User's Guide

Import Image 2 Lotus Notes

1.7.0

Robert Ibsen Voith/Voith's CODE

Printed on 31 March, 2010

Welcome to Import Image 2 Lotus Notes

This is the User's Guide for Import Image 2 Lotus Notes (abbreviated *II2LN*) and its main purpose is to unveil the possibilities in II2LN and to enable you to make the most out of the product.

Copyright © 2002-2010: Voith's CODE
All rights reserved

Due to continued product development this information may change without notice. The information and intellectual property contained herein is confidential between Voith's CODE. and the client and remains the exclusive property of Voith's CODE. If you find any problems in the documentation, please report them to us in writing. Voith's CODE does not warrant that this document is error-free.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise without the prior written permission of Voith's CODE.

Microsoft Word, Microsoft Office, Windows®, Window 95™, Window 98™, Windows NT®, Windows 2000, Windows XP and MS-DOS™ are trademarks of the Microsoft Corporation.

Xalan, Xerces are trademarks of the Apache.org organization.

Voith's CODE
Granittveien 13
N-3140 Borgheim
Norway
+47 33 38 77 01
Internet E-Mail: support@vcode.no
Website: http://www.vcode.no

Contents

CHAPTER 1

Introduction

In This Chapter

What is Import Image 2 Lotus Notes ?

II2LN is a programmers tool, aimed at application developers primarily seeking to manipulate and import images into Lotus Notes. While it has a separate Command Line Interface, allowing you to use II2LN without Lotus Notes, it should *not* be regarded as a self-contained image product such as Photoshop or Paint Shop Pro.

It is shipped as a single Dynamic Link Library (VCII2LN.DLL) with four appurtenant run time files;

§ DXLTOOLS10.DLL - The run time DLL for the Lotus XML Toolkit

§ LCPPN23.DLL - The run time DLL for the Lotus Notes C++ API

§ POLYIMAGEPRO.DLL - The image library used to tag images with IPTC and EXIF information

§ HTMLSnap2.DLL - The library used to create web site snapshots

II2LN is mostly intended to be accessed via LotusScript. This documentation will therefore contain LotusScript examples.

The main purpose

The main purpose of Import Image 2 Lotus Notes:

§ Import any kind of image, including web snapshots, without any loss in quality.

§ Import the image into rich text fields or use variables!

§ Store image variables in fields

§ Manipulate the image with typical batch-like operations such as resizing

§ Extract-, manipulate and write information stored within the image, such as ***EXIF*** http://www.exif.org/ and ***IPTC*** http://www.iptc.org.

§ Add your own variables

§ Superimpose your own text such as your company name onto the image

§ Superimpose your own overlay image such as your company logo onto the image.

§ Import web site snapshots or any URL as an image

§ Run your image through HTML templates for advanced superimposing.

§ Add photo frames or borders to your image

§ Use II2LN without Lotus Notes - directly from its Command Line Interface!

Import any kind of image

II2LN support the following image formats:

§ JPG
§ GIF
§ BMP
§ TIFF
§ PhotoCD
§ Photoshop
§ WBMP
§ PNG
§ PCX
§ PAX
§ TLA
§ WMF
§ EMF
§ APM
§ TGA

The image quality is not reduced during import!

Additionally, II2LN can with the support of ***Ghostscript*** (see "Support for EPS, PS and PDF files - Ghostscript Support" on page 170) import and manipulate the following formats;

§ EPS
§ EMF
§ PS
§ PDF

Import to rich text fields or use variables

II2LN can import an image into a rich text field in two different ways;

1. The image can be imported into a sole rich text field. For example, you can specify that you want an image to be imported to the rich text field *Image*. The field Image will after the import contain your image as a full fidelity Notes Embedded Image.
2. The image can replace a specified variable. If your rich text field already contain data, you can have II2LN search for a variable and replace the variable with the imported image. This powerful feature let you create and work with *templates* of a document, and have II2LN replace only the variable at runtime. You use the import option ***ReplaceWithVariable*** (on page 109) to control this feature.

   See more about variables ***here*** (see "Variables" on page 153).

Store image variables in fields

Wien II2LN work with an image, it retrieves many different variables, such as ***The original image variables*** (on page 154), ***The import image variables*** (on page 155), The EXIF variables (on page 156) and ***The IPTC variables*** (on page 157). All these variables can be stored in your Notes document along with the image! You have total control on whether to store all information in one field, or rather use a separate field per variable. You can also control which field name prefix you want to use, so II2LN better adhere to your coding standards!

Manipulate the image

II2LN can manipulate the image with batch-line operations. The operations are:

- Resize. Make an image smaller or larger
- Rotate. Rotate the image in any angle
- Sharpen. Sharpen the image
- Blur. Blur the image
- Flip. Flip the image either horizontally or vertically
- Ensure the image fits within an canvas format of for example 4:3 or 16:9

Extract EXIF and IPTC information

Several image formats such as JPG, TIFF and PNG may contain extra information stored directly in the image.

For example will many digital cameras inject technical information such as which camera took the picture, the date it was taken, what aperture and shutter speed that was used and whether the flash was used or not. This information is often stored in a format known as ***EXIF*** http://www.exif.org/ information.

Similar to embedded EXIF information images may contain embedded ***IPTC*** http://www.iptc.org information. IPTC is more focused on what the image contain, such as caption, headline and keywords. IPTC is widely used in news agencies, news papers and television.

Manipulate EXIF and IPTC information

Not only can II2LN *extract* EXIF and IPTC information from your images, it can also *write* EXIF and IPTC information to your images! Via the powerful import options ***ReplaceEXIF*** (on page 107), ***ReplaceITPC*** (see "ReplaceIPTC" on page 108), ***AppendIPTC*** (on page 69) and ***InsertIPTC*** (on page 88) can you set almost all EXIF and IPTC values.

II2LN will automatically update some of the values for you too, if you for example has use the ***AutoOrientation*** (on page 73) import option, II2LN will ensure that the EXIF value for Orientation is correctly set.

II2LN will separate between three distinct operations on EXIF or IPTC values:

§ Replace (ReplaceEXIF and ReplaceIPTC)

  When you replace a value you will completely remove any potential existing value for the specified tag. In other words, the old values are lost.

§ Append (AppendIPTC)

  When you *append* a value, you will append (- or suffix) the value to any existing value. In other words, the existing value will be *first*, and the new value *after*.

  If the tag doesn't exist or is empty, the Append operation will function as replace, and set the value for the tag

§ Insert (InsertIPT)

  When you insert a value, you will inject the value in front of any existing values. In other words, the existing values will be last.

  If the tag doesn't exist or is empty, the Insert operation will function as replace, and set the value for the tag

This gives you pretty good control over where to put your new values.

Remember that you can also clear any existing meta data (both EXIF and IPTC) by using the ***ClearExistingMetadata*** (on page 76) import option before any Replace/Append/Insert options.

Import web site snapshots or any URL as an image

Starting from version 1.4 of II2LN, you can import a snapshot of a web page just as easy as you can import and manipulate real images!

What is a snapshot? Very briefly it is an image of the web site at the time of the capture.

In fact, any place where you normally would specify an image name, such as in the functions ***ImportImage,*** (see "ImportImage" on page 35) ***ImportImageAsResource*** (on page 38) or ***AttachImage,*** (see "AttachImage" on page 42) you can now use an URL instead! Any URL staring with the prefix `http://` or `file://` will do. You can also use URLs in import options such as ***OverlayImage.*** (see "OverlayImage" on page 91)

The downsize is that the import operation take a bit longer, since we have to connect to the internet and retrieve the web pages. However, you have many new import options to let you control the web grabbing process.

The Command Line Interface

Prior to release 1.7 of II2LN, Even though you could use many of II2LN-features from scriptable tools like Visual Basic, II2lN was a mostly a tightly Lotus Notes-coubled tool, all controlled via LotusScript.

In release 1.7 a command line interface is introduced, meaning that you can perform many of the II2LN operations without using Lotus Notes at all.

In addition to being able to process images much in the same way as the ***ProcessImage*** (on page 60)-function, II2LN.EXE is controlled via a set of parameters and add some cool benefits to II2LN.

The general syntax is;

II2LN.EXE <Source file name> [Switches]

The only mandatory parameter is the *Source file name*. However, you probably want to do something with your image!

Param	Description
Source File Name	The full path and file name of the file to process - or - a DOS wildcard specifier to select more images in one go! If your file name contains spaces, please enclose the whole file name in apostrophes.

Switches	Switches controlling what to do with the Command Line Interface. If not specified, II2LN.EXE will only convert your image! Note that if the values for the switches contains spaces, you must enclose the whole switch and value in quotes! `"-importoptions:UseHTMLTemplate:Text Bottom"`

Source File Name

A Source File Name for the command line interface is either;

1 A file name to a single image file. If you run II2LN.EXE from within the same directory as your images, you do not need to use full path and file name. Just file names will suffice. Otherwise you may specify full filenames. Remember to replace spaces with +-signs!

2 A DOS Wildcard character to imply that you want to process several files in one go. For example will the wildcard *.jpg grab all JPG files within the directory, but leave PNG and BMP files behind. If you want to include subdirectories, the use the switch `-recursive`.

3 A single string of space separated file names, such as *"File1.jpg File2.Jpg File3.Jpg"*. Note that each file name is space-seprated! This is how tools like Directory Opus transfer several file names to II2LN, so we for example can rotate multiple images in one go. Note that such a file list must be used together with the switch `-filelist`.

If your file name contains spaces, please enclose the whole file name in apostrophes.

A note about target file names and target paths

Attention! II2LN.EXE may easily destroy your original image!!!! If you don't specify use the switch `-target` to specify a specific target file or the siwtch `-targetpath` to specify a target directory, II2LN.EXE will update the very same image as you have specified as source image!

Some times this is what you want, when you for example rotate your images from within File Explorer. When you produce images for your new photoframe, this is probably *not* what you want!

If your file name contains spaces, please enclose the whole file name in apostrophes.

Finally, remember that both `-targetpath` and `-target` switches can contain image variables. For example will the following command line;

```
II2LN.EXE "C:\Temp\Test\*.JPG" "-
targetpath:C:\Temp\Test\Output" "-
target:$(ORIGINAL_IMG_SORT_TIMESTAMP) -
$(ORIGINAL_IMG_FILENAMEONLY).jpg" -slfn "-
importoptions:resize:800,600;CANVASFORMATWITHCOLO
R:4,3,000000;photoframe:C:\Temp\Test\Tape In the
Corners - 800x600.png;NOCLIPRECTONHTMLTEMPLATE:1"
```

... first and foremost process all JPG files in the C:\Temp\Test directory. Then the -targetpath specifies that all processed files should be placed into the C:\Temp\Test\Output-directory. Note that if you don't specify a -target-switch now, the processed file names will match the original file names. However, the -target switch is specified above, and it looks like this;

```
"-target:$(ORIGINAL_IMG_SORT_TIMESTAMP) -
$(ORIGINAL_IMG_FILENAMEONLY).jpg"
```

First and foremost, see how the whole switch and switch-values are placed within quotes. Second, the target switch uses two image variables which will be filled in for each processed image. The first one, the `$(ORIGINAL_IMG_SORT_TIMESTAMP)` will contain the file timestamp on format YYYYMMDDHHMMSS, which is super for sorting. This import variable is kind of special, since it will hold the EXIF Date Time if that is present, otherwise the file's creation timestamp. Finally the the `$(ORIGINAL_IMG_FILENAMEONLY)` will get the file name only. Since we have specified the -slfn (SkipLeadingFileNumbers), the file name won't contain the prefix file number any more either.

Switches

II2LN.EXE uses switches to make it easy and quick to use II2LN. You can easily turn options on or off by using switches. A switch may have the format of a flag, just to turn something on or off, such as `-log` or `-recursive`. A switch may also be followed by a value, such as `-loglevel:4`, where the value is placed immediately after the flag, separated by a colon.

If your switch-value contain spaces, such as often is the case when you specify a file name or path, you must enclose the whole switch *and* value in quotes. For example if you want to specify the value Transparent Text Bottom to the switch -usehtmltemplate, it should look like this;

"-usehtmltemplate:Transparent Text Bottom"

As you see from the samples above, a switch is always prefixed with a minus-sign (-) and often specified in lower-case.

Below is a table of the available switches;

Switch	Description
-target	The file name of the processed file. Note that this can be a full path and file name cuch as "`C:\Temp Files\New File.jpg`," or just a file name like "`New File.jpg`". This is nice if you process a single file and want to redirect it to another file. Also note that using II2LN with just a Source File Name and this switch, will make II2LN.exe convert the image to the new file type denoted by the file extension of your target file.
-targetpath	A full path to a target directory. This switch can be used with both single file operations and multiple file operations. It will simply tell II2LN to store the resulting file(s) in this directory.
-filelist	This switch is used to denote that the Source File Name is actually a list of space separated file names.
-recursive	If you have specified DOS Wildcard characters in the Source File Name, then you may process multiple files in one go. Normally II2LN.EXE only process files within the current directory. By using this switch, you tell II2LN.EXE to include any other files matching the wildcard-specification in any sub directories too.

-importoptions	Specify a semi-colon separated ***Import Option*** (see "Import Options" on page 18) list, just like for any other DLL functions. Note that II2LN.EXE uses ***ProcessImage*** (on page 60) behind the scenes, so the same limitations for certain import options apply.
-displaylog	After processing an image, you may see the content of the log to your console. The log will contain the same data as described in ***Logging*** (on page 184). Note that you can use import options like ***SetLogLevel*** (on page 121) together with the switch `-importoptions` or use the switch `-loglevel`
-leavelog	II2LN extracts the II2LN to a local file in your private temporary files directory. Normally it will delete the log file upon completion. You can with this switch tell II2LN.EXE to leave it,
-loglevel:<level>	Set the log level with a number from 0 to 5, where 3 is normal. 0 means that just errors will be displayed, while 5 means that you will see debugging messages too. Note that you can achieve the exact same thing using the SetLogLevel import option with the switch *-importoptions*.
-convert:<format>	Convert the source file to the target file and convert the image to the format specified by the <format> number. See ***here*** (see "The image format codes" on page 57) for valid codes. If you specify 99, II2LN will use the file extension of the target file as guide for what format to convert to. Note that this switch uses the ***ConvertImage*** (on page 46)-function within the DLL, so you can do the exact same thing with LotiusScript!
-copy	Copy the source file to the target file. Note that if a target file already exist with the same same, it will be overwritten. Note that this switch uses the ***CopyII2LNImage*** (on page 34)-function within the DLL, so you can do the exact same thing with LotiusScript!
-move	Move the source file to the target file. Note that if a target file already exist with the same same, it will be overwritten Note that this switch uses the ***MoveII2LNImage*** (on page 59)-function within the DLL, so you can do the exact same thing with LotiusScript!

-rotateleft -rl	Rotate the image 90 degrees to the left (counter clockwise). At the same time, any existing EXIF Orientation tag will be normalized so the image won't be turned around again by most other image tools. Note that this switch uses the ***RotateII2LNImage*** (on page 62)-function within the DLL, so you can do the exact same thing with LotiusScript! Finally note that the RotateII2LNImage function used super quick rotation algorithms which is faster that using the ***Rotate*** (on page 115) import option
-rotateright -rr	Similar to -rotateleft, but now the image is rotated 90 degrees to the right (clockwise)
-rotateexif -re	If the image contains EXIF Orientation tag, and that you are pretty certain that this information is correct, you can use this switch to auto-orientate images according to the EXIF Orientation tag value.
-usehtmltemplate or -uht	This switch will use the ***HTML Template*** (see "UseHTMLTemplate" on page 136) feature within the II2LN. You must specify the HTML Template sub directory name behind the colon, such as; `"-uht:Transparent Text Bottom"` Again, you can do the same, by specifying UseHTMLTemplate-import option together with the `-importoptions` switch
-pause	If you launch II2LN.exe from tools like File Explorer, the console may flash some messages and then disappear too fast for your to read. By using this switch, you will pause II2LN.exe after execution.
-skipleadningfilenumbers -slfn	Many times your original image files has some sort of sequence number in from of the file name, such as; 001 - This is the first file.jpg 023 - This is the 23rd file.png By using this switch, you will tell II2LN to remove the file number for the image variable $(ORIGINAL_IMG_FILENAMEONLY). This can be handy if you just want the bare-bone text as a variable. Note that you will get an additional variable, named $(ORIGINAL_IMG_FILENUMBERONLY)

Some Cool Uses of the Command Line Interface

This topic will share some cool uses of the command line interface

Rename a bunch of files according to EXIF data

When you retrieve your images from your digital camera, you often get files with some sort of prefix followed by a sequence number. Examples can be DSC102293.JPG and RIM45664.JPG. Of course you can sort your images in your directory according to creation- or last modified date. But what if you want to have the snapshot date as part of your file name?

This is an easy task for II2LN. Below you see a sample on how to achieve this;

```
ii2ln "c:\Photos\*.jpg" "-
targetpath:C:\Temp\NextBase" "-
target:$(ORIGINAL_IMG_SORT_TIMESTAMP)-
$(ORIGINAL_IMG_FILENAMEONLY).jpg" -copy
```

Since the command line interface can process multiple files if you use DOS wildcard characters (such as c:\Photos\`*.jpg in the sample above`), you can use the built-in feature of variable replacement to rename your files according to the imbedded EXIF information. Above you first and foremost see the `-targetpath` and `-target` switches. The targetpath switch tells II2LN to put all processed files in the `C:\Temp\NextBase`-directory, and the target-switch contains two variables that will be replaced at runtime. The first variable contains the EXIF Date Time on the YYYYMMDDHHMMSS-format (which is super for sorting) and the next variable contains the file name only.

Use II2LN to rotate your images directly from within Directory Opus.

Even though many cool tools exist for organizing your images, I have always found most of them missing some features. First of all, I like to use the file explorer called ***Directory Opus*** (http://www.gpsoft.com.au/). Right after I have downloaded images from my camera, I can now in the Images-view (similar to the Thumbnail view in standard explorer), quickly rotate the images to the left or right. Albeit many other tools also do this, they seldom normalize the EXIF Orientation tag too. This is important if you for example want to show your images on a Windows Media Center (WMC). The WMC will typically try to be very clever and auto-orient your images according to the EXIF orientation value. If this value hasn't been set properly by your image software, it will falsely rotate once more!!

Below you see the command line I use to rotate all selected images to the left

```
C:\Program Files (x86)\IBM\Lotus\Notes\II2LN.exe
"{allfileshort$}" -filelist -rl
```

And below the same for the right-rotation;

```
C:\Program Files (x86)\IBM\Lotus\Notes\II2LN.exe
"{allfileshort$}" -filelist -rl
```

These commands are all tied to the keys CTRL + ALT + LEFT ARROW and CTRL + ALT + RIGHT ARROW respectively.

Quickly Produce Images For Your Photoframe

I have a photoframe (a NextBase PhotoM@ail X25 by the way) which I like to display my photos on. Since most photoframes doesn't have loads of memory, it is fairly important that the images aren't too large, especially since the photoframes has to resize them after all. My goal was therefore to quickly convert all images placed in one folder into another folder. The processing includes;

1. Convert unsupported image types into standard JPG
2. Resize all images to optimal size of the photoframe (the NextBase has 800x600). No need to have wasted mega pixels on the photoframe!
3. Auto caption all images with the name of the file in a nice, semi-transparent box at the bottom. Include the EXIT Date Time tag too if found! This takes the fresh ***UseHTMLTemplate*** (on page 136) import option into play.

By simply having the following command in a batch file, this task is now a breeze;

```
ii2ln c:\Photos\*.* -targetpath:C:\Temp\NextBase
"-uht:Transparent Text Bottom" -
importoptions:Resize:800,600
```

Why not use some really cool photoframes too, by using the fresh ***PhotoFrame*** (on page 102) import option!

Mass-convert a bunch of files from one format to another

By using the `-convert` switch, you can easily force II2LN to convert for example JPG files to PNG files. Below you see how to do that;

```
ii2ln "c:\Photos\*.jpg" "-
targetpath:C:\Temp\NextBase" "-
target:$(ORIGINAL_IMG_SORT_TIMESTAMP)-
$(ORIGINAL_IMG_FILENAMEONLY).png" -convert:6
```

The `-convert:6` means that II2LN will convert the images to 24-bits PNG images. Take a look ***here*** (see "The image format codes" on page 57) for valid codes!

By the way, if you specify format 99 (like -convert:99) you will instruct II2LN to save the images according to the file extension of the target file!

The challenges with standard Notes

Importing images into Lotus Notes has been possible since Notes R3. Unfortunately the earlier versions have always had some severe limitations, such as:

§ You were limited to use the only the client and manually import images.

  This got better with R4 where the @FileImport-function was introduced

§ You were limited to use 256 color palettes.

  R3 and R4 had did not save the original imported image, but converted it to a special Notes bitmap with a specific color palette. This often distorted the image.

§ You were limited to use GIF, JPG or BMPs, and non-compressed TIFF 5.0

  R5 could finally retrain the original format of the image, but only if the image was GIF or JPG. All other formats are converted to Notes' own bitmap format (which is a type of TIFF encoding). However, the special Notes palette isn't used anymore, which was a great improvement

§ The C++ API Import method would only allow 256 colors.

  Starting from R5 the Notes C++ API got the method Import, but it was limited to the Notes palette!

§ Domino XML (DXL) image format was unofficial.

  The DXL support first arrived with R5 and enabled the users to export and import data with a Domino variant of XML, DXL. Unfortunately it was still kind of hard to work with images, since it involves encoding and decoding of Base64 streams.

The overall design

II2LN is declared as external functions in LotusScript. The operation of the ***functions*** (see "Function groups" on page 17) is controlled with the so-called ***Import Options*** (on page 18). The Import Options is a set of variables controlling how the functions should process the images or URLs.

Further, the functions operate on *existing documents*. This is extremely important to understand. II2LN can only import or attach images to documents that already exist in the Notes database. The reason for this is that the II2LN operate strictly in the so-called back-end context, making it possible to run II2LN on a server. Besides, Lotus Notes has always had serious limitations regarding updates of the rich text fields in the front-end context, and changes tools such as II2LN make, won't be visible until the document has been closed and re-opened.

Below you see a small sample on how II2LN is declared and called in LotusScript:

The following LotusScript code declare the external function *ImportImage* in the VCII2LN.DLL. The code is typically placed in the (Declarations) part:

```
Declare Function ImportImage Lib "VCII2LN.DLL" (_
Byval pstrNotesServer As String, _
Byval pstrNotesDatabase As String, _
Byval pstrNotesUNID As String, _
Byval pstrNotesField As String, _
Byval pstrFilename As String, _
Byval pstrImportOptions As String) As Long
```

The LotusScript code above implies that the VCII2LN.DLL is placed in a directory specified in your PATH environment variable. Please refer to the Designer Help in Lotus Notes to see how to fully use the Declare-statement in order to specify alternative places for the DLL.

When the external function is specified, it can be called just like any other LotusScript function. Below you see the function in action:

```
Sub Initialize
     Dim lRc As Long
     lRc = ImportImage(_
      "YourServerName",_
      "DatabaseName.nsf",_
      "UNID", _ ' The UNID
      "Body",_ ' The field name
      "Filename.jpg",_
      "Sharpen:50;Resize:200,200")
End Sub
```

The sample above will import the image `c:\temp\smile.jpg` into the field *Body* in the document specified by the UNID in the specified database.

The import options specify that the image will be sharpened and resized to 200 by 200 pixels!

If ImportImage succeeds, it will return 0. All other return codes indicate an error situation.

Function groups

II2LN is based on a set of available functions, that can be accessed via LotusScript. The functions are grouped together as this;

§ Identification functions - These functions let you identify the installed version of II2LN. This makes it easier for you to ensure that your code runs against a known version of II2LN. All these functions are based on the version number of II2LN.

  The functions are:

  § GetII2LNMajorVersion

  § GetII2LNMinorVersion

  § GetII2LNBuildNumber

  § GetII2LNVersion

§ Import functions - These functions actually import the image into Lotus Notes. You can control much of the import processing with the Import Options. The functions are:

  § ImportImage - Import the image as Notes Embedded Image.

  § ImportImageAsResource - Import the image as a Image Resource.

- **Attachment functions** - These functions will attach the image to Lotus Notes-documents much like the built-in attachment logic. However, the images may be processed the same way as when importing images. The functions are:
    - AttachImage - Attach the image with Import Options
- **Miscellaneous functions** - Functions not directly related to the groups above;
    - GuessImageFormat - Analyze the specified image to detect what image format it is.

Import Options

The Import Options specify how the images or URLs should be processed. Basically the Import Options is a *semi-colon* separated string with *commands* and *values*.

The command and values are separated by a *colon*, and if the command have more than one value, the individual values are separated by *commas*.

If you need to transport the separators such as semi-colon, colon or comma, you do this by encoding the values to `&semicolon;`, `&colon;` or `&comma;`

Below you see a full blown Import Options string:

"SetLogFilename:c:\temp\vcii2ln.log;SetLogLevel:5;ProcessTimeField:VCII2LN_ProcTime;Resize:640,480;AutoOrientation:1"

Without going into the extact details of each command, the string above specify the follwing:

1. Set the log filename to `c:\temp\vcii2ln.log`.
2. Set the log level to 5
3. Inject a field with the name VCII2LN_ProcTime into the Notes document containing the processing time of II2LN.
4. Resize the image to 640 by 480.
5. If the image contains orientation information, II2LN can automatically rotate the image into correct orientation

When you see such a string, look out for the semi-colons. They separate each command and values pair. In the sample string above we find the commands; SetLogFilename, SetLogLevel, ProcessTimeField, Resize and AutoOrientation. Only the Resize-command has more than one parameter, and the values are separated by a comma.

See ***this*** (see "All Import Options" on page 65) chapter for a full description of each of the import options.

Note that multiple import options can be repeated during the same call. This makes it possible to for example overlay text on more places in one go!

Replacable variables

II2LN extract data from the imported image. By default will II2LN extract information such as image type, resolution and dimensions. Additionally you can instruct II2LN to extract ***EXIF*** (see "The most common EXIF tags" on page 172) and ***IPTC*** (see "The most common IPTC tags" on page 180) values by specifying the import options ***ExtractEXIFToField*** (on page 77) and ***ExtractIPTCToField.*** (see "ExtractIPTCToField" on page 81)

All the extracted meta data is made available as variables with appurtenant values. You can then replace any of these variables with their real run time values at import- or attachment time.

