Chapter 5
"Upgrading Domino Applications"
In this package, you will find:

A Biography of the authors of the book
A preview chapter from the book, Chapter 5 "Upgrading Domino Applications"
A synopsis of the book’s content
Information on where to buy this book

About the Authors

Dick McCarrick
Dick McCarrick is a freelance technical writer. He is co-author of the book Upgrading to Lotus Notes and Domino 7, aimed at Lotus Notes/Domino end users and administrators. Previously Dick worked for Lotus/IBM for over 15 years, first as a member of the Notes/Domino Documentation team and later for the developerWorks Lotus (LDD) website. At Lotus/IBM, he played a variety of roles in documenting many major components of Domino and Notes. He also wrote and edited numerous technical articles, including the regular column, "Ask Professor INI." In his spare time, Dick's leisure activities include running, fishing, woodworking, and reading about the natural sciences. An avid astronomer, he is former director of the Bridgewater (Mass.) State College Observatory.

I would like to thank my wife Lisa for her unflagging support, both for my career and my life in general.

Stephen Cooke
Stephen Cooke began programming as a hobbyist on the Apple. He later became involved with PCs in a professional capacity while working in the Czech Republic. He went on to work for Notes CS, a Lotus Business Partner in Prague. He has twelve years of consulting experience and has been working for IBM since 2000. He currently focuses on helping customers with Lotus and WebSphere-related challenges. His written work has also appeared in IBM developerWorks.

I would like to thank Tim Speed for the invitation to participate in this book, and my family for their patience and encouragement. I would also like to thank the folks at Teamstudio for their quick response to requests for information, and the people whose efforts went into preparing this book for publication.
Timothy Speed

Timothy Speed is an infrastructure and security architect for IBM Software Services for Lotus (ISSL). Tim has been involved in Internet and messaging security since 1992. Tim also participated with the Domino infrastructure at the Nagano Olympics and assisted with the Lotus Notes systems for the Sydney Olympics. His certifications include MCSE®, CISSP, Lotus Domino CLP Principal Administrator, and Lotus Domino CLP Principal Developer. Tim also is certified in Domino ND6 and D7. Tim has also co-authored six books: The Internet Security Guidebook (ISBN: 0122374711); The Personal Internet Security Guidebook (ISBN: 0126565619); Enterprise Directory and Security Implementation Guide: Designing and Implementing Directories in Your Organization (ISBN: 0121604527); Internet Security: A Jumpstart for Systems Administrators and IT Managers (ISBN: 1555582982); SSL VPN: Understanding, Evaluating and Planning Secure, Web-based Remote Access (ISBN: 1904811078); and Upgrading to Lotus Notes and Domino 7 (ISBN: 1904811639). Knowledge is based on many different facets—what you know, knowing where information can be found, and who you know. The information in this book is a combination of all these facets. Data sources have been referenced in this book, these include references to people, URLs, and other books. But much of the knowledge that is in this book comes from very smart people. First and foremost I need to thank my wife for helping me with the book and providing some of the editing throughout the various chapters. I thank my daughter Katherine for tolerating me during the months that I worked on this book. I am very grateful to Dick McCarrick for being crazy enough to coauthor this book. Special thanks to David Barnes the Development Editor. Also, thanks to Lotus/IBM (and ISSL), Walter Larry Berthelsen, and Jack Shoemaker for allowing me to coauthor this book. Thanks to Barry Heinz for reading/reviewing this book before publishing. Thanks to the content authors Dick McCarrick and Stephen Cooke.

To Linda Speed—"just me"


Finally, sorry if I missed you on this book, I will get you in the next. Sorry, Circe, you don't get an acknowledgment in this book.

Raphael Savir

Raphael Savir has been a developer and consultant for Lotus Notes/Domino applications for 15 years. He has worked in numerous positions, focusing on performance and development topics. Raphael enjoys speaking on these topics, and has been fortunate in being able to do so frequently over the years. Now with LS Development Corporation (http://www.lsdevelopment.com), Raphael works directly with clients to make efficient and friendly Notes or web applications running on the Domino platform. Raphael has written several development articles over the years, but this is his first attempt to string together more than a few pages.

