed) filenames, the name of a file of JPEG markers, which controls the compression operation, a string specifying copyright information to be inserted into the compressed data stream, and, optionally, either one or three parameters that specify scaling of the quantization matrices given in the header files. No scaling is represented by 100. Smaller values reduce the quantization values, for example, 50 scales each value by 50/100 = 0.5. This leads to reduced distortion of the image but results in poorer compression. The values apply to the Y, Cb and the Cr components respectively. If only one value is given for a color image, the same value is used for all 3 components. Only the first value is used for monochrome images.

The compression headers supplied (GENY.HDR - for monochrome images and GENYUV.HDR for color images) include a JFIF (JPEG File Interchange Format) header at the Version 1.02 level. The JFIF specification states that " the color space to be used is YCbCr as defined by CCIR 601 (256 levels)". It turns out that the portion of the JFIF specification, which states that images shall not be gamma-corrected, is generally ignored. A gamma of 2.2 seems to be the appropriate value. The images supplied for compression with these headers should conform to these expectations.

To execute COMPRESS.EXE the user at in an OS/2 window and 2) be in the drive/directory where COMIT95 has been installed. Next the user enters the following command:

compress infile outfile hdrfile [comment [QY QCb QCr]]
where:

infile	input TIFF filename (can include drive and directory)
outfile	output JPEG filename (can include drive and directory)
hdrfile	JPEG header file (GENYUV.HDR for color, GENY.HDR for monochrome)
comment	comment to add (quoted string)
QY QCb QCr	quantization table scale factors. For color images, QY=QCb=QCr, if QCb and QCr are omitted. (default =100) Smaller values give finer quantization

Examples:

1) Compress a 24-bit color TIFF image (d:\tiff\color1.tif) creating a 24-bit color JPEG output image (d:\imageout\colorcp1.jpg). Add comment "Copyright info". Quantization table scale factors set to 100.

infile	24-bit color TIFF file d:\tiff\color1.tif
outfile	24-bit color JPEG file d:\imageout\colorcp1.jpg
hdrfile	GENYUV.HDR
comment	"Copyright info"
QY QCb QCr	100 (QY=QCb=QCr)

compress d:\tiff\color1.tif d:\imageout\colorcp1.jpg genyuv.hdr "Copyright information" 100

2) Compress an 8-bit grayscale TIFF image (d:\tiff\gray1.tif) creating an 8-bit grayscale JPEG output image (d:\imageout\graycp1.jpg). Add comment "Copyright information". Quantization table scale factors set to 100.

infile	8-bit grayscale TIFF file d:\tiff\gray1.tif
outfile	8-bit grayscale JPEG file d:\imageout\graycp1.jpg
hdrfile	GENY.HDR
comment	"Copyright information"
QY	100 (Note: monochrome uses QY only)

compress d:\tiff\gray1.tif d:\imageout\graycp1.jpg geny.hdr "Copyright info" 100

4.8 Image (GIF) Compression with GIFCOMPR.EXE (stand-alone)

The GIFCOMPR.EXE file palettizes an image and performs GIF compression. The parameters are the input and output (compressed) filenames.

To execute GIFCOMPR.EXE, 1) the user at in an OS/2 window and 2) be in the drive/directory where COMIT95 has been installed. Next the user enters the following command:

gifcompr infile outfile

where:

infile	TIFF filename (can include drive and directory)
outfile	GIF filename (can include drive and directory)

Examples:

1) Compress a 24-bit color TIFF image (d:\tiff\color1.tif) creating a 24-bit color GIF output image (d:\imageout\colorcp1.gif).

infile	24-bit color TIFF file d:\tiff\color1.tif
outfile	24-bit color GIF file d:\imageout\colorcp1.gif

gifcompr d:\tiff\color1.tif d:\imageout\colorcp1.gif

2) Compress an 8-bit grayscale TIFF image (d:\tiff\gray1.tif) creating an 8-bit grayscale GIF output image (d:\imageout\graycp1.gif).

infile	8-bit grayscale TIFF file d:\tiff\gray1.tif
outfile	8-bit grayscale GIF file d:\imageout\graycp1.gif

gifcompress d:\tiff\gray1.tif d:\imageout\graycp1.gif

4.9 Image Compression (TIFF with lossless JPEG) .

The COMPRTIF.EXE file compresses an image using lossless JPEG and wraps it in the same TIFF headers that were present in the original file. No color space transformation is performed. The TIFF headers describing the compressed data are constructed in accordance with the proposed revision of the TIFF 6.0 specification entitled "Draft TIFF Technical Note #2". Dated March 17, 1995. The image is encoded as a single strip. The parameters for COMPRTIF.EXE are the input and output (compressed filenames.

To execute COMPRTIF.EXE, 1) the user must be in an OS/2 window and 2) be in the drive/directory where COMIT95 has been installed. Next the user enters the following command:

comprtif infile outfile

where:

infile	TIFF filename (can include drive and directory)
outfile	TIFF filename (can include drive and directory)

Example:

1) Compress a 24-bit color TIFF image (d:\tiff\color1.tif) creating a 24-bit color TIFF output image (d:\imageout\colorct1.tif).

infile	24-bit color TIFF file d:\tiff\color1.tif
outfile	24-bit color lossless TIFF filed:\imageout\colorct1.gif

comprtif d:\tiff\color1.tif d:\imageout\colorct1.tif

5.0 Loading Data

5.1 Overview:

In the DLCT solution we are interested in storing a set of images, along with the associated indexing information into the Digital Library. The Loader component of the DLCT solution provides the functionality to store the images and indexing information into the Digital Library. The Loader allows you to create new Items in the Digital Library, update these Items later on, or delete these Items. Typically, in any application of DLCT, a large number of Items are to be stored in the Digital Library. To facilitate this process, the Loader is designed to perform a batch operation by reading input from external files. The Loader supports two file formats at this time. 1) The output of IMGPREP (specifically, the IMGLSTY.EXE program) is the IMAGES.LST file, which is a fixed field file and consists of one line per record. This file lists the names of the derivative files and their sizes, and this file format is known as the 'COMIT95' format. 2) A variable length field format file , with a 'tab' character delimiting the fields. This file format is known as 'Delimited' file format. It is also known as a Comma Separated Variable (CSV) file in database parlance. Note that the Loader only supports a 'tab' character as a separator character at this time.

Loading is performed in two steps:

Step 1: In this step, you create a new Item in Digital Library. At this time you add the Content Parts (derivative images) to this Item. If you plan to use DLIC for viewing the images over the web, then you have to add a Notepart (which is a special type of Content Part) to the Item. In this Notepart there will be an entry which acts as a Table-of-Contents for DLIC. In addition, you can create a text part composed of attribute-value pair of information from the index class. This text part is called the 'Pseudo' text part, and is indexed by a free-text search engine like Text Miner, and allows users to perform free-text queries against the attribute values of this Item. Without this pseudo text part, users can only perform a parametric query to search for Items.

The input to the Loader at this step is the IMAGES.LST file generated by the IMGPREP component of DLCT. This file, as described in earlier chapters, consists of one record for every original TIFF image processed. The record contains the file names of the derivative images that are to be loaded into the Digital Library. Using this file, for each record, the Loader creates an Item in Digital Library, and adds the derivative images as Content Parts for this Item. The IMAGES.LST file does not have fields for descriptive information about the items, and so, in the second step, you update the attribute values of the Items.

Step 2: The input file for this step is an Annotation file, which is simply an ASCII file with variable length fields per line. Each field is separated by the 'tab' character. One of the fields is a Primary Key which is used to locate the already existing Item in the Digital Library. The remaining fields are used to update the attribute values of this Item.

Typically, loading is performed in small batches of manageable number of items. The reason being that if things go wrong during the loading process, and the Items are not created properly, the incomplete Items will have to be deleted. After the problem that caused the failure is resolved, the Items can be added back into the Digital Library.

Figures 5 and 6 illustrate the two loading steps described above.

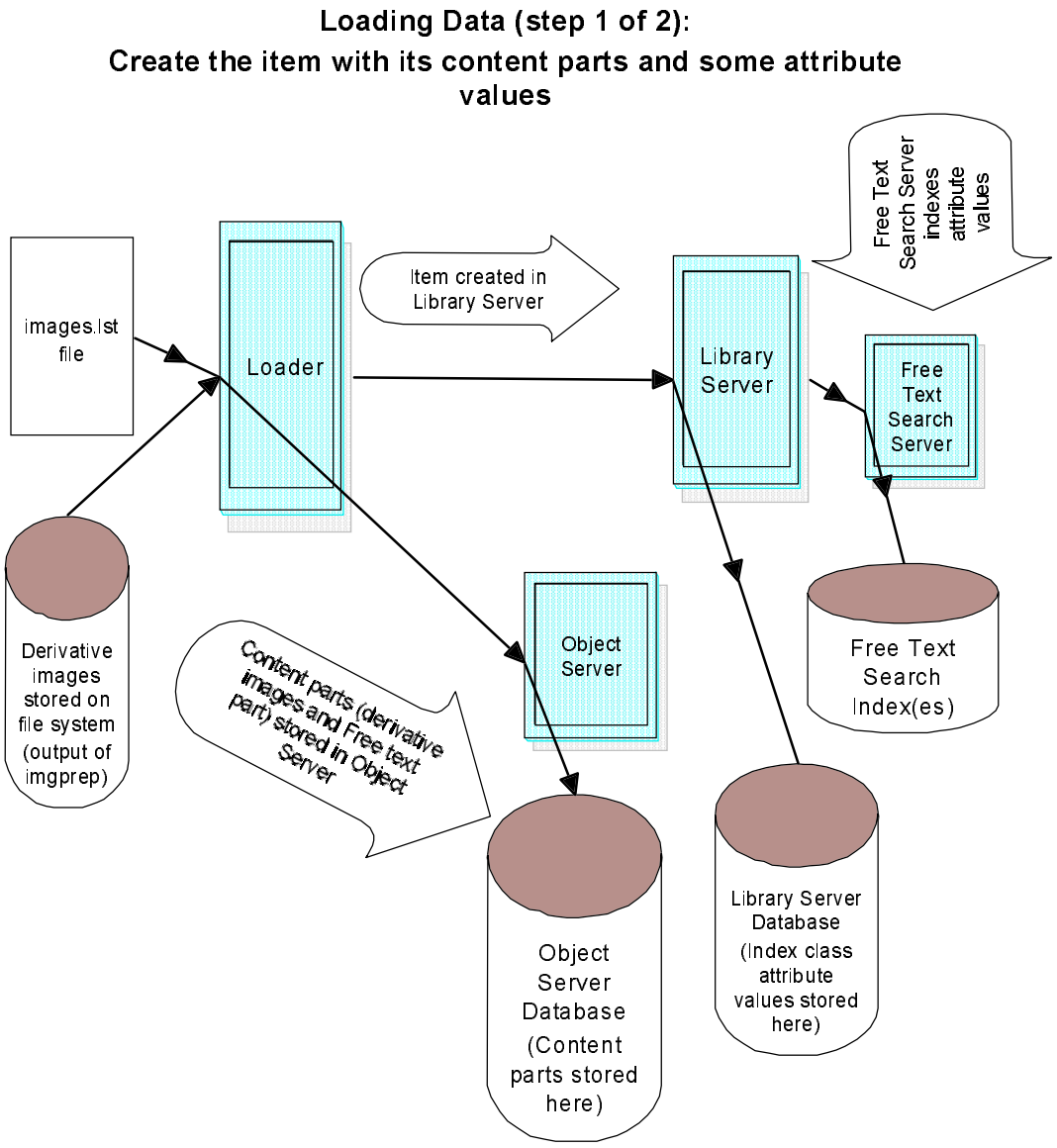

Figure 5: Step One in Process of Loading Data

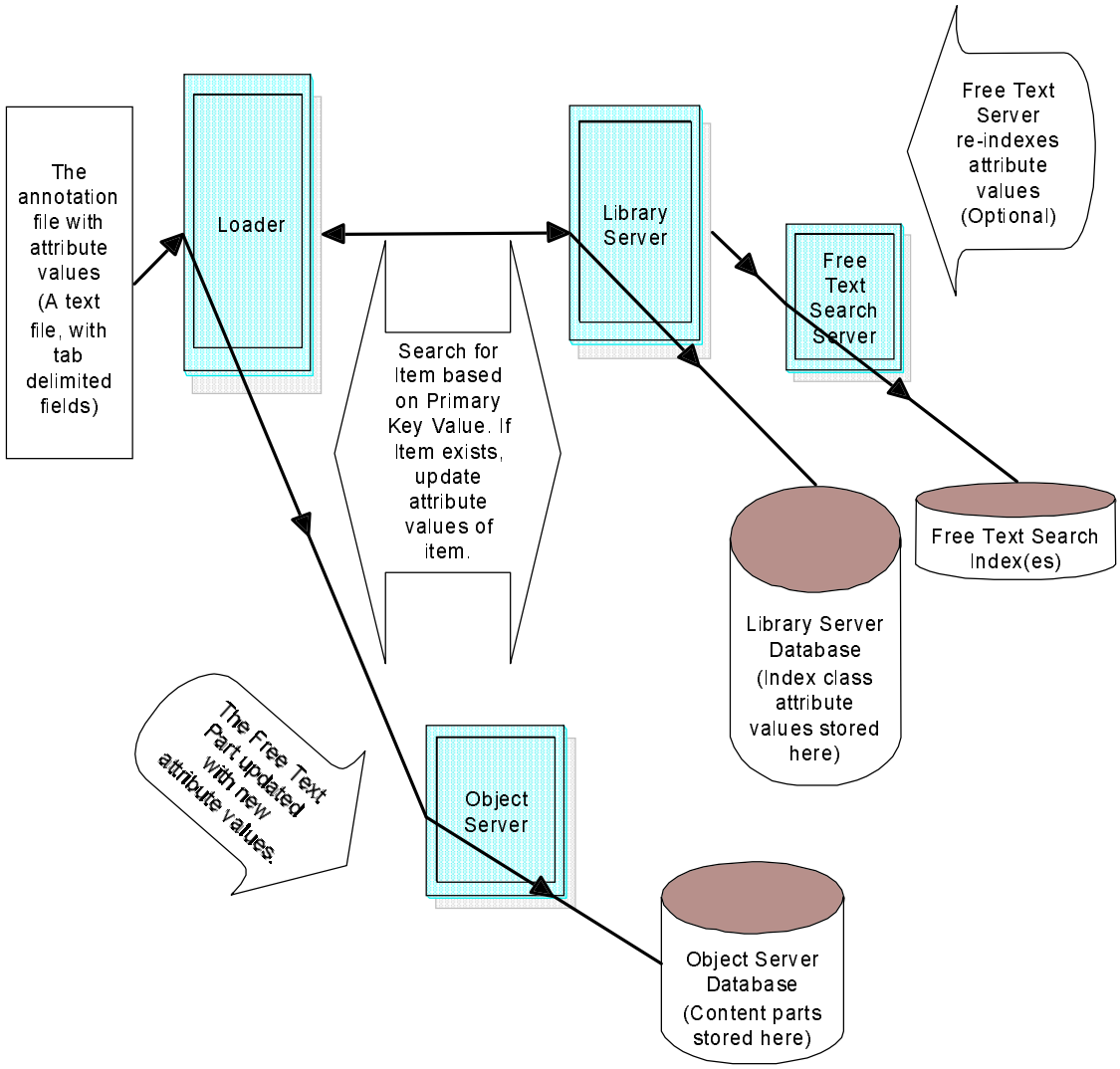

Figure 6: Step Two in Process of Loading Data

5.2 Parts of the Loader.

The IBM Digital Library Loader consists of the following parts:

The Loader Application Window

- The Message Console
- The Status Bar
- The Menu Bar

The Data Model Editor Main Window

- The Data Model Property Sheets
- The Apply button on each Property Sheet
- The Menu Bar

Figures 7 and 8 show the Loader and the Data Model Editor respectively.