A variable as just a string on a special format, like this; `$(Variablename)`.

See the ***Variables*** (on page 153) chapter for more information about available variables.

Starting from release 1.7 you can also add your own variables into the processing , by using the repeatable ***AddVariable*** (on page 67) import option.

What benefits do you get by registering ?

An unregistered version of II2LN will overlay the logo of Voith's CODE on the imported- or attached images. Registering the product will remove this watermark.

Finally, you will of course get prioritized support if you have any questions regarding II2LN or any other Voith's CODE-product.

Typographical Conventions

Before you start using this guide, it is important to understand the terms and typographical conventions used in the documentation.

For more information on specialized terms used in the documentation, see the Glossary at the end of this document.

The following kinds of formatting in the text identify special information.

Formatting convention	Type of Information
Triangular Bullet(Ø)	Step-by-step procedures. You can follow these instructions to complete a specific task.
Special Bold	Items you must select, such as menu options, command buttons, or items in a list.
Emphasis	Use to emphasize the importance of a point or for variable expressions such as parameters.
CAPITALS	Names of keys on the keyboard. for example, SHIFT, CTRL, or ALT.
KEY+KEY	Key combinations for which the user must press and hold down one key and then press another, for example, CTRL+P, or ALT+F4.

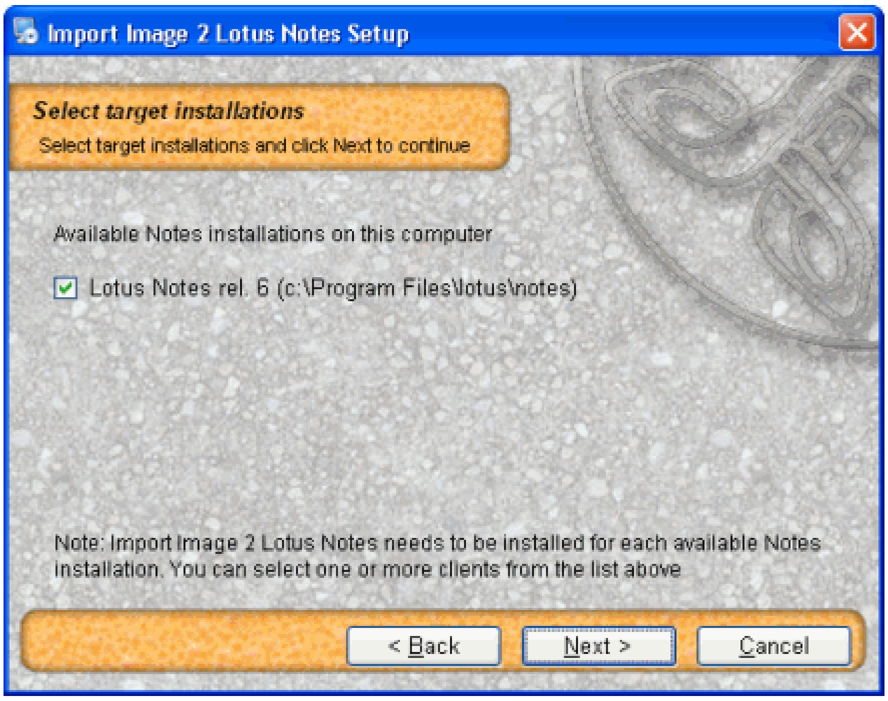
CHAPTER 2

Installation

In order to have Notes databases use II2LN-functionality, you must either install II2LN on the Notes client -or- on the Domino server. Note that server installation require a server-license.

Normally you will use the setup application to install II2LN. However, sometimes you will need to validate an install, for example if you have multiple Notes clients on a single workstation, or if you for some other reason suspect an erroneous installation. Please see ***here*** (see "Validate an Installation" on page 23) for a detailed list of the contents in the installer.

The setup application for II2LN will *attempt* to detect all your installed Notes clients and/or Domino servers. During installation you will clearly see which Notes and/or Domino versions II2LN attemt to install on.


Import Image 2 Lotus Notes Setup

Select target installations

Select target installations and click Next to continue

Available Notes installations on this computer

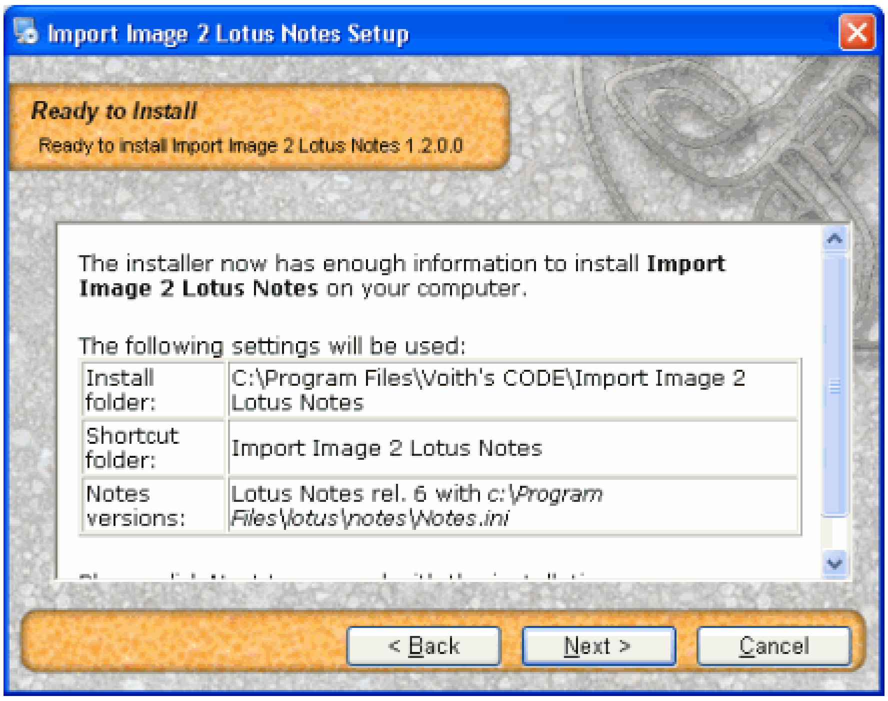
[✓] Lotus Notes rel. 6 (c:\Program Files\lotus\notes)

Note: Import Image 2 Lotus Notes needs to be installed for each available Notes installation. You can select one or more clients from the list above

Figure 1: Install - Notes client

In the screen shot above, you see that II2LN plan to install II2LN support for the found Lotus Notes rel. 6 installation. If you have multiple Notes versions, you can select yourself which versions you install II2LN support for.

The installer will show you a confirmation dialog before anything is installed or modified:


Import Image 2 Lotus Notes Setup
Ready to Install
Ready to install Import Image 2 Lotus Notes 1.2.0.0
< Back
Next >
Cancel

The installer now has enough information to install **Import Image 2 Lotus Notes** on your computer.

The following settings will be used:

Install folder:	C:\Program Files\Voith's CODE\Import Image 2 Lotus Notes
Shortcut folder:	Import Image 2 Lotus Notes
Notes versions:	Lotus Notes rel. 6 with *c:\Program Files\lotus\notes\Notes.ini*

Figure 2: Install - Confirmation

The most important aspect of the installation, is to place the files of II2LN in a directory that is referenced by the PATH environment variable on your machine. Otherwise LotusScript won't find the VCII2LN.DLL and it's two run time DLLs during execution.

In This Chapter

Requirements

II2LN is working together with Lotus Notes and therefore has the same requirements as Lotus Notes:

- Lotus Notes Release 5.0.3 or newer
  - A Notes configuration file (Notes.ini) must be available in the first 128 bytes of the system PATH environment variable
  - The Notes- or Domino program directory must also be specified in the PATH environment variable
- Machine and platform requirements; The same as for the Lotus Notes Release 5 client, which is:
  - Windows 95, 98 or ME - Windows NT 4.x SP3, Windows 2000, Windows XP, Windows Vista, Windows 7 -or- Windows 2000 server, Windows 2003 server or Windows 2008 server

- II2LN has been tested on both 32-bit and 64-bit Windows systems. However,bear in mind that II2LN still is a 32-bit component, and thus you will need 32-bit Lotus Notes client or 32-bit Lotus Domino server in order to utilize II2LN. II2LN will not work with 64-bit client or server!
- Internet Explorer 5 or newer
- Pentium processor
- RAM: On Windows 9x and Windows NT more than 32 MB. On Windows 2000 more than 64 MB and on Windows XP more than 128 MB.

Disk space: II2LN occupies approximately 7.0 MB.

Validate an Installation

The installer attempts to install all files correctly. However, there are scenarios which may confuse the installer, such as multiple installed Notes clients on a workstation, or a Notes client in addition to a Domino server on a server. In such cases you need to know the exact content of II2LN and where every file is located. This topic describes a completely manual install, and what you should look for to verify/validate an existing install.

Please contact Voith's CODE if you need to receive the II2LN files in a ZIP file

The II2LN Program Directory

II2LN installs files in several directories, but the so-called Program Directory is regarded as *home*. Typically this directory is placed somewhere along the Program Files directory structure. If you installed Windows with default directory structure, you'll find a C:\Program Files directory. Typically applications create a company subdirectory here, such as Microsoft, Adobe or Macromedia.The installer will try with a default company name of Voith's CODE for II2LN. In the company subdirectory C:\Program Files\<Name of any company> applications typically create an application directory. In our case the full application directory will be C:\Program Files\Voith's CODE\Import Image 2 Lotus Notes.

The following files are placed here:

Filename	Description	Filesize
UPDATE.EXE	II2LN contains a feature to download updates via Internet, and this program file does that job.	700 kB

UPDATE.ICO	Icon file for the Updater	29 kB
UPDATE.CLI	Data file for the Updater	23 KB
License.txt	The license file	6 kB
Import Image 2 Lotus Notes User's Guide.chm	The HTML Help version of the User's Guide	~1.3 MB
*.lss	Some sample LotusScript files	
HTML Templates	This is a directory containing other folders with HTML Templates. See ***UseHTMTemplate*** (see "UseHTMLTemplate" on page 136) import option for more information	

The Notes Program Directory

This is the directory where the main Notes files are placed. Search for NLNOTES.EXE and you are on the spot! You can also look in the Registry for the value of following key: HKEY_LOCAL_MACHINE\SOFTWARE\Lotus\Notes\Path.

Please note that each version of Lotus Notes has it's own subkey in the Registry. The subkey above points to the most recent installation!

If you install Notes R5 in the default directories, the Notes Program Directory will be C:\Lotus\Notes. If you do the same with Notes 6, the default program directory will be C:\Program Files\Lotus\Notes. Starting from Notes 8, the default program directory will be C:\Program Files\IBM\Lotus\Notes.

The following files are placed here:

Filename	Description	Filesize
VCII2LN.DLL	The main II2LN DLL file. This is the DLL you reference in all LotusScript code.	~700 kB
DXLTOOLS10.DLL	The Lotus DXL Runtime DLL. This is used to extract Notes data into the Lotus'ish XML format DXL	~ 2.6 MB
LCPPN23.DLL	The Lotus Notes C++ Runtime DLL. This is necessary when we want to program in C++ and let our application access Notes!	~ 2.0 MB
GDIPLUS.DLL	The GDIPLUS dll is only installed on pre Windows 2000 machines, and is a Microsoft Graphical Device support DLL, enhancing the quality of imaging on older operating systems	~ 1.6 MB
II2LN.EXE	The II2LN ***Command Line Interface*** (see "The Command Line Interface" on page 6) program	~300 kB

Very important: In order to have Notes C/C++ API based applications run correctly, these applications must know where to find the core Notes program files. The C and C++ API libraries use functions in all the DLLs! This is done by specifying the Notes Program Directory in the PATH environment variable. In Windows 95, 98 and ME, this is set in the AUTOEXEC.BAT file with the command

```
SET PATH=%PATH%;<full path to the Notes Program
Directory>
```

In Windows NT, 2000, XP, 2003 server and Vista the PATH environment variable is set from the System applet in the Control Panel. Be sure to have Administrators Rights in order to change this variable's content.

The Notes Data Directory

The Notes Data Directory the directory where Lotus Notes place it's database files (all the NSF and NTF files among others).

In its current version, II2LN does not install any files in this directory.

The Windows System Directory

The so-called System directory is where applications may place *system components* . In the previous years, this directory was heavily used by many applications, and you typically find thousands of files in this directory. However, in more recent years, Microsoft has recommended application developers not to use this directory, unless the component is meant for sharing between several applications. Typically the exact path to the system directory is C:\Window\System32 or C:\WINNT\System32 if you have installed Windows in it's standard directories. If you are uncertain what your system directory is, take a look at the PATH statement definition. The System directory is very often specified there, since this directory contains many many common and shared component files.

The following files are placed here:

Filename	Description	Filesize
PolyImagePro.dll	Graphical support DLL, mainly responsible for processing EXIF and IPTC tags	~ 2.6 MB

HTMLSNAP2.DLL	Graphical support DLL, mainly responsible for adding URL support to II2LN, meaning that you can import web sites as images.	~ 290 kB

Necessary Changes to the Registry

II2LN makes several modifications to the registry.

Note: Some of the settings are obfuscated, to keep your registration information a little bit more secure for prying eyes. This information is at the present time not possible to recreate in a manual matter.

The registry content for the key HKEY_LOCAL_MACHINE\SOFTWARE\Voith's CODE\Import Image 2 Lotus Notes contains the following names and values

Key	Value
Version	The version of II2LN installed.
Install\Data	Obfuscated
Install\Options	Obfuscated
Install\Path	The full path to the II2LN Program Directory
Install\PCode	Product Code. This is used if you want to upgrade the software via Internet. The PCode has the following meaning: VCII2LN_R<Release number>_L<Language number>. Currently we are on release 1, and the only available language is 9 meaning English. VCII2LN=Voith's CODE II2LN by the way!

The registry content for the key HKEY_CURRENT_USER\Software\Voith's CODE\Import Image 2 Lotus Notes contains the following names and values

Key	Value
Key	The registration key as received from Voith's CODE
HTMLTemplatePath	Optional key. Normally the II2LN installer places a `HTML Templates` directory in the program directory. This is for use with the ***UseHTMLTemplate*** (on page 136) import option. If you don't want your HTML templates in that directory, you can place the HTML Templates whereever you would like and reference the directory with this key

Very important about installation on a Domino server: A Domino server typically runs as a System Service, meaning that the Domino server will start whenever the machine starts. The Domino server will by default run in the context of the socalled System Account. In order to make the registration key available to the system account, the registration key value above must be specified in the following registry hive as well:

HKEY_USERS\.DEFAULT\Software\Voith's CODE\Import Image 2 Lotus Notes

Necessary Changes to the Notes.ini

The Notes.ini controls much of Notes/Dominos configuration.

In its current version, II2LN doesn't need to modify the Notes.ini

Register Common Components

II2LN contains a common module HTMLSNAP2.DLL which needs to be so-called *registered* in Windows. This means that the component are registered in the Registry with it's information, and thus made available to all other applications in Windows that would like to use their functions and features.

In order to register functions in Windows, ensure that the HTMLSNAP2.DLL is placed in the Windows System directory as described in the previous topics. Then open an command prompt (CMD.EXE from the Start->Run ...menu) and locate the system directory with the change directory command CHDIR (or CD for short). This can be done like this:

```
CD \Windows\System32
```

When in the system directory, issue the following command:

```
REGSVR32 <name of file to register>
```

For example:

```
REGSVR32 HTMLSNAP2.DLL
```

When enter is pressed, you should after a short while see a confirmation message of a successful registration. Do this for all the system files that has "This file needs to be registered in Windows" here.

You may also uninstall - or un-register the component by issuing the -u parameter, such as:

```
REGSVR32 -u HTMLSNAP2.DLL
```

CHAPTER 3

Functions

This chapter will describe each function in detail

In This Chapter

Identification functions

This function group consist of functions enabling you to identify the current release of II2LN installed on your machine or server.

It's main purpose is to make it possible to write code that knows the release level of II2LN, and thus what functions and import options that are available in that release. For example you can have a Notes database that utilize some of the newer features of II2LN. You can by using the identification functions ensure that your code has the needed release level of II2LN.

See ***here*** (see "What version of VCII2LN.DLL is installed ?" on page 186) on how to manually identify the current release of VCII2LN.DLL installed on your machine.

In the following descriptions of each function, we use the release number "1.2.0.8".

GetII2LNMajorVersion

Description:

Return the major version of the installed VCII2LN.DLL. If the full release is "1.2.0.8", this function return the number 1.

Declaration:

```
Declare Function GetII2LNMajorVersion Lib "VCII2LN.DLL" () As Integer
```

Returns:

The major version as an Integer

Usage:

```
Dim iMajorVersion As Integer
iMajorVersion = GetII2LNMajorVersion()
```

Introduced in version:

1.2.0.0

GetII2LNMinorVersion

Description:

Return the minor version of the installed VCII2LN.DLL. If the full release is "1.2.0.8", this function return the number 2.

Declaration:

Declare Function GetII2LNMinorVersion Lib "VCII2LN.DLL" () As Integer

Returns:

The minor version as an Integer

Usage:

```
Dim iMinorVersion As Integer
iMinorVersion = GetII2LNMinorVersion()
```

Introduced in version:

1.2.0.0

GetII2LNBuildNumber

Description:

Return the build number of the installed VCII2LN.DLL. If the full release is "1.2.0.8", this function return the number 8.

Declaration:

Declare Function GetII2LNBuildVersion Lib "VCII2LN.DLL" () As Integer

Returns:

The build number as an Integer

Usage:

```
Dim iBuildNumber As Integer
iBuildNumber = GetII2LNBuildNumber()
```

Introduced in version:

1.2.0.0

GetII2LNVersion

Description:

Return the full release string as seen in the ***Version*** (see "What version of VCII2LN.DLL is installed ?" on page 186) tab of the File Properties dialog box.

Declaration:

```
Declare Function GetII2LNVersion Lib
"VCII2LN.DLL" (_
Byval strOutBuf As String, _
Byval iLenOfOutBuf As Integer) As Integer
```

Parameters:

Param	Description
strOutBuf	String, in/out variable, will contain the release string after processing

iLenOfOutBuf	Maximum length of strOutBuf. II2LN will not fill in data beyond this length of strOutBuf.

Returns:

The length of the release string as an Integer. In addition, the release string is copied into the first variable strOutBuf. See notes below on how to strip off trailing blanks!

Usage:

```
' Make a string large enough for the version
string
Dim strVersionBuf As String * 255
Dim strTmp As String
' Hold the length of the returned buffer
Dim iLenOfReturnedBuf As Integer
iLenOfReturnedBuf =
GetII2LNVersion(strVersionBuf, 255)
' Strip of trailing blanks
strTmp = Left$(strVersionBuf, iLenOfReturnedBuf)
```

Notes:

This function return a string to LotusScript. Such logic is always a bit cumbersome to work with in LotusScript. First of all, you must define a String with a specific length. In the sample above, this is the number 255. If you don't declare the String in this way, you will declare a string with too small size for the function to work properly. When the GetII2LNVersion return the length of the content in the output variable (strOutBuf), use that number to strip off trailing blanks. Again, in the sample above, the variable strTmp contains the stripped release string.

Introduced in version:

1.2.0.0

CopyII2LNImage

Description:

CopyII2LNImage will copy the source image to the target image with all details. No image processing will be done what so ever.

Declaration:

```
Declare Function CopyII2LNImage Lib "VCII2LN.DLL"
(_
Byval pstrSourceFilename As String, _
Byval pstrTargetFilename As String, _
Byval pstrOptions As String) As Integer
```

Parameters:

Param	Description
pstrSourceFilename	The full filename of the image to copy.
pstrTargetFilename	The target filename. Note that if a target file already exist with the same same, it will be overwritten
pstrOptions	Import Options. Reserved for future use

Returns:

0 = OK
4110 to 4119 = Some error occurs during rotation

Usage:

```
lRc = CopyII2LNImage("c:\temp\Input
file.bmp","c:\temp\Output file.jpg")
```

The call above copies the source file to the target file.

Notes:

The target file name parameter can contain any of the image variables but the $(IMPORT_xxx)-series.

Introduced in version:

1.7.0.0

ImportImage

Description:

This is one of the core functions in II2LN and it import the specified image as a ***Notes Embedded Image*** (see "Notes Embedded Image - what is it ?" on page 166).

Declaration:

```
Declare Function ImportImage Lib "VCII2LN.DLL" (_
Byval pstrNotesServer As String, _
Byval pstrNotesDatabase As String, _
Byval pstrNotesUNID As String, _
Byval pstrNotesField As String, _
Byval pstrFilename As String, _
Byval pstrImportOptions As String) As Long
```

Parameters:

Param	Description
pstrNotesServer	The Domino server or blank ("") if local. The output from the NotesDatabase property *Server* works fine, such as: `db.Server`
pstrNotesDatabase	The filename of the database containg the document. Either the relative filename (such as mail\tester.nsf) or the extact filename (such as c:\program files\lotus\notes\data\mail\tester.nsf) may be used. The output from the NotesDatabase property *FilePath* works fine, such as: `db.FilePath`
pstrNotesUNID	The Document Unique ID of the document to work on. Note that this should be the textual string of the UNID. Note that you can use the NotesDocument's property *UniversalID*, such as: `doc.UniversalID`
pstrNotesField	The field name of the rich text field to import the image into.

pstrFilename	The full filename of the image to import. Starting from version 1.4 of II2LN you can also specify any `http://` or `file://` URL in this parameter
pstrImportOptions	Optional ***import options*** (see "All Import Options" on page 65) to control how the image is processed and imported

Returns:

The return code as Long. A return code of zero means successful import. See below for other return codes.

Usage:

```
Sub Initialize
     Dim lRc As Long
     lRc = ImportImage(_
      "YourServerName",_
      "Database.nsf",_
      "UNID",_
      "RichTextFieldName",_
      "C:\Temp\Smile.jpg",_
      "Sharpen:50;Resize:200,200")
End Sub
```

See ***this*** (see "Import an image from view context to the current document" on page 142) example for a complete demonstration of ImportImage

Notes:

Note that ImportImage can import the image either into a sole rich text field, or it can replace an already existing variable in the rich text field. Read more about this feature ***here*** (see "Import to rich text fields or use variables" on page 3).

All import logic is controlled by the import options, read more about what import options are ***here,*** (see "Import Options" on page 18) or see a list of available import options ***here.*** (see "All Import Options" on page 65)

Return Codes:

Code	Description
1001	Licence error. This PERSONAL license can only be used with Notes ID Name/Organisation. The installed license is locked to only one Notes user ID.You have probably tried to run II2LN with another Notes user ID.

1002	Licence error. This PERSONAL licence for N concurrent users can only be used within organization X. The installed license is limited to work on one Notes organization, You have probably tried to run the personal version of II2LN on another Notes organization.
1003	Licence error. This SERVER licence can only be used on server X. The installed license can only be used on the specified server name. Both servername and organization must match.
1004	Licence error. This SITE licence can only be used within organization X. The installed site-license can only be used with in the specified organization.
1005	Licence error. This DATABASE licence can only be used on database X. The licence is tied to a specified database, and it seems like you are using the component on another database
2001	There is an unspecified error with the internal DXL processing in II2LN. Please refer to the appendix about ***Troubleshooting*** (on page 184).
2002	The specified image is not a valid image. This error code will only be given if the import option StrictImageVerification is turned on.
2003	Error during processing of HTML Templates
ANY OTHE R	II2LN may return a whole set of errors. If errors occur, they will be logged in either the VCII2LN_Log field or in the optionally specified log file. See appendix about ***Troubleshooting*** (on page 184) for more info on logging

Introduced in version:

1.0.0.0

ImportImageAsResource

Description:

The ImportImageAsResource is used to import an image into Lotus Notes as an ***Image Resource*** (see "Image Resource - what is it ?" on page 161). This enables you to show the image directly in Notes views!!

Declaration:

```
Declare Function ImportImageAsResource Lib
"VCII2LN.DLL" (_
Byval pstrNotesServer As String, _
Byval pstrNotesDatabase As String, _
Byval pstrNotesUNID As String, _
Byval pstrNotesFieldThumbnail As String, _
Byval iThumbnailWidth As Integer,_
Byval iThumbnailHeight As Integer,_
Byval pstrNotesFieldImage As String, _
Byval pstrFilename As String, _
Byval pstrImportOptions As String) As Long
```

Parameters:

Param	Description
pstrNotesServer	The Domino server or blank ("") if local. The output from the NotesDatabase property *Server* works fine, such as: `db.Server`
pstrNotesDatabase	The filename of the database containg the document. Either the relative filename (such as mail\tester.nsf) or the extact filename (such as c:\program files\lotus\notes\ data\mail\tester.nsf) may be used. The output from the NotesDatabase property *FilePath* works fine, such as: `db.FilePath`
pstrNotesUNID	The Document Unique ID of the document to work on. Note that this should be the textual string of the UNID. Note that you can use the NotesDocument's property *UniversalID,* such as: `doc.UniversalID`

pstrNotesFieldThumbnail	The field name of the rich text field to hold the thumbnail image. Note that this image must not be visible in any way in the Notes document. Otherwise the referenced image will disappear!
iThumbnailWidth	The width in pixels of the thumbnail
iThumbnailHeight	The height in pixels of the thumbnail
pstrNotesFieldImage	The field name of the rich text field to import the image into.
pstrFilename	The full filename of the image to import. Starting from version 1.4 of II2LN you can also specify any `http://` or `file://` URL in this parameter
pstrImportOptions	Optional ***import options*** (see "All Import Options" on page 65) to control how the image is processed and imported

Returns:

Same return codes as ***ImportImage*** (on page 35).

Usage:

```
lRc = ImportImageAsResource(_
db.Server,_
db.FilePath,_
strUNID, _
"Thumbnail",_
120,_
90,_
"Image",_
strOriginalFile,_
strImportOptions)
```

The sample above import the image stored in the variable *strOriginalFile*. First a thumbnail of the image will be created in the rich text field *Thumbnail*, and the thumbnail will have the size 120 x 90 pixels. Remember, the Thumbnail-field should not be visible in the forms presenting the document, otherwise the thumbnail image will disappear when the document later is modified!

Secondly, II2LN will import the image to another rich text field named *Image*. The Image field will contain a ***Notes Embedded Image*** (see "Notes Embedded Image - what is it ?" on page 166) of the image, and can be used in the form!