I would like to thank my beautiful and encouraging wife, Lizzie, for helping me get through a brutal year of work and writing.
About the Reviewer

Barry Heinz

Barry Heinz is a Senior IT Architect with the IBM Software Group, specializing in Lotus software for large enterprises. He has been working with Lotus Notes and Domino since 1992.
Domino 7 Lotus Notes Application Development

If you're reading this book, you're probably already familiar with the Domino server. You know about all the powerful productivity features offered by this product and you know how much your company relies on it to communicate, collaborate, and manage its collective store of corporate knowledge.

This book is intended to help you with developing applications on the latest release of the Domino platform. This book has been written by Notes/Domino 'insiders'. Collectively, we possess decades of Notes/Domino experience; we've been with the product since Notes 1.0, and since then have worked directly with customers to help them with their Notes/Domino upgrade and deployment issues.

What This Book Covers

*Chapters 1 and 2* will help you understand the new features in Notes and Domino 7.

*Chapter 3* shows how to use DB2 as a data store for Domino databases so as to bring the scalability features of DB2 and the flexibility of SQL into Domino applications. The chapter shows how to install, configure, map, and then access Domino data stored in DB2.

*Chapter 4* will show you how to make the best use of new features added in Domino Designer 7 to better manage Lotus Notes and Domino applications. Specifically we will be covering Autosave, Agent Profiling, and remote Java debugging.

*Chapter 5* shows how to ensure that critical applications continue to run smoothly after you upgrade your Notes/Domino installation, while taking advantage of the new features and functionality release 7 has to offer.

*Chapter 6* will tackle issues you need to consider when upgrading your @Formula language to Notes/Domino. We first detail a backup strategy and then take a tour through the new Notes/Domino @Formulas and the potential upgrade issues they raise.

*Chapter 7* runs through the process of upgrading Domino-based agents and LotusScript; we also cover the use of TeamStudio Analyzer, which is a third-party tool to assist with your upgrade. The second half of the chapter runs through the new features available to LotusScript developers in Domino Designer 7.

*Chapter 8* examines Domino-based web services and you will see the Java implementation of one such web service. We cover the various tools Domino Designer 7 provides for interacting with WSDL and finish by examining the role UDDI plays in facilitating the adoption of web services.

*Chapter 9* covers using best practices to optimize your Domino applications for performance; specifically we will see how to efficiently code database properties, views, and forms/agents to work well in a Domino environment.

In Chapter 10, you will learn to use the new programming features offered in Lotus Notes/Domino 7 by actually implementing them in code.

In Chapter 11, we will examine two important new features, Domino Domain Monitoring (DDM) and Agent Profiles, which are critical for troubleshooting your Notes/Domino applications. Additionally, the chapter runs through several tips and techniques for identifying and correcting problems in your Notes/Domino 7 applications.

In Appendix A, we review several vendor tools that you can use to help upgrade your applications to Lotus Notes/Domino 7. These include Angkor by Atlantic Decisions, PistolStar Password Power 8 Plug-ins by PistolStar, Inc, CMT Inspector from Binary Tree, and FT Search Manager from IONET.
This chapter takes a closer look at several new features in Lotus Notes 7 and Lotus Domino Designer 7 clients that raise specific upgrade issues. In this chapter, we will identify some of those new features, show you how to implement them, and what to watch out for or think about. For a complete description of all the new features in Domino Designer 7, see Chapter 4.

When upgrading applications, you should keep two goals in mind. The first is to ensure interoperability; that is making sure that your clients can use your applications at least as well after upgrading as before. The second goal is to identify the critical handful of features whose implementation will add enough sizzle or functionality to your application for your users to applaud the upgrade (and thus be motivated to ensure the upgraded applications are quickly accepted and adopted). For your users, this mitigates the nuisance of upgrading.

Notes/Domino 7 offers some tremendous back-end improvements over previous releases. On the user-visible front-end, the improvements are more incremental. This is good news in that your users won't need extensive retraining, but of course, it also narrows the field in terms of finding those sharp new features that will make them excited to upgrade.