Loader Functions:

- File\Quit
  To quit the application
- Actions\Logon
  To log on to the Library Server
  Supply the userid, password and the library server to logon.
- Actions\Logoff
  To log off from the Library Server
- Actions\Data Model Editor
  Invokes the Data Model Editor
- Actions\Add Item
  Adds a new Item to the Library
- Actions\Update Item
  Updates the attributes or content parts (or both) of an existing Item in the Library

- Actions\Delete Item
  Deletes an existing Item (including its content parts) from the Library
- Actions\Stop
  Stops any activity (logon, Add Item, Update Item, Delete Item)
- Actions\Clear Messages
  Clears the Message Console
- Options\Verbose Progress Messages
  If selected, (default) the Loader displays progress messages of the Message Console. If off, messages are displayed only when there are errors
- Options\Log Messages to File (dlct.log)
  If selected, logs a copy of all messages seen on the Message Console to the file dlct.log, in the directory the Loader is launched from
- Options\Preview Only (to verify input file)
  If selected, the actual Item actions (Add, Update and Delete) are not performed. Use this option to check whether your input file has any errors, before you start the actual add, delete or updates
- Help\About
  Displays the version information

5.3 Data Model Editor Functions.

- File\New
  Creates a new instance of a Data Model
- File\Load
  Loads an instance of the Data Model previously saved to disk
- File\Save
  Saves the current instance of Data Model in memory to disk. If this instance was loaded from a disk file, then this menu action saves it back to the same file
- File\Save As
  Saves the current instance of the Data Model in memory to disk, but offers the option of selecting a different file name
- File\Verify
  Verifies that the data entered in the Data Model has no errors.
  This step is needed if a new Data Model has been created

- File\Close
  Closes the Data Model Editor. Note that the data model information is still in memory and can be used by the Loader

5.4 Using the Loader.

The Loader is controlled by a combination of the action command (Add, Delete or Update Item) and the Data Model. First and foremost, the action commands determine what the Loader attempts to do. The information in the Data Model, along with the mapping information determine how that action will be performed.

1. Add Item: This command is used to create new Items in Digital Library. In processing this command, the Loader uses the primary key information from the Data Model to determine which key will serve as the primary key. Then, it uses the mapping information from the Data Model, and for each record in the input file:
   a) Queries the Library Server for an Item with the index class and primary key specified in the Data Model
   b) If an Item exits, the Loader skips to the next record. In other words, you cannot add an Item if it already exists.
   c) If no Item exists, the Loader creates an Item using the specified index class and primary key value. If fields from the input file have been mapped to the attributes of the index class, these attribute values are also used at the time of creation. If fields from the input file have been mapped as Content Part file names, using the information from the 'Content Part' property sheet of the Data Model, the Loader creates the Content Parts for this Item. If the option of creating an free-text part for this Item has been specified, the Loader creates a pseudo text-part, adds it as a Content Part at part ID 155 to the Item, and puts this part on the index queue of the Free-Text Search Engine. If the option of creating a Notepart for DLIC has been specified, then the Loader creates the Notepart, using information from the 'Content Parts' property sheet.
   d) If any of the above steps fail, the Loader will try and delete the partial Item created. This gives you an opportunity to retry this command after fixing the cause of the failure.

2. Update Item: This command is used to update information about an existing Item. In processing this command, the Loader, similar to the Add Item command, uses the primary key value to query the Library Server for the Item.
   a) If an Item is not found, the Loader skips to the next record. You cannot update an Item that does not exit.

b) If an Item is found, the Loader, determines what update action to take. The Loader determines from the mapping information whether any of the fields in the input file have been mapped to attribute values in the index class. If so, the Loader updates the attribute values of the item with the new attribute values read in from the input files. The attributes that have not been mapped as coming from this input file do not change. If the pseudo free-text part has been created during the Add Item command, the Loader deletes the old free-text part, creates a new one with the update attribute values, and re-indexes the content part.

c) Caution: If you map a field in the input file to a Content Part file name, the Loader will try and update the Content Part. This operation is NOT supported in DLCT version 1.0, and can cause random parts to be added to your Item, and you will have to delete your Item. So, make sure that during an Update command, no field from the input file maps to an Content Part file name.

d) Also note that, during Update, the Notepart is not updated. The Notepart is created only during the Add Item phase, and never updated.

3. Delete Item: This command is used to delete an Item along with all the Content Parts. In processing this command, the Loader, similar to the Add and Update Item command uses the primary key information from the Data Model, and queries the Library Server.

a) If no Item is found, the Loader skips to the next record. Since an Item with this primary key does not exist, it cannot be deleted.

b) If an Item is found, the Loader deletes the Content Parts of this Item (including the Notepart and the pseudo text part if created) and then the Item itself.

c) Note that if you map fields from the input file to attribute values or Content Part file names, they are ignored, except for the primary key value.

5.5 Load Time Configuration

Using the Data Model Editor, you can define the Data Model for your application. Once defined, this Data Model must be saved to disk if you intend to reuse it later. When you save the Data Model to disk, you should save it in a file with an extension of **ltc** which stands for Load Time Configuration (LTC). As discussed in section 5.4, loading data into Digital Library is a two step process, the creation process (using the Add Item command) and the update process (using the Update Item command). You

have to define a Data Model for each step. The Data Model defined for the update process can be different than that of the create process. For example, in the update process, since you cannot update Content Parts in DLCT Version 1.0, you can specify the number of Content Parts to be zero, when creating the Data Model. Even if you did define a non-zero number of Content Parts, as long as you did not map any field in the input file to a Content Part file name, the Loader will not care about updating the Content Parts. Thus, the term LTC is used to distinguish the actual Data Model, which is the organization of the Item inside Digital Library from a particular instance of the Data Model created and used during a particular step of the multi-step loading process.

5.6 Using the Loader

Steps for Creating a new Data Model:

1. Launch the Loader (/usr/lpp/dlct/bin/loader)
2. Log on to the IBM Digital Library
3. Invoke the Data Model Editor
4. In the Data Model Editor, create a new Data Model
5. Enter the relevant information in the "General" property sheet (refer to figure 8)
   a) From the drop down list box, choose the index class name that the items in this data model will be indexed by.
   b) Enter the number of content parts that you have planned for each item of this archive
   c) If you plan to use a Free Text Search Engine such as Text Miner (also known as Search Manager), check this checkbox
   d) If you want to add the "attribute value pairs" so that the attribute values are free-text searchable, check the "Create/Update Free Text part from attribute values" checkbox
   e) If you plan to use the Digital Library Internet Connection (DLIC) to view the contents over the web, check the "Add TOC Entry in Note Part for DLIC" check box
   f) Click "Apply" button
6. Based on the options you selected step 5, new property sheets will be added to the Data Model.

7. Select "Attributes" sheet (refer to figure 9).
8. Select the attribute that will server as the "Primary Key". This attribute is highlighted with a red background. By default, the first attribute of any index class is set to be the primary key, and is highlighted in red.
9. If you selected "Create/Update Free Text Part from Attribute Values," you will see a second column labeled "FT" with 'X' marks in that column. This is the column where you can decide which attribute value pairs are to be used in creating the pseudo text document used for free text searching. By default, all attributes are selected. You should click on the attributes that you don't want included in the pseudo text document. An 'X' in a column means that you want this attribute-value pair to be included in the pseudo text document.
10. When finished, click the "Apply" button. This applies your information from this property sheet to the Data Model.
11. Next, select the "Content Parts" sheet (refer to figure 10). This sheet has four columns, and as many rows as the number of content parts you entered in step 5.b.
    a) The first column is "Part ID" and lets you assign any part id you like to each content part. By default, it numbers each part from one, but you can change the part ID. Do not use a part ID of 155; it is reserved for the pseudo text part. Also, do not use a part ID of '0' because it has a special significance to Digital Library.

    b) The second column is "Part Class". Here you assign the content class for each part. You can either enter the content class directly by typing in the cell, or double click in the cell to bring a list box with all the content classes defined for your Library Server. You can scroll this list and select (by double clicking) the content class you want assigned. You might find it helpful to resize the cell to a larger size.

    c) The third column is "Part Type" and by default has the value "BASE". You will not need to change this.

    d) The fourth column is labeled "TOC Entry". In this line, enter the value that will be used by the PageBuilder in the DLIC. Typically, the value is a string such as "PART1.JPG" or "FULL.JPG", which is an identifier for the part.

e) When you are finished, click the "Apply" button. This saves the values you entered on this sheet into the data model.

12. Next, select the FT setup sheet (refer to figure 11). This sheet will not show up if you did not check the "Uses Free Text Services" check box in step 5. c) above.

a) Enter the information requested. You will not need to change the name from "SM".

b) The type field is "ENU," for United States English.

c) Enter the Search Service Name.

d) Enter the Index name.

e) If you plan on having additional content parts that are text parts, and would like these text parts to be indexed, then enter the Content Class that represents this class (e.g.: ASCII).

f) Click the "Apply" button to apply information from this sheet to the data mode.

13. Next, select the "Mapper Setup" sheet (refer to figure 12). The Mapper allows you to map data fields from internal files into attribute values of an index class or content part file names.

a) Enter the input file name. You can click on the browse button to select the file from a File Dialog Box. There are currently two input file formats supported: A 'Delimited' format that has fields separated by tabs, or a COMIT95 format, which is the output of the COMIT95 program.

b) Select the correct input file format for the above file.

c) If you selected "COMIT95" format, then you do not need to enter anything for the number of columns, since the COMIT95 files, by definition, have 27 columns. If you selected the "Delimited" file format, then enter the number of columns in that file.

d) Enter the path to content files. You can use the browse button to choose any content part file. This sets the path to that directory. The mapper prefixes this path information to the actual content part file name read from input file.

e) Click "Apply."

14. Next, select the "Mapping" sheet. This is where, you can visually map each field of the input file to the field in the index class or a filename for a content part. This sheet will have as many rows as the number of columns specified in step 13. c), above. For example in the COMIT95 file case, there will be 27 rows. The number of columns in this grid are equal to the sum of the attributes in the index class and the number of content parts. In this sheet, for the attribute or content part you want to load, you select which field from the input file this value comes from. Figures 13 and 14 show a partial mapping for the sample data set.

15. When finished, click "Apply."

16. Once you have defined your data model, you must verify that all the information you entered seems reasonable. Select File\Verify. You will get the results of your verification on the Message Console. If you don't get "Successful verification", fix the problem, by following the hints in the message. Once you have verified successfully, the Item Action Buttons (Add, Delete, Update) are enabled.

17. You might want to save your data model at this time to disk. Choose File\Save As.

Using a previously defined data model.

Launch the Loader and logon to the Digital Library, and then open the Data Model Editor. Select "File\Load" from the menu to load an existing Data Model. You can change the Data Model if you like, but remember to verify the data at each time.

Adding or Creating New Items.

1. Launch the Loader and logon to Digital Library, and either create a new Data Model, or load an existing Data Model. This Data Model must be verified.

2. Select the appropriate options (verbose, log to file, preview only).

3. Click "Add Item" button.

Updating Items

Same as above, but choose "Update Item" command.

Deleting Items.

Same as above, but choose “Delete Item” command.

Verifying input file

Choose the “Preview Only” option, and then choose either of “Add Item”, “Update Item” or “Delete Item”, which will cause the input files to be read, parsed and prepared for performing the actual operation, but the actual operation will not be invoked.