All the imports are controlled via the import options in the string *strImportOptions*.

Notes:

In order to display Image Resources directly in Notes views, a few tricks has to be performed. First and foremost, the Thumbnail-field above can't be visible in any form! Secondly, II2LN will create a set of hidden fields in the Notes document for you. The autogenerated fields are always prefixed with the fieldname in the parameter *pstrNotesFieldThumbnail*. In the example above, this was "Thumbnail". Below we use XYZ to indicate this prefix

The fields are:

Field	Description

XYZFilename	This text field will contain the Image Resource filename that was the result of the import. In order to keep correct referencing and not having a problem with duplicate filenames, the Image Resource filename is automatically generated by II2LN. The format of the filename is;
	
	<NoteID><PrefixName>.<File Extension>
	
	For example;
	`000056ABXYZ.jpg`

XYZHeight	This number field contain the real height of the thumbnail image. Remember, II2LN will always calculate the best fit ***inside*** the specified X and Y parameters!
XYZWidth	This number field contain the real width of the thumbnail image.

II2LN create these extra fields to make it much easier for you to create a Notes view displaying the image resource. Learn all about that ***here*** (see "How to display my images in a Notes view" on page 162).

Introduced in version:

1.2.0.0

AttachImage

Description:

AttachImage may process the image just like ***ImportImage*** (on page 35) or ***ImportImageAsResource,*** (see "ImportImageAsResource" on page 38) but it will only attach the image to the specified rich text field.

Declaration:

```
Declare Function AttachImage Lib "VCII2LN.DLL" (_
Byval pstrNotesServer As String, _
Byval pstrNotesDatabase As String, _
Byval pstrNotesUNID As String, _
Byval pstrNotesField As String, _
Byval pstrFilename As String, _
Byval pstrImportOptions As String) As Long
```

Parameters:

Param	Description
pstrNotesServer	The Domino server or blank ("") if local. The output from the NotesDatabase property *Server* works fine, such as: `db.Server`
pstrNotesDatabase	The filename of the database containg the document. Either the relative filename (such as mail\tester.nsf) or the extact filename (such as c:\program files\lotus\notes\ data\mail\tester.nsf) may be used. The output from the NotesDatabase property *FilePath* works fine, such as: `db.FilePath`
pstrNotesUNID	The Document Unique ID of the document to work on. Note that this should be the textual string of the UNID. Note that you can use the NotesDocument's property *UniversalID,* such as: `doc.UniversalID`
pstrNotesField	The field name of the rich text field to attach the image to.

pstrFilename	The full filename of the image to attach. Starting from version 1.4 of II2LN you can also specify any `http://` or `file://` URL in this parameter
pstrImportOptions	Optional ***import options*** (see "All Import Options" on page 65) to control how the image is processed before it's imported.

Returns:

Same return codes as ***ImportImage*** (on page 35).

Usage:

```
lRc = AttachImage(db.Server,_
db.FilePath,_
strUNID, _
"Image",_
strFilename,_
strImportOptions)
```

The call above will attach the file specified in *strFilename* to the rich text field *Image* in the document with the UNID stored in *strUNID*. Before the image is attached, it is processed as specified in the import options in *strImportOptions*.

Introduced in version:

1.0.0.0

BrowseForDirectory

Description:

BrowseForDirectory is a special function that is included in II2LN to circumvent the issue with the Lotus Notes dialog boxes always starting in the Notes data directory. Perhaps you want to store a directory in your application, and be sure that the Browse For Directory dialog box pops up with your directory instead!

Declaration:

```
Declare Function BrowseForDirectory Lib
"VCII2LN.DLL" (_
Byval pstrTitle As String, _
Byval pstrDefaultDirectory As String, _
Byval pstrOutputBuffer As String,_
Byval iLenOutputBuffer As Integer) As Integer
```

Parameters:

Param	Description
pstrTitle	The title of the dialog box
pstrDefaultDirectory	The default directory
pstrOutputBuffer	The allocated string to hold the selected directory path
iLenOutputBuffer	The length of the output buffer above.

Returns:

0 = Always 0

Usage:

```
Dim iRc As Integer
Dim strOutputBuf As String * 255
Dim iLenOutputBuf As Integer
Dim strTmp As String

iRc = BrowseForDirectory("Choose directory for
your images"_
     , "D:\Temp", strOutputBuf, iLenOutputBuf)

strTmp = Left$(strOutputBuf, iRc) ' Strip of
trailing blanks

Messagebox strTmp
```

Notes:

Introduced in version:

1.6.0.3

ConvertImage

Description:

ConvertImage will let you convert an image from one format to another with ease. Note that the image is completely processed within the function, so there is no need to create or maintain Notes documents at all.

Declaration:

```
Declare Function ConvertImage Lib "VCII2LN.DLL"
(_
Byval pstrInputFilename As String, _
Byval pstrOutputFilename As String, _
Byval iOutputFormat As Integer, _
Byval pstrOptions As String) As Long
```

Parameters:

Param	Description
pstrInputFilename	The full filename of the image to convert Starting from version 1.4 of II2LN you can also specify any `http://` or `file://` URL in this parameter
pstrOutputFilename	The output filename after the conversion.
iOutputFormat	The format code you want to convert the file to. Note that if this is -1 or 99, ConvertImage will use the file extension of the output file as a guide to what format to convert to. See ***here*** (see "The image format codes" on page 57) for a list of valid format codes
pstrOptions	Optional options to control how the conversion is done. NOTE, this is NOT the same as Import Options, and those cannot be used here.

Returns:

0 = OK, image saved in desired format
1 = Input file not found or not loaded
2 = Error during save/convert

Usage:

lRc = ConvertImage("c:\temp\Input file.bmp","c:\temp\Output file.jpg",-1
, strImportOptions)

The call above will convert the input file to the output file name. Note that the output format is -1, and thus we use the output file extension of "jpg" as a guide to what format to save as. Note that the input file isn't changed at all.

Notes:

This function disables a list of import options, due to the fact that we don't work with a Notes document in a database. The disabled import options are;

AttachOriginalFile
AttachPreviewFile
ExtractEXIFToField
ExtractIPTCToField
ExtractImageVariablesToField
ProcessTimeField
ReplaceWithVariable
SetLogFilename
SetLogLevel
UseImageResourceDatabase

Note that ConvertImage won't convert images to postscript-based files, such as eps, pdf or ps.

Introduced in version:

1.6.0.0

GetImageDimension

Description:

The GetImageDimension will analyze the specified file and extract the dimensions from the file. The information you get are;

§ Width

§ Height

§ BitDepth

§ DPI X

§ DPI Y

Declaration:

```
Declare Function GetImageDimension Lib "VCII2LN.DLL" (_
Byval pstrFilename As String, _
iX As Integer, _
iY As Integer, _
iBitDepth As Integer, _
iDpiX As Integer, _
iDpiY As Integer) As Long
```

Parameters:

Param	Description
pstrFilename	The full filename of the image to extract dimensions from. Starting from version 1.4 of II2LN you can also specify any `http://` or `file://` URL in this parameter
iX	The output parameter containing the Width (X) in pixels
iY	The output parameter containing the Height (Y) pixels
iBitDepth	The output parameter containing the bitdepth
iDpiX	The DPI of the image in X direction
iDPI	The DPI of the image in Y direction

Returns:

-1 if the image format was not recognized. Otherwise 1.

Please note that the DPI variables may contain 0 if no DPI information was found in the image.

Usage:

```
Dim lRc As Long
Dim iX As Integer
Dim iY As Integer
Dim iBitDepth As Integer
Dim iDpiX As Integer
Dim iDpiY As Integer

lRc = GetImageDimension(strFileName,  iX, iY,
iBitDepth, iDpiX, iDpiY)

If lRc = 1 Then
     Messagebox "The image dimensions are X=" &
Cstr(iX) & " Y=" & Cstr(iY) & _
     " BitDepth=" & Cstr(iBitDepth) & " DpiX=" &
Cstr(iDpiX) & _
     " DpiY=" & Cstr(iDpiY), 0, "Image
Dimensions"
Else
     Messagebox "Not a valid image file", 0,
"Image Dimensions"
End If
```

Introduced in version:

1.5.0.0

GetImageFromClipboard

Description:

TheGetImageFromClipboard check whether the clipboard contain an image or not. If it does, it will be saved to the specified target file.

Why would you use this function? Imagine you have a Notes application where you want to replace an imbedded image which have been pasted into a document. You want to make sure that you can replace that image with one resized to the same size constrains. By using the Cut- or Copy method in the NotesUIDocument LotusScript class you can easily put the selected element on the clipboard. Then you will use the GetImageFromClipboard function to save the image to a temporary file, and then the ***GetImageDimension*** (on page 48) to get the image size. Now that you know the size of the image would will replace, you could use ***ProcessImage*** (on page 60) to resize the new image to the old size constraints, and finally import it with NotesUIDocument's Import method.

Declaration:

```
Declare Function GetImageFromClipboard Lib
"VCII2LN.DLL" (_
Byval pstrTargetImageFileName As String, _
iPreferMetaFile As Integer, _
iIncludeAlphaChannel As Integer, _
Byval pstrImportOptions As String) As Integer
```

Parameters:

Param	Description
pstrTargetImageFileName	The full filename of the saved clipboard image
iPreferMetaFile	Many applications save several image formats when copy data to clipboard. II2LN will by default search for the images in this sequence; DIB, BMP and ENHANCED METAFILE. By specifying 1 for this value, you change the search sequence to; ENHANCED METAFILE, DIB and BMP Default value is 0
iIncludeAlphaChannel	0 = Don't include alpha channel (transparency information in image) 1 = Include alpha channel, if present in clipboard image. Default value is 0

Returns:

Code	Description
0	OK, image captured from clipboard and saved to target file
1	Clipboard data isn't recognized as an image
2	Unknown output format
3	Clipboard capture ok, but save to file failed

Usage:

```
Dim ws As New NotesUIWorkspace
Dim uidoc As NotesUIDocument
Dim strOutFile As String

strOutFile = "c:\temp\My captured image.psd"

Set uidoc = ws.CurrentDocument
rc = Messagebox("You have SELECTED an image in
the current document?", 36, "II2LN")
If rc <> 6 Then Exit Sub
Call uidoc.Cut
iRc = GetImageFromClipboard(strOutFile,0,1,"")
If iRc = 0 Then
     Messagebox "All ok, clipboard captured to "
& strOutFile
Else
     Messagebox "GetImageFromClipboard error" &
Cstr(iRc)
End If
```

Notes:

Note that II2LN will try to create the target image based upon the file type you specify. This means that you will get an JPEG file if you have an file extension of .jpg or .jpeg, and you will get a GIF file if you specify a file extension of .gif. The supported file types are;

JPG, GIF, BMP, PNG, PSD, PSP, TGA and TIFF

Please note that if you want to use the Import trick as described in the introduction above, you should limit the file types to JPG, GIF or BMP, because the Import method in NotesUIDocument only support a very limited number of images.

Introduced in version:

1.5.0.2

GetLastII2LNError

Description:

GetLastII2LNError will retrieve the last error that occured during processing. You will receive both error number and the error message. If no error occured, you will receive error number 0 and a message indicating that everything is ok.

Declaration:

```
Declare Function GetLastII2LNError Lib
"VCII2LN.DLL" (_
Byval pstrErrorMessageBuffer As String, _
Byval iLengthOfErrorMessageBuffer As Integer) As
Integer
```

Parameters:

Param	Description
pstrErrorMessageBuffer	The allocated string to hold the error message.
iLengthOfErrorMessage Buffer	The length of the error message buffer above.

Returns:

0 = OK, all is well
Any other number means that an error has occured

Usage:

```
Dim iRc As Integer
Dim strErrorBuf As String * 255 ' Create a string for 255 chars
Dim iLenErrorBuf As Integer ' Hold the length of the returned buffer
Dim strTmp As String

iRc = GetLastII2LNError(strErrorBuf, iLenErrorBuf)
strTmp = Left$(strErrorBuf, iRc) ' Strip of trailing blanks
Messagebox strTmp
```

Notes:

Introduced in version:

1.7.0.0

GetTIFFFrameCount

Description:

A TIFF file can contain multiple images within a single TIFF file. Each image is stored in a frame. This function will return the number of frames within a single file. This function is often used together with the ***LoadTIFFFrame*** (on page 90) import option.

Declaration:

```
Declare Function GetTIFFFrameCount Lib "VCII2LN.DLL"
(pstrSourceFile as String) As Integer
```

Returns:

The number of frames as an Integer

Usage:

```
Dim iFrames As Integer
iFrames = GetTIFFFrameCount("c:\temp\TiffSamp.tif")
```

Introduced in version:

1.3.0.0

GuessImageFormat

Description:

The GuessImageFormat will analyze the specified file and determine what kind of image file it is. This analysis is performed on the actual image data, and the function is much smarter than looking as just the file extention. It is for example possible to store a JPG file with the filename TEST.GIF, and still GuessImageFormat will detect the image as JPG.

Declaration:

```
Declare Function GuessImageFormat Lib
"VCII2LN.DLL" _
(Byval pstrFilename As String) As Integer
```

Parameters:

Param	Description
pstrFilename	The full filename of the image to check. Starting from version 1.4 of II2LN you can also specify any `http://` or `file://` URL in this parameter

Returns:

-1 if the image format was not recognized. Otherwise one of the ***image format codes*** (see "The image format codes" on page 57).

Usage:

```
Dim iImageFormat as Integer
iImageFormat = GuessImageFormat(strFileName)
```

The iImageFormat will contain the number indicating the image format

Introduced in version:

1.1.0.0

The image format codes

Code	Description
0	JPG, 24 bits
1	JPG, 8 Bits
2	BMP, 24 bits
3	BMP, 8 bits
4	PCX, 24 bits
5	PCX, 8 bits
6	PNG, 24 bits
7	PNG, 8 bits
8	TIF, 24 bits
9	TIF, 8 bits
10	TGA, 24 bits
11	TGA, 8 bits
12	WMF. 24 bits
13	EMF, 24 bits
14	PSD, 24 bits
15	PSD, 8 bits
16	WBMP, 24 bits
17	TIF, 1 bit
18	PNG, 1 bit
19	BMP, 1 bit
20	PCD, 24 bits
21	GIF, 1, 8 or 24 bits
22	EPS
23	PS
24	PDF
25	PNG, 32 bits
26	TIF, 32 bits
27	TGA, 32 bits
26	PCD, 32 bits
27	JPG, 24 bits

Note, from release 1.6.0.0 support for image formats 22 to 27 were added.

GuessImageFormatStrict

Description:

The GuessImageFormatStrict will analyze the specified file and determine what kind of image file it is just like ***GuessImageFormat,*** (see "GuessImageFormat" on page 56) but with even more through check.

Also see the import option ***StrictImageVerification*** (on page 132) to protect the other import, attach and process functions.

Declaration:

```
Declare Function GuessImageFormatStrict Lib
"VCII2LN.DLL" _
(Byval pstrFilename As String) As Integer
```

Parameters:

Param	Description
pstrFilename	The full filename of the image to check. Starting from version 1.4 of II2LN you can also specify any `http://` or `file://` URL in this parameter

Returns:

-1 if the image format was not recognized. Otherwise one of the ***image format codes*** (see "The image format codes" on page 57).

Usage:

```
Dim iImageFormat as Integer
iImageFormat =
GuessImageFormatStrict(strFileName)
```

The iImageFormat will contain the number indicating the image format

Introduced in version:

1.5.0.1

MoveII2LNImage

Description:

MoveII2LNImage will move the source image to the target image with all details. No image processing will be done what so ever.

Declaration:

```
Declare Function MoveII2LNImage Lib "VCII2LN.DLL"
(_
Byval pstrSourceFilename As String, _
Byval pstrTargetFilename As String, _
Byval pstrOptions As String) As Integer
```

Parameters:

Param	Description
pstrSourceFilename	The full filename of the image to move.
pstrTargetFilename	The target filename. Note that if a target file already exist with the same same, it will be overwritten
pstrOptions	Import Options, reserved for future use

Returns:

0 = OK
4120 to 4129 = Some error occurs during rotation

Usage:

```
lRc = MoveII2LNImage("c:\temp\Input
file.bmp","c:\temp\Output file.jpg")
```

The call above moves the source file to the target file.

Notes:

The target file name parameter can contain any of the image variables but the $(IMPORT_xxx)-series.

Introduced in version:

1.7.0.0

ProcessImage

Description:

ProcessImage will let you process the image without importing or attaching the image to Notes at all. This makes it possible to do whatever you want to do with the result.

Declaration:

```
Declare Function ProcessImage Lib "VCII2LN.DLL"
(_
Byval pstrInputFilename As String, _
Byval pstrOutputFilename As String, _
Byval pstrImportOptions As String) As Long
```

Parameters:

Param	Description
pstrInputFilename	The full filename of the image to process. Starting from version 1.4 of II2LN you can also specify any `http://` or `file://` URL in this parameter
pstrOutputFilename	The output filename after the process. Note that II2LN by default will convert the image to JPG (or GIF if the image is originally a GIF). If you want to control the output type, use the OutputFormat Import Option
pstrImportOptions	Optional ***import options*** (see "All Import Options" on page 65) to control how the image is processed before it's imported. See Notes below for a list of import optiopns that won't be processed in ProcessImage

Returns:

Same return codes as ***ImportImage*** (on page 35).

Usage:

```
lRc = ProcessImage("c:\temp\Input
file.bmp","c:\temp\Output file.jpg",
strImportOptions)
```

The call above will process the input file according to the specified import options and save the result as specified in parameter 2

Notes:

This function disables a list of import options, due to the fact that we don't work with a Notes document in a database. The disabled import options are;

AttachOriginalFile
AttachPreviewFile
ExtractEXIFToField
ExtractIPTCToField
ExtractImageVariablesToField
ProcessTimeField
ReplaceWithVariable
UseImageResourceDatabase

Introduced in version:

1.5.0.0

RotateII2LNImage

Description:

RotateII2LNImage will rotate your image as specified by your parameters. This function is super quick too!

Declaration:

```
Declare Function RotateII2LNImage Lib
"VCII2LN.DLL" (_
Byval pstrInputFilename As String, _
Byval pstrOutputFilename As String, _
Byval iRotateType as Integer, _
Byval iAllowCropping as Integer, _
Byval iUseOrientation as Integer, _
Byval iNormalizeOrientation as Integer, _
Byval pstrOptions As String) As Integer
```

Parameters:

Param	Description
pstrInputFilename	The full filename of the image to rotate.
pstrOutputFilename	The output filename after the rotation. If the file extension is different from the input file extension, your image will be converted too.
iRotateType	How do you want to rotate your image? 0 = Rotate right (clockwise) 1 = Rotate left (counter clockwise) 2 = Flip horizontally 3 = Mirror
iAllowCropping	The rotate operation on non-conforming JPEG images will be combined with a cropping operation to remove visual distortion (if necessary) when iAllowCropping is set to 1. In this case the rotate operation is not truly lossless as reversing the rotation operation will result in an image with altered dimensions Specify 0 to NOT allow cropping and 1 to allow cropping

iUseOrientation	If the image contains EXIF information and has the EXIF tag Orientation set, the rotation will be according to value of the tag. This means that you can automatically rotate your images from for example a camera, to the correct rotation. Specify 0 to NOT use orientation and 1 to use it
iNormalizeOrientation	When you rotate in image, a rotation often doesn't correct any potentially existing EXIF Orientation tags. This may lead to issues where you see your rotated images fine on your computer, while for example your Media Center rotates your images once more - just becase they try to pay attention to the EXIF orientation tag! To ensure that your rotation also is registered in the EXIF Orientation tag, you can specify this parameter to 1 and use 0 if you don't want it to change the Orientation tag
pstrOptions	Import Options, reserved for future use

Returns:

0 = OK
4100 to 4109 = Some error occurs during rotation

Usage:

```
lRc = RotateII2LNImage("c:\temp\Input
file.bmp","c:\temp\Output file.jpg", 1, 0, 0, 1)
```

The call above rotates your image to the left (counter clockwise) and also updates the EXIF Orientation tag. We do not allow cropping and do not pay attention to any existing EXIF Orientation tag.

Notes:

The target file name parameter can contain any of the image variables but the $(IMPORT_xxx)-series.

Introduced in version:

1.7.0.0

CHAPTER 4

All Import Options

The ***import options*** (on page 18) control how you want to process the image when you import the image. The understanding and proper use of import options are therefore important to understand.

This chapter describes each available import option.

Please note that an import option may or may not have parameters. Some parameters may be optional, and II2LLN will fill in default values. In the following topics, the optional parameters will be listed in square brackets, like this

```
<mandatory param 1>, <mandatory param 2> [,
<optional param 1>]
```

The <optional param 1> is the optional parameter, and the import option will work without specifying it!

Also note that each import option can be repeatable, which means that you can specify the import option more than once in the string of import options. This makes it for example possible to overlay text at multiple places in the image in one go

In This Chapter

AddVariable

Description:

II2LN can extract most meta data directly from you image, such as EXIF and IPTC information. In addition many other ***variables*** (on page 153) are also available for you to use. With the repeatable AddVariable import option you can add your own variables too. This will make it easy to for example reference your very own variable $(MYTITLE) in a Notes document, or in a HTML Template. This gives you great flexibility to add your own dynamic content!

Declaration:

```
AddVariable:<Variable name>,<Variable value>
```

Parameters:

Param	Description
<Variable Name>	The variable name without prefix `$(` and suffix `)`. The variable name will automatically be known to II2LN as $(Variable Name). For example TEST will become $(TEST).
<Variable value>	Any value you want to keep in the variable.

Notes:

```
AddVariable:MYTITLE,This is some title!
```

This will make the variable $(MYTITLE) available for other import options such as ***ReplaceWithVariable*** (on page 109) and ***UseHTMLTemplate*** (on page 136). When II2LN sees the variable `$(MYTITLE)` it will be replaced with the value `This is some title!`.

Introduced in version:

1.7.0.0

Repeatable

Yes

AttachImportFile

Description:

II2LN main purpose is to import images for display, either as ***Notes Embedded Images*** (see "Notes Embedded Image - what is it ?" on page 166) or ***Image Resources*** (see "Image Resource - what is it ?" on page 161). However, there are times when you need to attach the files as ordinary attachments. For example in web-solutions, you can reference an attachment via the $File-URL.

This import option will attach the imported (ie. the modified) file to the specified field name.

Declaration:

```
AttachImportFile:<fieldname>
```

Parameters:

Param	Description
<fieldname>	The Notes rich text field name to attach the import image to

Notes:

Note that II2LN will clear any existing content in the specified field before attaching the file.

Introduced in version:

1.0.0.0

Repeatable

No

AppendIPTC

Description:

Append the specified value to any existing IPTC tag content. In other words, the new value will be suffixed to existing values.

If the specified tag doesn't exist, it will be created.

Declaration:

```
AppendIPTC:<tag name>, <tag value>
```

Parameters:

Param	Description
<tag name>	The IPTC tag name to append the specified value to
<tag value>	The value to append to the tag name

Notes:

Please see ***here*** (see "Manipulate EXIF and IPTC information" on page 5) for a description of the Replace, Append and Insert logic regarding EXIF and IPTC values

Introduced in version:

1.2.5.0

Repeatable

No

AttachOriginalFile

Description:

II2LN main purpose is to import images for display, either as ***Notes Embedded Images*** (see "Notes Embedded Image - what is it ?" on page 166) or ***Image Resources*** (see "Image Resource - what is it ?" on page 161). However, there are times when you need to attach the files as ordinary attachments. For example in web-solutions, you can reference an attachment via the $File-URL.

This import option will attach the original file to the specified field name.

Declaration:

```
AttachOriginalFile:<fieldname>
```

Parameters:

Param	Description
<fieldname>	The Notes rich text field name to attach the original image to

Notes:

Note that II2LN will clear any existing content in the specified field before attaching the file.

Introduced in version:

1.0.0.0

Repeatable

No

AttachPreviewFile

Description:

This import option enables you to create an preview image of the currently processed image. This means that you can have a separate attachment in your Notes document, which is optionally resized.

Declaration:

```
AttachPreviewFile:<file name>, <field name>,
[<X>, <Y>, <Resize Method>]
```

Parameters:

Param	Description
<file name>	The forced file name of the preview image. Note that this is the file name only, excluding the file extension. For example you will specify "Preview", not "Preview.jpg". The file extension is determined from the imported file name. The Notes field *PreviewFilename* will contain the real file name.
<field name>	The Notes rich text field name to attach the preview image to
<X> and <Y>	The desired X and Y of the preview image. If not specified, the preview image will have the size width and height as the imported image.

<Resize method>	You can optionally specify any of the available ***resize methods*** (see "ResizeMethod" on page 122). By default the resize method is set to 3.

Notes:

Sometimes you need to create a preview of an image, maybe before the image is finally processed in your application. This can for example be if your application allows for image manipulation before image processing, or if you just want to have a quick peek at the image before importing it. In such cases you want to keep the original image as intact as possible, but the original image may be too large to be used as a preview in your application (think about a full sized digital camera image, at full size it's not very "preview like"!).

II2LN offers two methods of keeping the original image intact, but show a resized preview. The ***OutputSize*** (on page 101) import option will just scale the Notes Embedded Image (thus the image in the document is full size, it's just the visible image that is resized), while this import option, AttachPreviewFile, will attach a resized copy of the image.

Note that II2LN will clear any existing content in the specified field before attaching the file.

This import option will automatically create a field with the name *PreviewFilename* in the Notes document, containing the filename of the attachment.

Introduced in version:

1.2.1.0

Repeatable

No

AutoOrientation

Description:

Many modern digital cameras store meta-data as ***EXIF*** (see "Extract EXIF and IPTC information" on page 4) tags within the image. If the camera has a sensor detecting the orientation of the camera, the orientation information may be stored as the so-called *EXIF Orientation tag*. If this tag is present in the image, II2LN can automatically rotate the image into correct orientation by specifying the AutoOrientation:Yes.

Note that you also can control the rotation manually, by specifying the ***Rotate*** (on page 115) import option.

Declaration:

AutoOrientation:<Yes>

Parameters:

Param	Description
<Yes>	Specify Yes or 1 to activate autoorientation.

Notes:

If the image does not contain the EXIF Orientation tag, the image is imported as-is, and no auto rotation is imposed on the image.

Also note that this import option will correct the EXIF tag for orientation.