To help you identify which features offer the most visible and immediate value to your users, we'll take a quick tour of several features that we feel offer the most "bang for the buck" from the perspective of an end-user. First, let's examine several high-profile Lotus Notes client features added in release 7.
Lotus Notes 7 Client Features

The following list describes several of the more user-visible features that have been added or enhanced in the Lotus Notes 7 client. These features can comprise a compelling reason for your users to upgrade:

- **AutoSave** saves your work without user intervention. For example, with AutoSave enabled, if your computer crashes, you will be able to reboot and recommence working at roughly the same point where you left off in any open documents.

- **Mail and the Resource & Reservations database** are enhanced but not radically changed. On the back-end, however, the Resource & Reservations database has been dramatically upgraded to better avoid scheduling conflicts.

- **Message Disclaimer**, a highly sought after feature, allows users and/or administrators to add a message disclaimer to every outgoing email. This is done through policy documents. The disclaimer is added after the user sends the outgoing email message, as opposed to a signature that the user sees before sending.

- **Save/Close all windows** lets you close all window tabs from the File menu (via the option Close All Open Window Tabs). You can also save the state of your open windows, either from the File menu (manually) or as a default setting under User Preferences (which makes it automatic). This means that when you reopen Notes, all these window tabs will be loaded for you. Note that it is only the tab window references that are loaded, not the actual views or documents. So when you click on one of these tabs, there may be a slight delay as the view is refreshed or the document is reloaded. The alternative would be that you would have to wait for all of these views and documents to be loaded just to get into Notes, which would be unreasonable.

Of the preceding features, AutoSave in particular is likely to be of interest to your users, so we will look at it in a bit more detail later in this chapter.

New Domino Designer Client Features

Some of the important and valuable new/upgraded features for the Domino Designer 7 include the following:

- **AutoSave** (mentioned above, and described in more detail later in this chapter).

- **Agent Profiler** allows a developer to see how long every call in their agent is taking, and how many times each of those calls occurs. This is an invaluable tool for performance monitoring and troubleshooting, and we'll look at it in more detail in Chapter 11, *Troubleshooting Applications*. 
Domino Domain Monitoring (DDM) is perhaps the single most important feature in Notes/Domino 7. It provides an extensive list of troubleshooting and performance monitoring tools, a subset of which is relevant for application developers. We will examine this in more detail in Chapter 11.

Significant improvements to referencing profile documents in views have been made. In addition to changing the color scheme for a view, you can now develop applications that are much more user-defined and dynamic. This will be described in detail later in this chapter.

Right-clicks in views brings up the action bar buttons. This is an incremental improvement, but in terms of overall usability in the product, it is a nice feature to have.

Shared view columns allow a developer to create complex and widely used view columns and then apply the shared design to multiple views. Any changes to that shared column design will automatically be reflected in all the views that reference that column.

In-place design element editing of labels is a very handy way for developers to change the names of forms, views, agents, and so on, without having to open and resave the design element. While highlighting a design element, you simply click to enter in-place editing, and you can use the Tab key to move along the row to edit different labels for the same design element. This feature works much the same as InViewEditing does for Notes clients.

Web Services are described in a later section.

DB2 design integration is also discussed in a separate section.

AutoSave

As mentioned previously, AutoSave allows you to automatically save your work periodically without having to do so manually. If your computer crashes, you’ll be able to resume your work at the point AutoSave last ran. This helps avoid the situation where you lose hours of work because you forgot to save as you went along, which has probably happened to everyone at least once!

Two things have to happen to enable the AutoSave feature: the user has to turn on the setting, and the application developer must enable AutoSave for the form the user is currently working on. For users to enable AutoSave on their clients, they must select File | Preferences | User Preferences. On the Basics tab in the Startup Options section, there is a new setting AutoSave every n minutes, where n is the number of minutes between auto-saves. This interval can be from 1 to 999 minutes. (You must close and reopen Notes for AutoSave to be enabled.)
For the developer, AutoSave must be enabled on a form-by-form basis. Open the form in Domino Designer, and select Design | Form Properties. Then check the new option Allow Autosave.

With AutoSave enabled, when you are working in a document with this form and experience a computer crash or power outage, you will see the following dialog box on restarting Notes:

![IBM Lotus Notes dialog box](https://www.packtpub.com/domino-7-app-dev/book)
If you choose **Yes**, then you'll see a popup similar to the following:

![Recover Unsaved Documents dialog box](image)

Enough information is provided in the dialog box for you to make an informed decision about whether or not to recover the document(s).