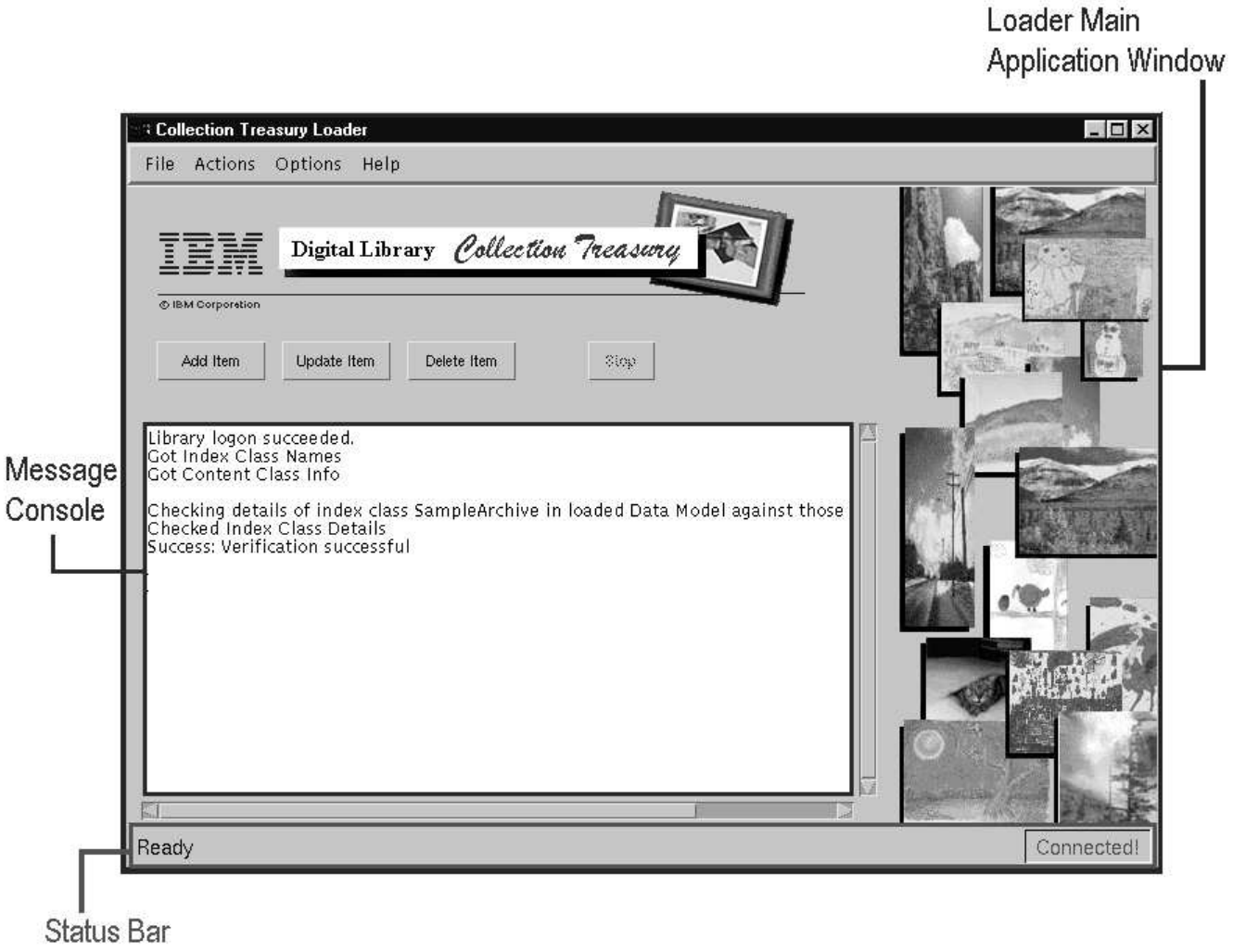

Figure 7: The Loader

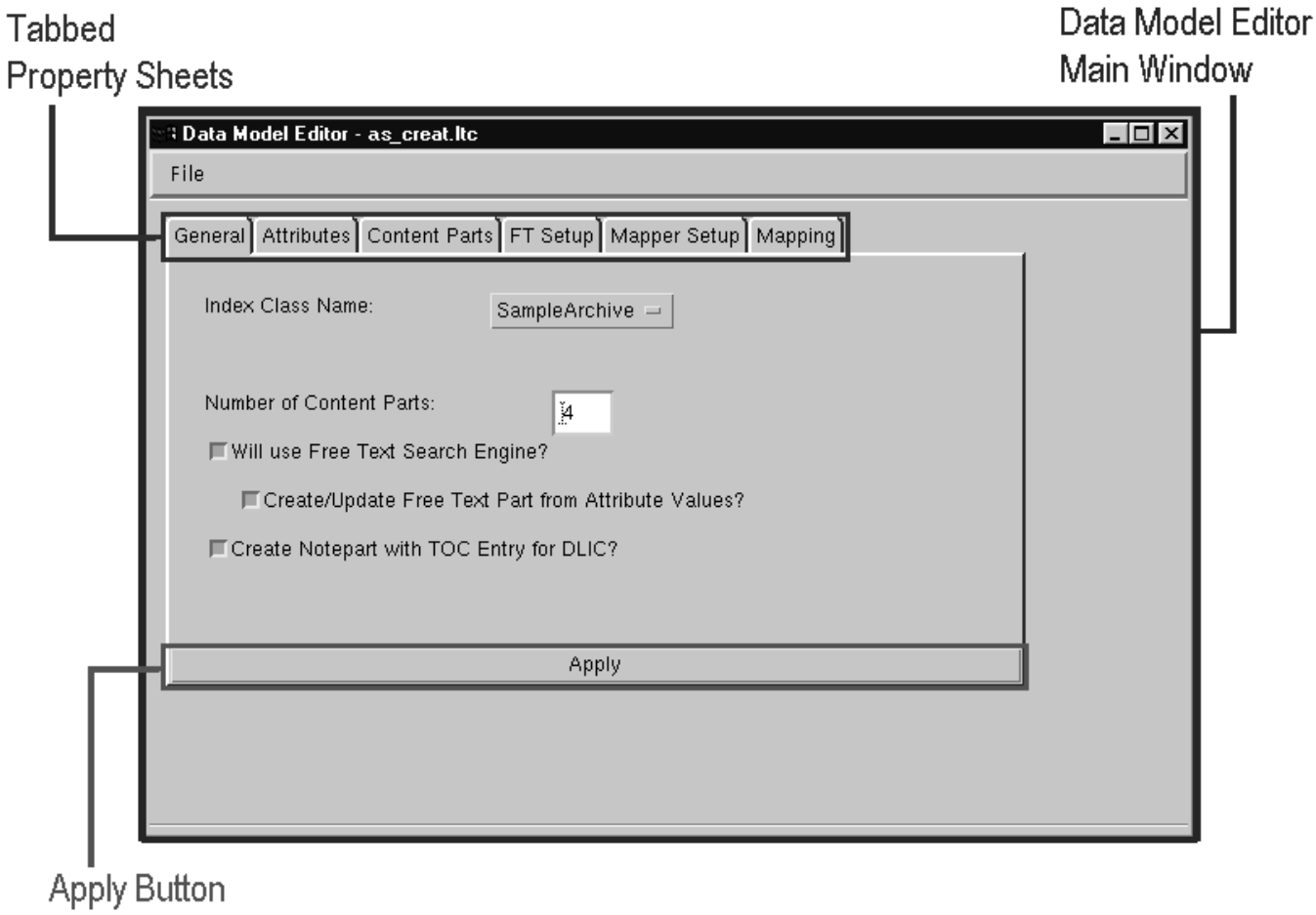

Figure 8: The Data Model Editor with the 'General' property sheet

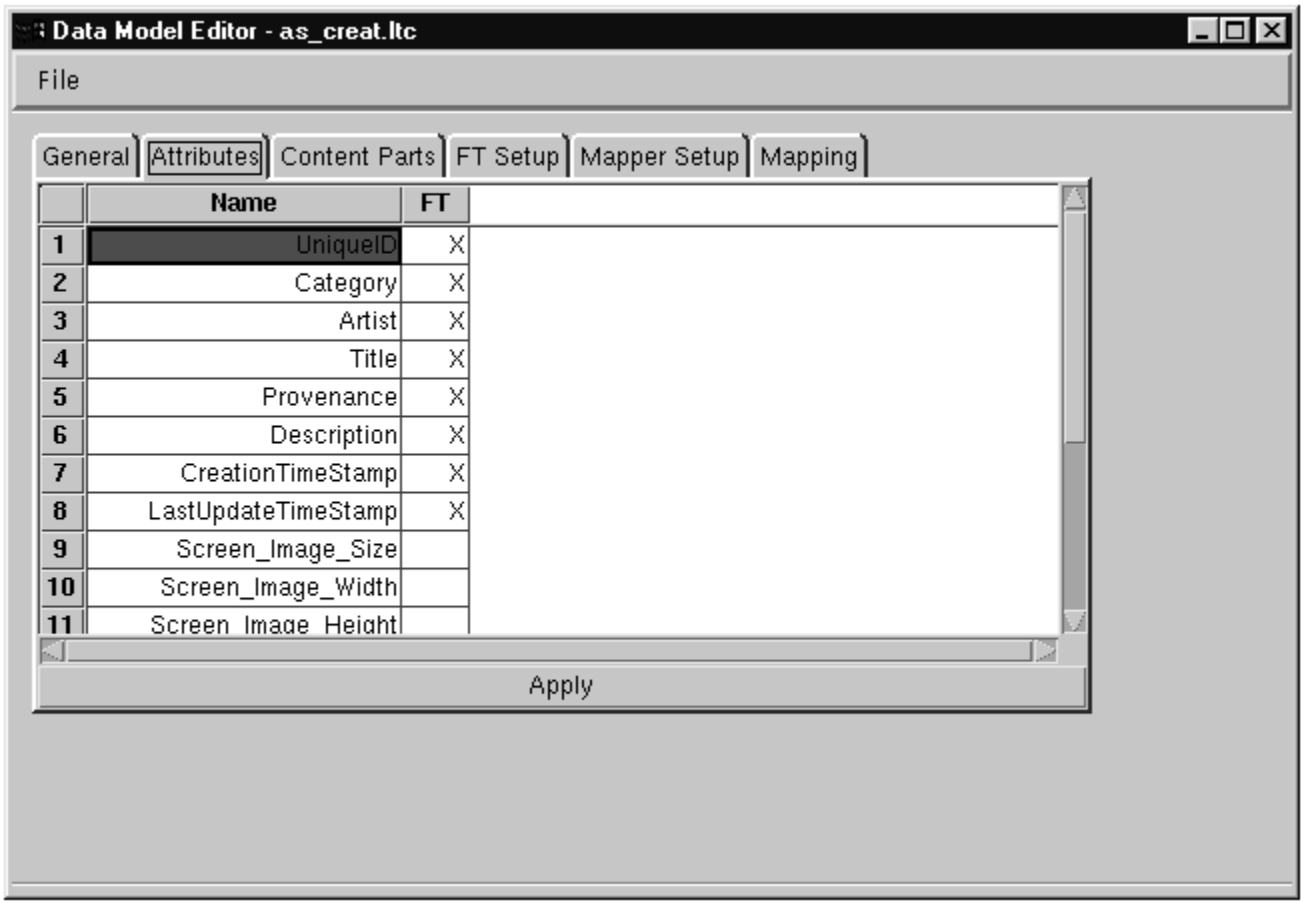

Figure 9: The 'Attribute' property sheet

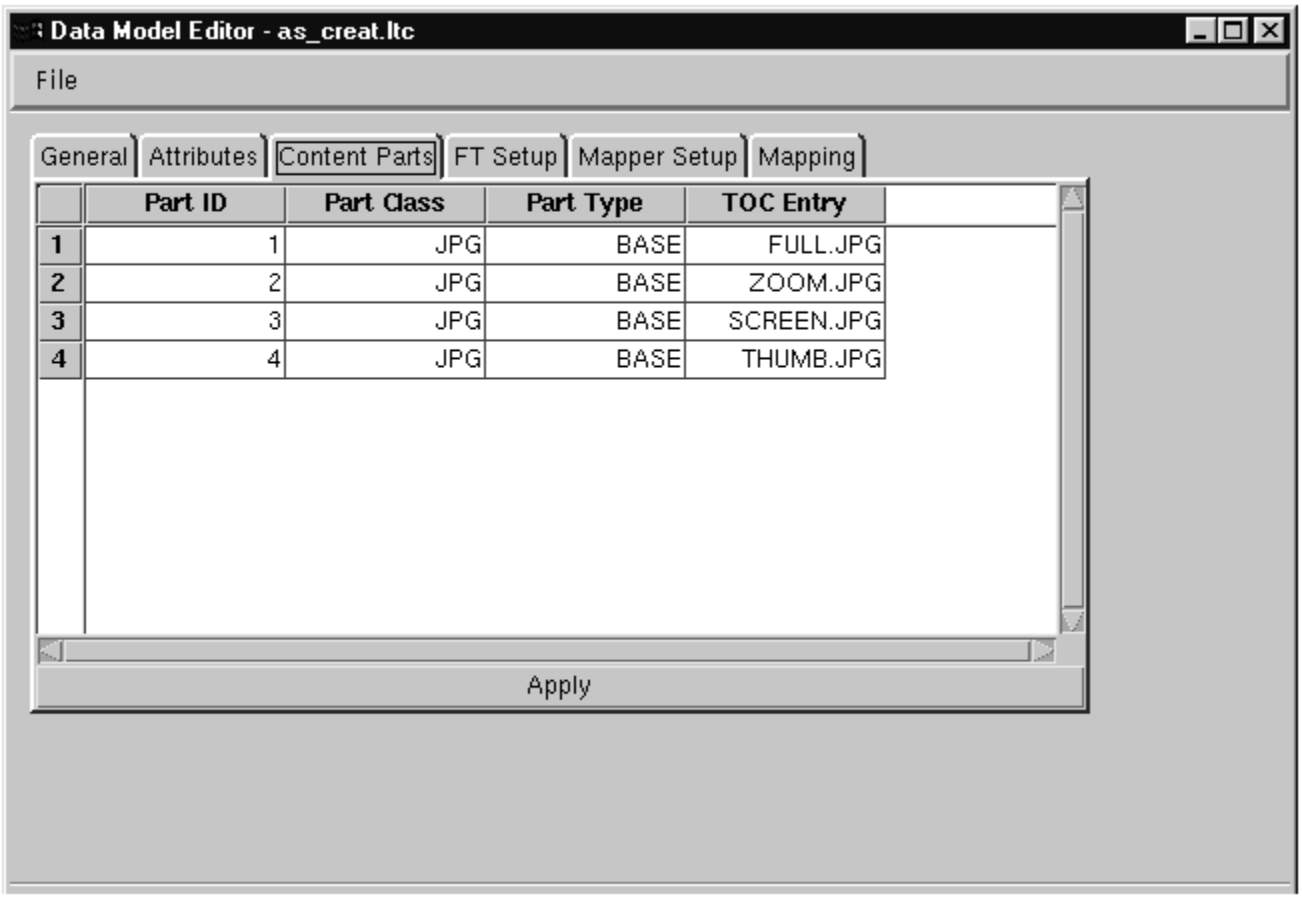

Figure 10: The 'Content Parts' property sheet

© IBM Corporation

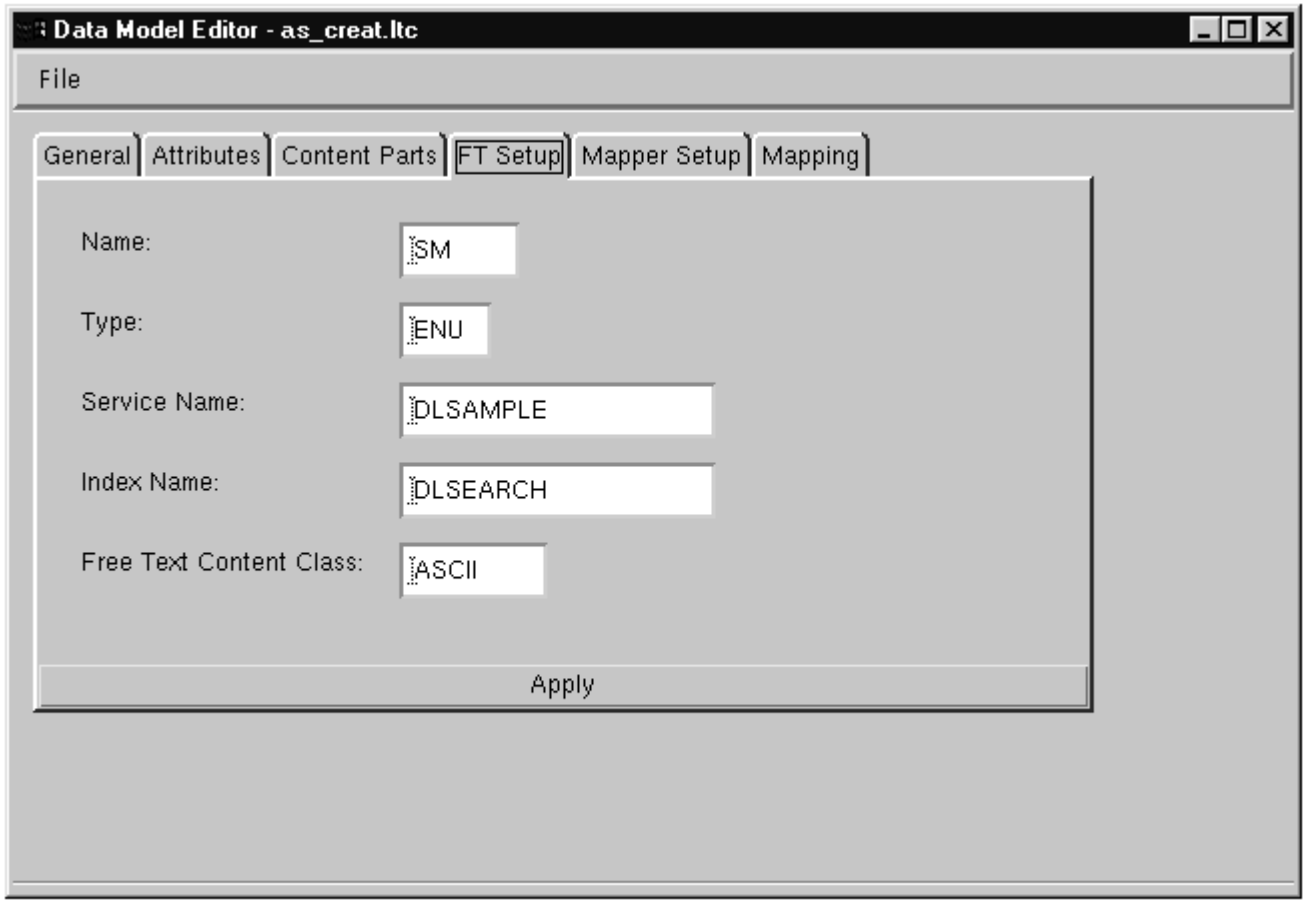

Figure 11: The 'FT Setup' property sheet