Introduced in version:

1.2.0.0

Repeatable

No

Blur

Description:

The Blur import option let you blur -or- smooth, the image.

Declaration:

```
Blur:<level>
```

Parameters:

Param	Description
<level>	From 0 (no blur, which is the default) to 100 (much blur). For example; `Blur:50`

Notes:

Just as for the ***Sharpen*** (on page 133) import option, it's difficult to foresee how much blur to impose on an image. It's therefore adviced to determine the blur-level in an interactive image application such as PaintShop Pro or Photoshop.

Introduced in version:

1.2.0.0

Repeatable

Yes

CanvasFormatWithColor

Description:

The CanvasFormatWithColor import option can enforce that the image fits within a certain format such as 4:3 or 16:9. This operation will not alter the aspect ration of the image, and in the case where the image is smaller than the specified format, white space may occur on either side of the original image. You can however control the color of the "white space"

Declaration:

```
CanvasFormatWithColor:<Width>,<Height>,[<Backgrou
nd color>]
```

Parameters:

Param	Description
<Width>	The width of the format, such as 4 or 16.
<Height>	The height of the format, such as 3 or 9
<Background color>	Optional parameter to control the background color of the potential white space surrounding the image. Specify the color according to the ***How to specify colors ?*** (on page 161).

Notes:

The main usage for this option is if the result-image is to be used within an application that automatically assumes a certain format. For example may several Flash applications assume that any image displayed within it's image browser, adhere to the 4:3 canvas format. If your image is not adhering to the 4:3 format, the Flash application will stretch the image to fit 4:3, thus destroying the aspect ratio of the image. You can easily avoid that by using the CanvasFormatWithColor import option!

Also note that this import option is repeatable, meaning that its position in the sequence of specified import options, may affect what the result looks like. For example can you ensure that the original image is adhering to a certain format, by specifying CanvasFormatWithColor very early in the sequence of import options. Or, you can rather ensure the format after you have resized and otherwise processed the image, by specifying CanvasFormatWithColor later in the sequence. See more on troubleshooting sequence of import options in ***Change the sequence of the import options*** (on page 185).

Introduced in version:

1.5.0.3

Repeatable

Yes

ClearExistingMetadata

Description:

Use this import option to clear any existing EXIF and IPTC information. Note that this import option must be specified before any other EXIF- or IPTC manipulation import options such as ReplaceEXIF, ReplaceIPTC, AppendIPTC or InsertIPTC.

Declaration:

```
ClearExistingMetadata:1
```

Parameters:

Param	Description
1	Instruct II2LN to clear existing metadata

Notes:

Note that II2LN will only clear the meta data in the import file. The meta data in the original file is never touched

Introduced in version:

1.2.5.0

Repeatable

No

ExtractEXIFToField

Description:

By specifying the ExtractEXIFToField import option you can automatically store the ***EXIF*** (see "Extract EXIF and IPTC information" on page 4) information in either one field, or several fields, in your document.

By having the option you use a multivalue text field or a separate field per EXIF tag- and value pair, most uses should be covered.

Declaration:

```
ExtractEXIFToField:<fieldname> [, <split>|<both>]
```

Parameters:

Param	Description
<fieldname>	The field name in your Notes document to store the multivalue text field, containing all the EXIF tag- and value pairs.
<split>	The optional SPLIT instruct II2LN to create a separate field per EXIF tag- and value pair. II2LN will use the <fieldname> as a prefix for all the EXIF tags. For example, if the <fieldname> is ***EXIF***, the orientation tag has this field name: `EXIF_Orientation` Note that all fields are text

<both>	The optional BOTH instruct II2LN to both create one EXIF field *and* create a separate field for each tag

Notes:

See***The most common EXIF tags*** (on page 172) for a list of EXIF tags

Introduced in version:

1.1.0.0

Repeatable

No

Store tags and values in one field versus many fields

Sample with ONE field:

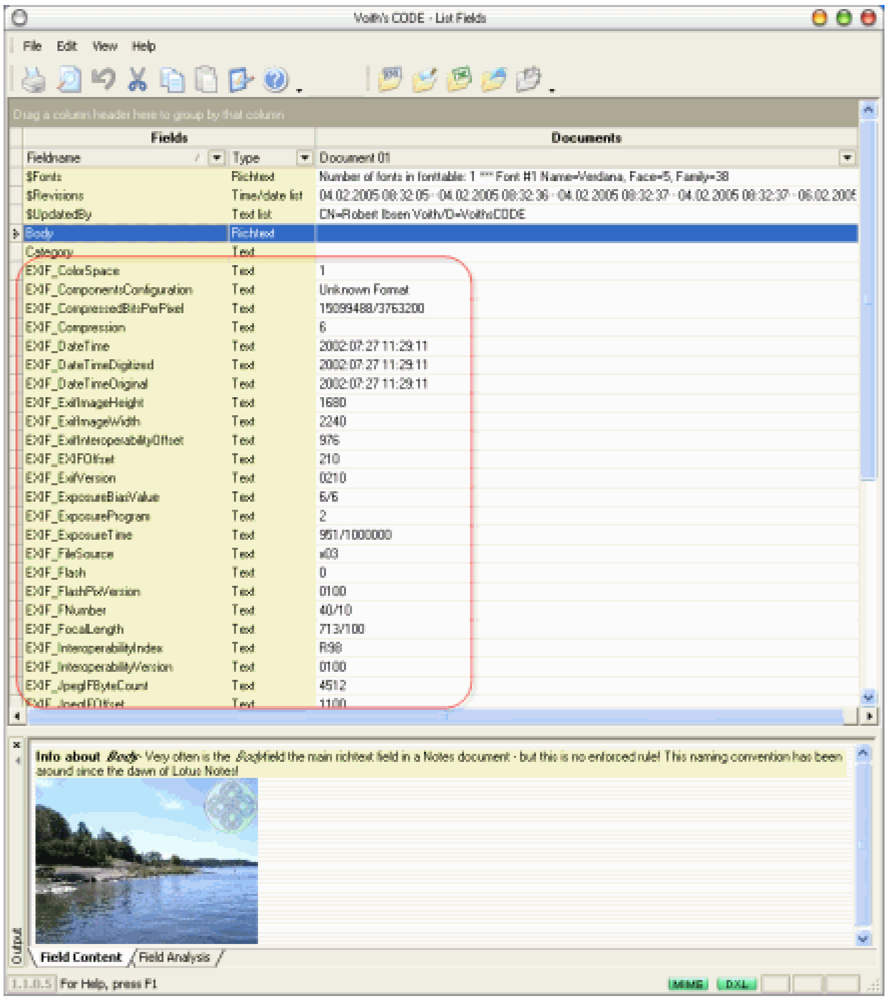
If you specify the import option `ExtractEXIFToField:EXIF`, the Notes document will contain *one* multivalue text field with the name *EXIF*. The content of the EXIF field looks like this;

```
Make|CASIO
Model|QV-4000
Orientation|8
XResolution|72/1
YResolution|72/1
ResolutionUnit|2
Software|Ver1.00
DateTime|2004:02:14 14:54:38
YCbCrPositioning|1
EXIFOffset|210
ExposureTime|1183/1000000
FNumber|56/10
ExposureProgram|2
ExifVersion|0210
DateTimeOriginal|2004:02:14 14:54:38
DateTimeDigitized|2004:02:14 14:54:38
ComponentsConfiguration|Unknown Format
CompressedBitsPerPixel|15099488/3763200
ExposureBiasValue|0/6
MaxApertureValue|20/10
MeteringMode|5
Flash|0
FocalLength|713/100
MakerNote|Unknown Format
FlashPixVersion|0100
ColorSpace|1
ExifImageWidth|2240
ExifImageHeight|1680
ExifInteroperabilityOffset|976
FileSource|x03
InteroperabilityIndex|R98
InteroperabilityVersion|0100
Compression|6
XResolution|72/1
YResolution|72/1
ResolutionUnit|2
JpegIFOffset|1100
JpegIFByteCount|8432
```

Note that the EXIF tag- and it's appurtenant value is separated by a vertical bar (|). You can use this in your @formulas to separate the tag- and value.

Sample with separate fields:

If you specify the import option `ExtractEXIFToField:EXIF,SPLIT`, II2LN will create a separate text field for each EXIF tag. Below you see a screen dump from ***List Fields*** http://www.listfields.com, showing you the fields in a Notes document:



Fields		Documents
Fieldname	Type	Document 01
$Fonts	Richtext	Number of fonts in fonttable: 1 *** Font #1 Name=Verdana, Face=5, Family=38
$Revisions	Time/date list	04.02.2005 08:32:05 · 04.02.2005 08:32:36 · 04.02.2005 08:32:37 · 04.02.2005 08:32:37 · 06.02.200
$UpdatedBy	Text list	CN=Robert Ibsen Voith/O=VoithsCODE
Body	Richtext	
Category	Text	
EXIF_ColorSpace	Text	1
EXIF_ComponentsConfiguration	Text	Unknown Format
EXIF_CompressedBitsPerPixel	Text	15099488/3763200
EXIF_Compression	Text	6
EXIF_DateTime	Text	2002:07:27 11:29:11
EXIF_DateTimeDigitized	Text	2002:07:27 11:29:11
EXIF_DateTimeOriginal	Text	2002:07:27 11:29:11
EXIF_ExifImageHeight	Text	1680
EXIF_ExifImageWidth	Text	2240
EXIF_ExifInteroperabilityOffset	Text	976
EXIF_EXIFOffset	Text	210
EXIF_ExifVersion	Text	0210
EXIF_ExposureBiasValue	Text	6/6
EXIF_ExposureProgram	Text	2
EXIF_ExposureTime	Text	951/1000000
EXIF_FileSource	Text	x03
EXIF_Flash	Text	0
EXIF_FlashPixVersion	Text	0100
EXIF_FNumber	Text	40/10
EXIF_FocalLength	Text	713/100
EXIF_InteroperabilityIndex	Text	R98
EXIF_InteroperabilityVersion	Text	0100
EXIF_JpegIFByteCount	Text	4512
EXIF_JpegIFOffset	Text	1100

Info about ***Body*** - Very often is the *Body* field the main richtext field in a Notes document - but this is no enforced rule! This naming convention has been around since the dawn of Lotus Notes!

You see many EXIF fields.

ExtractIPTCToField

Description:

The ExtractIPTCToField import option instruct II2LN to extract any potential ***IPTC*** (see "Extract EXIF and IPTC information" on page 4) information in the image to either one field, or a separate field per IPTC tag- and value pair. This is very similar to the ***ExtractEXIFToField*** (on page 77) import option.

Declaration:

```
ExtractIPTCToField:<fieldname> [, <split>|both]
```

Parameters:

Param	Description
<fieldname>	The field name in your Notes document to store the multivalue text field, containing all the IPTC tag- and value pairs.
<split>	The optional SPLIT instruct II2LN to create a separate field per IPTC tag- and value pair. II2LN will use the <fieldname> as a prefix for all the IPTC tags. For example, if the <fieldname> is *IPTC*, the caption tag (IPTC tag 120) has this field name: `IPTC_Caption_Abstract` Note that all fields are text.
<both>	The optional BOTH instruct II2LN to both create one IPTC field *and* create a separate field for each tag

Notes:

The IPTC tags are referenced by its IPTC tag name. See ***The most common IPTC tags*** (on page 180) for a list of IPTC tags. Note that you also can use the IPTC names to modify IPTC values with the ***ReplaceIPTC,*** (see "ReplaceIPTC" on page 108) ***AppendIPTC*** (on page 69) and ***InsertIPTC*** (on page 88) import options.

Also see the ***ExtractEXIFToField-samples*** (see "Store tags and values in one field versus many fields" on page 79) to see how the data is stored when using one field versus many fields.

Below you see a sample of IPTC tags from an image:

```
Urgency|1
ObjectName|Ski slopes across winter landscape
Keyword|Ski slopes, Norway
SpecialInstructions|Contact Voith's CODE for
further instructions
DateCreated|20050207
City|Borgheim
ByLine|Robert Ibsen Voith
Credit|Robert Ibsen Voith
Headline|Ski slopes
OriginalTransmissionReference|II2LN
Country_Primary_LocationName|Norway
Writer_Editor|Robert Ibsen Voith
Caption_Abstract|Ski slopes in Norway, late
autumn
CopyrightNotice|Voith's CODE
```

Introduced in version:

1.1.0.0

Repeatable

No

ExtractImageVariablesToField

Description:

The ExtractImageVariablesToField import option instruct II2LN to store all basic (not IPTC and EXIF, which is controlled by separate ***ExtractEXIFToField*** (on page 77) and ***ExtractIPTCToFields*** (see "ExtractIPTCToField" on page 81) import options) information about the image in either one field, or a separate field per image variable tag- and value pair.

Declaration:

```
ExtractImageVariablesToField:<fieldname> [,
<split>|<both>]
```

Parameters:

Param	Description
<fieldname>	The field name in your Notes document to store the multivalue text field, containing all the image variables tag- and value pairs.
<split>	The optional SPLIT instruct II2LN to create a separate field per image variable tag- and value pair. II2LN will use the <fieldname> as a prefix for all the image variable tags. For example, if the <fieldname> is *"ImgVar"*, the image variable ORIGINAL_IMG_FILESIZE has this field name: `ImgVar_ORIGINAL_IMG_FILESIZE` Note that all fields are text.
<both>	The optional BOTH instruct II2LN to both create one image variable field *and* create a separate field for each tag

Notes:

Please refer to the ***The original image variables*** (on page 154) and ***The import image variables*** (on page 155) for a list of image variables. In these topics you will see that the variables has the format `$(Variable name)`, such as `$(ORIGINAL_IMG_CREATION_TIMESTAMP)` and `$(IMPORT_IMG_HEIGHT)`. When II2LN create real Notes fields with the variable names as real Notes field names, II2LN can't use the dollar and parenthesis characters, so they are removed from the name when we create field names.This means that we will have the variable names `ORIGINAL_IMG_CREATION_TIMESTAMP` and `IMPORT_IMG_HEIGHT`.

Also see the ***ExtractEXIFToField-samples*** (see "Store tags and values in one field versus many fields" on page 79) to see how the data is stored when using one field versus many fields.

Introduced in version:

1.5.0.0

Repeatable

No

FavorPhotoshop

Description:

When you extract- or manipulate EXIF and IPTC information, the images may use several techniques, protocols and tricks to store this information. By using this import option you tell II2LN to favor the way Adobe Photoshop uses to store and retrieve such information. Use this if you need compability with Photoshop.

Declaration:

```
FavorPhotoshop:1
```

Parameters:

Param	Description
1	Instruct II2LN to favor the Photoshop way of storing and retrieving meta data

Notes:

Introduced in version:

1.2.5.0

Repeatable

No

Flip

Description:

The Flip import option let you flip the image either horizontally or vertically.

Declaration:

```
Flip:<direction>
```

Parameters:

Param	Description
<direction>	Either H for horizontal flip or V for vertical flip, such as: `Flip:H` or `Flip:V`

Introduced in version:

1.1.0.0

Repeatable

Yes

ForceImportFilename

Description:

When II2LN work with modified files, it uses the original filename and file extension by default. Note that II2LN makes a copy of the original file, and places that in your temporary directory (which may be overridden by ***WorkingPath)*** (see "WorkingPath" on page 138). No modifications are ever done directly to the original file.

Therefore, if you decide to attach the import file with the ***AttachImportFile*** (on page 68) import option, it will normally have the original name and extension.

If you want to use a specific file name , you can instruct II2LN to do so, by using the ForceImportFilename import option.

Declaration:

```
ForceImportFilename:<filename>
```

Parameters:

Param	Description
<filename>	The file name to be used when attaching

Notes:

Note that the import file always will retain its original file extension. For example, if the original file `C:\Temp\Smily.gif` is imported, and you specify `ForceImportFilename:Image`, the actual file imported will be `Image.gif`. Note the same file extension.

Introduced in version:

1.0.0.0

Repeatable

No

ImpersonateAs

Description:

If you run II2LN on a Domino server, you need to specify a Notes name to impersonate as. This is required by the internal DXL processing within II2LN.

Declaration:

```
ImpersonateAs:<Person or Org Name>
```

Parameters:

Param	Description
`<Person or Org Name>`	The to impersonate as, on the format <Name/Organization>. For example; John Doe/Acme -or- NotesSrv1/Acme

Notes:

When you run II2LN on a Notes client, it will automatically use the currently logged in Notes User ID. This is not possible on a server, so II2LN has to impersonate an ID.

Introduced in version:

1.1.0.0

Repeatable

No

InsertIPTC

Description:

Insert the specified value in front of any existing IPTC tag content. In other words, the new value will be prefixed before any existing values.

If the specified tag doesn't exist, it will be created.

Declaration:

```
InsertIPTC:<tag name>, <tag value>
```

Parameters:

Param	Description
<tag name>	The IPTC tag name to insert the specified value to
<tag value>	The value to replace to the tag name

Notes:

Please see ***here*** (see "Manipulate EXIF and IPTC information" on page 5) for a description of the Replace, Append and Insert logic regarding EXIF and IPTC values

Introduced in version:

1.2.5.0

Repeatable

Yes

LeaveDXLBackup

Description:

When II2LN prepare to import- or attach an image to a document, it first create a backup of the document. The backup is in the form of a DXL import file. If II2LN processing succeeds, the backup file is automatically deleted. If an error occurs, II2LN will try to rollback to the original document, so no harm has been inflicted on the original document. If you want to keep the DXL backup file, you can specify the LeaveDXLBackup import option.

Declaration:

LeaveDXLBackup:<Yes>

Parameters:

Param	Description
<Yes>	Specify YES or 1 to make II2LN leave the DXL backup file after successful processing

Notes:

Note that the DXL backup file is always placed in your temp directory. If you want to specify a specific place for this file, and all other temporary files, use the ***WorkingPath*** (on page 138) import option.

Introduced in version:

1.0.0.0

Repeatable

No

LoadTIFFFrame

Description:

TIFF images may contain several separate images within one single file. Each image is stored in a *frame*. With the LoadTIFFFrame import option you can specify which page (and thus which image) to load.

Declaration:

LoadTIFFPage:<PageNumber>

Parameters:

Param	Description
<PageNumber>	Starting with 0 for the first image and up to the maximum image number

Notes:

Use the function ***GetTIFFFrameCount*** (on page 55) to retrieve the number of frames within a TIFF file.

Introduced in version:

1.3.0.0

Repeatable

No

OverlayImage

Description:

The OverlayImage let you overlay an image of your choosing on the specified image. This is very similar to the ***OverlayText*** (see "WriteText" on page 96) import option. You control placement, optional resizing and transparency.

This import option has much more control over the overlay operation. If you want to add a photo-frame to your image, take a look at the ***PhotoFrame*** (on page 102)-import option.

Declaration:

```
OverlayImage:<filename> [, <where>,
<transparent>, <X>, <Y>, <ResizeMethod>,
<transparent color>, <tolerance>
```

Parameters:

Param	Description
<filename>	Full path and file name to overlay image. Note that this can be an entirely other image format than the original image. If this is the only parameter you specify, the overlay image will retain its original size, and be placed in the Middle/Middle position. No transparency is controlled at all. Starting from version 1.4 of II2LN you can also specify any `http://` or `file://` URL in this parameter
<where>	Control where you want to place the image. The placement number range from 0 (Upper/Left) to 8 (Bottom/Right), with the exact same logic as the <where> parameter for the OverlayText. See ***The <where> parameter*** (on page 98).
<transparent>	From 0 (invisible) to 100 (solid). See ***The <transparent> parameter*** (on page 100) on OverlayText.

<X> and <Y>	If you want to resize the overlay image to fit within the X and Y parameters, specify <X> and <Y>. Note that the overlay image is resized with the simplest resize method unless you additionally specify the next parameter. See more on X and Y specification below.
<ResizeMethod>	If you specify the <X> and <Y> you can optionally impose a resize method for the overlay image. If the value is SLAVE, you will use the same resize method as the import image. The resize method number is the same as you set with the ***SetResizeMethod*** (see "ResizeMethod" on page 122) import option.
<transparent color>	If your overlay image contains areas that should be transparent on the import image, you can use this parameter to specify the color that should act as a transparent color. Specify the color according to the ***How to specify colors ?*** (on page 161).
<tolerance>	If you specify the <transparent color> parameter above, you can even specify the color-tolerance to use. The default value is 0 which means strict color matching. If you for example specify the color Magenta (FF00FF), this can be the one and only color being used as a transparency color. If you increase the <tolerance>, you can embrace more and more colors similar to FF00FF. The upermost limit is 65535, which means ALL colors are used, and the overlay image then becomes completely transparent!

More on how to specify the <X> and <Y> parameters.

If you specify X and Y parameters equal to -1, you indicate to II2LN to resize the overlay image to the same size as the original image. In other words, the overlay image will overlay the complete imported image.

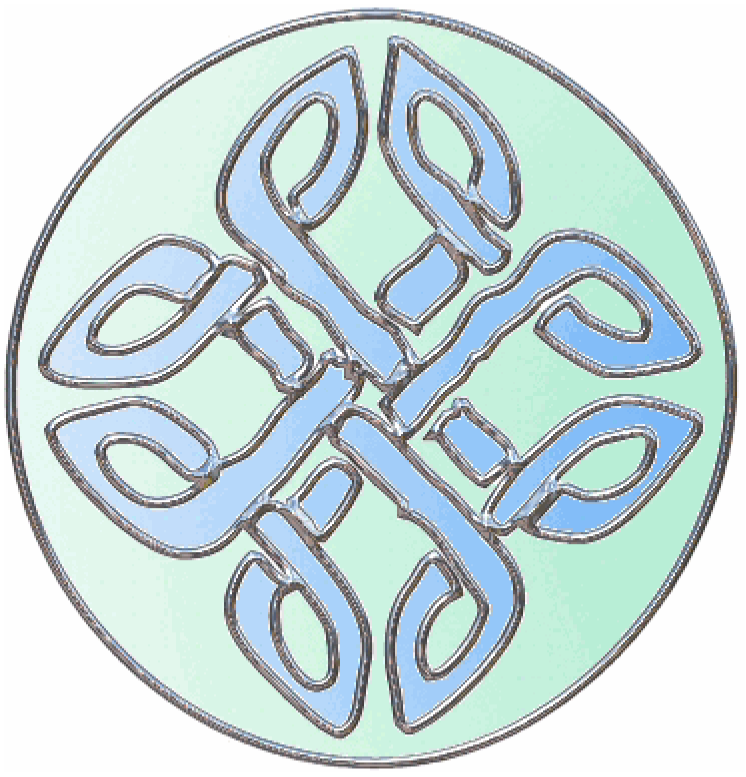
If you specify X and Y parameters equal to 0, you indicate to II2LN to use the original size of the overlay image. You may need to do so, in order to being able to specify some of the other optional parameters.

A sample:

We have the original digital camera JPG image with original dimensions 2240 x 1680:

The image above will be resized to 200 by 200 pixels.

Then we have a logo we want to superimpose onto the image above. The logo is a GIF file with original dimensions 374 x 384:

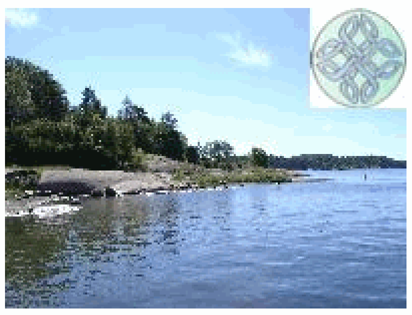


As you see the logo is circular, with some white space around the logo.

Our first attempt to overlay the logo on the image uses this import options string:

```
"Resize:200,200;OverlayImage:c:\temp\Logo.gif,2,1
00,50,50,7"
```

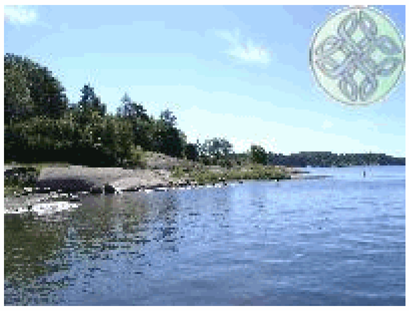
The import option string above will first resize the original image, and then overlay the logo. The OverlayImage will place the logo in the ***upper right corner (2)*** (see "The <where> parameter" on page 98), have ***solid transparency*** (see "The <transparent> parameter" on page 100) on, and resize the logo to 50 by 50 pixels with ***resize method*** (see "ResizeMethod" on page 122) 7. Our first result looks like this;



The logo is resized and placed in the upper right corner, but the white space in the logo GIF file is still visible. We specify the additional <transparency color> parameter and end up with an import option string like this:

```
"Resize:200,200;OverlayImage:c:\temp\Logo.gif,2,1
00,50,50,7,FFFFFF"
```

The result looks like this;

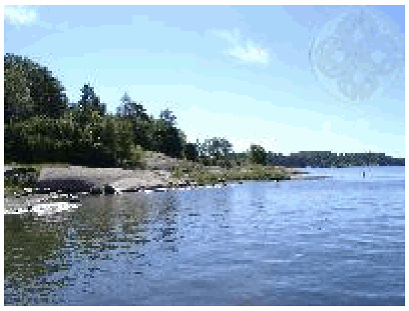


Now the white space around the logo has been transparent! Cool!

The final test adjust the <transparent> parameter to impose a more professional look for the logo:

```
"Resize:200,200;OverlayImage:c:\temp\Logo.gif,2,2
5,50,50,7,FFFFFF"
```

Below you see the result of the import options above;



Introduced in version:

1.2.0.0

Repeatable

Yes

WriteText

Description:

The OverlayText (also known as WriteText) import option let you superimpose a text onto the image at import- or attachment time. You completely control what and where to write the text.

Declaration:

```
OverlayText:<where>, <font name>, <font size>,
<text> [, <color>, <transparency>, <X>, <Y>]
```

Parameters:

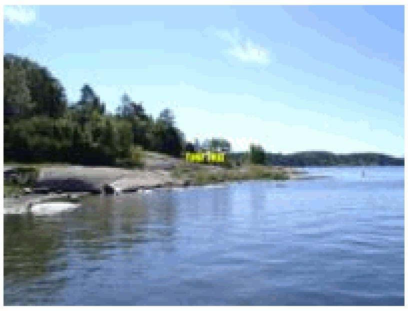
Param	Description
<where>	From 0 to 8. See ***The <where> parameter*** (on page 98) below
<font name>	Any available font name. Note that if II2LN is installed on a server, you must select a font available on the server
<font size>	Any font size in pixels
<text>	The text to write on the image. Note, if the text contains semi-colon, colon or comma, use the codes `&semicolon;`, `&colon;` and `&comma;` repectivly. These codes will be replaced by II2LN.
<color>	Use any RGB color on the II2LN ***color format*** (see "How to specify colors ?" on page 161).
<transparency>	From 0 (completely transparent, ie. invisible!) to 100 (solid, no transparency at all). See ***The <transparent> parameter*** (on page 100) below.
<X> and <Y>	Force exact positioning of the text. If specified, it overrides the <where> parameter.

