Migrating Data *from* Notes/Domino

By Chuck Connell

I believe in the Lotus products. I worked at Lotus for five years, serving as the development manager for the Notes API and helping to plan the first Lotusphere. I have coded, installed, and maintained many Notes/Domino applications, spoken at industry conferences, and written articles about these topics. In spite of my professional love affair with this product set, I understand that organizations sometimes make a valid decision to move their data *out* of the Notes/Domino world, and onto another platform. It is my job as a consultant to help them do so.

This article gives a summary of the methods for migrating Notes/Domino applications (NSF files) to other platforms. I have focused on methods that produce general output that can serve as input to many other platforms, rather than tools that target just one other system. The export techniques are arranged in approximate order of easier to harder.

Notes View to Excel

Many applications and platforms can accept an Excel spreadsheet as an import format. If the application cannot import an Excel file directly, Excel can often export another format that *can* be read by your target application. So getting your Notes data into Excel is a valuable step in many migration projects.

Conveniently, Notes can export any view to the old Lotus 1-2-3 format, which Excel can read with high fidelity. (You must use a version of Excel prior to 2007, since that version dropped support for 1-2-3 files.)

- Go to any Notes view.
- Select some or all of the documents.
- File / Export / Save as Type = 1-2-3. Give the file the extension WK1.
- Open the WK1 file directly into Excel (prior to 2007).

You are limited to 65,000 rows (records), 256 columns (fields), and 32,000 characters per cell (single data item).

You can also try using the comma-separated-value or tabular-text (tab separated) export formats, which can be imported directly to any version of Excel. I have had personal experience, however, that these export formats can be problematic. If there are commas, or quotes, or tabs, or too many spaces in the data items, the export/import process can yield incorrect results. I have found the 1-2-3 format to give the best reliability.

One gotcha to watch out for is the width of the view columns in Notes. Be sure to make each one wider than the widest data item in that column, or you may find some items cut off during the migration process.

You can export plain text fields in this way, since any plain text field can fit into a single spreadsheet cell.

You cannot export rich-text fields with this method, since rich-text cannot appear in a view.

Whole Document to RTF

If the Notes documents you want to migrate make significant use of formatted rich-text, and there are not too many documents, you can consider the RTF export format. This export option is only available at the Notes UI, however, so it must be done manually. For up to a few hundred documents, this is a viable option. For thousands of documents, it is not realistic.

To try this export format, open a Notes document. Pull down File / Export / Save as Type = RTF. The resulting RTF file can be opened directly into Microsoft Word, with very high fidelity of the rich-text formatting.

LotusScript to Tab-Separated Text

This export method uses the same idea as the built-in tab-separated text option, but does it correctly. A tab-separated text file is a nearly universal export/import format. There is no limit on the number of rows or columns, or the size of each data item. The only restriction is that the data itself cannot contain any tab characters, which is easy to achieve by just changing any such occurrence to a few spaces.

To produce correct tab-separated text files, I like to write my own export code using LotusScript. This is quite easy to do, and can be packaged as a pull-down action that operates on selected documents, or an entire view, or an entire database.

The LotusScript looks something like this:

```
TabChar = Chr$(9)

DataFile = Freefile()
Open "C:\temp\export.txt" For Output As DataFile

Do <for all documents>
      Field1  = DataDoc.GetFirstItem("FieldName1").Text
      Field2  = DataDoc.GetFirstItem("FieldName2").Text
      Field3  = DataDoc.GetFirstItem("FieldName3").Text

      Print #DataFile, Field1 + TabChar + Field2 + _
            TabChar + Field3
End Do
```

In addition to generating tab-separated output, this method has the advantage that you can "fix" data at the same time. It is common for migration projects to run into dirty data such as missing last names, invalid social security numbers, impossible birthdates, etc. The LotusScript code can correct these problems while doing the export, or at least flag them, like this:

```
If Len(SSN)<>11 Then Error 1001, "Found bad SSN."
```

The resulting tab-separated file can be imported directly into many application platforms, opened with any version of Excel, or imported to Excel and then exported from there to another format.

Tab-separated files can contain rich-text fields from the Notes documents, if you are willing to lose the formatting. Just use the method NotesRichtextItem.GetUnformattedText() to get the plain contents of a rich-text field.

NSF File to PDF

While the RTF export method (described above) provides high-fidelity migration for formatted rich-text, it is not practical for a large number of documents. When there are many documents, and the migration must run programmatically, I have turned to PDF as an output method. Since PDF files can capture formatting at least as detailed as Notes rich-text, this type of output can, in theory, result in no loss of fidelity.

But how to convert Notes to PDF? There are at least two third-party tools that do so, including Primeapple and Swing. Both are packaged as LotusScript shared library calls. I tested the Primeapple product with very good results, including fonts, colors, indenting, tables, embedded pictures, and attachments. Sample LotusScript that calls the Primeapple tool looks like this:

```
Selection = |Select @Text(@DocumentUniqueID) = "| + _
NotesDocument.UniversalID + |"|

PdfPath = "c:\temp\PdfConvert.pdf"

PdfOptions = + _
"Database=" + NotesDatabase.FileName + ";" + _
"LeftMargin=10;" + _
"TopMargin=10;"   + _
"EnableEditing=True;" + _
"EnablePrinting=True;" + _
"PageMode=Thumbnails;" + _
"LogFile=c:\temp\PdfConvert.log;" + _

Call DoPDF(Selection, PdfPath, PdfOptions) 'Primeapple tool
```

The first part sets a selection formula that will be used by the conversion tool to find the particular document (by Universal ID) that you want to convert. The second part sets the filename for the PDF output. The third section sets various options for the PDF conversion, and there are many other options not shown here. Finally, a single line calls the Primeapple tool and converts the entire Notes document to a corresponding PDF file.