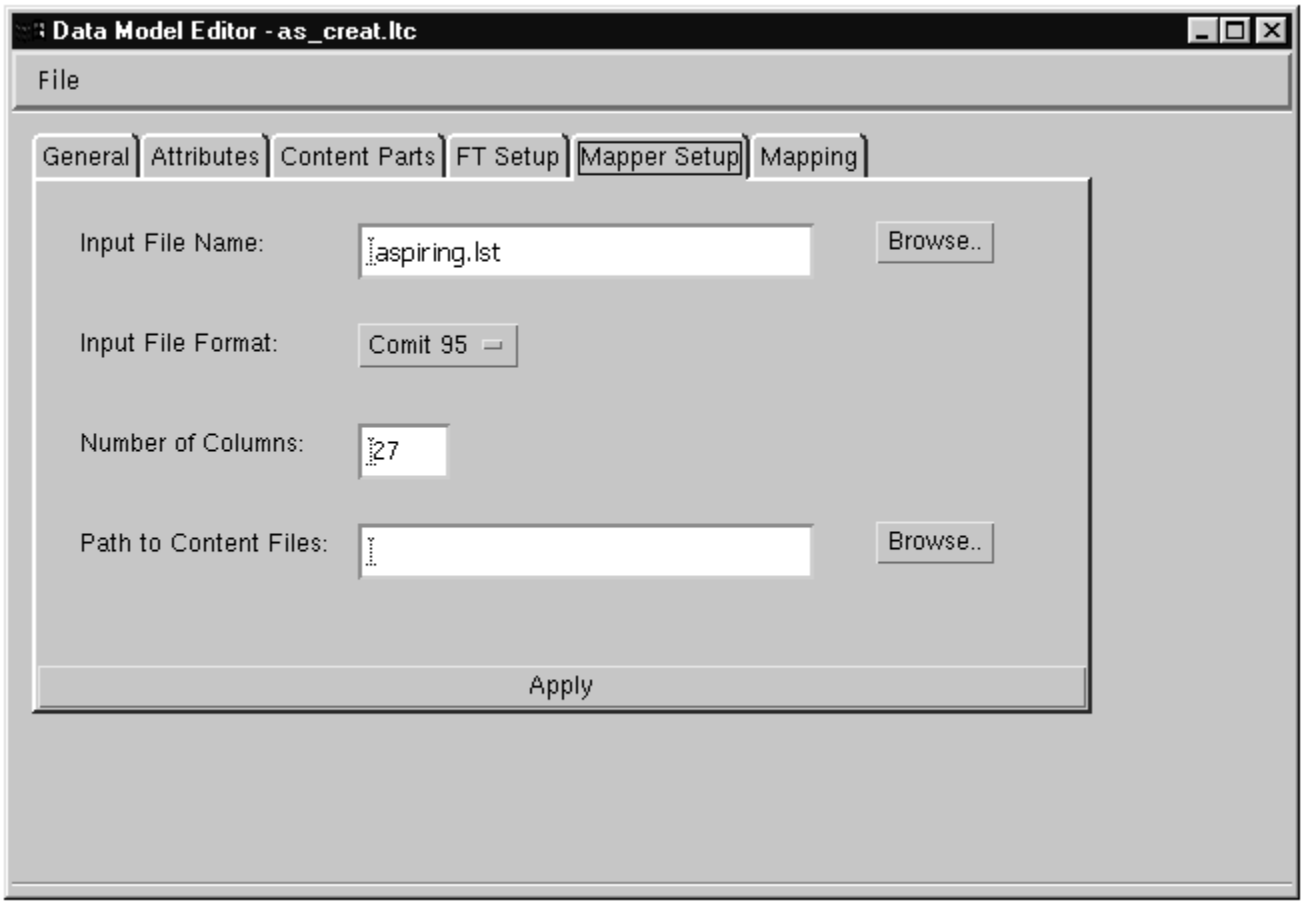

Figure 12: The 'Mapper Setup' property sheet

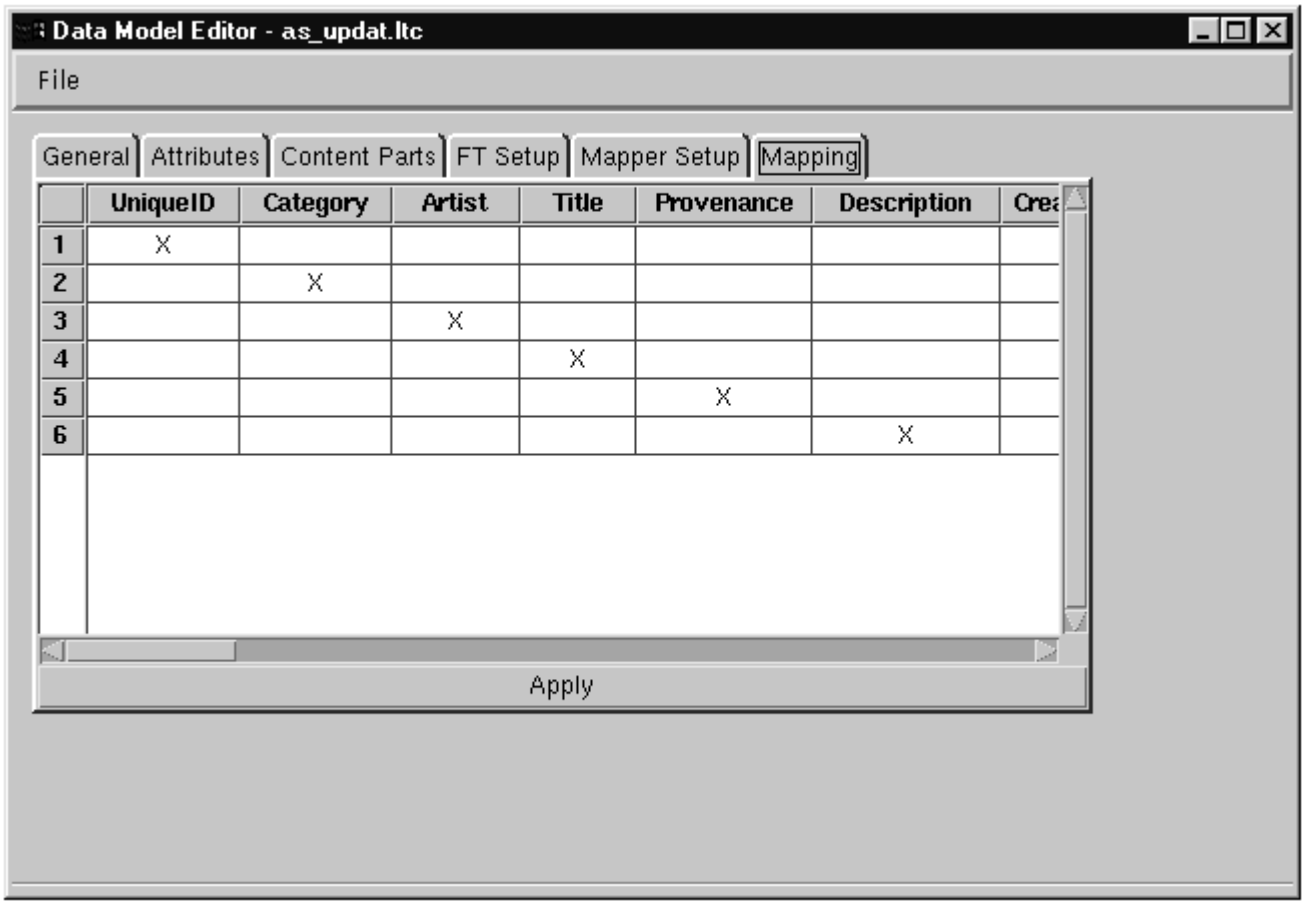

Figure 13: The 'Mapping' property sheet, showing partial mapping for an annotation file

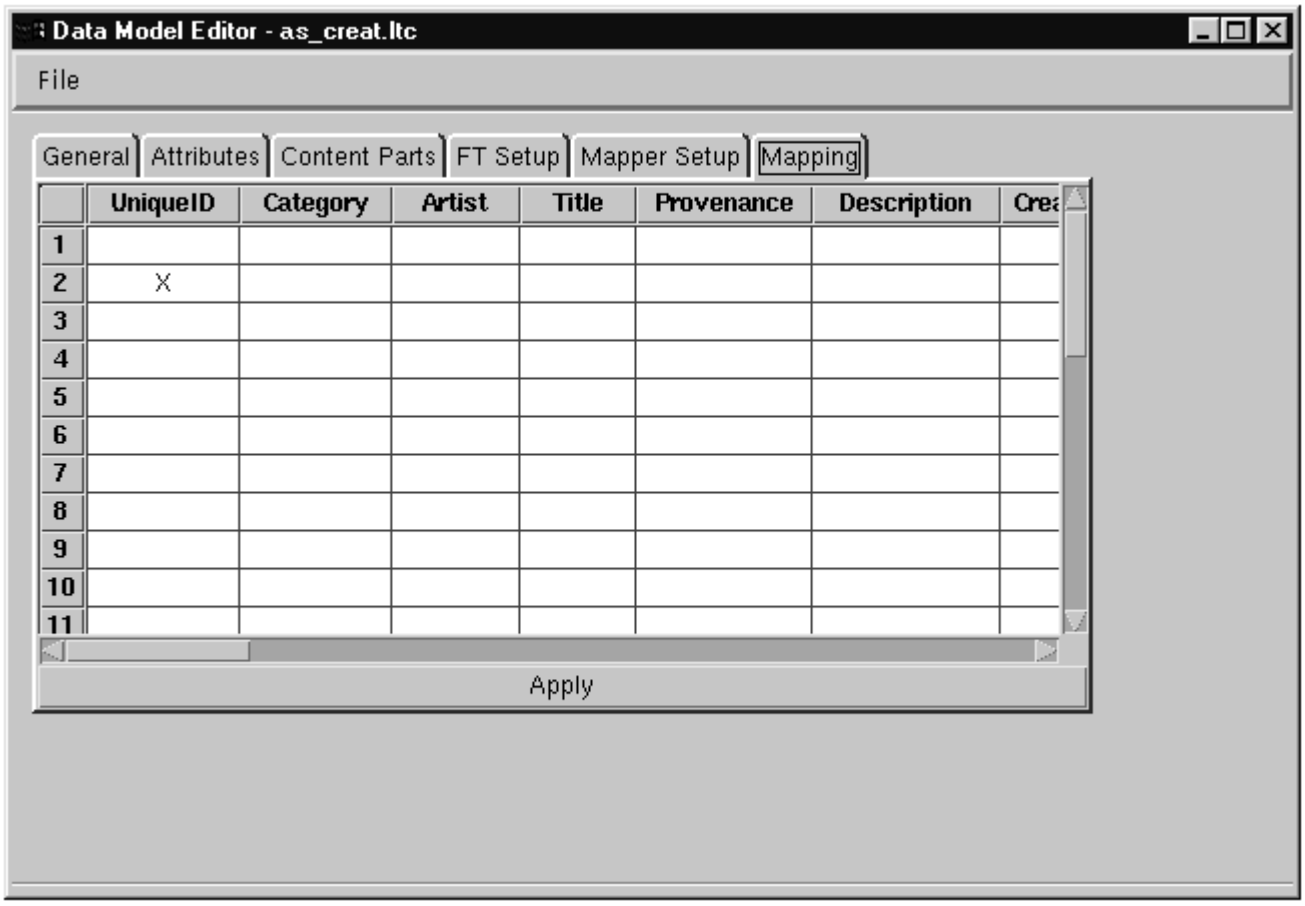

Figure 14: The 'Mapping' property sheet showing partial mapping from the images.lst file

6.0 The DLCT Sample Application

The home page (ARCHIVE.HTML) of the sample application is illustrated in figure 15.