Notes:

Note that the sequence of OverlayText in the complete string of import options is very important. If you for example specify the OverLay text before the ***Resize*** (on page 112) import option, you will overlay the text on the original, and potentially huge image. The text will therefore look very small. Below you see a sample with an ordinary digital camera image, where we import the image specifying the import option string:

```
"OverlayText:4,Impact,72,Your
Text,FFFF00;SetResizeMethod:7;Resize:200,200"
```

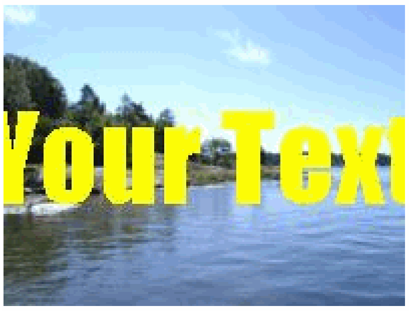
The import string above, will *first* overlay the text "Your text" with font *Impact* and font size *72* on the image. *Then*, it will resize the image and the result is that the overlayed text looks very small:



Oops, 72 point text looking that small ?!? That is of course because 72 points on a full scale digital camera image isn't really that big.

In the following sample, we have just re-ordered the import options to;

```
"SetResizeMethod:7;Resize:200,200;OverlayText:4,I
mpact,72,Your Text,FFFF00"
```



Now the text gets it real size in 72 points, because the image was resized to 200 by 200 pixels first!

Introduced in version:

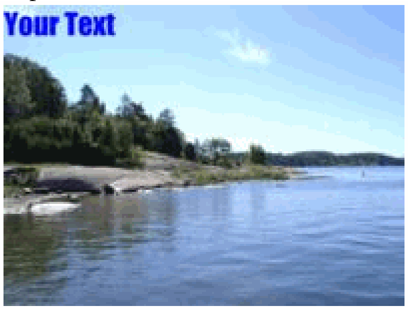
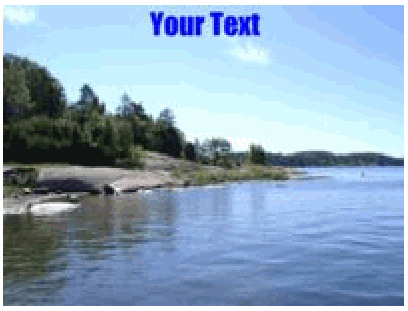
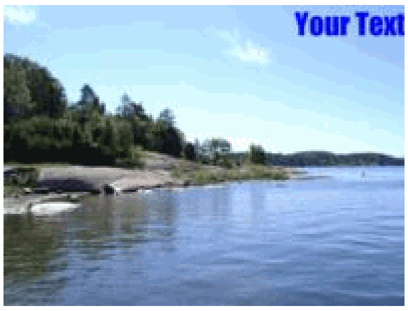
1.0.0.0

Repeatable

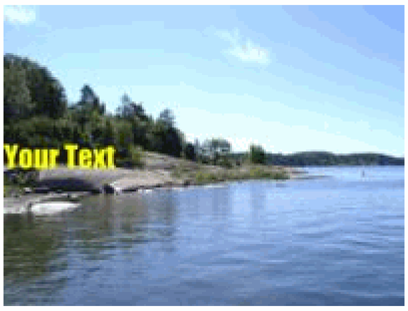
Yes

The <where> parameter

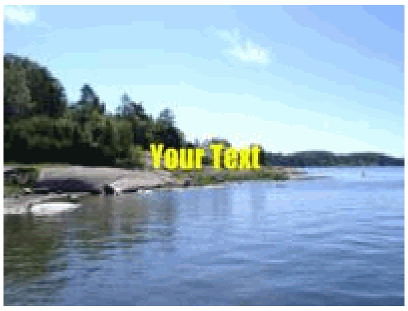
The <where> parameter determine where the text is placed on the image. Use the numbers below to place the text on the image.

Where	Description
0	Top/Left 
1	Top/Middle 
2	Top/Right 

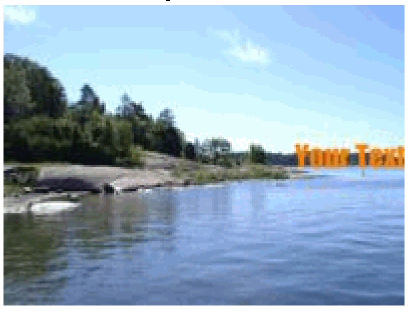
3 Middle/Left



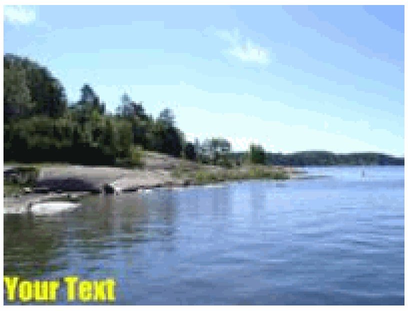
4 Middle/Middle

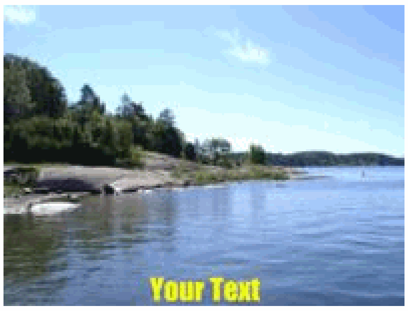
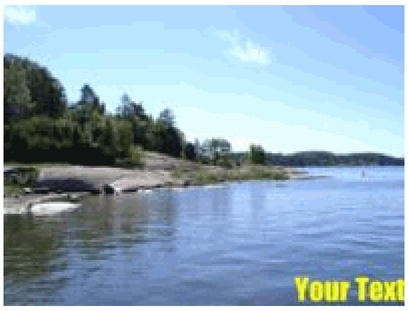


5 Middle/Right



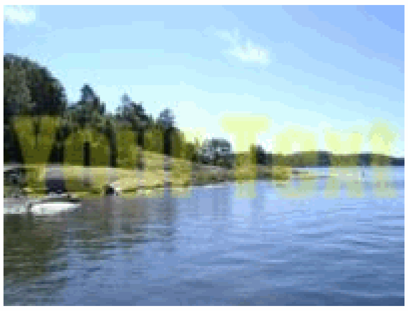
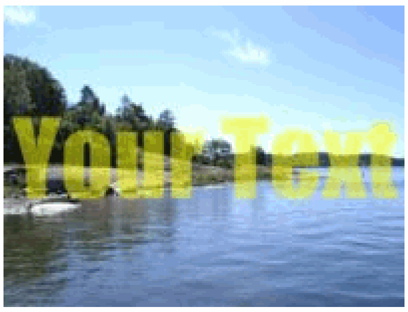
6 Bottom/Left



7	Bottom/Middle 
8	Bottom/Right 

The <transparent> parameter

Some <transparency> samples

Transp.	Sample
25	
50	

75	
100	

OutputSize

Description:

Images imported as ***Notes Embedded Image*** (see "Notes Embedded Image - what is it ?" on page 166) has two methods of setting the size of the visible image. First and foremost, you can decide to really resize the image with the ***Resize*** (on page 112) import option. This alters the imported image permanently. If you rather want to keep the original image intact, you can resize only the visible image. This is done with the OutputSize import option.

Declaration:

```
OutputSize:<X>,<Y>
```

Parameters:

Param	Description
<X>	The width of the resized image
<Y>	The height of the resized image

Notes:

Resizing the image with OutputSize has similar effect to the manual operation of resizing. If you paste an image into a rich text field, and select the image, you will see small grippers in each corner of the image. Click and hold down the mouse button to resize the image visually. The image looks resized, but it isn't.

Also note that the <X> and <Y> is exposed as variables, the `$(IMPORT_WIDTH)` and `$(IMPORT_HEIGHT)` respectively.

Introduced in version:

1.0.0.0

Repeatable

No

PhotoFrame

Description:

The Photoframe import option will allow you to stamp a frame or border image ontop of your image. This is similar to the ***OverlayImage*** (on page 91) import option, but this import option will rather use the transparency within the frame image itself. This means that you can easily use existing frames or create your own transparent frame images with tools like Photoshop or Paint Shop Pro!

This import option can help you make real cool images for your photoframe shows!

Declaration:

```
Photoframe:<file name of frame image>
```

Parameters:

Param	Description
<file name of frame image>	Full path and file name of the frame image to use

Notes:

Since a photo frame is an image with transparent parts by nature, only some file types would normally be used with this import option. The most popular format now is PNG. Note that the original image that you will stamp your frame onto, can be of any kind!

Please note that the current version of this import option will simply resize the photo frame image according to the original image. This means that you will get pretty distorted frames if you use a small, portrait like photo frame on a huge wide-screen image.

Tools like Paint Shop Pro contains several photo borders that you can use as basis for your own borders.

1. Create a transparent raster-based image close to the size of your original image. Why not create a set of photo-frames with the suffix 800x600, 1024x768 etc, and perhaps also with an identifier of portrait or landscape? In PSP, use for example "Image dimensions" equal to 800 x 600, and choose "Image characteristics" equal to *Raster Background*. Finally choose "Color" equal to *Transparent*.
2. When you have your new, transparent image, choose the menu *Image -> Picture frame ...*
3. Choose any frame of your liking, but be sure to choose *Frame inside of your canvas*. This will make PSP keep your original size of 800 x 600.
4. Use the PNG Optimizer to export the transparent image. Choose *File -> Export -> PNG Optimizer*. On the second tab named Transparency, choose *Single Color Transparency* and *Existing image or layer transparency*.

You photo-frame is ready to be used!

Introduced in version:

1.7.0.0

Repeatable

Yes

ProcessTimeField

Description:

If you want to measure the speed of II2LN, you can have II2LN create a number field in your document. The ProcessTimeFields import option specify the name of the field

Declaration:

```
ProcessTimeField:<fieldname>
```

Parameters:

Param	Description
<fieldname>	The Notes field name holding the processing time as a decimal number.

Notes:

Note that the field type always will be Number.

This import option is useful if you want to track the differences in processing time when you experiment with import options.

Introduced in version:

1.0.0.0

Repeatable

No

Reflection

Description:

The Relection import option add an image reflection to the image.

Declaration:

```
Reflection:<SurfaceColor>,<ReflectYPos>,<Reflecti
onHeight>,<ReflectionRatioStart>,<ReflectionRatio
End>,<FadeMode>,<TextureIntensity>,<TextureScale>
```

Parameters:

Param	Description
<SurfaceColor>	The surface color which the reflection will blend with. Use any RGB color on the II2LN ***color format*** (see "How to specify colors ?" on page 161). Default color is white (FFFFFF)
<ReflectYPos>	Vertical position of the reflection on the source image, measured in percent. Valid values 1 to 100. If you specify 70, the reflection will start 70 % down on the image. Default value is 75
<ReflectionHeight>	The height of the reflection area, in either pixels or percent. You specify pixels by suffixing with "px", such as "200px". Percent values are valid from 1 to 100. Note that the reflection height will be added to the existing height of the image. Default value is 25 of the image height
<ReflectionRatioStart>	Where do the fade between the reflection and the surface color start? Values are specified as decimal numbers between 0.0 and 1.0, where 0.0 means to edge, and 1.0 bottom edge. The number can even be negative, which means that the reflected image will always have some of the surface color in it. This value should be lower than the RefectionRatioEnd value. Default value is 0.25

<ReflectionRatioEnd>	Where do the fade between the reflection and the surface color end? This value should be greater than RefelectionRatioStart. Values are specified as decimal numbers between 0.0 and 1.0, where 0.0 means to edge, and 1.0 bottom edge. Default value is 0.75
<FadeMode>	How should the fading between the reflection and the surface be processed? You may specify the following values; 0 = Linerar 1 = Sine, fast 2 = Sine, slow Default value is 1
<TextureIntensity>	The intensity of the applied texture. Values are specified as decimal numbers between 0.0 and 1.0. Default value is 0.0
<TextureScale>	The size of the texture applied to the reflection. Default value is 10

Introduced in version:

1.6.0.0

Repeatable

Yes

ReplaceEXIF

Description:

Replace the specified value for the specified EXIF tag. In other words, any existing value will be overwritten - and lost.

If the specified tag doesn't exist, it will be created.

Declaration:

```
ReplaceEXIF:<tag name>, <tag value>
```

Parameters:

Param	Description
<tag name>	The EXIF tag name to replace the value
<tag value>	The new value, replacing any existing value

Notes:

Please see ***here*** (see "Manipulate EXIF and IPTC information" on page 5) for a description of the Replace, Append and Insert logic regarding EXIF and IPTC values

Introduced in version:

1.2.5.0

Repeatable

Yes

ReplaceIPTC

Description:

Replace the specified value for the specified IPTC tag. In other words, any existing value will be overwritten - and lost.

If the specified tag doesn't exist, it will be created.

Declaration:

```
ReplaceIPTC:<tag name>, <tag value>
```

Parameters:

Param	Description
<tag name>	The IPTC tag name to replace the value
<tag value>	The new value, replacing any existing value

Notes:

Please see ***here*** (see "Manipulate EXIF and IPTC information" on page 5) for a description of the Replace, Append and Insert logic regarding EXIF and IPTC values

Introduced in version:

1.2.5.0

Repeatable

Yes

ReplaceWithVariable

Description:

ReplaceWithVariable indicate to II2LN that the imported image should replace a variable in the rich text field. This is a powerful feature which enables you to use ***document templates*** already containing information.

ReplaceWithVariable also enables II2LN to replace variables in any field in the document

Declaration:

```
ReplaceWithVariable:<Variable>
```

Parameters:

Param	Description
<Variable>	The variable to search for in the rich text field. If you for example specify ***$(Image)***, then II2LN would replace that variable with the image

Notes:

Below the effect of the ReplaceWithVariable import option is demonstrated:

1 The original image is a car-picture

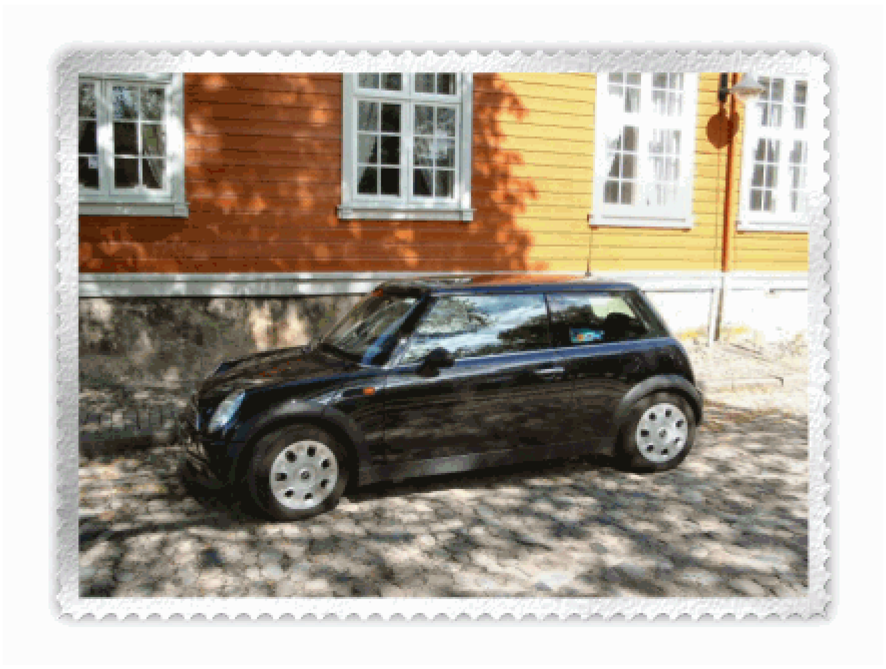
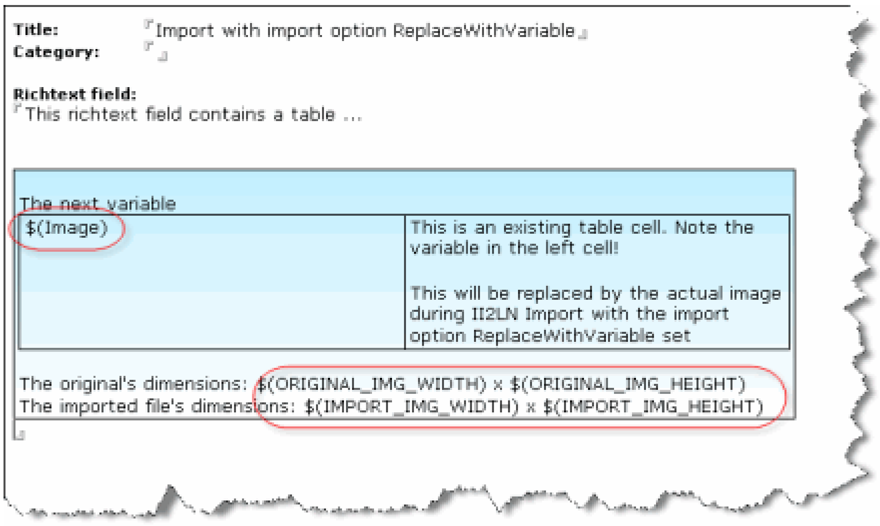


Figure 3: The Mini - Original Image

2 Then we have an Notes document with some content in the rich text field



Title: Import with import option ReplaceWithVariable
Category:

Richtext field:
This richtext field contains a table ...

The next variable

$(Image)	This is an existing table cell. Note the variable in the left cell! This will be replaced by the actual image during II2LN Import with the import option ReplaceWithVariable set

The original's dimensions: $(ORIGINAL_IMG_WIDTH) x $(ORIGINAL_IMG_HEIGHT)
The imported file's dimensions: $(IMPORT_IMG_WIDTH) x $(IMPORT_IMG_HEIGHT)

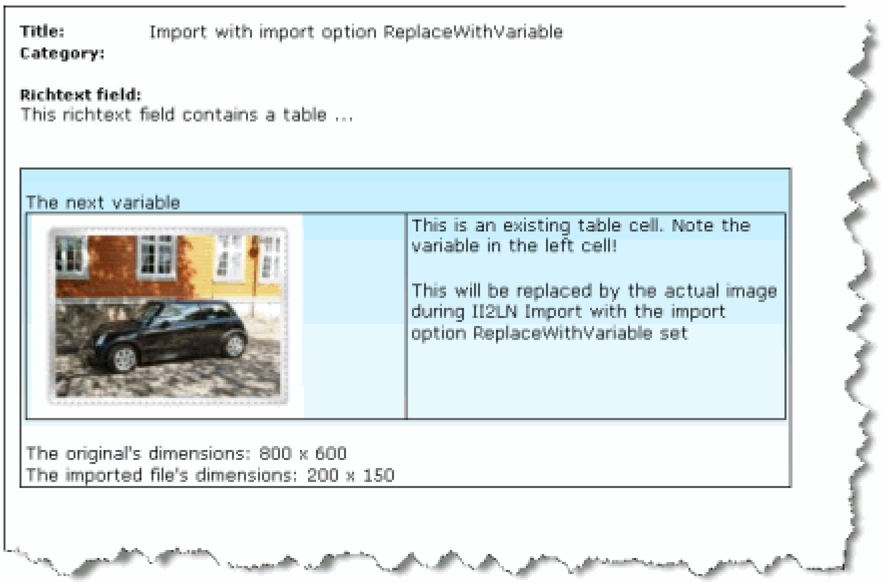
Note the use of variables in the image above. All variables will be replaced by actual content during import

3 Import the image with the following ***ImportImage*** (on page 35) call and the ReplaceWithVariable set

```
lRc = ImportImage(_db.Server,_
db.FilePath,_
doc.UniversalID, _
"Body",_
strFileName,_
"ReplaceWithVariable:$(Image);Resize:200,200")
```

The LotusScript code above will import the image with the filename stored in *strFileName*, into the *Body* rich text field. Note how we specify that we want to replace the $(Image) variable. Finally we ***resize*** (on page 112) the image during import to 200 x 200 pixels

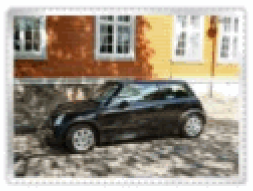
4 Below you see the nice result!



Title: Import with import option ReplaceWithVariable
Category:

Richtext field:
This richtext field contains a table ...

The next variable

	This is an existing table cell. Note the variable in the left cell! This will be replaced by the actual image during II2LN Import with the import option ReplaceWithVariable set

The original's dimensions: 800 x 600
The imported file's dimensions: 200 x 150

Note that the $(Image) variable has been replaced by the imported (and reszied) image. Additionally the four image-variables at the bottom has been replaced by their real values! See more on available variables ***here.*** (see "Variables" on page 153)

Introduced in version:

1.0.0.0

Repeatable

No

Resize

Description:

One of the more powerful import options in II2LN is Resize. With this import option you can reduce or enlarge an image to fit within specified X and Y rectangles. You can for example specify that an image should resize to 200 by 200 pixels. II2LN will automatically find the best fit for your image, without loosing the aspect ratio. In other words, your image will never become stretched or compressed because you didn't set the X and Y correctly.

Note that you can ensure that the Resize import option don't enlarge the image, if it is smaller than the specified width and height you specify. You control this with the ***SetResizeNoEnlargement*** (on page 123) import option.

Declaration:

```
Resize:<X>,<Y>
```

Parameters:

Param	Description
<X>	The maximum width of the resized image
<Y>	The maximum height of the resized image

Notes:

All image manipulation ***may*** reduce the quality of the image. If you for example reduce a large image to a small thumbnail, you might feel that the thumbnail looks "harsh" or "bad". Most modern image processing systems therefore let the user choose which resizing method to use, offering ***high quality, but slow***-methods in one end, and ***not-so-good quality, but quick***-methods in the other end. II2LN has several resize methods, so you can choose quality before speed or vise versa. See more about resize methods ***here.*** (see "ResizeMethod" on page 122)

Note that you also has the opportunity to set the Dots Per Inch (DPI) for an image. This is controlled with the ***SetDPI*** (on page 116) import option.

Note that you can use the "find best fit within specified dimensions"-logic to ensure that images always have equal width or height. This is accomplished by specifying a huge number for the value you don't care about. For example; if you want all thumbnails to have an equal width of 80 pixels, you can use this resize specification:

```
Resize:80,9999
```

Since the resize import option will calculate the best fit within the rectangle 80 x 9999, the image will have 80 pixels width and unknown height. Again, the resize operation won't destroy the aspect ration.

Introduced in version:

1.0.0.0

Repeatable

No

ResizeMetric

Description:

II2LN already know how to resize your images with pixels and the ordinary ***Resize*** (on page 112) import option takes care of this. Sometimes you need to resize the image to a certain print size, such as 10 x 15 cm. photograph. The ResizeMetric import option has been added to make that happen. Just as for the Resize import option, II2LN will find the best fit within the specified boundaries without loosing aspect ratio.

Note that you set the ***DPI*** (see "Dots Per Inch - DPI" on page 160) directly as one of the parameters to this import option.

Declaration:

```
ResizeMetric:<X>,<Y>, <DPI>
```

Parameters:

Param	Description
<X>	The maximum width of the resized image in millimeters
<Y>	The maximum height of the resized image in millimeters
<DPI>	The Dots Per Inch in both the X and Y direction

Notes:

This import option will override the ***Resize*** (on page 112) and ***SetDPI*** (on page 116) import options.

This import option will automatically update the EXIF tags controlling X- and Y resolution.

Introduced in version:

1.2.5.0

Repeatable

No

Rotate

Description:

The Rotate import option let you rotate the image at any angle. You may also specify the background color to fill in the blank areas resulting from a rotation.

Remember that you also can specify the ***AutoOrientation*** (on page 73) import option, to auto-rotate the image into correct orientation.

Declaration:

```
Rotate:<angle> [, <background color>]
```

Parameters:

Param	Description
<angle>	Angle in degrees. Use integers. For example: `Rotate:13` will tilt the image 13 degrees
<background color>	The background color of the areas that are left blank as the result of a rotation. Read more about how to specify colors in II2LN ***here*** (see "How to specify colors ?" on page 161).

Notes:

Note that rotation to 0, 90, 180 and 270 degrees are considerably faster than other angles. Also note that the direction of rotation is towards right, meaning II2LN follows the 360 degrees circle.

II2LN will optimize multiple Rotate import options, so only *one* real rotation is performed. Finally the Rotate import option will be executed after all other import options.

Introduced in version:

1.2.0.0

Repeatable

Yes

SetDPI

Description:

You can force the DPI (Dots Per Inch) image resolution to be set for the import file.

Declaration:

SetDPI:<DPI>

Parameters:

Param	Description
<DPI>	`The DPI for both both X and Y dimensions. For example;` `SetDPI:72 or SetDPI:300`

Notes:

Please see ***this*** (see "Dots Per Inch - DPI" on page 160) chapter for a deeper description of DPI.

Introduced in version:

1.2.4.0

Repeatable

No

SetGhostscriptParameters

Description:

Use this import option to add additional Ghostscript parameters. You have the ability to tweak the output almost to your content, and with this import option you can add additional Ghostscript-options and thus control the conversion in detail.

Declaration:

SetGhostscriptParameters:<parameters>

Parameters:

Param	Description
<parameters>	The additional Ghostscript parameters

Notes:

In order to utilize this import option you must download and install Ghostscript according to the step-by-step list ***here*** (see "Support for EPS, PS and PDF files - Ghostscript Support" on page 170).

One typical usage of this import option is because you want to enhance the quality of the conversion of a PDF file. By experimenting we have found that the following parameters are fine - especially if you also use the II2LN's Resize import option at the same time;

```
-r300 -dTextAlphaBits=4 -dGraphicsAlphaBits=4
```

As you see, the parameters are entered exactly as they would be entered on a manual GSWIN32C.EXE usage. The parameters that II2LN transfer by default is:

```
-dSAFER -q -dNOPAUSE -dBATCH -sDEVICE=jpeg
```

Refer to the GhostScript documentation or search for "Ghostscript commandline options" to learn more about the available options.