To keep the AutoSave database itself clean and fast, and to prevent the user from receiving repeat warnings about the same old documents, AutoSave documents are held in a special Notes database (`.nsf` file) until they are no longer needed (because the documents have been successfully saved or recovered). The database is named with a convention that allows multiple users on the same machine to have different AutoSave databases. The filename of the database is `as_` followed by the first initial and then the entire last name of the user. So for user Fred Merkle, the AutoSave database would be called `as_fmerkle.nsf`.

**Things to Consider when Enabling AutoSave in a Form**

For the developer, enabling AutoSave is very easy, but there are some potential issues that you need to think about first. For example, if you enable this feature on a complex form that has a lot of event-driven code, you may not get satisfactory results. For simple forms (or forms that are complex only because they have many fields), AutoSave should work very well.
To illustrate this point about complex forms not necessarily working properly with AutoSave, imagine this scenario. You have an application applying for security clearance. A user creates a document and saves it, and the document is then reviewed by a manager. That manager can change the status from Submitted to Reviewed, Accepted, Rejected, and so on. When the status changes, an email is sent to all interested parties informing them of this change. The program tracks whether the status changes during a user session by way of a \texttt{PostOpen} event. It saves in memory the value of the status as the document was opened. Then, as the document is saved, the \texttt{QuerySave} event compares the current value to what is being held in memory (and this is the key). If the value is different, an email message is generated that says, for example, "Dear Fred Merkle, your request for security clearance Alpha has been reviewed and the status has been changed to REJECTED for the following reason: [explanation follows]".

If a manager experienced a crash while in the middle of reviewing a security request in this application, and then rebooted Notes and used AutoSave to recover their document, edits would be preserved (which of course is how AutoSave is supposed to work). However, AutoSave cannot preserve the in-memory value of the status field. In our example, this would create a problem, because the notification email would not be sent out. But in many applications, the forms do not use sophisticated in-memory tracking, and so AutoSave will work smoothly. In fact, even in our example, AutoSave will save your work; it just won't preserve the in-memory information. So although the work flow process would be compromised, at least your data would still be preserved.

The key here is that the developer needs to think about each form and whether or not the potential for data loss outweighs any potential compromises in how the application functions. In many cases, the answer is an easy yes, and so enabling this feature makes sense.

### Referencing Profile Documents in Views

In Notes/Domino 6, you can reference a profile document from a view in order to determine the appropriate color choices. The mail template does this, and this allows a user to specify that emails from (for instance) Fred Merkle should appear in blue, while messages from Alice Merkle should appear in orange. This is a powerful feature for enhancing the user interface of any application.

Notes/Domino 7 takes this a step further and allows you to actually populate a view column with icons based on choices made in a profile document. We'll go through a simple example, and then we'll look at how to employ this feature.
Imagine that you've got a Sales Tracking application. There are many thousands of companies and customers and products. Each month, your management team chooses a small set of customers and products that should receive special attention. Wouldn't it be nice if there were a column that had a special icon that would display whether the record referenced that customer or product? With this new feature, you can do exactly that.

Your first steps will be to create a profile document form, and then create the appropriate labels and fields within that form. In our simple example, we might have a Company field and a Product field. These two fields might be keyword lists that reference the list of companies and products respectively.

Next, you need to create a special computed text field in this profile form that resolves to the formula you'd like to see in your view column. For example, you might want the view column formula to read as follows:

```plaintext
@if(CompanyName = <company name chosen in profile document> | ProductName = <product chosen in profile document>; 59; 0)
```

This would display a dollar-bill icon if the company name in a record was the same as the value chosen in your profile document, or if the product chosen matched the product value in your profile document.

To make this formula, your profile-document formula might read as follows (and note the use of quotation marks):

```plaintext
"c := Company;
p := Product;

vCompanyName := \"" + CompanyName + \"\";
vProductName := \"" + ProductName + \"\";
@if(c = vCompanyName | p = vProductName; 59; 0)"
```

In the preceding formula, Company and Product refer to the values in the documents in the view, CompanyName and ProductName (and therefore vCompanyName and vProductName) refer to the values in the profile document.