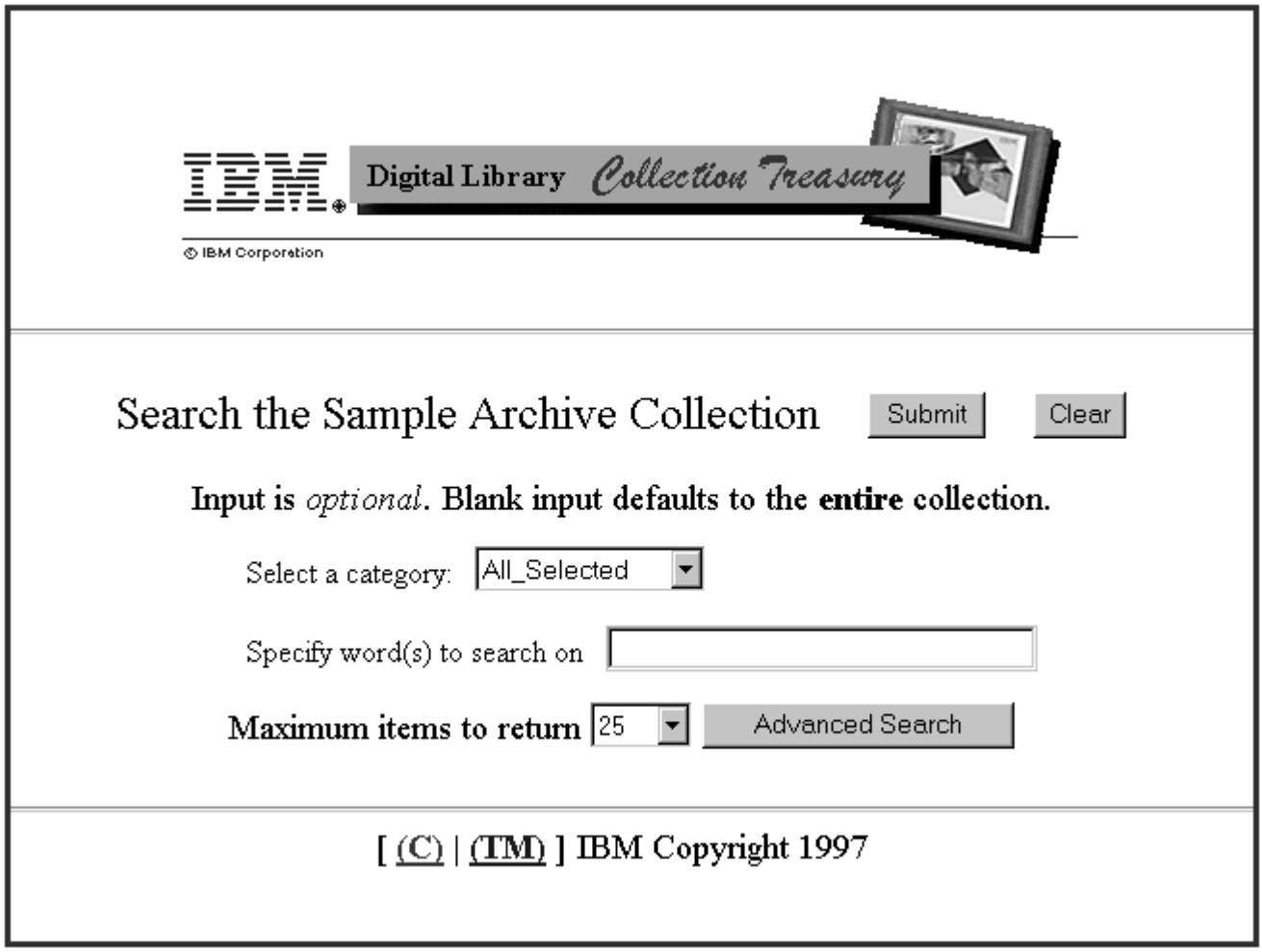

Figure 15: Home Page for Sample Application

As specified on the home page, input is optional. Clicking on the submit button with maximum results set to 25 will cause the entire Sample Archive Index Class to be searched and a random selection of 25 items to be displayed on the results page. The results page from such a search is illustrated in figure 16.

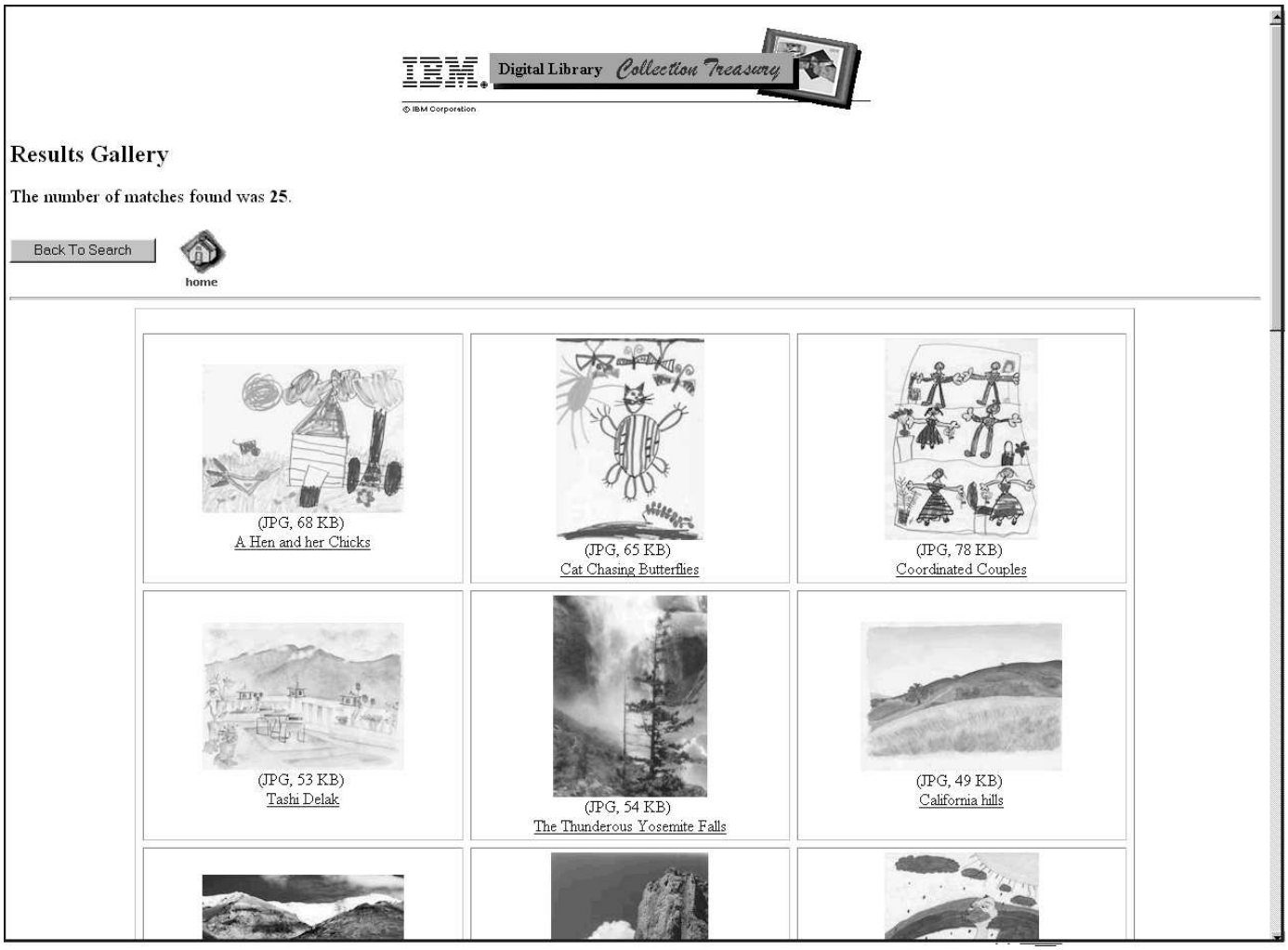

Figure 16: Search Results Page

To limit the search, a category can be selected. To limit further the advanced search page can be used. Clicking on the 'Advanced Search' button from the home page brings up the advanced search page. This page is illustrated in figure 17.

Figure 17 also illustrates the free text search and parametric search areas where values can be input. A sample query is shown in figure 18.

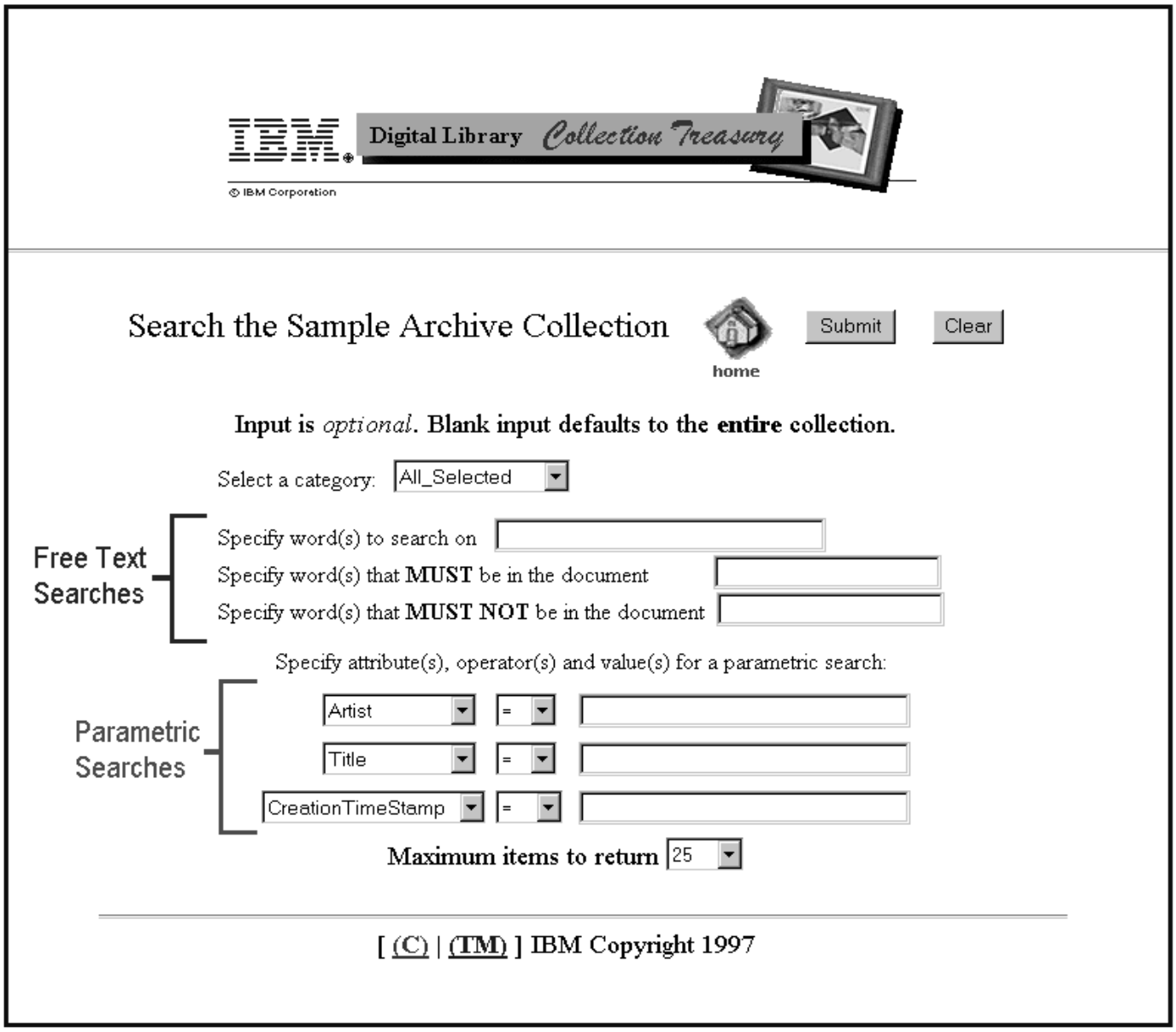

Figure 17: Advanced Search Page

The wild card character, %, is available within an attribute value text field and only works with the ~ attribute operator

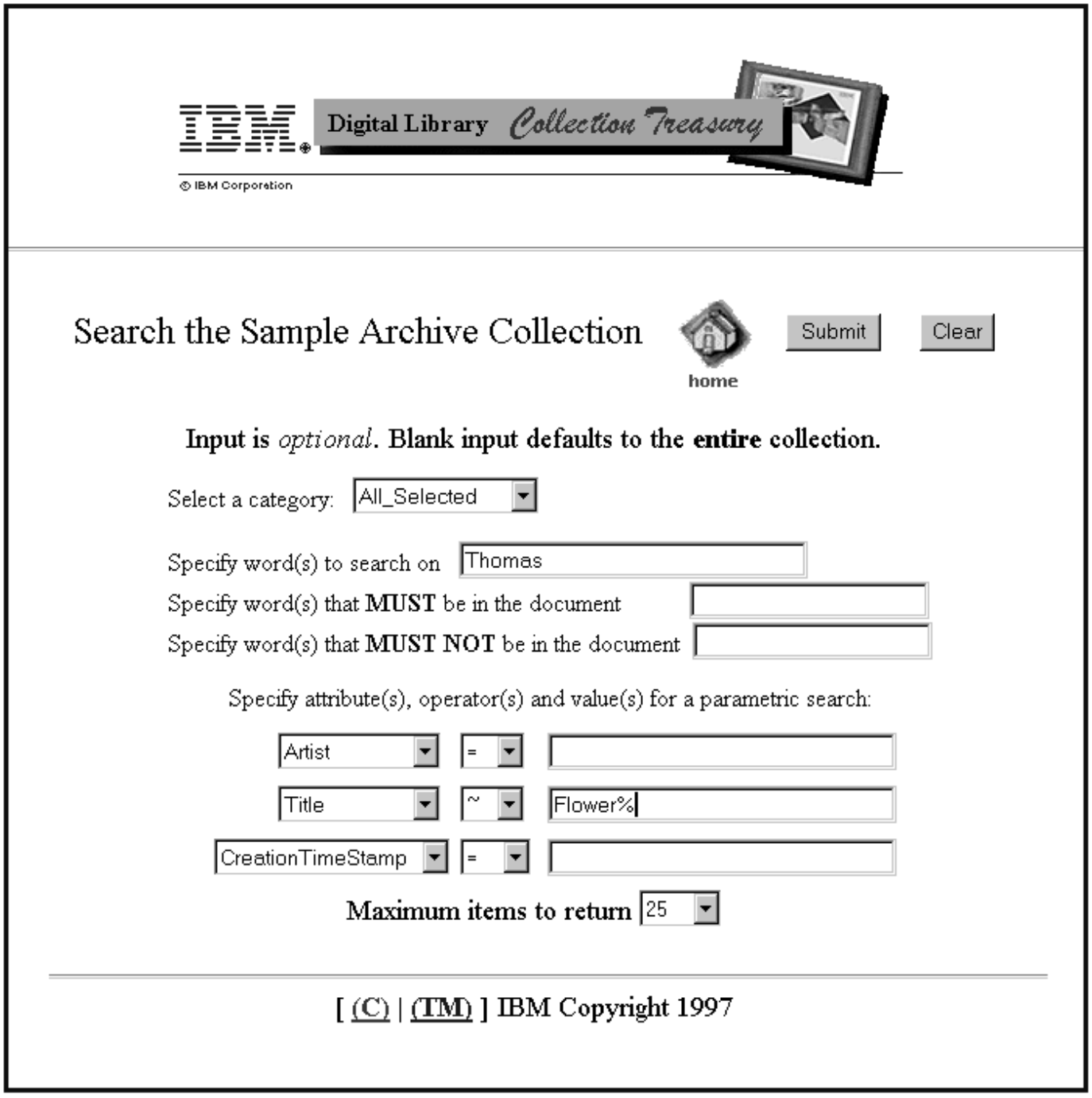

Figure 18 Sample Query

The results of the query shown in figure 18 are illustrated in figure 19.

Figure 19: Results Page From a Search Using The Wild Card Character

From the results page, the user can click a thumbnail to see a 'screen size' version of the same image, as illustrated in figure 20 on the following page.

From the 'screen size' page, clicking on the Zoom-In button will bring up a bigger / higher quality version of the same image. A Zoom-In version of the image is depicted in figure 21.

