
**Level:** All
**Works with:** Notes/Domino
**Updated:** 01-Aug-2003

by David Byrd
and Timothy Speed

You have been designing a Lotus Notes database for the last 48 hours on your laptop—adding and updating design elements, adding graphics, and developing with LotusScript—making the ultimate "cool" application. It is about 11 PM on a Friday night. You are tired and ready to go home. You replicate your changes from your laptop to a Domino server, gulp down that last drop of coffee, and trip on the stairs on your way out the door. You are fast asleep, but in the middle of a dream, you wake up. You have an epiphany about a change that you must put into your cool database—now! It's 4 AM. You dial up the server and access the database that you have been working on. You open Domino Designer and look for your design elements directly on the server. To your horror, all of your new design elements are gone! All your work from the last two days is missing. All that is left are the standard design elements from the discussion database template. At first, you are profane, then you deny that your work is not there. Neither solves the problem, so finally you decide that it must be hackers or even the Klingons. What has happened?

Answer: You created a new database based on a template, and the inheritance of the template name was left in the database properties. The design server task ran at 1 AM and replaced the design based on the template inheritance name. The good news is that nothing is broken; this is all normal. In this case, let's hope you did not replicate the changes from your laptop to the server. Otherwise, all of this work could be lost.

This two-part article series reviews some of the issues and problems that you can experience with Domino templates. When you create a database from a template, regardless if it is a Notes standard template such as mail.ntf or a custom designed template, the database inherits design elements from the template. If configured correctly, databases that inherit their designs from master templates receive the latest changes through the nightly design server task. Templates can be used for many different purposes:

- To create a new database based on a template theme (discussion, database library, address book, and so on).
- To update application design.
- To update a single (NSF) application with elements from more than one template.
- To transfer a design to another developer via a template.
- To invoke a signature onto application design elements for ECLs (Execution Control Lists).
- To update several applications with the same design (for example, mail files).
- To reduce disk space with the new Single Copy Template feature that leverages a single copy of design elements for multiple databases.

This article introduces you to templates and explains in detail how they work. We also show you how to build a tool to help manage your templates. (The complete code for this tool is available for download from the Sandbox .) In Part 2 of this series, we'll discuss template best practices and other tools you can use. This article series

assumes some familiarity with Notes/Domino programming, although it should still be useful even if you're a relative novice to using Notes/Domino templates. Also, we suggest you read the *LDD Today* article "Enhancements to Notes/Domino 6.0.1 and 6.0.2," which includes descriptions of the latest template features.

## A template primer

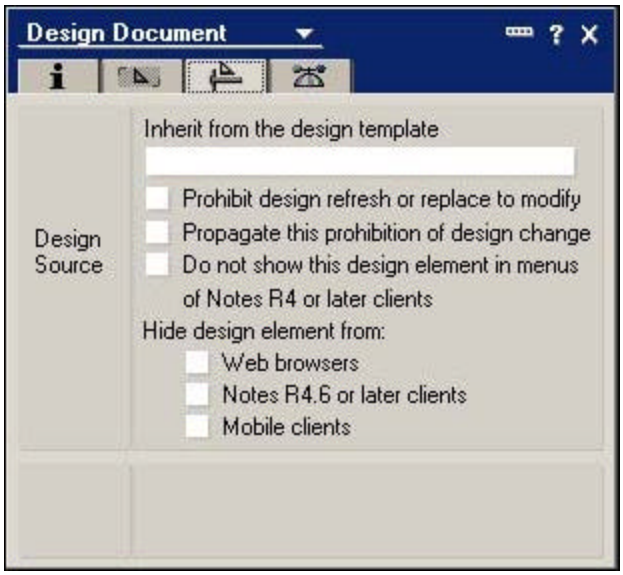
Database templates allow you a great amount of control over the management and deployment of applications. Each template can control (either directly or indirectly) the design of many different databases. The database itself has an overall inheritance template setting, but this can be refined through multiple inheritance techniques:

- *Single database template*
  This setup is the most basic because a database inherits only from a single template. The template name is defined in the database design properties.
- *Single database template with additional external design elements*
  This setup is very similar to the first except that additional design elements are added to the database which then inherit from additional templates beyond the main database level template. The advantage of this is that you can use a single template as a code library for reuse in various applications.
- *Templates that inherit from another template*
  This setup is also similar to the first, but adds to the twist that the templates are themselves Notes databases. A template can inherit from another template directly as in the first scenario or through the addition of other design elements as in the second. The combination of these inheritances is carried into the main database.
- *Script libraries*
  This setup is useful for controlling the design of agents. If you make a change to an agent, in some cases, you need to re-enable and reschedule the agent. If you make a change to a script library that is being used by an agent, you don’t need to re-enable and/or reschedule that agent