Keep in mind that when you use one of these tools to convert a Notes document to PDF, the entire document is converted, losing the individual field features. So a Notes pick-list field is converted to whatever the current value of the field is at the time of conversion. The field no longer works as a pick-list within the PDF file.

Another important point is that PDF files are often treated as read-only. If your goal is for users of the new platform to just *look at* the migrated Notes documents, this is fine. If you want users to edit the documents, you will have to provide them with an application that can edit PDF such as Abode Acrobat or Nitro PDF. Alternatively, you can use PDF as an intermediate format for conversion to Microsoft Word, as explained below.

NSF to PDF to Word

Since editing PDF files can be awkward for users, Notes documents are often more conveniently expressed as Microsoft Word files. I was unable to find a good tool for converting Notes/Domino files directly into Microsoft Word, but you can use PDF as an intermediate format. By converting Notes documents to PDF, as explained above, and then converting the PDF to Word, you can achieve surprisingly good fidelity. I recently used this method for a large, highly formatted set of documents, and my customer was happy with the results.

I recommend Nitro PDF Express for converting PDF to Word. It is inexpensive and, more importantly, does batch conversion. You can select all the PDF files in a directory and then convert them to Word, with one command.

As with Notes-to-PDF conversion, the result is a single Word file for each Notes document. All fields visible in Notes are present in Word, but the functionality of individual fields is lost. So picklists no longer have drop-down selection and field validation no longer works.

XML

Notes/Domino provides powerful programming tools to export any document, data, or design elements to XML format. You can access these XML classes from either LotusScript or Java, where relevant classes include NotesDxlExporter, NotesXslTransformer, and others.

The XML output that you can create in this way is very high-fidelity, which you can test by creating a round-trip. Generate an XML output file for a complex Notes document, and then import that same XML file back into Notes and examine the resulting document with the Notes client. I have done this, and the result is surprisingly good.

The XML programming interface allows you a lot of control over which parts of the NSF file are exported, so you can get just what you want in the XML output. You can export one document at a time, all the documents meeting certain criteria, or an entire database. You can include (or exclude) pictures, attachments, forms, views, etc. A drawback to using XML, however, is that it transfers some of the migration work down the river. This is particularly true for formatted rich-text. You can create an XML output that tells you everything about a Notes rich-text field, but you then have to interpret the XML later and map it to your new platform.

Nevertheless, because the Notes XML classes are so powerful (and fast), and with the availability of many third-party tools for XML processing, you should consider XML as part of your migration toolbox.

Java API

The Notes/Domino Java API contains approximately the same core feature set as the LotusScript API. You can use Java to create tab-separated text files, and to call the XML export classes. But Java has several key advantages over LotusScript as a Notes programming method.

- You can run Java programs standalone, without being contained in a Notes database. This is not possible with LotusScript.
- Java gives you access to the full power of the Java programming language. As much as I enjoy working with LotusScript, I must admit that Java is a stronger programming method.
- Many modern application platforms contains their own Java API, so it is likely that you can write a Java program that reads from Notes and writes to the new platform in the same code block.

The drawbacks to Java are that it is somewhat harder than LotusScript to learn, and it cannot call the Primeapple PDF routine, if you plan on using PDF as an intermediate migration format.

C API

The C-language API was the first true programming interface for Lotus Notes, coming before LotusScript or Java. The C API is also the most powerful interface to NSF files, and can do some things that cannot be done by any other method. Along with the power however comes a drawback – the C API is not for the faint of heart. Some straightforward tasks, such as simply listing the documents in a view, require a series of tricky setup calls and complicated data structures. Fortunately, most everything needed for a data migration project usually can be done through one of the simpler interfaces.

One situation where you might want to consider using the Notes C API is when the target platform has its own C-language interface. In this case, you can write a single C program that reads from Notes and writes to the new platform.

**

To perform a full-fidelity migration from Notes/Domino to another platform, there are two additional points to keep in mind.

Attachments – In many applications, the file attachments to a Notes document are just as important, perhaps more so, than the Notes data fields. Some Notes applications are essentially containers for the "real" information in the attachments. In these situations, your migration project must include proper handling of the attachments, by extracting all of them and then making sure they are inserted into corresponding locations in the new platform. Fortunately,

most of the methods described above allow attachments processing, including XML, LotusScript, Java, and C. The PDF method also includes attachment handling, in that the LotusScript code creating the PDF files can extract attachments and re-attach the same files to the PDF output.

Document links – Doc links can be a thorny issue during a migration project. Some other platforms simply do not have the concept of document links, which are so convenient in Notes applications. For platforms that do include a feature similar to doc links, you have to record two pieces of information about each Notes doc link during the export process: where the doc link was embedded, and where it pointed to. You have to assign some kind of document ID number to each exported document, and reference this ID when you resolve the doc links in the new platform. Notes provides internal document IDs, which can be helpful in this process, but the new platform will have its own document ID system, so you need to map these references properly. XML, LotusScript, Java, and C all provide the tools to perform these operations.

About the Author: *Chuck Connell is president of CHC-3 Consulting. He helps organizations use, and sometimes migrate from, Lotus products.*

For More Information

Notes/Domino documentation, including LotusScript and Java classes:
publib.boulder.ibm.com/infocenter/domhelp/v8r0/index.jsp

A sample Notes/Java program shell that shows how to get started with this interface:
www.chc-3.com/downloads.php

An article I wrote about migrating from Notes to Documentum:
www.chc-3.com/pub/MigratingNotesToDocumentum.pdf

Primeapple DominoPDF:
www.primeapple.co.uk/

Nitro PDF Express (PDF to Word):
www.nitropdf.com/express/overview.htm