© IBM Corporation

ZOOM-IN

Flowers with Bugs

Figure20: Screen Size Image Resulting From a Thumbnail Selection

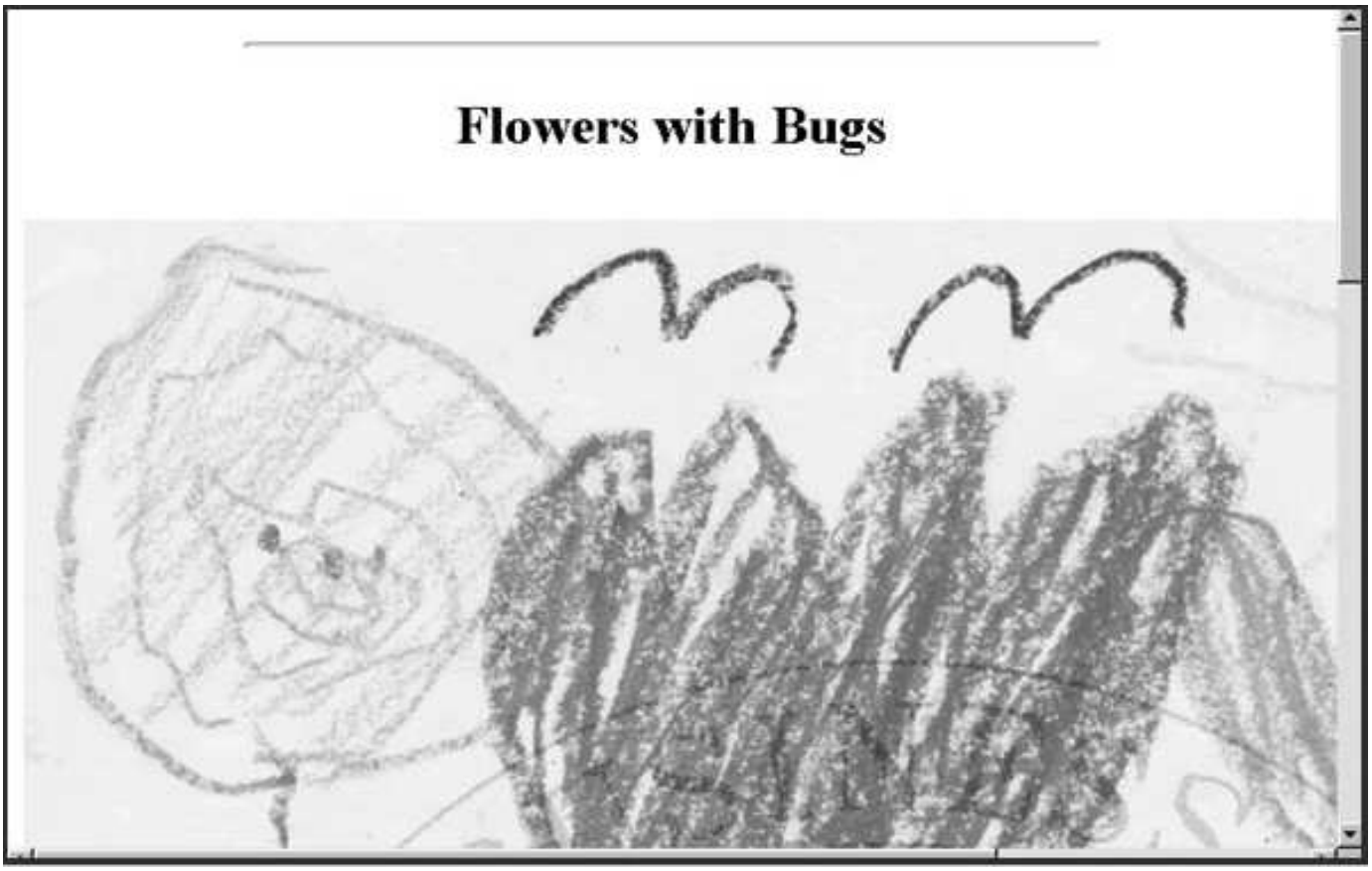

Figure 21: Zoom-in Image Derived From a Screen Size Image

7.0 Customizing the Web Pages

The DLCT end user interface consists of a home page (ARCHIVE.HTML) and six templates. The six templates are:

- ARCHIVE.WPB

  This is the template version of the home page, arhive.html. It is used when the search is initiated from the home page, results have been displayed and the user clicks on the 'Back to Search' button

- ADVANCED.WPB

  This template is used to generate the advanced search page. It is brought up by clicking on the 'Advanced Search' button from the home page or when a search has been initiated from the advanced search page, results have been displayed, and the user has clicked on the “Back to Search” button.

- ARCHIVERESULTS.WPB

  This template is used when the user clicks on the “Submit” button in either the home page or advanced search page.

- ARCHIVESCREEN.WPB

  This template brings up the screen size image of a thumbnail. It is invoked when a thumbnail on the results page is clicked

- ARCHIVEZOOM.WPB

  This template brings up the zoomed-in image. It is invoked by clicking on the 'Zoom-In' button on the archives screen page.

- ARCHIVETEXT.WPB

  This template is used to display the text part associated with an image. It is brought up by clicking on the hypertext, which is displayed below the thumbnails on the results page

A template looks a lot like standard HTML except for the Perl tags. A template is associated with each page of the sample application. Templates are an extension of the Page Builder and EUIS components of DL Internet Connection. For an example of some of the common template elements, see the *IBM Digital Library Application Programming Guide and Reference for the Internet Connection, Chapter 4: The Page Builder*. The separation of the templates from Page

Builder and EUIS is intended to make it easier to create new Web Pages and to modify existing ones. Unlike standard cgi-bin programming, state information is not stored in hidden fields of the dynamically generated web page. Instead state information is maintained on the web server via the cache manager program. Unlike the DL Sample Application, the DLCT templates and the archive.html page have the index class hard coded into them. As mentioned earlier, this index class is "Sample Archive." It must be replaced with your chosen index class name in all templates and HTML. The FRNCLIET.INI file must be modified and "**SampleArchive**" must be replaced with your index class. The advanced search template attribute list boxes must be modified to contain the attributes you want to select for your index class. The hypertext that is displayed under the thumbnails on the results page is created within the **archiveresults** template. A variable called "**templateDocumentTitle**" is used to create it and the code must be updated to reflect the attribute values you wish to display. The code is shown in Figure 22.

```
##################################################
# Build the Template Title
##################################################
$template DocumentTitle=" ";
$artist=$CMBQResultDocumentAttributesDataG{"Artist"};
$title=$CMBQResultDocumentAttributesDataG{"Title"};

if ($artist ne " ") {$templateDocumentTitle .=$author.".";}
if ($title ne " ") {$templateDocumentTitle .=$title .".";}
if ($templateDocumentTitle eq " ") {
  $templateDocumentTitle.=$CMBQResultDocumentAttributesDataG{"Unique ID"};
}

    $result.=
"
<A HREF=".&defineURL("documentContent",$docID,"archivetext").">".
      $templateDocumentTitle."</A></TD>";
```

Figure 22: Building the Template Title

The code depicted in Figure 1, is also in **archivescreen**, **archivezoom** and **archivetext**. It is used to create the page headings.

The DLCT templates use the pseudo-filenames defined in Table 2. If you choose a

different name, the templates must be updated. For instance, if you choose a pseudo name of THUMBNAIL.JPG for your thumbnail images, you must replace THUMB.JPG in the **archiveresults** template with THUMBNAIL.JPG.

The GIF images used for the banner and the buttoms can easily be replaced by creating your own and inserting them in place of the existing ones.

To make major modifications to the templates or to add new ones, refer to the *IBM Digital Library Application Programming Guide and Reference for the Internet Connection.*(See Appendix A. for ordering information.)

Appendix A: Where to Find More Information

Order Number	Digital Library Publication
GC26-8621	Introducing the IBM Digital Library Solution
SC26-8623	Planning and installation Guide
SC26-8626	Managing Digital Library with System Administration and Database Utilities
SC26-8658	System Administration Procedures
SC26-8651	Application Programming Guide for Windows
SC26-8624	Application Programming Guide for OS/2
SC26-8654	Application Programming Guide for AIX and IRIX
SC26-8668	IBM Digital Library Application Programming Guide and Reference for the Internet Connection
SC26-8652	Application Programming Reference, Volume 1
SC26-8653	Application Programming Reference , Volume 2: Search Manager
SC26-8657	C++ Application Programming Guide and Reference
SC26-8625	Messages and Codes
GC26-8872	Multimedia Server for AIX

Appendix B: Loader Sample Input

B.1: Definition of IMAGES.LST file

The IMAGES.LST file contains one row for each scanned image that COMIT95 has processed. The number of characters per record is 251; 250 charcters plus LF/CR.

The information entered into the fields is assumed to be left justified. Spaces not needed to complete field entries are assumed to be filled with blank characters.

The columns of the IMAGES.LST file are:

1. USI: This column is 20 characters wide. It contains the original TIFF image name of the scanned image and must be unique to the repository. (eg. st001413.tif)
2. SC_NAME: 20 characters wide; The name of the scanned image minus the extension. (eg. st001413)
3. D1_NAME: 20 characters wide; The name of the first derivative image created from the scanned image which is the full size image. (eg. ft001413.JPG)
4. D1_FORMAT: 4 characters wide; The format of the first derivative image.
5. D1_SIZE: 8 characters wide; The size of the first derivative image in kilobytes.
6. D1_WID: 8 characters wide; The width of the first derivative image in pixels.
7. D1_HGT: 8 characters wide; The height of the first derivative image in pixels.
8. D1_UPDATE: 2 characters wide; The desired update status of the first derivative image.
9. D2_NAME: 20 characters wide; The name of the second derivative image created from the scanned image which is the zoom image. (eg. zt001413.JPG)
10. D2_FORMAT: 4 characters wide; The format of the second derivative image.
11. D2_SIZE: 8 characters wide; The size of the second derivative image in kilobytes.
12. D2_WID: 8 characters wide; The width of the second derivative image in pixels.
13. D2_HGT: 8 characters wide; The height of the second derivative image in pixels.
14. D2_UPDATE: 2 characters wide; The desired update status of the second derivative image.
15. D3_NAME: 20 characters wide; The name of the third derivative image created from the scanned image which is the screen image. (eg. st001413.JPG)
16. D3_FORMAT: 4 characters wide; The format of the third derivative image.
17. D3_SIZE: 8 characters wide; The size of the third derivative image in kilobytes.
18. D3_WID: 8 characters wide; The width of the third derivative image in pixels.
19. D3_HGT: 8 characters wide; The height of the third derivative image in pixels.
20. D3_UPDATE: 2 characters wide; The desired update status of the third derivative image.
21. D4_NAME: 20 characters wide; The name of the fourth derivative image created from the scanned image which is the thumbnail image. (eg. tt001413.JPG)
22. D4_FORMAT: 4 characters wide; The format of the fourth derivative image.
23. D4_SIZE: 8 characters wide; The size of the fourth derivative image in kilobytes.
24. D4_WID: 8 characters wide; The width of the fourth derivative image in pixels.
25. D4_HGT: 8 characters wide; The height of the fourth derivative image in pixels.
26. D4_UPDATE: 2 characters wide; The desired update status of the fourth derivative image.
27. CD_LABEL: 10 characters wide; The original CD disk label where this scan can be located.

The valid values of the format parameters (D1_FORMAT, D2_FORMAT, D3_FORMAT, and D4_FORMAT) are:

TIF, which indicates a TIFF image,
JPG, which indicated a JPEG-compressed image,
GIF, which indicates a GIF image.

The valid values of the update parameters (D1_UPDATE, D2_UPDATE, D3_UPDATE, and D4_UPDATE) are:

E, which indicates that this image should be the first entry of this derivative image in the repository.
R, which indicates that this image is a replacement for a copy of this derivative image in the repository.
I, which indicates that this image not be loaded as an entry of this derivative image which already exists in the repository.

B.2 IMAGES.LST File for Sample Data

```
sg000010.TIF        SG000010            fg000010.jpg        jpg
389     2131    1600    E zg000010.jpg        jpg 255     1400
1051    E sg000010.jpg        jpg 100     700     526     E
tg000010.jpg        jpg 12      200     150     E DDRIVE

sg000011.TIF        SG000011            fg000011.jpg        jpg
320     2131    1600    E zg000011.jpg        jpg 216     1400
1051    E sg000011.jpg        jpg 83      700     526     E
tg000011.jpg        jpg 11      200     150     E DDRIVE

sg000012.TIF        SG000012            fg000012.jpg        jpg
394     2131    1600    E zg000012.jpg        jpg 257     1400
1051    E sg000012.jpg        jpg 101     700     526     E
tg000012.jpg        jpg 12      200     150     E DDRIVE

sg000013.TIF        SG000013            fg000013.jpg        jpg
312     2290    1600    E zg000013.jpg        jpg 195     1400
978     E sg000013.jpg        jpg 75      700     489     E
tg000013.jpg        jpg 11      200     140     E DDRIVE

sg000014.TIF        SG000014            fg000014.jpg        jpg
354     2131    1600    E zg000014.jpg        jpg 235     1400
1051    E sg000014.jpg        jpg 91      700     526     E
tg000014.jpg        jpg 11      200     150     E DDRIVE

sg000015.TIF        SG000015            fg000015.jpg        jpg
315     2131    1600    E zg000015.jpg        jpg 216     1400
1051    E sg000015.jpg        jpg 82      700     526     E
tg000015.jpg        jpg 11      200     150     E DDRIVE

sg000016.TIF        SG000016            fg000016.jpg        jpg
406     2131    1600    E zg000016.jpg        jpg 264     1400
1051    E sg000016.jpg        jpg 105     700     526     E
tg000016.jpg        jpg 13      200     150     E DDRIVE

sg000017.TIF        SG000017            fg000017.jpg        jpg
```

```
407     2131    1600    E zg000017.jpg          jpg 264     1400
1051    E sg000017.jpg          jpg 104     700     526     E
tg000017.jpg        jpg 12      200     150     E DDRIVE

sg000018.TIF        SG000018            fg000018.jpg        jpg
179     1600    2440    E zg000018.jpg          jpg 107     918
1400    E sg000018.jpg          jpg 36      459     700     E
tg000018.jpg        jpg 6       131     200     E DDRIVE

sg000019.TIF        SG000019            fg000019.jpg        jpg
353     2352    1600    E zg000019.jpg          jpg 206     1400
952     E sg000019.jpg          jpg 81      700     476     E
tg000019.jpg        jpg 11      200     136     E DDRIVE
```

B.3 Annotation (Tab Delimited) File for Sample Data

```
"SG000010"   "Product Team"     "Pearce and Cheung"      "Tom -  artist of the team"
      "Taken with an Olympus Digital Camera, June 20 1997"     "Tom - always a
smile on his face from ear to ear"

"SG000011"   "Product Team"     "Pearce and Cheung"      "Tammy - made the team
laugh""Taken with an Olympus Digital Camera, June 20 1997"     "Tammy - the amazing
super mom of twin daughters and an artist son"

"SG000012"   "Product Team"     "Pearce and Cheung"      "Teri - paid all the bills"
      "Taken with an Olympus Digital Camera, June 20 1997"     "Teri - her mouth-
watering delicious Truffles gave the team that oomf"

"SG000013"   "Product Team"     "Pearce and Cheung"      "Tom, Denice and Sarah"
      "Taken with an Olympus Digital Camera, June 20 1997"     "Committed to 95"

"SG000014"   "Product Team"     "Pearce and Cheung"      "Sundeep, Tammy and Sarah"
      "Taken with an Olympus Digital Camera, June 20 1997"     "The Solution
Pioneers"

"SG000015"   "Product Team"     "Pearce and Cheung"      "Sundeep and Tammy"
      "Taken with an Olympus Digital Camera, June 20 1997"     "The technical duo -
Loaded with Data Modeling ideas"

"SG000016"   "Product Team"     "Pearce and Cheung"      "Denice - what a team
player!"     "Taken with an Olympus Digital Camera, June 20 1997"     "Denice -
adores cats, big and small"

"SG000017"   "Product Team"     "Pearce and Cheung"      "Sundeep - calm, cool and
collected"   "Taken with an Olympus Digital Camera, June 20 1997"     "Sundeep -
would rather be sailing"

"SG000018"   "Product Team"     "Pearce and Cheung"      "Sarah - the mediator"
      "Taken with an Olympus Digital Camera, June 20 1997"     "Sarah - kept the
team on track"

"SG000019"   "Product Team"     "Pearce and Cheung"      "Team that made it all
happen!"     "Taken with an Olympus Digital Camera, June 20 1997"     "Outside at
STL, the team breaks for a photo"
```