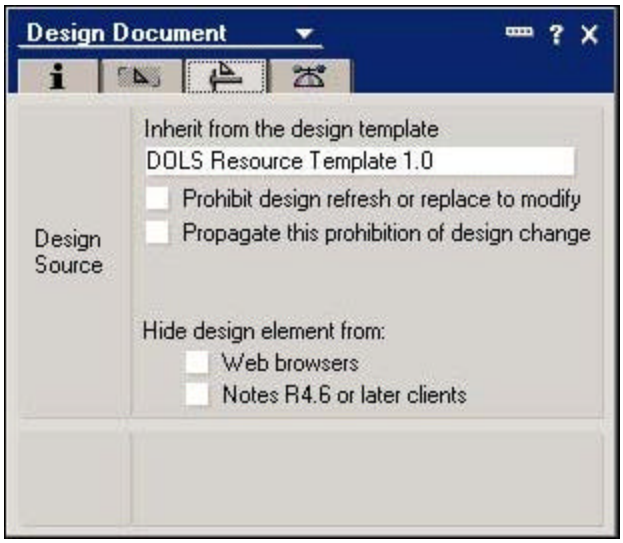
You can also set inheritance at the design element level. These settings allow you even tighter control over the database. Most database design elements allow you to override the template name defined in the database design properties. Right-clicking the design element from Domino Designer lets you view and change these settings. The setting shown in the Design Document dialog box changes based on design element type. For instance, the database icon design element only allows you to prohibit design refresh or replace or modify. Other design elements include specific features, such as Run Agent as Web user for the agent design element. Also, you can hide individual design elements at runtime from either Notes clients or Web browsers selectively. This is a heavily used feature by developers building Notes applications for use through both clients. The last template-related feature on the Design Document dialog box is the Prohibit and Propagate settings. These prevent the design element from being updated by a template refresh or replace. The "Propagate this prohibition of design change" setting causes the design elements to inherit this setting from the template.

This is the standard Design Document for most design elements.

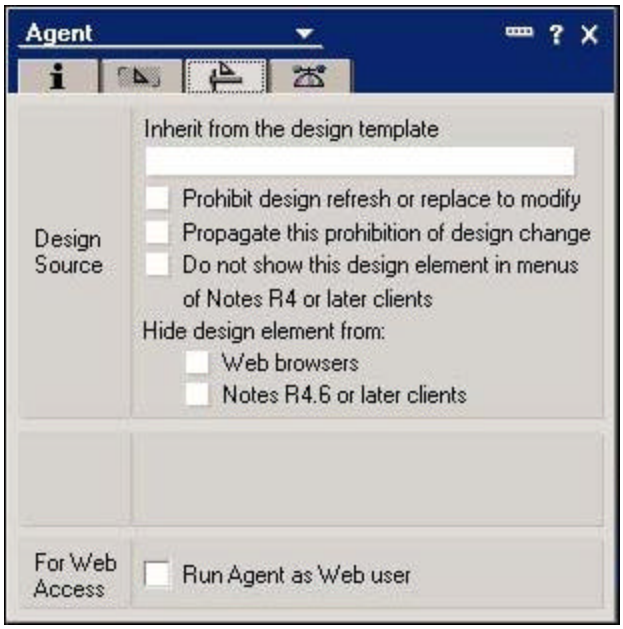
 




This dialog box demonstrates specific inheritance from an external template for only this design element, in this case a frameset. This design type did not exist in R4 so the "Do not show this design element in menus of Notes R4 or later clients" setting is not necessary and does not display:




In R5, the Agent dialog box adds the Run Agent as Web User setting to the mix:

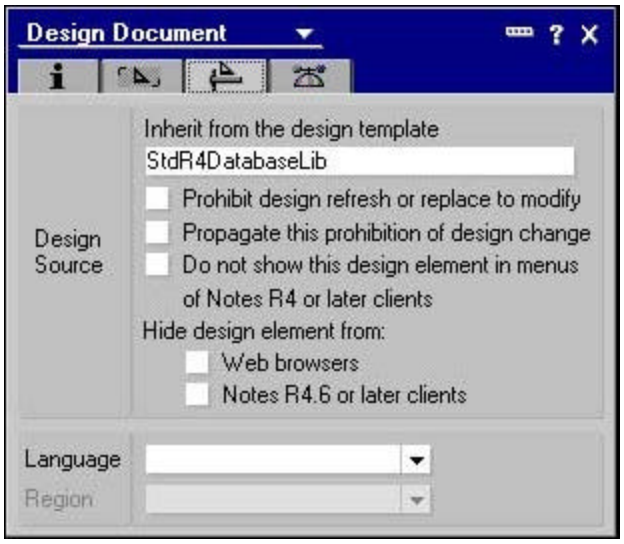



However, in Notes/Domino 6 this Run Agent as Web User option is configured as part of the Agent Properties box:


The next Design Document dialog box is specific to the Global database elements. These include the Using Database document, About this Database document, and database script:

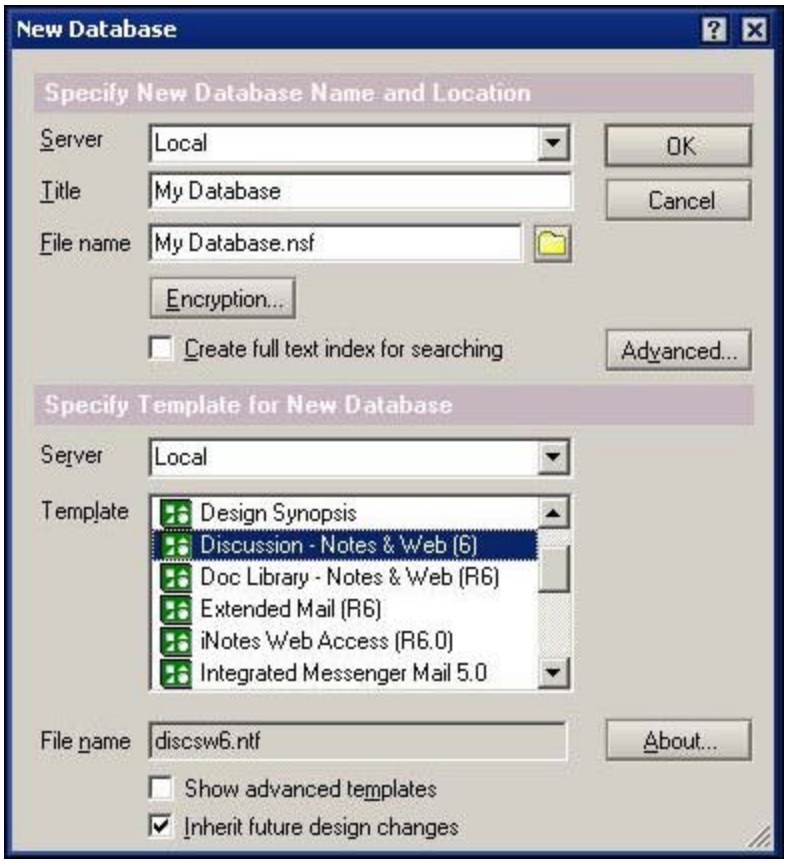

The Icon design element gets a custom Design Document dialog box as well. This design element stores various pieces of information about the database besides the icon. This includes information such as the Inherit from name and database launch properties. Using a tool such as NotesPeek displays this additional information:


A single database can also inherit from several templates. Here is an example: Select the Design tab, and enter the name of the template you want this design element to inherit from in the “Inherit from the design template” field. Whenever you refresh a database, these elements are pulled automatically from the templates you have designated:

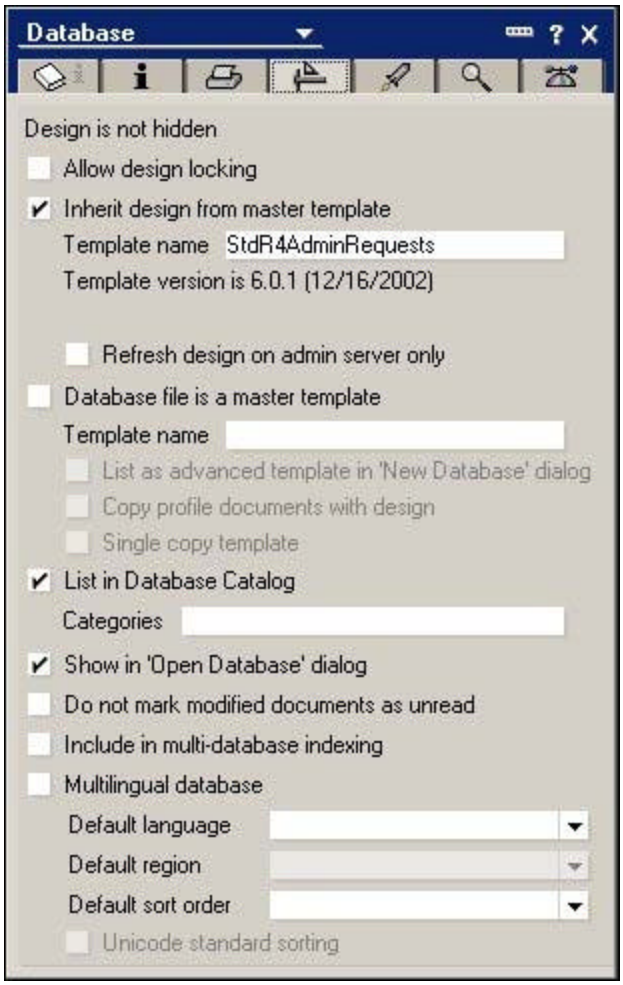



## Template settings

So how can you use a template out-of-the-box? Let’s say you want to create a discussion database. In Domino Designer, select File – Database – New. This displays the following dialog box:




Most of you have created databases this way, easy and quick. Let's review a few of the settings and properties for a database and template. In the following screen, you can see the settings that can control a template or a database for inheritance:

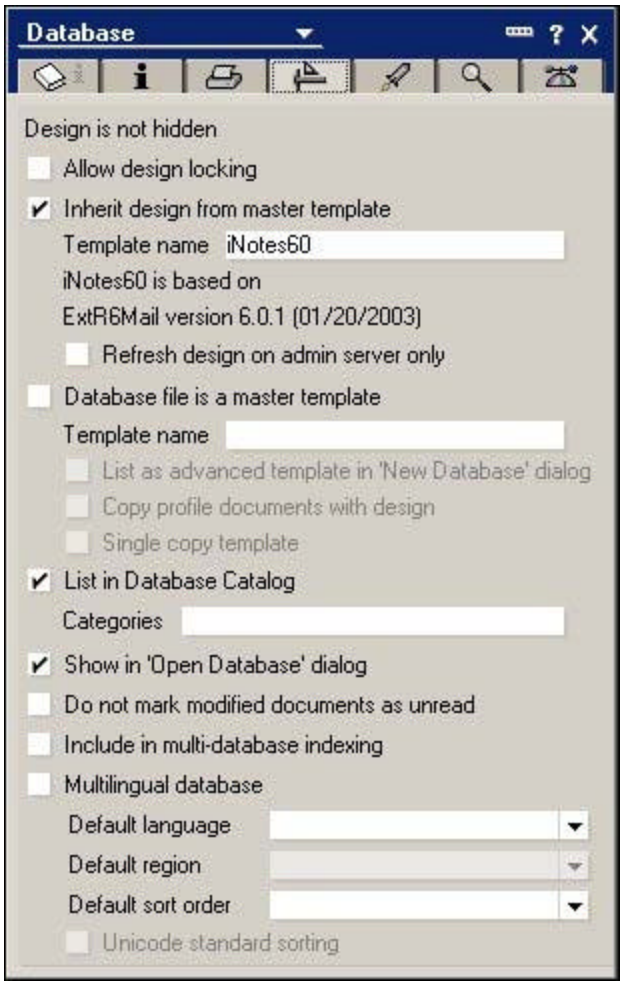



This is where you can get into trouble right from the start. When you create a new application, be sure to review the Inherit design from template setting. If you use a template to create a custom application, change the name of the Inherit design from template value in the design properties immediately. (Most servers run the design task daily to refresh designs.) After you create a custom database, you can subsequently use it as a template to create other applications.

One item to note is the template version information below the inherited master template name. This new feature (introduced in Notes/Domino 6.0.1) lets you maintain a shared field called $TemplateBuild, allowing for easier template identification. The field's properties must be set via background with LotusScript or formula language. (The two key fields containing this data are $TemplateBuildName and $TemplateBuildDate. The Notes/Domino 6 discussion forum contains sample LotusScript code for retrieving this information written by Thomas Gumz.)