A nice tricks to experiment with Ghostscript parameters is therefore to use GSWIN32C.EXE directly in a Command Prompt Session (run CMD.EXE). You could for example use this parameters:

```
gswin32c -dSAFER -q -sPAPERSIZE=a4 -r600 -
dJPEGQ=100 -dTextAlphaBits=4 -
dGraphicsAlphaBits=4 -dNOPAUSE -dBATCH -
sDEVICE=jpeg -sOutputFile=OutputJPEGFileName.jpg
InputPDFFileName.pdf
```

Introduced in version:

1.2.5.0

Repeatable

No

SetGhostscriptProcessor

Description:

Normally II2LN will try to load the existing Ghostscript installation on the machine where II2LN is installed. Normally this imply that the Ghostscript processor command line tool `GSWIN32C.EXE` is placed in a directory specified in the environment variable PATH. However, if this isn't an option, you may specify the exact path and file name with this import option

Declaration:

SetGhostscriptProcessor:<path and file name>

Parameters:

Param	Description
<path and file name>	Full path and file name to the Ghostscript processor

Notes:

In order to utilize this import option you must download and install Ghostscript according to the step-by-step list ***here*** (see "Support for EPS, PS and PDF files - Ghostscript Support" on page 170).

Introduced in version:

1.2.5.0

Repeatable

No

SetLogFilename

Description:

By default II2LN logs to a special field in the document; `VCII2LN_Log`. By specifying the SetLogFilename import option, you also direct the log to a specified file. One super addition with the log file, is that you can really see how much time each part of the II2LN processing takes.

Declaration:

SetLogFile:<log file name>

Parameters:

Param	Description
<log file name>	Full path and filename to the log file. For example `C:\Temp\VCII2LN.Log`

Notes:

Note that you can determine how much get written to the log by using the ***SetLogLevel*** (on page 121) import option.

Introduced in version:

1.0.0.0

Repeatable

No

SetLogLevel

Description:

By specifying the SetLogLevel import option, you can control how much information that is written to the log field (`VCII2LN_Log`), and optionally written to the log file specified by the ***SetLogFilename*** (on page 120) import option.

The range is from 0, which means no logging at all, to 5, which dumps debug-like information. Log level 3 is normal and default value in II2LN

Declaration:

SetLogLevel:<level>

Parameters:

Param	Description
<level>	From 0 to 5. 3 Is default

Notes:

Introduced in version:

1.0.0.0

Repeatable

No

ResizeMethod

Description:

II2LN can ***resize*** (on page 112) an image during import or attachment. To control the quality and speed of the resize operation, you can specify the SetResizeMethod import option.

It has 7 different resize methods, ranging from 0, *fastest, but no-so-good quality* to 7, *high quality, but slow.*

The default value is -1 (or equal to not specifying SetResizeMethod at all). The default resize method is very quick, but not with very high quality. The smaller resize operation you perform, the lesser quality!

Declaration:

```
SetResizeMethod:<method number>
```

Parameters:

Param	Description
<method number>	From 0 to 7

Notes:

The available resize methods are:

Method	Description
-1	Very simple resize, the default. Very quick, not so good quality
0	Simple resize
1	Bi-cubic resize
2	Simple/Decimate Resize
3	Bicubic/Decimate Resize
4	Resample Resize
5	Lanczos Filter Image Resize
6	Catrom Filter Image Resize
7	Mitchell Filter Image Resize

Introduced in version:

1.1.0.0

Repeatable

No

SetResizeNoEnlargement

Description:

II2LN can ***resize*** (on page 112) an image during import or attachment. This is controlled with the ***Resize*** (on page 112) and ***SetResizeMethod*** (see "ResizeMethod" on page 122) import options. But what happen when you have a small image and instruct it to resize to something which is *bigger* than the original image?? II2LN will then by default *enlarge* the image to the desired size. Sometimes this is ok, and sometimes its not. With the SetResizeNoEnlargement, you can now instruct II2LN to to enlarge images if the original image is smaller than the size you specify with Resize.

Declaration:

```
SetResizeNoEnlargement:<on>
```

Parameters:

Param	Description
<on>	Specify 1 to enable this import option and 0 to disable it. The default setting is 0

Notes:

Introduced in version:

1.5.0.0

Repeatable

No

SetWebGrabBrowserSize

Description:

When you specify an URL as a image file name, II2LN will connect to the internet and retrieve the URL as an image

By using this import option you can simulate a certain size of the browser window when the snapshot is retrieved. This can be nice to use if you want to see how the web pages look when shown in a browser with for example 800 or 1024 pixels width.

Declaration:

SetWebGrabBrowserSize:<width>,<height>

Parameters:

Param	Description
<width>	The width of the browser window, in pixels
<height>	The height of the browser window, in pixels

Notes:

Note that if the page doesn't fill the complete specified window size, II2LN will add blank space with the background color of the web page. This may give strange results if the web site is fully controlled with CSS or Flash.

Introduced in version:

1.4.0.0

Repeatable

No

SetWebGrabClipRect

Description:

When you specify an URL as a image file name, II2LN will connect to the internet and retrieve the URL as an image.

If you just want to grab a certain peace of a web page, you can use this import option to specify the rectangle you want to "clip out", thus "ClipRect"

Declaration:

SetWebGrabClipRect:<X>,<Y>,<Width>,<Height>

Parameters:

Param	Description
<X>	The X parameter in pixels
<Y>	The Y parameter in pixels
<Width>	The width in pixels
<Height>	The height in pixels

Notes:

Introduced in version:

1.4.0.0

Repeatable

No

SetWebGrabDelay

Description:

When you specify an URL as a image file name, II2LN will connect to the internet and retrieve the URL as an image

By using this import option you can define the delay-time II2LN should wait before it attempt to grab a web page. This is useful if you have a page which you know has to load certain slow or huge elements such as Flash presentations.

Declaration:

SetWebGrabDelay:<millisec>

Parameters:

Param	Description
<millisec>	The delay specified in milli seconds

Notes:

Use this import option in conjunction with SetWebGrabEableActiveX to grab pages containing Flash.

Introduced in version:

1.4.0.0

Repeatable

No

SetWebGrabEnableActiveX

Description:

When you specify an URL as a image file name, II2LN will connect to the internet and retrieve the URL as an image

By using this import option you can indicate to II2LN that the web page you are about to grab, contain Flash presentations. If you try to grab a page containing Flash without this option, II2LN will only display an empty element.

Declaration:

SetWebGrabEnableActiveX:<on>

Parameters:

Param	Description
<on>	Specify 1 to enable this import option and 0 to disable it. The default setting is 0

Notes:

Introduced in version:

1.4.0.0

Repeatable

No

SetWebGrabEnableJava

Description:

When you specify an URL as a image file name, II2LN will connect to the internet and retrieve the URL as an image

By using this import option you can indicate to II2LN that the web page you are about to grab, contain Java. If you try to grab a page containing Java without this option, II2LN will only display an empty element.

Declaration:

SetWebGrabEnableJava:<on>

Parameters:

Param	Description
<on>	Specify 1 to enable this import option and 0 to disable it. The default setting is 0

Notes:

Introduced in version:

1.4.0.0

Repeatable

No

SetWebGrabEnableScript

Description:

When you specify an URL as a image file name, II2LN will connect to the internet and retrieve the URL as an image

By using this import option you can indicate to II2LN that the web page you are about to grab, contain scripts. If you try to grab a page containing scrips without this option, II2LN may only show empty elements.

Declaration:

SetWebGrabEnableScript:<on>

Parameters:

Param	Description
<on>	Specify 1 to enable this import option and 0 to disable it. The default setting is 0

Notes:

Introduced in version:

1.4.0.0

Repeatable

No

SetWebGrabTimeout

Description:

When you specify an URL as a image file name, II2LN will connect to the internet and retrieve the URL as an image

By using this import option you can define the maximum time II2LN should process a web page. This is nice to do if you try to work with huge and/or slow web pages.

Declaration:

SetWebGrabTimeout:<millisec>

Parameters:

Param	Description
<millisec>	The timeout specified in milli seconds

Notes:

If the timeout time is reached (the web grabbing times out ...), II2LN will grab whatever which is currently visible on the page. This means that you may experience images missing certain elements.

Introduced in version:

1.4.0.0

Repeatable

No

SetWebGrabZoom

Description:

When you specify an URL as a image file name, II2LN will connect to the internet and retrieve the URL as an image

By using this import option you can set the zoom level from 1 (default) to 5. This normally means that an web page snapshot is enlarged.

Declaration:

SetWebGrabEnableZoom:<level>

Parameters:

Param	Description
<level>	Specify from 1 (default value) to 5

Notes:

Introduced in version:

1.4.0.0

Repeatable

No

StrictImageVerification

Description:

In order to protect itself from malfunction, II2LN always verify the specified image files. This is done by binary analysis of the bytes within the file, and thus II2LN can easily detect if a GIF file with a JPG file extension, really is a GIF file. Normally II2LN process most image types without problems.

If you have problems with your images, you might turn on even stricter image verification. If your image doesn't load, or ***GuessImageFormat*** (on page 56) reports "not a valid file", the file cannot be processed by II2LN.

Declaration:

StrictImageVerification:<OnOff>

Parameters:

Param	Description
<OnOff>	Specify YES or 1 to turn ON the strict image verification or NO or 0 to turn it off. This verification is by default OFF.

Notes:

The reason for this import option came August in 2007 when we received some JPG files which had crashed Notes. The images could not be opened in Photoshop or Paint Shop Pro, so there definitely was something wrong with the files. The supposed-to-be JPG files turned out to be TIFF files with some sort of embedded JPG encoding. After some research it turns out that there are several sub-JPG formats out there, with the similar construction. Since none of these TIFF-mixed-with-JPG is regarded as a standard, there are very limited support for them out there. As a curiosity Microsoft Vista went into an endless loop of crashing its main Windows Explorer process, when it tried to create a thumbnail view of the image while it was on the desktop. So not even Vista knew how to handle this image.

Introduced in version:

1.5.0.1

Repeatable

No

Sharpen

Description:

The Sharpen import option will process the image so it seems sharper.

Declaration:

```
Sharpen:<level>
```

Parameters:

Param	Description
<level>	From 0 (no sharpening which is the default) to 100 (much sharpening). For example: `Sharpen:50`

Notes:

It is difficult to foresee how much an image should be sharpened or not in advance. Clearly II2LN's sharpen logic must be used when your images is of the "same kind", regarding hue, saturation and colors.

Introduced in version:

1.2.0.0

Repeatable

Yes

UseImageResourceDatabase

Description:

II2LN has the ability to store images as ***image resources*** in Lotus Notes. This is typically used when you want to ***show images in Notes views*** (see "How to display my images in a Notes view" on page 162), and thus use the***ImportImageAsResource*** (on page 38) import option. There is however a limit to how many images that can be stored as resources, and the maximum number is in the range 5-7000 images, depending on the length of the field names you choose to use.

In order to overcome this limitation, one option is to spread the image resources on multiple databases. You only need to use multiple databases for the image resources themselves, and the original database retain its original design and function. The original database can thus refer to other databases for the resources. You can test this yourself by manually inserting a resource in any rich text field via the Notes client (use menu Create -> Insert image resource ...). You can insert resources from any database!

To alleviate this challenge, you can specify the *UseImageResourceDatabase* import option. II2LN will then completely automatically create and maintain the image resource databases for you.

Declaration:

```
UseImageResourceDatabase:<Database name prefix>
```

Parameters:

Param	Description
<Database name prefix>	The prefix for the database filenames created by II2LN. For example: `UseImageResourceDatabase:ImgRes`

Notes:

II2LN will use the prefix parameter when it create a new image resource database. Every database name will then have a sequence number as suffix. For example;

```
UseImageResourceDatabase:ImgRes
```

This will make the first database have the name ImgRes001.nsf. The next database will have the name ImgRes002.nsf and so on.

Also note that the image resource databases are created with the same Access Control Lists as the original database.

Introduced in version:

1.3.0.0

Repeatable

No

UseHTMLTemplate

Description:

II2LN contains several methods of adding content to your image, such as the ***OverlayImage*** (on page 91) and ***OverlayText*** (see "WriteText" on page 96) import options. While these options can do real cool stuff to your image, they just have a certain amount of flexibility. What if you want to just grab some cool HTML and CSS stuff of some web site, and with small modifications, process your image in that HTML/CSS? Meet UseHTMLTemplate!

With UseHTMLTemplate you will be able to feed your own image, alongside with any of your image meta data through the HTML template, and grab the result as an image.

Declaration:

```
UseHTMLTemplate:<Template subdirectory>
```

Parameters:

Param	Description
<Template subdirectory>	In order to keep it somewhat simple to specify which template to use, just specify the folder name of the template you want to use. See more in the Notes below

Notes:

A HTML template can be a single HTML file, or a multitude of files, such as HTML, CSS, JS, Flash compontens etc. Each available template is stored within a separate folder in the HTML Templates-sub directory in the ***II2LN program directory*** (see "The II2LN Program Directory" on page 23). In order to prepare and make enable a HTML page for use in II2LN, follow these steps.

1 Create from scratch or grab the HTML and all appurtenant files to a new subdirectory beneath the HTML Templates directory. Name the directory something you would like to refer the UseHTMLTemplate import option to later, such as for example Transparent Text Bottom.

2 Rename the main HTML file to Index.htm. Test that index.htm really works by launching it in your local browser. In order to see the desired effect, you would probably need to use some local image file and reference that directly in the img-tag with the `file:///` prefix.

3. Remove any left- or top white-space before proceeding. By default II2LN will clip the page to the processed image's size on pixel precession. So if you have left- and/or top-gutter (white space), the clipping may cut of the resulting image too much at the right- and bottom side. If you explicitly don't want II2LN to clip the resulting image, then specify the Import Option;
   `NOCLIPRECTONHTMLTEMPLATE:1`

   To ensure that your body content doesn't have any gutter, use css on the body-style like this;
   `<BODY style='padding:0;margin:0;'>`
   Of course the css-style can be in the css instead.

4. Locate the spot in the HTML file which import the image. This is typically an img-element or some CSS-code which reference an image as background image. The whole file-reference should be replaced by the special variable `$(FULLPATH)`. This variable will point to the processed II2LN image during processing. Remember that any other II2LN import options like Rotate and Resize can be processed before applying the template!

5. Use all image variables (such as ***$(ORIGINAL_IMG_blabla)*** (see "The original image variables" on page 154) and ***$(EXIF_Blabla)*** (see "The EXIF variables" on page 156) and ***$(IPTC_Blabla)*** (see "The IPTC variables" on page 157)) to fill in variables from your original image meta data. Special HTMLTemplate variables also exist. Use these variables to ensure that you can point to other HTML/CSS elements
   `$(HTMlTEMPLATEROOTPATH)` points to the root path for all your HTML Templates
   `$(HTMLTEMPLATEPATH)` contains the sub-directory name of the chosen template, scuh as "Transparent Text Bottom"
   `$(HTMLTEMPLATEFULLPATH)` points to the full path to the chosen template, a combination of $(HTMlTEMPLATEROOTPATH) and $(HTMLTEMPLATEPATH)
   `$(HTMLTEMPLATEFINDEXFILEPATH)` points to the full path of the chosen Index.htm file

6. Even use your own dynamically added variables with the ***AddVariable*** (on page 67) import option.

7. Re-reference all appurtenant files such as css and js, with the use of the $(HTMLTEMPLATExxx-variables above.

When you have done all this, you can simply use the UseHTMLTemplate import option like this;

UseHTMLTemplate:<Subdirectory Name of Template>

Finally, note that this import option is not repeatable, and it will be processed as one of the very last import options, meaning that you can preprocess your image with all other import options such as rotate, resize, overlaytext etc.

Introduced in version:

1.7.0.0

Repeatable

No

WorkingPath

Description:

By default II2LN will place all temporary files in your temporary directory. The exact placement of this directory differs from Windows version to Windows version, and from user to user. Typically you find your temporary directory in the environment variable TEMP. On a Windows 2000 or XP system, this is often set to something like `%USERPROFILE%\Local Settings\Temp`. The %USERPROFILE% variable expands to something like `C:\Documents and Settings\<user logon name>\Local Settings\Temp\`.

However, it's not always you want to use this directory for II2LN's temporary files. You can specify another directory with WorkingPath!

Declaration:

WorkingPath:<path>

Parameters:

Param	Description
<path>	Full path to directory where II2LN will store temporary files. For example; `C:\Temp\II2LN`

Notes:

Introduced in version:

1.1.0.0

Repeatable

No

CHAPTER 5

Examples

This chapter will use- and describe II2LN in some real world scenarios. Hopefully you find the examples inspiring!

In This Chapter

Import an image from view context to the current document

The following sample will be implemented as an LotusScript agent and is intended to be launched from a Notes view. Declare the ImportImage function as specified ***here*** (see "ImportImage" on page 35), and enter the following code in the Initialize-section of the agent:

```
' First declare everything and set the current
database

Dim session As New NotesSession

Dim db As NotesDatabase

Set db = session.CurrentDatabase

Dim lRc As Long

Dim strFileName As String

Dim collection As NotesDocumentCollection

Dim doc As NotesDocument

Dim ws As New NotesUIWorkspace

' Ask the user for an image filename. Just accept
one filename

filenames = ws.OpenFileDialog(   False, _

"Image to be imported","All
files|*.*|Jpegs|*.jpg",_

"c:\temp")

If Not(Isempty(filenames)) Then

     Forall filename In filenames

          strFileName = filename

          Exit Forall

     End Forall

End If

If strFileName <> "" Then
```

```
      ' Set up a collection of the
UnprocessedDocuments, ie. work only with the
selected Notes documen

     Set collection = db.UnprocessedDocuments

     Set doc = collection.GetFirstDocument()

     While Not(doc Is Nothing)

            // Perform the ImportImage function
with the import option ReplaceWithVariable and
Resize

           lRc = ImportImage(_

           db.Server,_

           db.FilePath,_

           doc.UniversalID, _

           "Body",_

           strFileName,_


     "ReplaceWithVariable:$(Image);Resize:640,48
0")

           Set doc = Nothing



            ' Don't bother to process more than
one selected Notes document

            exit while

     Wend

End If
```

Create an e-mail containing an image

The following sample will be implemented as an LotusScript agent and is intended to be launched from a Notes view. Declare the ImportImage function as specified ***here*** (see "ImportImage" on page 35), and enter the following code in the Initialize-section of the agent:

```
' First declare everything and set the current
database

Dim lRc As Long

Dim session As New notessession

Dim doc As NotesDocument

Dim db As notesdatabase

Dim RtItem As NotesRichTextItem

Dim strUNID As String


Set db = session.CurrentDatabase


' Create a new document in the current database

Set doc = New NotesDocument(db)


doc.Form = "Memo"

doc.SendTo = "someuser@somecompany.com"

doc.Subject = "Here's the document you wanted"

Set RTItem = New NotesRichTextItem( doc, "Body")

Call doc.Save(True, False)

' Note that the document is saved, and the
document unique ID is retrieved.

strUnid = doc.UniversalID
```

```
' Now, forget about the doc while ImportImage
does its job



' Set the doc variable to Nothing while
ImportImage do its job. This is the most
important step here. Since II2LN always work on
the backend document, the reference that the doc
variable holds, will not be in sync with the
modified document later on.

Set doc = Nothing



' Do the ImportImage job. You may wonder why you
don't save the document after ImportImage?
Remember, II2LN do this for you. All you need to
now now, is open the document again.

lRc =
ImportImage(db.Server,db.FilePath,strUnid,"Body",
"C:\ImageToSend.tif","SetLogFilename:c:\temp\temp
.log;SetLogLevel:5;Resize:640,9999;LoadTIFFPage:0
")

' Reconnect to the document, by retrieving the
document by the unique id. This means that
LotusScript will get the document again, and now
the links within LotusScript are ok.

Set doc = db.GetDocumentByUNID(strUnid)



' Send and remove temporary document if you don't
need it.

Call doc.Send(False)

Call doc.Remove(True)
```

Create a preview of PDF attachments

Envision a database containing documents with Acrobat PDF attachments. By using II2LN together with ***Ghostscript,*** (see "Support for EPS, PS and PDF files - Ghostscript Support" on page 170) you can extract the PDF attachment and make an image of the first page automatically. The agent below will do that. Please note that the agent look for the PDF attachments in the Body field. For each PDF attachment found, the agent will create two new rich text fields, rtDisplay_n and rtAttachment_n, where "n" represent the number of the attachment. If you want to display the image, please add the necessary rtDisplay_n fields to your Notes form.

```
'Process PDF Attachments with II2LN:

Option Public

Option Declare

' Functions from II2LN

Declare Function ImportImage Lib "VCII2LN.DLL" (_

Byval pstrNotesServer As String, _

Byval pstrNotesDatabase As String, _

Byval pstrNotesUNID As String, _

Byval pstrNotesField As String, _

Byval pstrFilename As String, _

Byval pstrImportOptions As String) As Long


' General Windows functions

Declare Function GetTempPath Lib "kernel32.dll"
Alias "GetTempPathA" (Byval nBufferLength As
Long, Byval lpBuffer As String) As Long


Sub Initialize

 Dim session As New NotesSession

 Dim db As NotesDatabase
Dim col As NotesDocumentCollection

Dim doc As NotesDocument

Dim docNext As NotesDocument

Dim strUNID As String

Dim lRc As Long

Dim strTempPath As String

Dim strFilePath As String

Dim body As NotesRichTextItem

Dim listAttachmentsInDoc List As String

Dim strImportOptions As String

Dim listImportOptions List As String

Dim strIO As String

Dim strIOParam As String

Dim strTmp As String

Dim iCounter As Integer


' Get the temporary path for the user. This is
the directory where the temp files will be stored

strTempPath = GetTemporaryPath()


On Error Resume Next


' Prepare to loop through all the selected
documents

Set db = session.CurrentDatabase

Set col = db.UnprocessedDocuments

Set doc = col.GetFirstDocument
```

```
 ' For each selected document ....


While Not doc Is Nothing

  ' Remember the UNID of the currently selected
document

  strUNID = doc.UniversalID



  ' Get the next document now - because when we
use II2LN, we detach the doc variable!

  Set docNext = col.GetNextDocument(doc)



  ' Find and extract the attachment. If found,
then process the current document with II2LN -
but

  ' only if the document has attachments at all

  If doc.HasEmbedded Then

   ' Ensure list of filenames is empty

   Erase listAttachmentsInDoc



   ' Get the handle to the richtext item "Body"

   Set body = doc.GetFirstItem("Body")



   ' Loop through all the attachments in the
document

   iCounter = 0

   Forall attachment In body.EmbeddedObjects

    If attachment.Type = EMBED_ATTACHMENT Then

     ' Construct temporary file path for
attachment

     strFilePath = strTempPath &
attachment.Source
```

```
     ' Extract attachment

     Call attachment.ExtractFile(strFilePath)


     ' If our list of filenames for this document
doesn't contain the filename, then add it

     If Not
Iselement(listAttachmentsInDoc(strFilePath)) Then

      listAttachmentsInDoc(Cstr(iCounter)) =

strFilePath

      iCounter = iCounter + 1

     End If


     Call attachment.Remove

    End If

   End Forall


   ' Save the document - this will essentially
save the document *without* the attachment

   Call doc.Save(True,False)


   ' Ok, we don't need to reference the doc-
variable anymore before II2LN processing,

   ' so we get rid of it

   Set doc = Nothing


   ' Loop through all attachments

   iCounter = 1

   Forall strFileName In listAttachmentsInDoc
```

```
    ' Clear list of II2LN options

    Erase listImportOptions



    ' Ok, now we have a list of attachment
filenames, and they are all extracted to temp
directory

    ' Time to work with II2LN!



    ' Using the list structure below makes it
easy to add new import options!

    listImportOptions("SetResizeMethod") = "7"

    listImportOptions("SetLogLevel") = "5"

'    listImportOptions("Sharpen") = "50"

    ' The SetGhostscriptProcessor is not
necessary if you find GSWIN32C.EXE in one of your
PATH directories

    listImportOptions("SetGhostscriptProcessor")
= "E:\Programs\gswin32c.exe"    ' The exact path
and filename to GS

    listImportOptions("SetGhostscriptParameters")
= "-r300 -dTextAlphaBits=4 -dGraphicsAlphaBits=4"
' Enhance PDF conversion!!

    listImportOptions("Resize") = "1024,0"

    listImportOptions("AttachOriginalFile") =
"rtAttachment_" & Cstr(iCounter)

    ' Straighten out the listImportOptions list
to a string, which I can use with II2LN

    Forall impopt In listImportOptions

     strIO = Listtag(impopt)

     strIOParam = impopt

     strTmp = strIO & ":" & strIOParam

     If strImportOptions = "" Then

      strImportOptions = strTmp
     Else
      strImportOptions = strImportOptions & ";" &
strTmp
     End If
    End Forall


    ' Process the image with II2LN
    lRc = ImportImage(_
    db.Server,_  ' The server
    db.FilePath,_ ' The database
    strUNID, _ ' The UNID
    "rtDisplay_" & Cstr(iCounter),_ ' The field
name
    strFileName,_ ' The image to import
    strImportOptions) ' The import options


    ' If OK, delete the temp file
    If lRc = 0 Then
     Kill strFileName
    End If
    iCounter = iCounter + 1
   End Forall
  End If ' end if doc.HasEmbedded


  ' Advance to the next selected document in the
view
  Set doc = docNext
 Wend
 Exit Sub
```

```
End Sub



Function GetTemporaryPath() As String

 Dim strTempPath As String*255

 Dim lStringLength As Long

 strTempPath = String(255, 0)  ' Initialize
string

 lStringLength = GetTempPath(255, strTempPath)

 GetTemporaryPath  = Left$(strTempPath,
lStringLength)

End Function
```

CHAPTER 6

Variables

The variables has the format `$(variablename)` and can be placed anywhere in the Notes document, including rich text fields. When II2LN process the document, the real values from the document will replace the specified variables. Note that II2LN replace the values as text strings.