Appendix C: Sample COMIT95.INI File

C.1 Example of COMIT95.INI file:

```
[current]
option_set_list=list1

[option_set_lists]
list1=option_set_1,option_set_2,option_set_3,option_set_4

[option_set_1]
;  Creates Full Size JPEG
;  MODIFY (if needed):
;    1) compress_usage=perform&save,e:\imgprep\out
;    2) default_copyright="Copyright (C) 1997 by xxxxxxxxxx"
reduce_usage=omit
sharpen_usage=omit
rotate_usage=omit
cstransform_usage=omit
watermark_usage=omit
compress_usage=perform&save,e:\imgprep\out
output_format=jpeg
default_copyright="Copyright (C) 1997 by xxxxxxxxxx"
mono_jpeg_hdr=geny.hdr
color_jpeg_hdr=genyuv.hdr
mono_jpeg_qscale=100
color_jpeg_qscale=100,100,100

[option_set_2]
;  Creates Half-size JPEG with watermark
;  Reduce (2x) and compress image adding watermark
;  MODIFY (if needed):
;    1) output_size=1400,1400
;    2) watermark_usage=perform&tsave,e:\temp
;    3) watermark_img=e:\images\wm\wmmask.tif
;    4) compress_usage=perform&save,e:\imgprep\out
;    5) default_copyright="Copyright (C) 1997 by xxxxxxxxxx"
reduce_usage=perform&discard
output_size=1400,1400
sharpen_usage=omit
rotate_usage=omit
cstransform_usage=omit
watermark_usage=perform&tsave,e:\temp
watermark_img=e:\images\wm\wmmask.tif
mono_watermark_parms="/i 17 /g .4 /s .9 /w .01 /v .02"
color_watermark_parms="/i 12 /g .4 /s .9 /w .01 /v .02"
compress_usage=perform&save,e:\imgprep\out
output_format=jpeg
default_copyright="Copyright (C) 1997 by xxxxxxxxxx"
mono_jpeg_hdr=geny.hdr
color_jpeg_hdr=genyuv.hdr
mono_jpeg_qscale=100
color_jpeg_qscale=100,100,100

[option_set_3]
```

```
; Creates Zoom JPEG (watermarked)
; Reduce compressed image with watermark
; MODIFY (if needed):
;   1) source_dir=e:\temp
;   2) output_size=700,700
;   3) compress_usage=perform&save,e:\imgprep\out
;   4) default_copyright="Copyright (C) 1997 by xxxxxxxxxx"
source_dir=e:\temp
source_ext=wm2
reduce_usage=perform&discard
output_size=700,700
sharpen_usage=omit
rotate_usage=omit
cstransform_usage=omit
watermark_usage=omit
compress_usage=perform&save,e:\imgprep\out
output_format=jpeg
default_copyright="Copyright (C) 1997 by xxxxxxxxxx"
mono_jpeg_hdr=geny.hdr
color_jpeg_hdr=genyuv.hdr
mono_jpeg_qscale=100
color_jpeg_qscale=100,100,100

[option_set_4]
; Creates Thumbnail JPEG
; MODIFY if needed:
;   1) compress_usage=perform&save,e:\imgprep\out
;   2) default_copyright="Copyright (C) 1997 by xxxxxxxxxx"
reduce_usage=perform&discard
output_size=200,200
sharpen_usage=omit
rotate_usage=omit
cstransform_usage=omit
watermark_usage=omit
compress_usage=perform&save,e:\imgprep\out
output_format=jpeg
default_copyright="Copyright (C) 1997 by xxxxxxxxxx"
mono_jpeg_hdr=geny.hdr
color_jpeg_hdr=genyuv.hdr
mono_jpeg_qscale=100
color_jpeg_qscale=100,100,100
```

C.2 Description of Sample COMIT95.INI

The sample COMIT95.INI has 4 distinctive option sets - one for each derivative produced.

Option_set_1 produces a full-sized JPEG using the COMPRESS.EXE function of COMIT95 only. The compressed image is saved as a JPEG in the e:\out (input filename.JP1). Copyright information has been added and default quantization table scale factors are used (QY=QCb=QCr=100). It is setup to handle 8-bit gray or 24-bit color TIFF files as input.

Option_set_2 generates a zoomed, watermarked JPEG image using a combination of REDUCE.EXE, WMARK.EXE and COMPRESS.EXE. This zoomed image is to be 2X larger than the screen-sized image. The reduced, watermarked TIFF image is stored in a temporary drive\directory path (e:\temp\input filename.WM2) which is discarded after the next step (option_set_3). The final reduced, watermarked, compressed JPEG image is saved as a JPEG in e:\out. (input filename.JP2). Copyright information has been added and default quantization table scale factors are used (QY=QCb=QCr=100). This can handle 8-bit gray or 24-bit color TIFF files as input. The watermark mask is in e:\images\wm\wmlogo.tif. Note the watermark options of /i, /g, /s, /w and /v. These were the actual values that we used to produce the "Aspiring Artists" category in the sample data.

Option_set_3 creates a screen-sized, watermarked JPEG image using REDUCE.EXE and COMPRESS.EXE. Notice that we used as input, the reduced, watermarked TIFF image produced by option_set_2. Thus, in this step, we only needed to reduce and compress the image as the watermark had already been applied. The compressed image is saved as a JPEG (e:\out\input filename.JP3). Copyright information has been added and default quantization table scale factors are used (QY=QCb=QCr=100). It is setup to handle 8-bit gray or 24-bit color TIFF files as input.

Option_set_4 produces the fourth and final derivative - the thumbnail JPEG image. Using the original TIFF image as input, the image was reduced (REDUCE.EXE) and saved as a compressed (COMPRESS.EXE) JPEG (e:\out\input filename.JP4). Our sample thumbnails were created using an output size of 200 x 200.

C.3 Valid Descriptors for COMIT95.INI

The descriptors used within COMIT95.INI file are divided into 3 categories: 1) general 2) function_usage and 3) function specific. General descriptors are placed at the top of the option_set and are optional. Function_usage descriptors follow the general and tell COMIT95 what functions it is to perform and what to do with the output from each of the functions. The function specific descriptors are exactly that. They provide specific information to COMIT95 which is required to perform the various functions. If a function is to be performed, the function specific descriptors should also be specified.

General descriptors:

Descriptor	Description	Format
source_dir	Used to specify a directory for the image that is used as input to a sequence of operations. If this is used, should be the	**source_dir = d:\input - or - source_dir=d:**

	first line of the option_set_list.	
source_ext	Used to specify a file extension for the image that is used as input to a sequence of operations. If this is used, should be the second line of the option_set_list.	**source_ext=wm2**

Function_usage descriptors:

Descriptor	Description	Format
reduce_usage	Used to specify whether reduce function is to be performed or omitted, the output discarded, temporarily saved, or saved and if saved, in what directory. Default is perform&discard.	**reduce_usage=omit** **reduce_usage=perform&discard** **reduce_usage=perform&save,e:\out** **reduce_usage=perform&tsave,e:\out**
sharpen_usage	Used to specify whether sharpen function is to be performed or omitted, the output discarded, temporarily saved, or saved and if saved, in what directory. Default is perform&discard.	**sharpen_usage=omit** **sharpen_usage=perform&discard** **sharpen_usage=perform&save,e:\out** **sharpen_usage=perform&tsave,e:\out**
rotate_usage	Used to specify whether rotate function is to be performed or omitted, the output discarded, temporarily saved, or saved and if saved, in what directory. Default is perform&discard.	**rotate_usage=omit** **rotate_usage=perform&discard** **rotate_usage=perform&save,e:\out** **rotate_usage=perform&tsave,e:\out**
cstranform_usage	Used to specify whether colorspace transformation function is to be performed or omitted, the output discarded, temporarily saved, or saved and if saved, in what directory.	**cstransform_usage=omit** **cstransform_usage=perform&discard** **cstransform_usage=perform&save,e:\out** **cstransform_usage=perform&tsave,e:\out**

	Default is perform&discard.	
watermark_usage	Used to specify whether watermark function is to be performed or omitted, the output discarded, temporarily saved, or saved and if saved, in what directory.	**watermark_usage=omit** **watermark_usage=perform&discard** **watermark_usage=perform&save,e:\out** **watermark_usage=perform&tsave,e:\out**
	Default=perform&discard.	
compress_usage	Used to specify whether compress function is to be performed or omitted, the output discarded, temporarily saved, or saved and if saved, in what directory. If the output directory is not specified, output will be placed in the directory where COMIT95 resides	**compress_usage=omit** **compress_usage=perform&discard** **compress_usage=perform&save,e:\out** **compress_usage=perform&tsave,e:\out**
	Default is perform&save.	

Function specific descriptors:

Descriptor	Description	Format
For use with: Reduce		
output_size	Specifies the width in pixels and the height in pixels of the reduced image. If the output size is larger than the input image, no reduction is performed and the size of the image is unchanged. Default values are the width and height of the original image.	**output_size=1000,2000**

For use with: Sharpen

color_sharpen method	Determines if the color or monochrome sharpening algorithm should be applied to a color image. It is ignored for a monochrome image. Default is color	**color_sharpen_method=color** **- or -** **color_sharpen_method=mono**
sharpen	Specifies how much sharpening is to be performed. It is expressed in terms of the peak response of the sharpening filter. If sharpening is to be performed, the minimum allowed value is 0.5. For monochrome images, the maximum value is 9.5. (suggested values: 1.0 thru 1.5) For color images, the maximum value is 4.5 (suggested value 3.0). When the sharpening value is a negative number no sharpening is performed. Default = -1. .	**sharpen=1.0**

For use with: Rotate

rotate	Specifies a clockwise rotation of the image from its original orientation in the TIFF file. Values = yes, 0, 90, 180, 270. Value of **yes** indicates that rotation is to be done in accordance with the TIFF orientation tag. Values of 0, 90, 180 and 270 specify the number of degrees the image is to be rotated clockwise . Default = yes	**rotate=yes** **rotate=0** **rotate=90** **rotate=180** **rotate=270**

color_space	Specifies the name of the color space of the output image for the color space transformation step. Must be specified in the DISP.INI file. Default=smpte65	**color_space=smpte65**

For use with: Watermark

watermark_img	Names the grayscale image to be used to watermark the image. Sometimes referred to as the watermark mask. No default. If watermarking is to be performed, this must be supplied.	**watermark_img=e:\wmlogo.tif**
mono_watermark_parms	Parameters for the watermarking procedure for monochrome images. A string in double quotes specifies the desired flags just as they appear on the WMARK.EXE function as described in Section 4.6.	**mono_watermark_parms="/i 17 /g .4 /s .9"**
color_watermark_parms	Parameters for the watermarking procedure for color images. A string in double quotes specifies the desired flags just as they appear on the WMARK.EXE function as described in Section 4.6.	**color_watermark_parms="/i 12 /g .4 /s .9"**

For use with: Compression

output_format	Format of the output from the image compression process.	**output_format=jpeg** **output_format=gif** **output_format=tiffjpeg**

default copyright	Specifies a copyright notice to be placed in the JPEG compressed file if the input does not contain a Copyright tag. If the input does contain a Copyright tag, the information specified by the tag is stored in the compressed file. The string is enclosed in double quotes. This descriptor is ignored if the **output_format=gif** or **output_format= tiffjpeg**. The default is a null string.	**default_copyright= "Default text"**
override_copyright	If the override_copyright descriptor equals yes, the copyright information in the Copyright tag of the input file will be discarded. In this case the copyright information placed in the JPEG compressed file will be the string specified by the "default_copyright" descriptor. This descriptor is ignored if the **output_format=gif** or **output_format= tiffjpeg**. The default is no.	**override_copyright=yes** **-or-** **override copyright=no**
mono_jpeg_hdr	Name of the JPEG header file to be used to compress monochrome images. This descriptor is ignored if the **output_format=gif** or **output_format= tiffjpeg.**	**mono_jpeg_hdr=geny.hdr**
color_jpeg_hdr	Name of the JPEG header file to be used to compress color images. This descriptor is ignored if the **output_format=gif** or **output_format= tiffjpeg.**	**color_jpeg_hdr=genyuv.hdr**

mono_jpeg_qscale	Used to change the amount of distortion in the compressed image file by adjusting the scaling of the quantization matrices in the JPEG header. A value of 100 specifies that the tables in the header are to be used without alteration. A value of **N** specifies that each matrix element is to be multiplied by N/100. Smaller values give less distortion, but provide poorer compression. This descriptor is ignored if the **output_format=gif** or **output_format= tiffjpeg.** Default is 100.	**mono_jpeg_qscale=100**
color_jpeg_qscale	Used to change the amount of distortion in the compressed image file by adjusting the scaling of the quantization matrices in the JPEG header. A value of 100 specifies that the tables in the header are to be used without alteration. A value of **N** specifies that each matrix element is to be multiplied by N/100. Smaller values give less distortion, but provide poorer compression. This descriptor is ignored if the **output_format=gif** or **output_format= tiffjpeg.** Default is 100, 100, 100.	**color_jpeg_qscale=100,100,100** **-or-** **color_jpeg_qscale=100**

Implementation Guide

Appendix D: Image Preparation Command File

D.1 Example File IMGPREP.CMD

```
IMGPREP.CMD
* This REXX utility processes all TIFF files in a given directory (inpath)
  and stores the derivative files in (imgpath) in preparation
  for LOADING the Collection Treasury system. The preparation includes:
  1. reduction scaling
  2. sharpening
  3. rotation (if needed)
  4. color transform (if needed)
  5. add water mark
  6. JPEG compression
  7. rename of output files
  8. generate an images.lst file
*/
call RxFuncAdd 'SysFileTree','RexxUtil','SysFileTree'
arg help
/* Ensure "inpath" is correct */
inpath = 'e:\imgprep\in'
/* Ensure "imgpath" is correct */
imgpath = 'e:\imgprep\out'
if help = '?'
then do
say ' Batch utility to prepare images for Colection Treasury loading. '
say ' Usage: imgprep '
say ' Default SOURCE directory is: ' inpath
say ' Default OUTPUT directory is: ' imgpath
say ' Note: Check the comit95.ini file for default parameters. '
say '       The save directory in comit95.ini should be set to: ' imgpath
say '       Ensure the INIPATH is set correctly ie. x:\comit95 '
say '       where x:\comit95 contains the comit95.ini file. '
say '       '
```

(Continued on next page)

```
exit
end
ftemplate = inpath'\????????.tif'
rc = SysFileTree(ftemplate,files,FO)
"erase comit95.log"
"erase images.lst"

n=0
m=0
p=0
/* loop to process and generate derivative images */
do i=1 to files.0
 "comit95" files.i  ' /n >>comit95.log'
 if rc > 0
 then do
 say 'IMGPREP processing comit95 error on image:'files.i
 say 'continue processing next image.'
 n=n+1
 end
 if rc < 0
 then do
 say 'IMGPREP comit95 aborted due to error.'
 signal ENDLOOP
 end
 if rc = 0
 then do
 m=m+1
 /* rename the derivative images */
 say 'Renaming derivative files.'
 ftemp = imgpath'\*.JP1'
 "ren" ftemp ' F*.JPG'
 ftemp = imgpath'\*.JP2'
 "ren" ftemp ' Z*.JPG'
 ftemp = imgpath'\*.JP3'
 "ren" ftemp ' S*.JPG'
 ftemp = imgpath'\*.JP4'
 "ren" ftemp ' T*.JPG'
 end
 /* create and append to images.lst file */
 "imglsty" files.i  ' >>comit95.log'
 if rc > 0
```

```
 then do
 say 'IMGPREP processing imglst error on image:'files.i
 say 'continue processing next image.'
 p=p+1
 end
end

ENDLOOP:
say 'IMGPREP batch processing ended.'
say 'Number of images correctly processed:'m
if n > 0
then do
say 'Number of images that failed processing by comit95:'n
say 'Please check the file comit95.log for details of error.'
end
if p > 0
then do
say 'Number of images that failed processing by imglst:'p
say 'Please check the file comit95.log for details of error.'
end
```

D.2 Customizing IMGPREP.CMD

To customize IMGPREP.CMD, two changes must be made:

1. **INPATH** must be the correct drive\directory path where your scanned TIFF images are stored. These are the TIFF images that are ***input*** to the image preparation process. You can only specify one drive/directory per run.