Databases are associated with templates via a design property. In the following screen, you see that this mail file inherits its design from the iNotes60 template:




Note that you should sign the template and individual design elements. When a user opens a database or any design element, the signature is checked against the workstation ECL (Execution Control List) prior to allowing its use. Proper planning of the workstation ECLs is beyond the scope of this article; for more information, see the Notes Client help.

## Refreshing the database design

There are three methods available to refresh/replace an existing databases design:

- Design server task
- Convert server task
- The Notes client

**Design task**

The Domino server design task runs by default every day at 1 AM. It performs a design refresh against all databases that refresh from any template. The design task searches for a matching name (for example, iNotes60.ntf) in all of the templates in the Notes data directory of the server. After the design process has a handle on the template, it compares each database element between the database (NSF) and the template (NTF or NSF). If a newer design element is found, it replaces the design element in the database. It is important to note that this is only a refresh, not a replacement. This could be at the database or the design element level.

When the design task first starts up, it scans all databases building a list of templates available. This process sometimes produces errors if two templates are configured with the same name, for example:

04/28/2003 01:00:56 AM WARNING: Both nntpdi50.ntf and nntpdi46.ntf claim to be Design Template

'StdR5.0NNTPDisc'

To avoid this, make sure that you don't have any template names duplicated. If you do, then the refresh design processes (manual or server-based) use the first design element based on the alphabetical file name of the template.

After the template scan is finished, the design task processes the databases on the server looking for elements that have a template name associated with them. If the template cannot be found, the following error message is displayed:

04/28/2003 1:01:06 AM Warning: Cannot locate design note 'servers.gif' in 'DOLS Resource Template' template

This error can be a cause for concern if the template is on the server. However, you may also see this message in non-error situations, especially with some of the system databases. If a matching template is found, then the design elements are compared by name to see if they should be updated. If the template contains a different version, the design element is added, updated, or removed from the database unless the "Prohibit design refresh or replace to modify" setting is turned on. The server log displays a message similar to the following as it is processing these changes.

04/28/2003 1:01:46 AM Updating 'DOLS Configuration Settings' into database 'Extended Mail (R6)' from template 'DOLS Resource Template'
04/28/2003 1:01:46 AM Updating 'DOLS Download Control' into database 'Extended Mail (R6)' from template 'DOLS Resource Template'
04/28/2003 1:01:46 AM Updating 'DOLS Download Instructions' into database 'Extended Mail (R6)' from template 'DOLS Resource Template'
04/28/2003 1:01:46 AM Updating 'DOLS Load Download Page' into database 'Extended Mail (R6)' from template 'DOLS Resource Template'
04/28/2003 1:01:46 AM Updating 'DOLS Offline Configuration' into database 'Extended Mail (R6)' from template 'DOLS Resource Template'
04/28/2003 1:01:46 AM Updating 'DOLS Request Offline ID' into database 'Extended Mail (R6)' from template 'DOLS Resource Template

If a design element mysteriously disappears from the database, this is the place to look!

**Convert task**
The convert server task is another tool used in the template world. This server task has historically been used to update mail databases, but can be used with any database. Unlike the design task, convert performs a replace and not a refresh operation. The task has several command line options to control its operation. In the following command example, the convert task replaces the design of a user's mail database regardless of the current setting indicated by the asterisk (*) with the Lotus Notes 6 mail template:

load convert mail\user.nsf * mail6.ntf

This option of having the templates located elsewhere allows you more control over deployment. Logging for the process is done through the status bar and in the Notes 6 client, does not occur as a background process.

**Notes client**
The Notes client also contains logic to perform both the refresh and replace operations. This is very valuable because this lets you update a database with a template and have the database and template reside on any server. (The design and convert server tasks require the template to reside locally on the server.) You can do this via the Notes workspace or by use of a bookmark. Select the database to refresh and right-click the icon. Then s elect Database - Refresh Design. This refreshes the design elements that have been changed.

When you refresh a design via your client, you may see errors in the status bar, for instance:

Both bubba.ntf and Joebob.ntf claim to the Design Template 'Accounting'

You also see this error in the log.nsf file after the design task runs. If this occurs, then the first database listed alphabetically refreshes the design. So in this case, the changes from bubba.ntf are used to update Accounting.nsf and any changes in Joebob.ntf are ignored.