If you use the ***ImportImage*** (on page 35) function, please note that the variable replacing only takes place when the ***ReplaceWithVariable*** (on page 109) import option is specified

In This Chapter

The current document variables

Variable	Description
$(REPLICAID)	The replica ID of the current database
$(NOTEID)	The Note ID of the current document. Note the Note ID is on 4 bytes hex format, such as: 00001FB5.
$(UNID)	The Document Unique ID of the current document.
$(IMPORT_FIELDNAME)	The Notes field name to import- or attach the file to

The original image variables

The following variables use values from the original (or unmodified) image.

Variable	Description
$(ORIGINAL_IMG_CREATION_TIMESTAMP)	The creation date and time of the image. The timestamp has the following format: `dd.mm.yyyy HH:MM:SS`
$(ORIGINAL_IMG_LASTWRITE_TIMESTAMP)	The last modified -or- last write date and time of the image. Same format as above.
$(ORIGINAL_IMG_SORT_TIMESTAMP)	The timestamp on a sortable format YYYYMMDDHHMMSS. If the image contains the EXIF Date Time tag, this variable will contain that value, otherwise it will contain the creation timestamp above. This makes this variable usable if you want to rename a file with variables!
$(ORIGINAL_IMG_FILESIZE)	The file size in bytes of the image. No formatting
$(ORIGINAL_IMG_FILENAME)	The filename only (ie. not the drive and path) of the image.
$(ORIGINAL_IMG_FULLPATH)	The full path and filename of the image (ie. including drive, path and filename)
$(ORIGINAL_IMG_WIDTH)	The width of the image in pixels
$(ORIGINAL_IMG_HEIGHT)	The height of the image in pixels
$(ORIGINAL_IMG_BITDEPTH)	The bit depth of the image
$(ORIGINAL_IMG_FORMATCODE)	The format code of the image, see ***The image format codes*** (on page 57)
$(ORIGINAL_IMG_FORMAT)	The image format, such as `Jpg` or `Gif`.
$(ORIGINAL_IMG_DPI_X)	Horizontal Dots Per Inch in the image
$(ORIGINAL_IMG_DPI_Y)	Vertical Dots Per Inch in the image

$(ORIGINAL_IMG_DPI_UNITCODE)	Which unit code does the Dot Per Inch variables use ? You will receive `1 (=Inch)`, `2 (=cm)` or `3 (=None)`
$(ORIGINAL_IMG_DPI_UNIT)	Which unit does the Dot Per Inch variables use ? You will receive `Inch`, `cm` or `None`

The import image variables

The following variables use values from the imported (or modified) image.

Variable	Description
$(IMPORT_TYPE)	The image format of the import file, such as `Jpg` or `Gif`.
$(IMPORT_IMG_CREATION_TIMESTAMP)	The creation date and time of the image. The timestamp has the following format: `dd.mm.yyyy HH:MM:SS`
$(IMPORT_IMG_LASTWRITE_TIMESTAMP)	The last modified -or- last write date and time of the image. Same format as above.
$(IMPORT_IMG_FILESIZE)	The file size in bytes of the image. No formatting
$(IMPORT_IMG_FILENAME)	The filename only (ie. not the drive and path) of the image.
$(IMPORT_IMG_FULLPATH)	The full path and filename of the image (ie. including drive, path and filename)
$(IMPORT_IMG_WIDTH)	The width of the image in pixels
$(IMPORT_IMG_HEIGHT)	The height of the image in pixels
$(IMPORT_IMG_BITDEPTH)	The bit depth of the image
$(IMPORT_IMG_FORMATCODE)	The format code of the image, see ***The image format codes*** (on page 57)
$(IMPORT_IMG_FORMAT)	The image format, such as `Jpg` or `Gif`.
$(IMPORT_IMG_DPI_X)	Horizontal Dots Per Inch in the image

$(IMPORT_IMG_DPI_Y)	Vertical Dots Per Inch in the image
$(IMPORT_IMG_DPI_UNITCODE)	Which unit code does the Dot Per Inch variables use ? You will receive `1 (=Inch)`, `2 (=cm)` or `3 (=None)`
$(IMPORT_IMG_DPI_UNIT)	Which unit does the Dot Per Inch variables use ? You will receive `Inch`, `cm` or `None`

The EXIF variables

If the import option ***ExtractEXIFToField*** (on page 77) is specified, II2LN will extract potential EXIF information. The ExtractEXIFToField import option will primarily extract the EXIF information to either one or multiple fields.

In addition, all EXIF tags will result in variables that can be used in the document, as long as the ***ReplaceWithVariable*** (on page 109) import option is specified.

The format of EXIF variables are:

`$(EXIF_xxx)`

where xxx represent a EXIF tag name. Below you find some examples of the EXIF variables:

`$(EXIF_Make), $(EXIF_Model), $(EXIF_Orientation), $(EXIF_XResolution)` and `$(EXIF_Flash)`

See ***The most common EXIF tags*** (on page 172) for a list of EXIF tags

The IPTC variables

If the import option ***ExtractIPTCToField*** (on page 81) is specified, II2LN will extract potential IPTC information. The ExtractIPTCToField import option will primarily extract the IPTC information to either one or multiple fields.

In addition, all IPTC tags will result in variables that can be used in the document, as long as the ***ReplaceWithVariable*** (on page 109) import option is specified.

The format of IPTC variables are:

```
$(IPTC_TAG_xxx)
```

where xxx represent a IPTC tag name. Below you find some examples of the IPTC variables:

`$(IPTC_TAG_Caption_Abstract),`
`$(IPTC_TAG_CopyrightNotice),` and
`$(IPTC_TAG_Headline)`

See ***The most common IPTC tags*** (on page 180) for a list of IPTC tags

CHAPTER 7

Appendix

In This Chapter

Dots Per Inch - DPI

The term Dots Per Inch (DPI) causes lots of confusion. Try to search Internet or UseNet groups, and you will see lots of discussions around this topic.

First and foremost, forget about DPI if the image don't touch a printer!! The DPI setting doesn't affect *screen display* of the file. The screen display is *only* controlled by the pixel dimension of the image (such as 640 x 480 pixels, 1600 x 1200 pixels and so on), and it's finally the screen resolution that determine how large the image will look.

What DPI do control is at what density the image is *printed*.

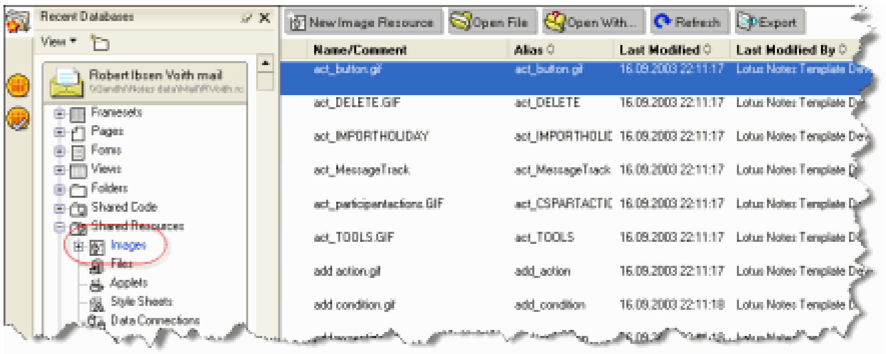
To attempt to explain the topic, let's assume we have an image which is 720 x 540 pixels. Note that the width of 720 pixels aren't completely taken out the air! This is because our math will be so much simpler!!

On your screen, this image will be exactly 720 pixels wide, so if you have an traditional LCD screen with 1024 x 768 pixels resolution, this image will occupy approximately 2/3 of your screen. Anyone with 1600 x 1200 displays around ? These screens will show the same image in approximately 1/2 the screen. And the cool thing is; The displayed image will be the same size no matter if the DPI is 1, 72, 96 or 300! So much for the screen display.

When it comes to printing, the DPI start to work. Note that the DPI is Dots Per *Inch*. This means how may dots the image contain per *inch*. If the DPI for example is 72, then the image will print with a size of 10 inches (720 / 72 = 10). If the image has 144 DPI, the image will print with a size of 5 inches (720 / 144 = 5). Note that the *image detail* isn't changed at all, only how many pixels we cram into each inch !!

Image Resource - what is it ?

When you design a Notes application, you have the option to imbedded images as resources with the Lotus Designer. Below you see a snapshot of the Image Resources in the mail database:



Name/Comment	Alias	Last Modified	Last Modified By
act_button.gif	act_button.gif	16.09.2003 22:11:17	Lotus Notes Template Dev
act_DELETE.GIF	act_DELETE	16.09.2003 22:11:17	Lotus Notes Template D
act_IMPORTHOLIDAY	act_IMPORTHOLIC	16.09.2003 22:11:17	Lotus Notes Template Dev
act_MessageTrack	act_MessageTrack	16.09.2003 22:11:17	Lotus Notes Template D
act_participantactions.GIF	act_CSPARTACTIC	16.09.2003 22:11:17	Lotus Notes Template D
act_TOOLS.GIF	act_TOOLS	16.09.2003 22:11:17	Lotus Notes Template D
add action.gif	add_action	16.09.2003 22:11:17	Lotus Notes Template Dev
add condition.gif	add_condition	16.09.2003 22:11:18	Lotus Notes Template D

Figure 4: Image Resources in the mail file

Traditionally the Image Resources are used to store design elements of the database.

By importing any image into a Notes database as an image resource, the image can be *referenced* by the rich text field instead of embedded. The image itself, will therefore not be stored in the rich text field, only a reference to the image resource.

The real benefit of storing the image as a resource, is that the image can be shown in ordinary Notes views. II2LN automatically take care about all the difficult parts for you!

How to specify colors ?

Several import options, such as ***Rotate*** (on page 115) and ***OverlayText*** (see "WriteText" on page 96) let you specify colors. All colors in II2LN are specified as a hexadecimal string on the following format:

```
RRGGBB
```

For example, black are specified as `000000` while white is specified as `FFFFFF`. The most common colors have the following codes:

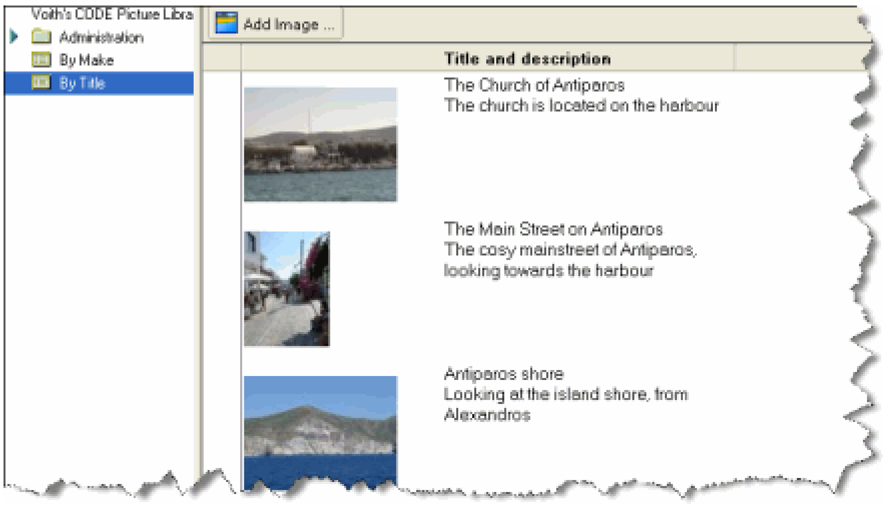
Color	Code
Black	000000
White	FFFFFF

Red	FF0000
Blue	0000FF
Lime Green	00FF00
Pink	FF00FF
Yellow	FFFF00
Orange	FF7F00
Dark Blue	000080
Dark Orchid	9932CD
Brown	A62A2A

If you don't have a color chart available, try to search the internet for "RGB color chart". For example; ***1 NetCentral's Helpful RGB Color Selection Chart*** www.1netcentral.com/rgb-color-chart.html

How to display my images in a Notes view

II2LN has a powerful feature that let you import images and make them visible directly in the Notes views. Below you see a snapshot of this;



The Church of Antiparos
The church is located on the harbour
The Main Street on Antiparos
The cosy mainstreet of Antiparos, looking towards the harbour
Antiparos shore
Looking at the island shore, from Alexandros

Figure 5: Image Resources in Notes views

The snapshot above shows imported images directly in the Notes views! How do you do that ?

1 Use the function ***ImportImageAsResource*** (on page 38) to import and prepare the image for display in Notes views.

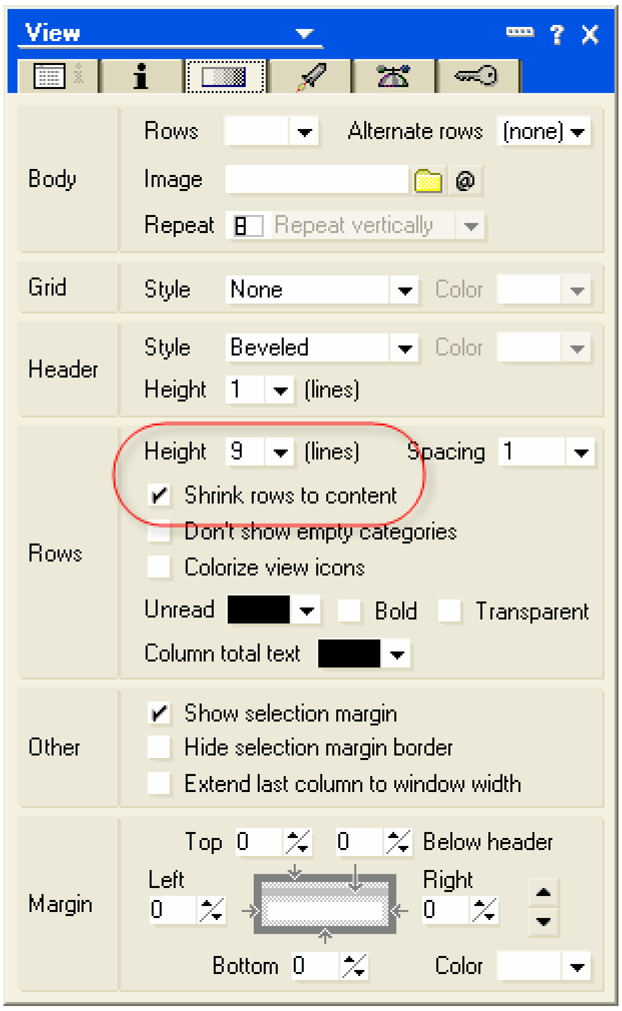
2 The ImportImageAsResource function will automatically create a set of extra fields in the Notes document. These fields are prefixed with the fieldname you specified as thumbnail-fieldname. If that was "XYZ", you will have these three extra fields;

1. XYZFilename - the filename of the ***Image Resource*** (see "Image Resource - what is it ?" on page 161) itself.
2. XYZWidth - the width of the thumbnail in pixels
3. XYZHeight - the height of the thumbnail in pixels

3 II2LN will also automatically create a rich text field ("XYZ" in our scenario) and establish the reference to the image resource.

4 Create a standard Notes view.

1. Ensure that your view properties has the Height set to 9 lines, and that you have checked the "Shrink rows to content"



View

Body — Rows: ; Alternate rows: (none); Image: ; Repeat: Repeat vertically

Grid — Style: None; Color:

Header — Style: Beveled; Color: ; Height: 1 (lines)

Rows — Height: 9 (lines); Spacing: 1
- [✓] Shrink rows to content
- [ ] Don't show empty categories
- [ ] Colorize view icons

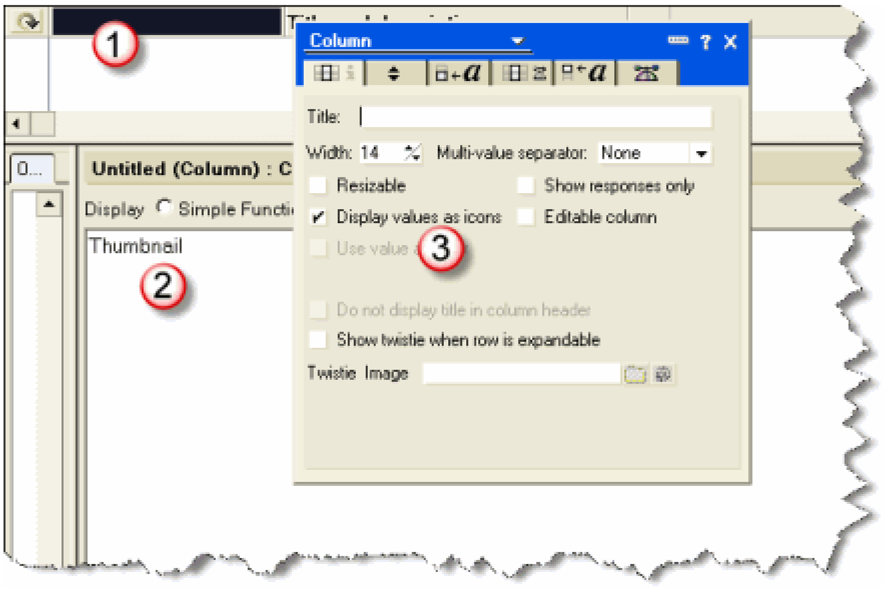
Unread: ■ ; [ ] Bold ; [ ] Transparent
Column total text: ■

Other
- [✓] Show selection margin
- [ ] Hide selection margin border
- [ ] Extend last column to window width

Margin — Top: 0; Below header: 0; Left: 0; Right: 0; Bottom: 0; Color:

This enables our view to show images taller than one line!

2. Determine where you want your thumbnail to be displayed. In the snapshot below, the thumbnail is in the first column:



Title:
Width: 14
Multi-value separator: None
Resizable [ ]
Show responses only [ ]
Display values as icons [✓]
Editable column [ ]
Use value [ ]
Do not display title in column header [ ]
Show twistie when row is expandable [ ]

3. Create a column that is wide enough to display your thumbnail
4. The formula for your column is the field name you specified as the thumbnail-field name. In our example it would have been "XYZ".
5. Make the column display it's value as icons
6. Add another "height adjuster column". This is just a column that will add some empty spaces so we gain enough room for our image. The column can typically be the last column -unless- you make a categorized view. Then this column has to be just in front of the thumbnail column.

   To have a formula add some space for you, you can for example use this formula:

   ```
   REM {Ensure the row is high enough};
   n := @Round (ThumbnailHeight / 40 +0,5);
   s := @Repeat(" @"; n);
   @Explode(s; "@")
   ```

   Note that the field ThumbnailHeight represents the autogenerated field with the thumbnail prefix above! In our scenario this field would have the fieldname "XYZHeight".

5 In order to make Notes refresh properly and thus display your imported image in the view, add the following LotusScript code after the ImportImageAsResource-call:

```
Dim ws As New NotesUIWorkspace
Call ws.ViewRefresh
Call ws.ReloadWindow
```

Notes Embedded Image - what is it ?

Beware! These kind of images is also known as "inline images".

Lotus Notes can import images to rich text fields through the Notes client or via the C/C++ API. If the imported image is a GIF or a JPG, Notes will fortunately store the image within the Note in its original format. If the image is anything else, Notes will convert the imported image to a stripped down TIFF variant.

The Notes client also accept images through copy and paste. When you for example see an image on a web page and copy that to the clipboard, you can easily paste the image into a rich text fields in Notes. The pasted image is however converted to the stripped down TIFF version at paste-time!

Unfortunately you can't really do much with embedded images in the Notes client. You don't have any options to view or save the images as files. You need special tools, such as ***List Fields*** http://www.listfields.com to export and make such images available again.

Let's have a look at a sample. In the Notes document below, a winter-image has been imported into Lotus Notes:

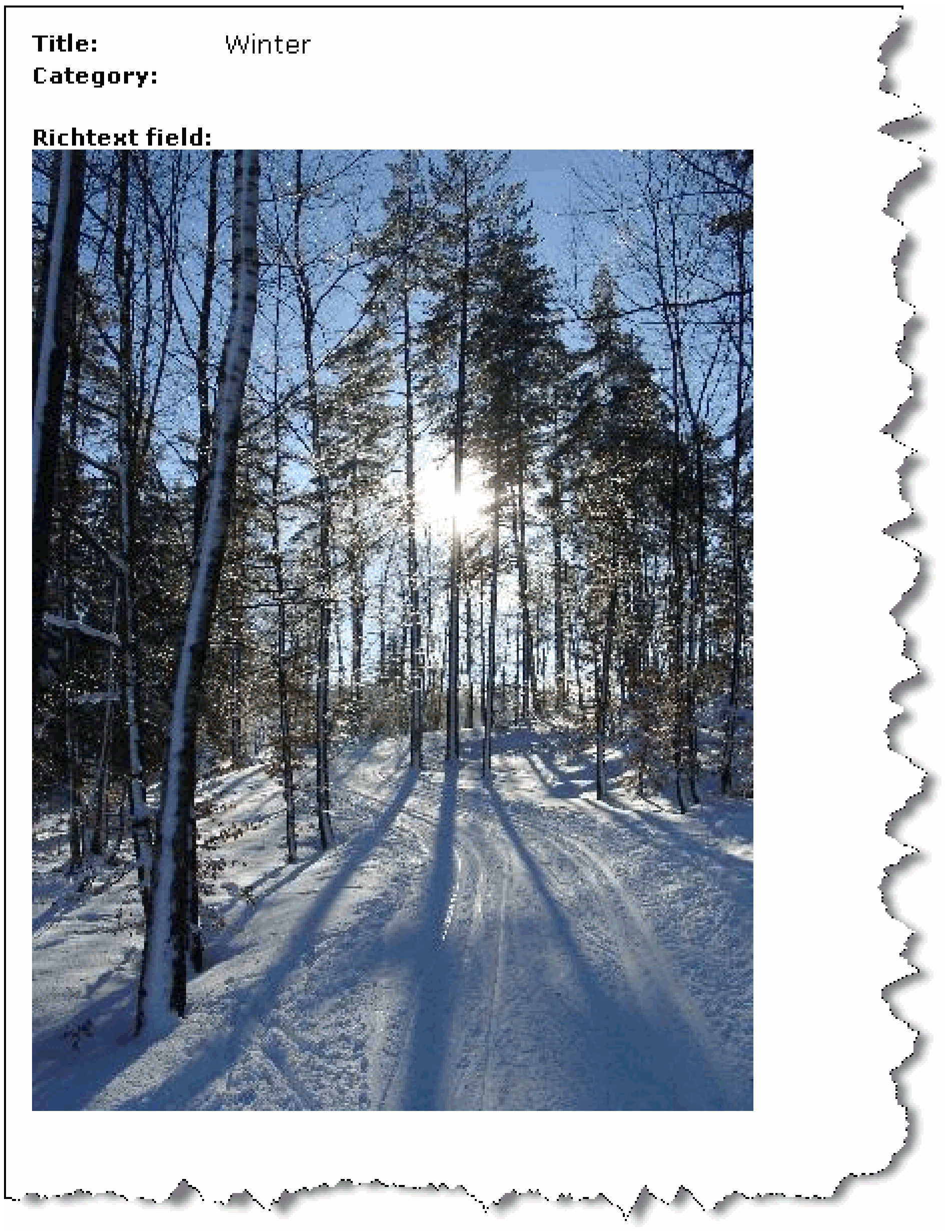
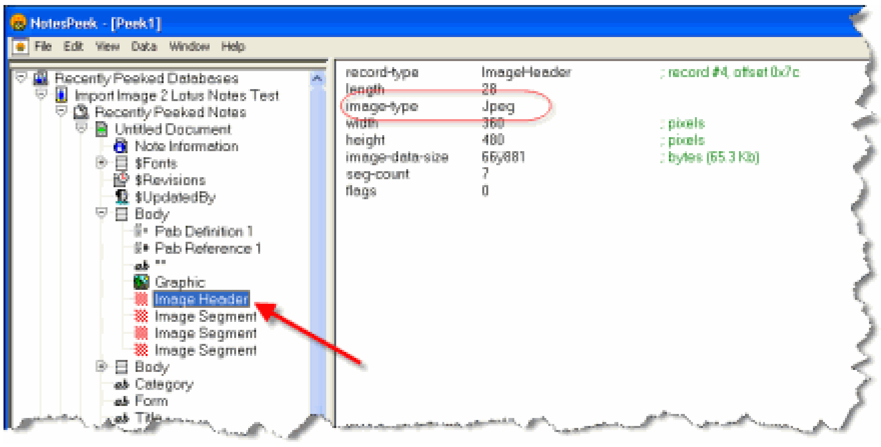


Figure 6: A Notes Embedded Image

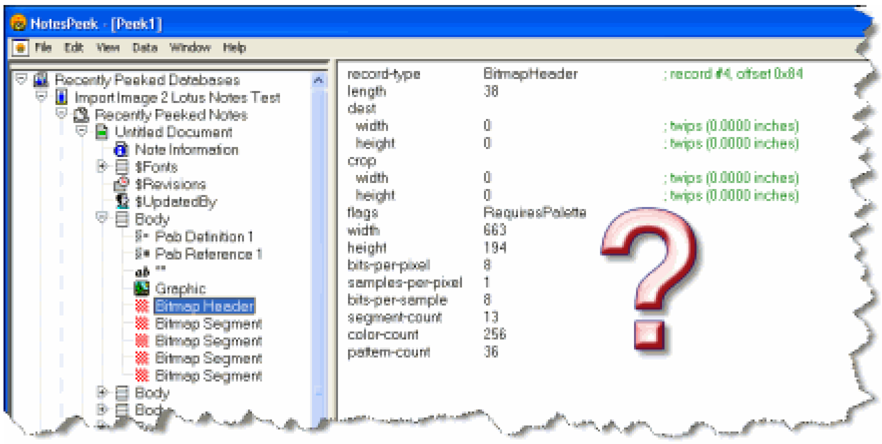
If you right-click on the image, you won't get any "image properties" or similar. If you look at the Notes document with ***NotesPeek,*** (see "NotesPeek" on page 169) you can see deep down in the graphic-related CD-records that you have a JPG, GIF or other image. Below you see how to identify the image type with NotesPeek:



```
record-type        ImageHeader      ; record #4, offset 0x7c
length             28
image-type         Jpeg
width              360              ; pixels
height             480              ; pixels
image-data-size    66,881           ; bytes (65.3 Kb)
seg-count          7
flags              0
```

Figure 7: NotesPeek looks at JPG

Below you see what the same screenshot looks like when you have an unknown image:



```
record-type        BitmapHeader     ; record #4, offset 0x84
length             38
dest
  width            0                ; twips (0.0000 inches)
  height           0                ; twips (0.0000 inches)
crop
  width            0                ; twips (0.0000 inches)
  height           0                ; twips (0.0000 inches)
flags              RequiresPalette
width              663
height             194
bits-per-pixel     8
samples-per-pixel  1
bits-per-sample    8
segment-count      13
color-count        256
pattern-count      36
```

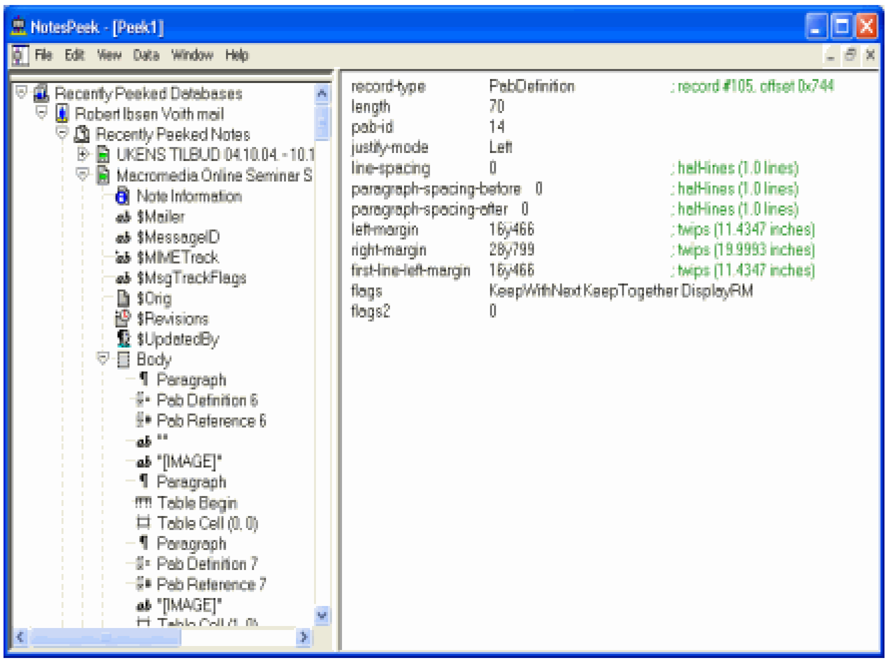
Figure 8: NotesPeek looks at unknown

From the CD-records you can't exactly tell what kind of image this originally was. We do see however, several data about the image. And when we know that such images are a stripped down version of TIFF, we are able through some serious programming to decode the image into something that we can view, edit or save. This is how List Fields process these images

NotesPeek

For years, NotesPeek has been the de facto standard application to dive into the inner details of Notes databases. The name with the suffix "peek" reminiscence the old days when programmers frequently had to read content at memory locations by "peeking", and writing to memory locations with "poking". In addition the icon x-raying the "Notes-folks" identify the purpose of NotesPeek quite good:

NotesPeek was written by Ned Batchelder, a former Lotus employee. The user interface is clean and intuitive using the treelist to navigate in databases, documents and fields. The content is shown in the main area to the right. Below you see a screenshot of NotesPeek.



```
record-type              PabDefinition                ; record #105, offset 0x744
length                   70
pab-id                   14
justify-mode             Left
line-spacing             0                            ; halflines (1.0 lines)
paragraph-spacing-before   0                          ; halflines (1.0 lines)
paragraph-spacing-after  0                            ; halflines (1.0 lines)
left-margin              16,466                       ; twips (11.4347 inches)
right-margin             28,799                       ; twips (19.9993 inches)
first-line-left-margin   16,466                       ; twips (11.4347 inches)
flags                    KeepWithNext KeepTogether DisplayRM
flags2                   0
```

Figure 9: NotesPeek

You will find all sort of information here, and as you see from the screenshot above, you can really dive into richtext to see how the CD records are sequenced and what they contain.