2. **IMGPATH** must be the drive/directory path where you want to save your derivative images. These are your *output* derivative images that you will load into DLCT. As with INPATH, you can only specify one drive/directory per run.

INPATH statement format

inpath='drive/directory'

Example:

inpath = 'd:\scanimage=\in'

Note: The space before and after the "=" sign."

IMGPATH statement format:

imgpath = 'drive/directory'

Example:

imgpath = 'd:imageout'

Note: The space before and after the "=" sign**.**

Appendix E: Collection Treasury Application Load Time Configuration Planning Worksheets

E.1: Planning worksheets for the sample data using SampleArchive index class

Worksheet 1 of 3

Purpose:

(a) Create item with content parts

1. Index Class

Index Class Name: SampleArchive

Primary Key: UniqueID

2. Content Part

Number of Content Parts: 4

Part Ids and Part Classes:

Part ID	Part Class	Remarks
1	JPG	The Full Image.
2	JPG	The Zoom Image
3	JPG	The Screen Image
4	JPG	The Thumbnail Image

3. Options

Will use Free Text Search Engine ? Y

Create Free Text Part from Attribute Values ? Y

Create TOC entry in Notepart for DLIC ? Y

4. Free Text Search

Search Service Name: DLSAMPLE

Search Index Name: DLSEARCH

Content Class for Parts to be indexed: ASCII

5. End User Access using Internet Connection

Entries for the Table of Content (TOC) notepart

Part ID	TOC entry name
1	FULL.JPG
2	ZOOM.JPG
3	SCREEN.JPG
4	THUMB.JPG

6. Data Loading

(For Item Creation)

Input file name: aspiring.lst

File Format: Comit 95

Number of Columns: 27 (default)

Path to Content Files:_____

Mapping Information:

Field Number in Input File	Maps to Index Class Field or Content Part	Remarks
2	UniqueID	The Primary Key
3	part1	The file name for the 1st derivative
9	part2	The file name for the 2nd derivative
11	Zoom_Image_Size	size in (KB) of 2nd derivative
12	Zoom_Image_Width	width in pixels of 2nd derivative
13	Zoom_Image_Height	height in pixels of 2nd derivative
15	part3	The file name for the 3rd derivative
17	Screen_Image_Size	size in (KB) of 3rd derivative
18	Screen_Image_Width	width in pixels of 3rd derivative
19	Screen_Image_Height	height in pixels of 3rd derivative
21	part4	The file name for the 4th derivative
23	Thumb_Image_Size	size in (KB) of 4th derivative
24	Thumb_Image_Width	width in pixels of 4th derivative
25	Thumb_Image_Height	height in pixels of 4th derivative

File name to save this configuration: as_creat.ltc

Worksheet 2 of 3:

Purpose:

(b) Update item attributes (and/or the optional Free Text Part)

1. Index Class

Index Class Name: SampleArchive

Primary Key: UniqueID

2. Content Part

Number of Content Parts: 0

Part Ids and Part Classes: Not Applicable for this Load Time Configuration

3. Options

Will use Free Text Search Engine ? Y

Create Free Text Part from Attribute Values ? Y

Create TOC entry in notepart for DLIC ? N (Because already created)

4. Free Text Search

Search Service Name: DLSAMPLE

Search Index Name: DLSEARCH

Content Class for Parts to be indexed: ASCII

5. End User Access using Internet Connection

Not Applicable for this Load Time Configuration

6. Data Loading

(For Item Attribute Updates and Free Text Part Updates)

Input file name: aspiring.txt

File Format: Delimited

Number of Columns: 6

Path to Content Files:_____

Mapping Information:

Field Number in Input File	Maps to Index Class Field or Content Part	Remarks
1	UniqueID	Primary field to search on
2	Category	Update these 5 fields
3	Artist	
4	Title	
5	Provenance	
6	Description	

File name to save this configuration: as_updat.ltc

Worksheet 3 of 3:

Purpose:

(c) Delete item (and its content parts)

For Deleting Items, the same Load Time Configuration as that used to create the item can be used.

E.2: Blank Worksheets

Note: The number of rows required in the tables below are dependent upon the number of parts and the columns in the input files to the Loader. Depending on your particular configuration, you may need more or less rows in the tables.

Worksheet 1 of 3:

Purpose:

(a) Create item with content parts

1. Index Class

Index Class Name: _____

Primary Key: _____

2. Content Part

Number of Content Parts: __

Part Ids and Part Classes:

Part ID	**Part Class**	**Remarks**

3. Options

Will use Free Text Search Engine ? _

Create Free Text Part from Attribute Values ? _

Create TOC entry in Notepart for DLIC ? _

4. Free Text Search

Search Service Name: _____

Search Index Name: _____

Content Class for Parts to be indexed: _____

5. End User Access using Internet Connection

Entries for the Table of Content (TOC) notepart

Part ID	**TOC entry name**

6. Data Loading

(For Item Creation)

Input file name: _____

File Format: _____

Number of Columns: _____

Path to Content Files:_____

Mapping Information:

Field Number in Input File	Maps to Index Class Field or Content Part	Remarks

File name to save this configuration: _____

Worksheet 2 of 3:

Purpose:

(b) Update item attributes (and/or the optional Free Text Part)

1. Index Class

Index Class Name: _____

Primary Key: _____

2. Content Part

Number of Content Parts: _

Part Ids and Part Classes: Not Applicable for this Load Time Configuration

3. Options

Will use Free Text Search Engine ? _

Create Free Text Part from Attribute Values ? _

Create TOC entry in notepart for DLIC ? N (Because already created)

4. Free Text Search

Search Service Name: _____

Search Index Name: _____

Content Class for Parts to be indexed: _____

5. End User Access using Internet Connection

Not Applicable for this Load Time Configuration

6. Data Loading

(For Item Attribute Updates and Free Text Part Updates)

Input file name: _____

File Format: _____

Number of Columns: ___

Path to Content Files: _____

Mapping Information:

Field Number in Input File	Maps to Index Class Field or Content Part	Remarks

File name to save this configuration: _____

Worksheet 3 of 3:

Purpose:

(c) Delete item (and its content parts)

For Deleting Items, the same Load Time Configuration as that used to create the item can be used.

Appendix F: Error Codes

The Loader is a client of Digital Library, and so that for most part, the error codes and messages it displays are those returned by the components of the Digital Library, and are documented in the IBM Digital Library Messages and Codes, Version 1 (part number SC26-8625-00). The Loader puts out its own error codes, in the range 10000 to 10100, along with the Digital Library error codes, which are numbered to be less than 10000.

The explanation of these error codes are given below. The string in parenthesis following the error code is the internal representation of the error message.

10000: (kEuEcAddItemFailed)

Failed to add an item. Look at the Digital Library return codes for more information.

10001: (kEuEcAddItemFailedAddNotepartFailedDeleteItem)

During an 'Add Item', the Loader could not add the Note part, which is used for the Table of Contents (TOC) entry for the Digital Library Internet Connection (DLIC). The Loader then tried to delete this incomplete item, but was unable to do that also. Look at the Digital Library return codes for more information.

10002: (kEuEcAddItemFailedAddNotepartDeletedItem)

During an 'Add Item', the Loader could not add the Note part, which is used for the Table of Contents (TOC) entry for the Digital Library Internet Connection (DLIC) piece. The Loader then deleted the item it had created. Look at the Digital Library return codes for more information.

10003: (kEuEcAddItemFailedAddFreetextpartDeletedItem)

During an 'Add Item', the Loader could not add the Free Text part, which is built up of the attribute value pairs of the item. The Loader then deleted this incomplete item. Look at the Digital Library return codes for more information.

10004: (kEuEcAddItemFailedAddFreetextpartFailedDeleteItem)

During an 'Add Item', the Loader could not add the Free Text part, which is built up of the attribute value pairs of the item. The Loader then tried to delete this incomplete item, but was unable to do so. Look at the Digital Library return codes for more information.

10005: (kEuEcAddItemFailedDuplicateItem)

During an 'Add Item', the Loader could not create the item because an item with the same primary key value already exists.

10010: (kEuEcUpdateItemFailed)

During an “Update Item”, the Loader could not update the item. Look at the Digital Library return codes for more information.

10011: (kEuEcUpdateItemFailedGetAttributes)

During an “Update Item”, the Loader could not update the item because it could not obtain the attributes of the item. Look at the Digital Library return codes for more information.

10012: (kEuEcUpdateItemFailedDeleteFreetextpart)

During an “Update Item”, the Loader could not update the item because it could not delete the Free Text Part in order to update it. Look at the Digital Library return codes for more information.

10013: (kEuEcUpdateItemFailedAddFreetextpart)

During an “Update Item”, the Loader could not update the item, because it could not add the updated Free Text Part, although it deleted the existing Free Text Part. Look at the Digital Library return codes for more information.

10020: (kEuEcDeleteItemFailed)

During an “Delete Item” the Loader failed to delete the item. Look at the Digital Library return codes for more information.

10031: (kEuEcCheckIndexClassMismatchNAttrib)

The Loader checks the index class information saved in a Data Model against that available from the current Library Server and found a mismatch in the number of attributes. This would normally occur if the index class or it’s key fields were modified after this Data Model was created.

10032: (kEuEcCheckIndexClassAttribMismatchName)

The Loader checks the index class information saved in a Data Model against that available from the current Library Server and found a mismatch in the name of at least one attribute. This would normally occur if the index class or it’s key fields were modified after this Data Model was created.

10033: (kEuEcCheckIndexClassAttribMismatchType)

The Loader checks the index class information saved in a Data Model against that available from the current Library Server and found a mismatch in the Type of at least on attribute. This would normally occur if the index class or it’s key fields were modified after this Data Model was created.

10034: (kEuEcCheckIndexClassAttribMismatchId)

The Loader checks the index class information saved in a Data Model against that available from the current Library Server and found a mismatch in the class ID. This is not considered a fatal error. This would normally occur if the index class or its key fields were modified after this Data Model was created.

Appendix G: Microline Toolkit License Agreement and Warranty

Microline Component Toolkit 3.0™
License Agreement and Warranty

Microline Software, 1095 E. Duane Ave. Suite 207, Sunnyvale, CA 94086
PLEASE READ THIS LICENSE AGREEMENT BEFORE USING THIS SOFTWARE.

This license is an agreement between you and Microline Software.

1. License Grant
Subject to the terms set forth in this License, you may use the computer software (hereafter "Software") contained in the Microline Component Toolkit package. You may use the Software to create applications and applets for use and distribution free of charge.

By downloading, using or copying this Software, you agree to abide by the intellectual property laws, and all other applicable laws of the United States, and the terms of this License.

You may distribute this Software to a third party in unmodified form. You may not modify the Software and then distribute it to a third party. The Software contains code that limits its functionality. You may not enable the restricted functionality without permission from Microline Software. You may not attempt to decompile, disassemble, or reverse engineer the Software.

If the package this license is contained in does not also contain a separate Key Class license, there is a Professional Development Kit for the Microline Component Toolkit provided separately by Microline Software for a fee which enables expanded functionality in the Software. You may not attempt to remove the restrictions within the Software to obtain the functionality of the Professional Development Kit without obtaining the Key Class license from Microline Software.

2. Copyright and Title
The Software manuals and other documentation and its copyrights are owned by Microline Software and are protected by United States copyright laws and international treaty provisions. The Software may not be copied or distributed without this license agreement and copyright notice included on all copies in a manner which makes the agreement and notice available to users prior to use. You may not copy or distribute the Software after receiving a notice from Microline Software terminating your distribution privileges.

3. Disclaimer of Warranty
THIS SOFTWARE AND REFERENCE MATERIALS ARE PROVIDED AS IS, WITHOUT WARRANTY AS TO THEIR PERFORMANCE, MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR AGAINST INFRINGEMENT. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THIS PROGRAM IS ASSUMED BY YOU.

MICROLINE SOFTWARE SHALL NOT BE LIABLE FOR INDIRECT, SPECIAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM THE USE OF THIS PRODUCT. SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATIONS MIGHT NOT APPLY TO YOU.

Microline Software shall have no liability or responsibility for Software altered, modified, or converted by you or a third party, damages resulting from accident, abuse, or misapplication, or for problems due to the malfunction of your equipment or software not supplied by Microline Software.

4. Termination
This License is in effect until terminated and terminates without notice from Microline Software if you fail to comply with any of its provisions. Upon termination you shall destroy the Software and all copies or portions thereof.

5. U.S. Government Restricted Rights
This Software and documentation are provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1) of the Rights in Technical Data and Computer Software Clause at DFARS 252 227-7013 or subparagraphs (c)(1)(ii) and (2) of Commercial Computer Software – Restricted Rights at 48 CFR 52 227-19, as applicable, supplier is Microline Software, 1095 E. Duane Ave. Suite 207, Sunnyvale CA 94086

This License represents the entire statement of the understandings and agreements among the parties.