## Duplicate template names tool

Obviously, it would be useful to find duplicate template names before you start the refresh process. To do this, you can use a third-party tool. Or you can write your own. The following example shows the code for a tool to find duplicate template names. (You can download this code from the Sandbox.)

To start, create a new database named search.nsf to use a blank template. Then in Domino Designer create a new agent in this database and name it Search. Set the agent runtime type to Action List Selected so the agent appears in the Actions menu. Then select LotusScript in the Formula type list. In the Declarations object of the agent, insert the following code:

```
Dim WhatServer As String
Dim dbd As NotesDbDirectory
Dim tmpDb As NotesDatabase
Dim Tdb As NotesDatabase
Dim SelectDBtype As String
Dim CountDb As Long
Dim view As NotesView
Dim ServerError As String
Dim currentdb As NotesDatabase
Dim item1 As NotesItem
Dim item2 As NotesItem
Dim doc As NotesDocument
Dim DefaultValue As String
Dim CheckTemplateValue As String
Dim AllOK As String
Dim NotesDatabase As NotesDatabase
Dim ServerPath As String
Dim dbtype As Variant
Dim DisplayServer As String
```

In the Initialize object of the agent, place this code:

```
Sub Initialize (This line is shown for reference only)
    Dim profSession As New NotesSession
    Dim session As New NotesSession
askagain:
    WhatServer$ = Inputbox$("Enter Server name - 'exit' -to end program   - enter nothing for Local")
' Leave blank for Local
    If WhatServer$ = "exit" Then
        Exit Sub
    End If
    If WhatServer$ = "" Then
        DisplayServer$ = "Local"
    Else
        DisplayServer$ = WhatServer$
    End If


    AllOK$ = Lcase(Inputbox$("You have selected ---- " & DisplayServer$ & " -- Is this Correct? Yes/No"))
    If AllOK$ = "yes" Then
'  All Done
    Else
        Goto askagain
```

```
    End If

NextStuff:
' The next two lines will get a handle on the Notes data directory
    Set dbd = New NotesDbDirectory( WhatServer$)
'  Use as a generic counter - how many DBs are processed.
    CountDb = 1
DBorTemplate:

' The next lines are used to determine if you want to process templates or databases.

    SelectDbtype$ = Lcase(Inputbox$("Select Database or Template"))
    If SelectDbtype$ = "database" Then
        dbType = DATABASE
        Set tmpDb = dbd.GetFirstDatabase( dbtype)
    Elseif SelectDbtype$ = "template" Then
        dbType = TEMPLATE
        Set tmpDb = dbd.GetFirstDatabase( dbtype)
    Else
        Goto DborTemplate
    End If

'  Here's the meat of the code. This gets a handle on each database in the Notes data directory and extracts the
template information.

    Do While Not ( tmpDb Is Nothing )
        Serverpath$ = tmpdb.FilePath
        Call tmpDb.Open( WhatServer$,ServerPath$)
        Serverpath$ = tmpdb.FilePath
        Set notesDatabase = New NotesDatabase( WhatServer$, ServerPath$ )

' The next lines are optional, but useful for troubleshooting your code:

        If notesDatabase.IsOpen Then
            Print( "Successfully opened " & notesDatabase .Title )
        Else
            Print( "Unable to open database" )
        End If
        Print Serverpath$
        Set Tdb = session.CurrentDatabase
        Set doc = tdb.CreateDocument
        doc.Subject = "New Template Entry"
        doc.form = "New Template Entry"
        doc.ServerName = WhatServer$
        doc.RepID=tmpDb.ReplicaID
        doc.pathname = Serverpath$
        doc.dbtype = SelectDbtype$
        doc.templatename = tmpDB.DesignTemplateName
        doc.ActualDBtitle = tmpDB.Title
        CheckTemplateValue$ = tmpDB.DesignTemplateName

' These lines are used as part of the user interface to show which settings are enabled for the inheritance:

        If tmpDB.DesignTemplateName = "" Then
            doc.x= ""
        Else
            doc.x= "x"
```

```
        End If
        doc.databaseisatemplate = tmpDB.TemplateName
        If tmpDB.TemplateName= "" Then
            doc.y= ""
        Else
            doc.y= "x"
        End If
        Call doc.Save( True, True )
        Call tmpDb.Close
nextentry:
        Set tmpDb = dbd.GetNextDatabase

' Finally, these optional lines are used for troubleshooting and/or process counting:

        CountDb = CountDb + 1
        Print CountDb
    Loop

End Sub  (This line is show for reference only)
```

The next step is to create a form with the following fields in your search.nsf database.