The final step is to create a column in your view that is marked with the following Column Properties settings:

- **User definable** selected

- **Profile Document** named, for example, (Profile-Interest)

- Programmatic Name set to reference the computed text field from your profile document form, for example, $Interest

and in the formula box for this column, enter a formula such as @Random:

This will be overwitten with your profile-document's computed text formula, but it serves as a placeholder. If you substitute a simpler formula (such as 0), you will break the function, and all your documents in the view will display as replication/save conflicts.

We’d like to mention two final notes about using profile documents in views (whether they are just for color or for displaying icons). First, these profile documents cannot be per user, they must be per database. Second, every time the profile document is updated, every view that references that profile document must be rebuilt (not just refreshed). So this is a very handy feature for an administrator or database owner to be able to use, but it might be dangerous if all 5,000 users of your application have access to a button that lets them update the profile document in question.

**Web Services**

In an ongoing effort to make Domino more compliant with open standards, Notes/Domino 7 offers the ability to make your Domino server a hosted Web Service. This means that clients can make Web Service requests from your Domino server, just as they would from any Web Service.

In the Domino Designer 7 client, there is a new design element under **Shared Code** called **Web Services**. A Web Service design element is very similar in structure and function to an agent, and in fact, most of the user interface is the same. In particular, the **Security** tab is the same, providing restrictions on agent/Web Service activity, who can see the agent/Web Service, and whether or not this should run as a web user, as the signer of the design element, or under the authority of another user.

A significant feature of Web Services is that they return data to the requesting client as XML data.

**DB2 Design Integration**

In Notes/Domino 7, you have the ability to use a DB2 back-end to provide robust query capabilities across multiple Notes databases. There is some work required to set this up:
Upgrading Domino Applications

- DB2 must be installed. This does not come with Domino; it is a separate install.
- There is some Domino configuration required, mostly to set up as an authorized DB2 user account.

The Notes database(s) that you want to integrate with DB2 must be created as DB2-enabled. If you have an existing database that you want to convert, you can make a copy or replica copy and mark it as a DB2 database. (It is not possible to convert existing databases simply through your new server configuration, or through a server-console command.) In your Notes database, you can then create a DB2 Access View (DAV), which sets up the appropriate tables in the DB2 storage. Note that you may have to clean your data to avoid data-type conflicts. Also note that these will add to the size of your storage needs, but will not be reflected in the size of the database as recorded in any of the Domino housekeeping reports. However, this size is usually fairly nominal.

In your Notes database, you can now create views that use SQL statements to leverage these DAVs and which can display results from across multiple documents, even across multiple databases, for use within a single Notes view. A simple example would be a view that displays customer information as well as the internal sales rep's name from SalesActivity documents in the SalesTracking database, and also displays the sales rep's information, which comes from your SalesRepInfo database. A table similar to the one below will display in your view, with the first six columns coming from SalesTracking and the right-most column (Rep Phone) coming from SalesRepInfo.

<table>
<thead>
<tr>
<th>Company</th>
<th>Customer</th>
<th>City</th>
<th>State</th>
<th>Zip</th>
<th>Rep</th>
<th>Rep Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACME, Inc.</td>
<td>Alice Smith</td>
<td>Boston</td>
<td>MA</td>
<td>02100</td>
<td>Fred Merkle</td>
<td>617.555.5555</td>
</tr>
<tr>
<td>ACME, Inc.</td>
<td>Betty Rogers</td>
<td>Boston</td>
<td>MA</td>
<td>02100</td>
<td>Jane Merkle</td>
<td>617.444.4444</td>
</tr>
</tbody>
</table>

Tips when Using DB2

There is no direct performance gain from DB2-enabling your Notes database(s). Although the file structure of a DB2 database is far more efficient than Domino for storing large quantities of well-defined data, these gains cannot be realized by the combined setup of a DB2-enabled Domino database. On the other hand, if you have a business need for various views or reports that combine data from multiple sources (as with our simple example above), then you can consider DB2-enabling your databases as a very high-performance alternative to writing your own code to mimic this feature.