Support for EPS, PS and PDF files - Ghostscript Support

II2LN normally operate with binary image formats such as JPEG, GIF, TIFF and BMPs. However, another group of image formats exists - the interpreted vector oriented image formats. Examples of such file types are;

§ EPS - Encapsulated Postscript files.

§ PS - Postscript files

§ PDF - Adobe Acrobat files

§ EMF and WMF - Windows Metafile Thumbnails

II2LN can support most of the files in the formats above via the Ghostscript processor. Ghostscript has a special licence making it impractical (and illegal!) to include Ghostscript directly with II2LN. So in order to utilize Ghostscript you have to download and install Ghostscript your self. As soon as you have followed the installation steps below, II2LN will automatically detect its presence and utilize the Ghostscript support.

Note that there are two different "packages" or distributions of Ghostscript. The primary difference is how the package is licensed. ***AFPL Ghostscript*** http://www.cs.wisc.edu/~ghost/ is distributed with a license called the Aladdin Ghostscript Free Public License that allows free use, copying, and distribution by end users, but does not allow commercial distribution. AFPL Ghostscript was previously known as Aladdin Ghostscript. ***GNU Ghostscript*** http://www.ghostscript.com/awki is distributed with the GNU General Public License, which allows free use, and free copying and redistribution under certain conditions (including, in some cases, commercial distribution). GNU Ghostscript versions are usually released shortly after the next AFPL Ghostscript version.

As far as II2LN goes, it doesn't matter which version of Ghostscript you use.

Please note that II2LN can control the whereabouts of Ghostscript with the ***SetGhostscriptProcessor*** (on page 119) import option. Additionally you can add your own Ghostscript parameters with the ***SetGhostscriptParameters*** (on page 117) import option.

How to download and install Ghostscript

1 Go to the homepage of Ghostscript. Select either the ***APFL*** http://www.cs.wisc.edu/~ghost/ or ***GPL*** http://www.ghostscript.com/awki version of Ghostscript.

Please note that these web sites often are quite large and sometimes it may be difficult to navigate in them

2 Look for sentences such as "Obtaining the latest version of Ghostscript" or "download Ghostscript".

Note here that there are several versions of Ghostscript available. You are looking for the AFPL Ghostscript.

You will typically see multiple versions available. As a general advice, always go for the latest version! At the time of writing the latest current version is 8.50

3 Locate the Ghostscript package for Windows

The Windows installer typically has names on this format; gs854w32.exe.

4 Download the Windows installer

Remember where you store the installer program!

5 Install Ghostscript on your machine

By default the installer will only install the Ghostscript in the specified directory on the machine. Please note that this does not imply that the Ghostscript processor will be available for II2LN directly.

The default install directory is C:\Program Files\gs\gs8.54

6 Make Ghostscript available to II2LN

When II2LN encounter one of the file extentions EPS, PS, PDF, EMF or WMF, it will automatically try to connect to Ghostscript. In order to make II2LN able to locate Ghostscript, you have to perform one of the steps below

1. **Specify the Ghostscript bin-folder in your PATH environment variable**

   If you look in the sub-folder bin (if you installed to the default directories, this will be C:\Program Files\gs\gs8.54\bin), you'll see some EXE and DLL files. In order to process Ghostscript, your machine must have access to all these files.

   Add the `C:\Program Files\gs\gs8.50\bin` directory to your PATH environment variable.

2. **Copy the content of the bin-folder to a directory already in the PATH statement**

   As an alternative to the step above, you can copy the EXE and DLL files in the above bin-directory to a directory which you know exists in your PATH environment variable, such as `C:\Windows\System32`.

3. **Use the SetGhostscriptProcessor import option**

   As a final option, you can specify the full path and file name with the ***SetGhostscriptProcessor*** (on page 119) import option.

The most common EXIF tags

Baseline EXIF tags are those tags that are listed as part of the core of EXIF standards

Code	Hex	Name	Description
1		InteropIndex	
2		InteropVersion	
254	00FE	NewSubfileType	A general indication of the kind of data contained in this subfile.
255	00FF	SubfileType	A general indication of the kind of data contained in this subfile.
256	0100	ImageWidth	The number of columns in the image, i.e., the number of pixels per row.
257	0101	ImageLength	The number of rows of pixels in the image.
258	0102	BitsPerSample	Number of bits per component.

259	0103	Compression	Compression scheme used on the image data.
262	0106	PhotometricInterpretation	The color space of the image data.
263	0107	Threshholding	For black and white TIFF files that represent shades of gray, the technique used to convert from gray to black and white pixels.
264	0108	CellWidth	The width of the dithering or halftoning matrix used to create a dithered or halftoned bilevel file.
265	0109	CellLength	The length of the dithering or halftoning matrix used to create a dithered or halftoned bilevel file.
266	010A	FillOrder	The logical order of bits within a byte.
270	010E	ImageDescription	A string that describes the subject of the image.
271	010F	Make	The scanner/camera manufacturer.
272	0110	Model	The scanner/camera model name or number.
273	0111	StripOffsets	For each strip, the byte offset of that strip.
274	0112	Orientation	The orientation of the image with respect to the rows and columns.
277	0115	SamplesPerPixel	The number of components per pixel.
278	0116	RowsPerStrip	The number of rows per strip.
279	0117	StripByteCounts	For each strip, the number of bytes in the strip after compression.
280	0118	MinSampleValue	The minimum component value used.
281	0119	MaxSampleValue	The maximum component value used.
282	011A	XResolution	The number of pixels per ResolutionUnit in the ImageWidth direction.
283	011B	YResolution	The number of pixels per ResolutionUnit in the ImageLength direction.

284	011C	PlanarConfiguration	How the components of each pixel are stored.
288	0120	FreeOffsets	For each string of contiguous unused bytes in a TIFF file, the byte offset of the string.
289	0121	FreeByteCounts	For each string of contiguous unused bytes in a TIFF file, the number of bytes in the string.
290	0122	GrayResponseUnit	The precision of the information contained in the GrayResponseCurve.
291	0123	GrayResponseCurve	For grayscale data, the optical density of each possible pixel value.
296	0128	ResolutionUnit	The unit of measurement for XResolution and YResolution.
305	0131	Software	Name and version number of the software package(s) used to create the image.
306	0132	DateTime	Date and time of image creation.
315	013B	Artist	Person who created the image.
316	013C	HostComputer	The computer and/or operating system in use at the time of image creation.
320	0140	ColorMap	A color map for palette color images.
338	0152	ExtraSamples	Description of extra components.
33432	8298	Copyright	Copyright notice.

Extension EXIT tags are those tags listed as part of EXIF features that may not be supported by all EXIF readers, according to the EXIF specification

269	010D	DocumentName	The name of the document from which this image was scanned.
285	011D	PageName	The name of the page from which this image was scanned.
286	011E	XPosition	X position of the image.

287	011F	YPosition	Y position of the image.
292	0124	T4Options	Options for Group 3 Fax compression
293	0125	T6Options	Options for Group 4 Fax compression
297	0129	PageNumber	The page number of the page from which this image was scanned.
301	012D	TransferFunction	Describes a transfer function for the image in tabular style.
317	013D	Predictor	A mathematical operator that is applied to the image data before an encoding scheme is applied.
318	013E	WhitePoint	The chromaticity of the white point of the image.
319	013F	PrimaryChromaticiti es	The chromaticities of the primaries of the image.
321	0141	HalftoneHints	Conveys to the halftone function the range of gray levels within a colorimetrically-specified image that should retain tonal detail.
322	0142	TileWidth	The tile width in pixels. This is the number of columns in each tile.
323	0143	TileLength	The tile length (height) in pixels. This is the number of rows in each tile.
324	0144	TileOffsets	For each tile, the byte offset of that tile, as compressed and stored on disk.
325	0145	TileByteCounts	For each tile, the number of (compressed) bytes in that tile.
326	0146	BadFaxLines	Used in the TIFF-F standard, denotes the number of 'bad' scan lines encountered by the facsimile device.

327	0147	CleanFaxData	Used in the TIFF-F standard, indicates if 'bad' lines encountered during reception are stored in the data, or if 'bad' lines have been replaced by the receiver.
328	0148	ConsecutiveBadFax Lines	Used in the TIFF-F standard, denotes the maximum number of consecutive 'bad' scanlines received.
330	014A	SubIFDs	Offset to child IFDs.
332	014C	InkSet	The set of inks used in a separated (PhotometricInterpretatio n=5) image.
333	014D	InkNames	The name of each ink used in a separated image.
334	014E	NumberOfInks	The number of inks.
336	0150	DotRange	The component values that correspond to a 0% dot and 100% dot.
337	0151	TargetPrinter	A description of the printing environment for which this separation is intended.
339	0153	SampleFormat	Specifies how to interpret each data sample in a pixel.
340	0154	SMinSampleValue	Specifies the minimum sample value.
341	0155	SMaxSampleValue	Specifies the maximum sample value.
342	0156	TransferRange	Expands the range of the TransferFunction.
343	0157	ClipPath	Mirrors the essentials of PostScript's path creation functionality.
344	0158	XClipPathUnits	The number of units that span the width of the image, in terms of integer ClipPath coordinates.
345	0159	YClipPathUnits	The number of units that span the height of the image, in terms of integer ClipPath coordinates.

346	015A	Indexed	Aims to broaden the support for indexed images to include support for any color space.
347	015B	JPEGTables	JPEG quantization and/or Huffman tables.
351	015F	OPIProxy	OPI-related.
400	0190	GlobalParametersIFD	Used in the TIFF-FX standard to point to an IFD containing tags that are globally applicable to the complete TIFF file.
401	0191	ProfileType	Used in the TIFF-FX standard, denotes the type of data stored in this file or IFD.
402	0192	FaxProfile	Used in the TIFF-FX standard, denotes the 'profile' that applies to this file.
403	0193	CodingMethods	Used in the TIFF-FX standard, indicates which coding methods are used in the file.
404	0194	VersionYear	Used in the TIFF-FX standard, denotes the year of the standard specified by the FaxProfile field.
405	0195	ModeNumber	Used in the TIFF-FX standard, denotes the mode of the standard specified by the FaxProfile field.
433	01B1	Decode	Used in the TIFF-F and TIFF-FX standards, holds information about the ITULAB (PhotometricInterpretation = 10) encoding.
434	01B2	DefaultImageColor	Defined in the Mixed Raster Content part of RFC 2301, is the default color needed in areas where no image is available.
512	0200	JPEGProc	Old-style JPEG compression field. TechNote2 invalidates this part of the specification.

513	0201	JPEGInterchangeFormat	Old-style JPEG compression field. TechNote2 invalidates this part of the specification.
514	0202	JPEGInterchangeFormatLength	Old-style JPEG compression field. TechNote2 invalidates this part of the specification.
515	0203	JPEGRestartInterval	Old-style JPEG compression field. TechNote2 invalidates this part of the specification.
517	0205	JPEGLosslessPredictors	Old-style JPEG compression field. TechNote2 invalidates this part of the specification.
518	0206	JPEGPointTransforms	Old-style JPEG compression field. TechNote2 invalidates this part of the specification.
519	0207	JPEGQTables	Old-style JPEG compression field. TechNote2 invalidates this part of the specification.
520	0208	JPEGDCTables	Old-style JPEG compression field. TechNote2 invalidates this part of the specification.
521	0209	JPEGACTables	Old-style JPEG compression field. TechNote2 invalidates this part of the specification.
529	0211	YCbCrCoefficients	The transformation from RGB to YCbCr image data.
530	0212	YCbCrSubSampling	Specifies the subsampling factors used for the chrominance components of a YCbCr image.
531	0213	YCbCrPositioning	Specifies the positioning of subsampled chrominance components relative to luminance samples.

532	0214	ReferenceBlackWhite	Specifies a pair of headroom and footroom image data values (codes) for each pixel component.
559	022F	StripRowCounts	Defined in the Mixed Raster Content part of RFC 2301, used to replace RowsPerStrip for IFDs with variable-sized strips.
700	02BC	XMP	XML packet containing XMP metadata
32781	800D	ImageID	OPI-related.
34732	87AC	ImageLayer	Defined in the Mixed Raster Content part of RFC 2301, used to denote the particular function of this Image in the mixed raster scheme.

GPS Related EXIF tags

Code	Hex	Name	Description
0	0000	GPSVersion	
1	0001	GPSLatitudeRef	
3	0003	GPSLongitudeRef	
5	0005	GPSAltitudeRef	
6	0006	GPSAltitude	
8	0008	GPSSatellites	
9	0009	GPSStatus	
10	000A	GPSMeasureMode	GPS Measurement Mode
11	000B	GPSDOP	GPS Degree of Precision
12	000C	GPSSpeedRef	
13	000D	GPSSpeed	
14	000E	GPSTrackRef	
15	000F	GPSTrack	
16	0010	GPSImgDirectionRef	PS Image Direction Ref
17	0011	GPSImgDirection	GPS Image Direction
18	0012	GPSMapDatum	
19	0013	GPSDestLatitudeRef	GPS Destination Latitude Ref

23	0017	GPSDestBearingRef	GPS Destination Bearing Ref
24	0018	GPSDestBearing	GPS Destination Bearing
25	0019	GPSDestDistanceRef	GPS Destination Distance Ref
26	001A	GPSDestDistance	GPS Destination Distance
29	001D	GPSDateStamp	GPS Date Stamp
30	001E	GPSDifferential	GPS Differential

Please refer to ***EXIF.org*** http://www.exif.org/ for more information

The most common IPTC tags

All the IPTC tags except *Record Version* can be both read and written by II2LN. Use the Name as tag ID in the import options ***ReplaceIPTC*** (on page 108), ***AppendIPTC,*** (see "AppendIPTC" on page 69) ***InsertIPTC.*** (see "InsertIPTC" on page 88) The ***ExtractIPTCToField*** (on page 81) import options will extract the human readable values for each tag name below.

Tag	Name	Description
00	Record Version	Mandatory / The current IPTC Information Interchange Model = 4
03	ObjectType	Object Type Reference. 1:News, 2:Data, 3:Advisory x:yy
04	ObjectAttribute	Object Attribute Reference. 1:Current, 2:Analysis, 3: Archive .. x:yy
05	ObjectName	ObjectName. For example "Skislopes in Norway" 64 Bytes
07	EditStatus	EditStatus. Status of the objectdata accoding to the provider eg "Correction" 64 Bytes

08	EditorialUpdate	Editorial Update - (To a previous object) x:yyy (Num)
10	Urgency	Urgency, Editorial urgency 1 = most, 5 = normal, 8 = least x
12	SubjectReference	Subject Reference, IPTC:SubRefNum:SubName:SubMatter:SubDetailName 13 to 236 bytes
15	Category	Category. Identifies the subject of the object in the opinion of the provider AAA (Alpha)
20	SupplementalCategory	Supplemental Category. Further identifies the subject of the object in the opinion of the provider 32 bytes
22	FixtureIdentifier	Fixture Identifier. Identifies frequently occuring object data eg "Euroweather" 32 bytes (Alpha)
25	Keyword	Keywords, 64 bytes
26	LocationCode	Content Location Code, 3 char code indicating which country the event took place eg NOR AAA (Alpha)
27	LocationName	Content Location Name. Name of the country the event took place eg "Norway" 64 Bytes
30	ReleaseDate	Release Date. The earliest date the provider intends the object is to be used Date
35	ReleaseTime	Release Time. The earliest time the provider intends the object is to be used Time
37	ExpirationDate	Epiration Date. The latest date the provider intends the object is to be used Date
38	ExpirationTime	Expiration Time. The earliest time the provider intends the object is to be used Time
40	SpecialInstructions	Special Instructions. text 256 Bytes

42	ActionAdvised	Action Advised. 2 Digits - provider defined. eg 01 = Kill object xx
45	ReferenceService	Reference Service. Only used to refer to a previous 1:30 (Reference Service) 10 Bytes (Alpha)
47	ReferenceDate	Reference Date. Only allowed if 45 used
50	ReferenceNumber	Reference Number. Only allowed if 45 used 8 Bytes (Num)
55	DateCreated	Date Created. Date of the actual event - not when it was digitisedDate Date
60	TimeCreated	Time Created. Time of the actual event - not when it was digitised Time
62	DigitalCreationDate	Digital Creation Date. Date the digital representation of the objectdata was creatred Date
63	DigitalCreationTime	Digital Creation Time. Time the digital representation of the objectdata was creatred Time
65	OriginatingProgram	Originating Program. For example "Photoshop" or "II2LN" 32 Bytes
70	ProgramVersion	Program Version. Only use if 65 used eg "1.2" 10 Bytes
75	ObjectCycle	Object Cycle. Virtually only used in US for Morning Evening or both
80	ByLine	Byline. Photographer, for example "Your Name" 32 Bytes
85	ByLineTitle	ByLine Title. The title or role of photographer to be credited, for example House photographer, correspondent etc 32 Bytes
90	City	City. According to the practices of the provider eg "Oslo" 32 Bytes

92	SubLocation	SubLocation. According to the practices of the provider eg "Holmenkollen" 32 Bytes
95	Province_State	Province State. According to the practices of the provider eg "Vestfold", "Adger" 32 Bytes
100	Country_Primary_LocationCode	Country Primary Location Code. The country code of the subject matter eg "USA" or "NOR" AAA (Alpha)
101	Country_Primary_LocationName	Country Primary Location Name. The country code of the subject matter eg "Norway" 64 Bytes
103	OriginalTransmissionReference	Original Transmission Reference. A code meaning where the object was transmitted from 32 Bytes
105	Headline	Headline. Synopsis of the subject matter 256 Bytes
110	Credit	Credit. Identifies the provider - not necessarily the owner / creator 32 Bytes
115	Source	Source. Identifies the original owner / creator 32 Bytes
116	CopyrightNotice	Copyright Notice. The providers own copyright notice 128 Bytes
118	Contact	Contact. For further info on the object 128 Bytes
120	Caption_Abstract	Caption and/or Abstract. Full version / rest of the headline 2000 Bytes
122	Writer_Editor	Writer and/or Editor. Who wrote the caption 32 Bytes
125	Rasterised Caption	
130	ImageType	Image Type. Code representing B/w, Y comp of seps etc.Contact x:A

131	ImageOrientation	Image Orientation. P = Portrait, L = landscape, S = Square 1 Byte
135	LanguageIdentifier	Short code identifying the language. For example US for USA and NO for Norway

Troubleshooting

What do you do if the II2LN doesn't do what you expect it to do ? Or even worse, what do you do if II2LN crash the Notes client or Domino server ?

Logging

Fortunately II2LN has a built-in log system. By default will II2LN create a field with the name *VCII2LN_Log* in your Notes documents. This multivalue text field may contain the exact reason for the problem. However, the default level of logging is minimal, so you won't see any of the steps that II2LN perform.

You can change the log level by specifying the Import Option *SetLogLevel* with a value from 0 (no logging at all) to 5 (verbose logging). The import option will look something like this;

```
"SetLogLevel:5"
```

Finally you can export the log to a file on the file system, by specifying the import option *SetLogFilename*, like this;

```
"SetLogFilename:C:\Temp\Your log filename.log"
```

You can of course specify any filename you like.

Change the sequence of the import options

If II2LN doesn't do quite what you expected it to, check and verify the sequence of your import options. II2LN will process the import options in the order you have specified them.

Remember that the ***Rotate*** (on page 115) import option is processed after all other import options

How to install the licence for II2LN on a server

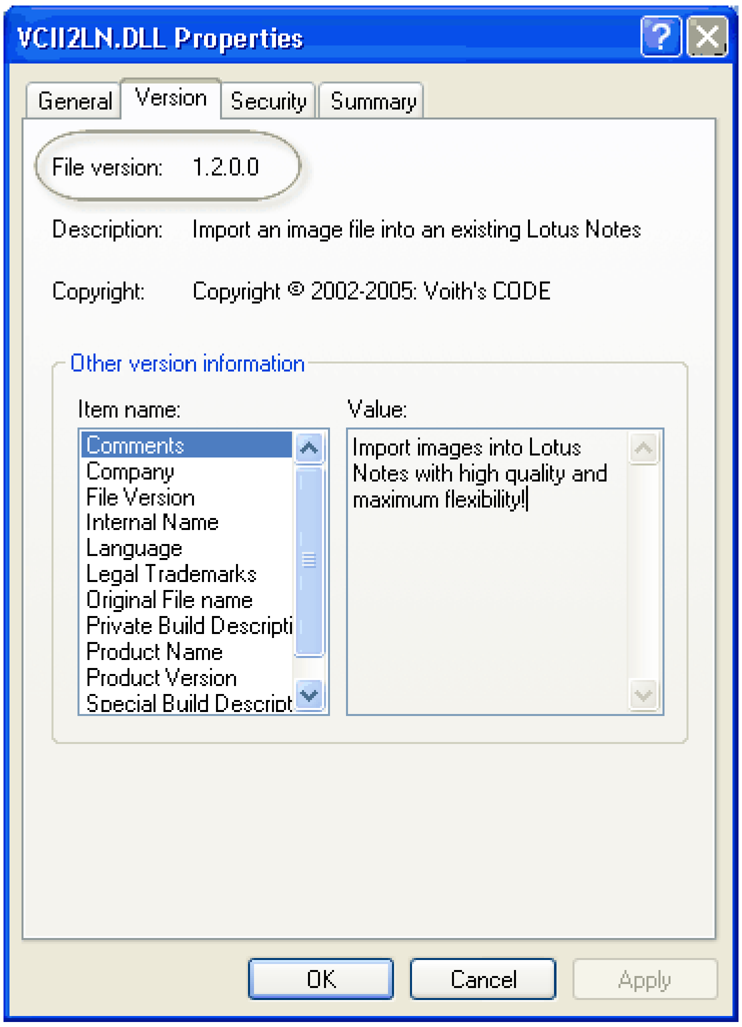
When II2LN is installed on a Domino server, you must ensure that the licence key is placed in the following registry-hive:

HKEY_USERS\.DEFAULT\Software\Voith's CODE\Import Image 2 Lotus Notes\Key

Normally the II2LN licence file is installed in the HKEY_CURRENT_USER hive.

What version of VCII2LN.DLL is installed ?

There are several ways to check which version of VCII2LN.DLL you have installed on your machine. The far easiest is to see the file properties of the file itself. Below you see a snapshot of the Version-pane in the File Properties dialog box:



File version: 1.2.0.0

Description: Import an image file into an existing Lotus Notes

Copyright: Copyright © 2002-2005: Voith's CODE

Comments
Company
File Version
Internal Name
Language
Legal Trademarks
Original File name
Private Build Descripti
Product Name
Product Version
Special Build Descript

Import images into Lotus Notes with high quality and maximum flexibility!

The version is specified at the top line

Index