- *Subject*
  This acts as an informational field.
- *ServerName*
  This is the name of the server where the data was extracted.
- *RepID*
  This is the replica ID of each database found in your search.
- *pathname*
  This is where the database is in the Notes data root path.
- *dbtype*
  This is a template or database.
- *templatename*
  This is the inheritance name of each database found in your search.
- *databaseisatemplate*
  This is the name of the template.
- *ActualDBtitle*
  This is the title in the database or template name.

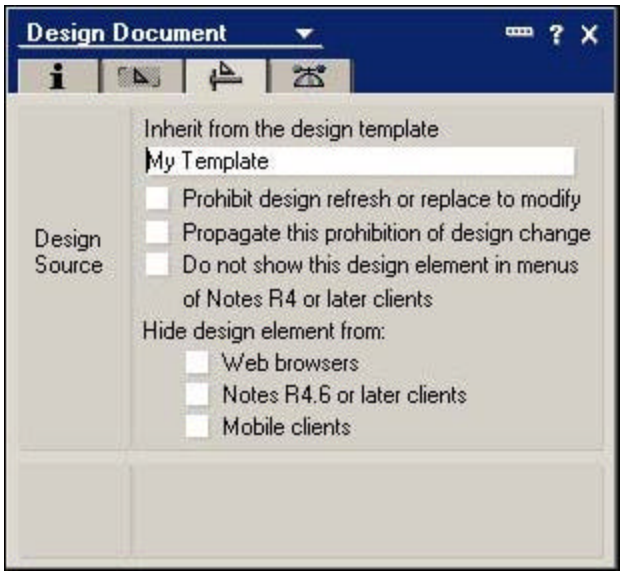
Also, now that this data is placed in a document you can create views and agents to report the data.

Another way to find design names, including field names, is the Notes C API. The preceding LotusScript code gives a summary of the overall database design names. This does not take into account the fact that each design element can have independent inheritance names. These template references can only be retrieved by scanning all design elements, checking for the $Class field. When scanning all the design elements within a database, the $Class field contains the design element level Inherit template name. If the $Class field does not exist, then the design element does not inherit from an external source explicitly. A good Notes C API application to use as a base for this type of application is the des_coll sample located in the directory notesapi\samples\dbdesign\des_coll. This example prints a list of all design elements located within a given database that can easily be extended to check for the existence of the $Class field printing its value.

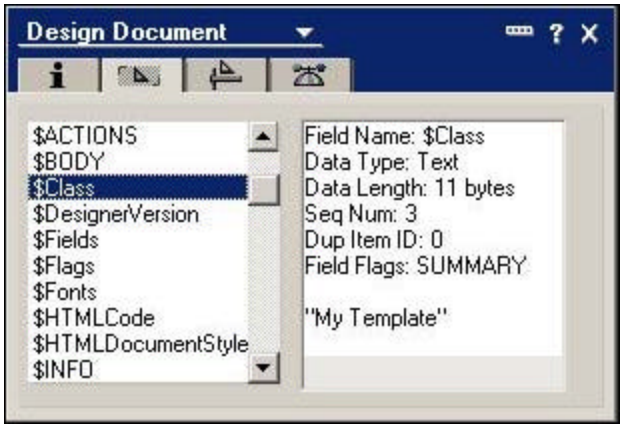
## Retrieving design element inheritance information

This section explains how to retrieve individual design element inheritance names. First, set the "Inherit from the design template" setting to My Template:




Next check the fields contained within the design element and examine the value of the $Class field. This should not contain the design template name:




Then clear the Inherit name field, removing the $Class field:


## Design elements that don't refresh

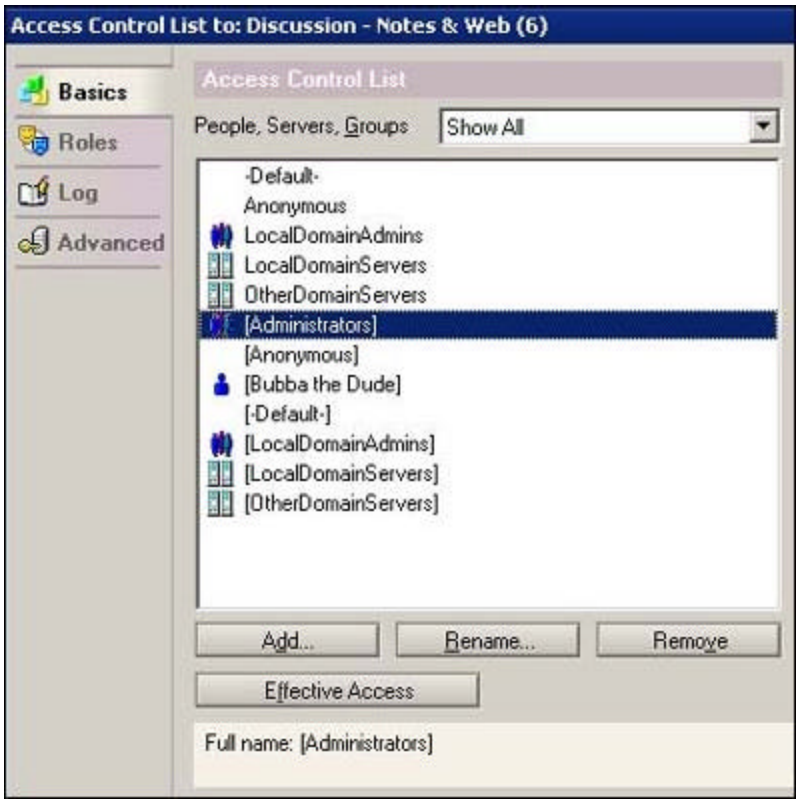
One issue that we run across is that developers don't always know which elements don't normally refresh. Here

is a list of design elements that do **not** refresh from a template:

- Database title
- Database type
- Disable background agents for this database setting
- Settings on the Info tab
- Settings on the Printing tab
- List in Database catalog setting
- Show in 'Open Database' dialog setting
- Template inheritance names (these values must match in the NSF and the NTF)
- Include in multi-database indexing setting
- Multilingual database data (default language, default region, default sort order)
- Full-text indexing options and settings
- Settings on the Advanced tab
- About This Database document
- New documents

## Setting ACLs via templates

One nice feature that you can take advantage of is that you can add ACL entries to a new database based on the entries in the template. In the template, add the ACL entry that you want added to the database. Put the ACL name in brackets, and then set access level and user type, for example:




Now when you create a new database based on this template, it places the ACL entry in the new NSF file. Note that you can add ACL entries to new databases by placing the name of the Person or Group in brackets, for example [Administrators]. One possible use of this is to add an Administrators group to mail.box (via mailbox.ntf) and to users mail files (via mail6.ntf).

## Templates: the developer's friends

In this article, we introduced you to the inner workings of Notes templates. We gave you a brief description of how templates work and what you can do with them, including controlling design changes and setting ACLs. We

also explained how you can retrieve design information, providing you with a tool to help you do this. In Part 2, we'll continue our tour of Notes templates, offering best practices tools and techniques. Stay tuned!

**ABOUT THE AUTHORS**

David Byrd is a Consulting IT Architect with IBM Software Services for Lotus (ISSL) from Atlanta, GA. David is fluent in virtually all areas of Lotus products and technologies ranging from C/C++ API development and application, security, and messaging architectures with heavy focus on enterprise-level messaging and directories. He has worked with Lotus Notes and Domino since the early 1990's and holds numerous certifications from Lotus, IBM, Redhat, Microsoft, and Novell. You can email David at david_byrd@us.ibm.com.

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. He also participated with the Domino infrastructure team at the Nagano Olympics and assisted with the Lotus Notes systems for the Sydney Olympics. His certifications include MCSE©, VCA (VeriSign Certified Administrator), Lotus Domino CLP Principal Administrator, and Lotus Domino CLP Principal Developer. Tim has also co-authored four books: *The Internet Security Guidebook* , ISBN: 0122374711, February, 2001; *The Personal Internet Security Guidebook* , ISBN: 0126565619, October, 2001; *Enterprise Directory and Security Implementation Guide: Designing and Implementing Directories in Your Organization* , ISBN:0121604527; and *Internet Security: A Jumpstart for Systems Administrators and IT Managers,* ISBN 1555582982. You can reach Tim at Tim Speed@us.ibm.com.