If you make a database copy (or replica copy) on a server or workstation that does not have DB2 installed, you will have to make this a regular NSF copy, and it will not have the capability to do the special SQL query views. However, your NSF data will be entirely preserved.

Template Management
When you upgrade your servers, you are likely to upgrade some or all of the standard templates: Domino Directory, Mail, Resource & Reservations, Discussion, and so on. There are three major steps you need to perform to ensure compatibility throughout your upgrade process:

1. Review code
2. Customize new templates
3. Recompile script libraries

These steps are logical and sensible, but easily overlooked in the hustle and bustle of upgrading servers, clients, and your own customized application code.

Reviewing Code
The first step is to know what code has changed between your old templates and the new (release 7) templates, and simply to test your client mix (as most customers will not upgrade every single client simultaneously) against these new features/templates. You can determine the code differences by running a utility that will break down all design changes. (See Appendix A for more information about tools.) After you determine what code has changed, you must perform some analysis to decide what is worth testing—it’s better to identify whatever errors or problems you may encounter before you upgrade.

Customizing New Templates
If you have customized your mail template, or any other standard template, you’ll need to take one further step. You should have documented all of your own customizations, and now you’ll need to analyze your customizations against the code changes in the new templates, and then apply your customizations appropriately to the new templates. In most cases, this presents no problems. However, sometimes design elements are added or removed, field names that you had reserved for your own use are now used by the template, or subs/functions which had been used in the old template are completely missing in the new template. So this too needs careful analysis.

Recompiling Script Libraries
Finally, for any application that has your own code, whether partly or wholly customized, you’ll want to recompile all the LotusScript under Notes/Domino 7. To do this, open the Script Libraries view in Domino Designer, and select

**Upgrading Domino Applications**

**Tools | Recompile All Scripts.** Depending upon the size of the application, this may take some time, as it has to parse through all your code twice, once to build up a dependency tree and again to find any code problems.

![About Compiling All LotusScript](image)

When the compiling is complete, you will be presented with a dialog box that lists the design elements with problems. You can go directly into that design element from the dialog box to fix the problems. In the following example, we have changed a script library sub that is referenced from two forms, Recompile Script1 and Recompile Script2:

![Compile all LotusScript in nd7features](image)

Note that if you click on one of these design elements in the dialog box and click **OK**, it will open that design element for you, but it won't put you directly into the event, button, sub, and so on that needs fixing. You'll still have to find that yourself. One way is to make a meaningless edit (such as inserting a space and then removing it) and then try to save. The form will now catch the error, so it will more helpfully point out where the error is coming from.

Note that after you finish with that form, and save and close it, you will not go back to the preceding dialog box. Instead, you’ll be back in the Script Libraries view. To return to the dialog box, you will have to select **Tools | Recompile All Scripts** again.

**A Final Note about Templates**

If you are not experienced with templates, be careful the first few times you work with them. Templates will push changes, by default, every night into all databases referencing the templates on that server. That means that if you make changes directly to the design elements in the application, you risk having your changes overwritten by the template. Worse, if you check off the setting to prohibit design template updates, then you risk having your design out of synch with template changes.

Under Design properties in the third tab, you can select the option **Prohibit design refresh or replace to Modify**. But this risks making your design out of synch with the template. Typically, you would do this for a troubleshooting view that does not need to reside in the template.

On the **Design** tab of Database properties, you can assign a template for this database. Doing so will refresh/replace all design elements in your database, as needed, from the template on that server every night.
Summary

In this chapter, we’ve discussed several new and enhanced Notes/Domino 7 features that raise particularly interesting application upgrade issues. These features include AutoSave, the ability to reference profile documents in views, Web Services, and DB2 integration. We also took a look at managing your Notes/Domino templates to accommodate the updates and enhancements made to them in Notes/Domino 7. This information will help ensure that your critical applications continue to run smoothly after you upgrade your Notes/Domino installation, while taking advantage of the new features and functionality release 7 has to offer.
Where to buy this book

You can buy Domino 7 Lotus Notes Application Development from the Packt Publishing website:  
http://www.packtpub.com/domino-7-app-dev/book

Free shipping to the US, UK, Europe, Australia, New Zealand and India.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.