

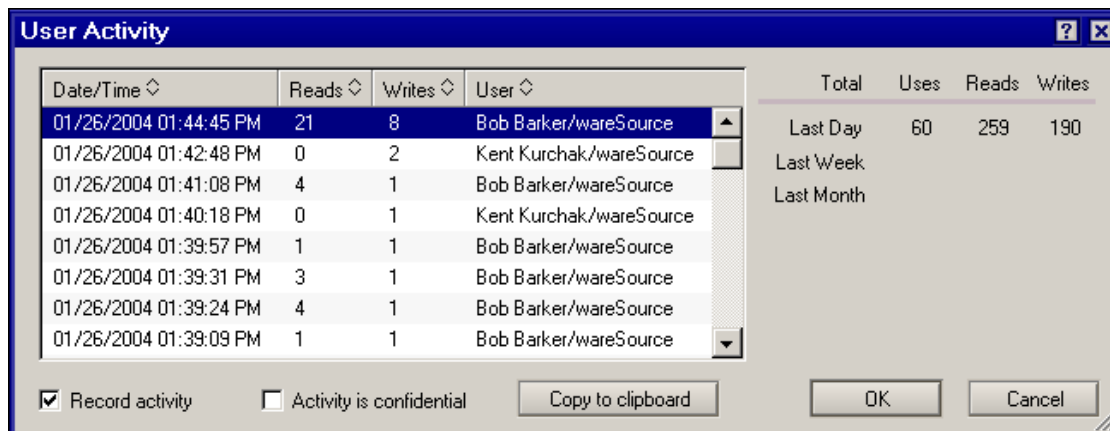
Notes 6.x Application Strategies: User Activity Tracking

It is essential for Domino administrators to know how Notes databases are being used. With this information, they can keep servers operating at peak efficiency, spread processing, disk, and network loads across multiple servers, and can better guarantee responsive performance for users. For administrators, Domino comes out of the box with activity logging and Domino Administrator includes powerful analysis tools to understand the logging data collected.

This article is the fourth in a series about application strategies for Notes/Domino 6 and later. It describes how to add granular, database-specific user activity tracking to the design of any Notes database. This detail tracking can tell you when a user opens a database as well as when he opens a document for reading or editing. Whereas general database logging is of interest to administrators, the kind of activity tracking shown in this article is of greater interest to database developers, managers, and compliance officers.

Built-in user activity logging

Before getting into database-specific logging, first let me review the type of logging that Domino includes for the purpose of server performance analysis and resource allocation. By default user activity is recorded on the per-database level with respect to access and document reads/writes. Any user with Reader access and greater to the database can read the User Activity log from the Database properties Info tab (click the User Detail button):



By default Record activity is selected. You can disable activity recording by de-selecting this option. Be aware of two things with respect to disabling activity recording in the User Activity dialog box:

- You only affect the User Activity dialog box; usage data is still recorded to the *Domino Server Log* database in the *Session* documents found in the *Database\Usage* and *\Sizes* views.

- The next time the StatLog server task runs (loaded by default at 5 AM), it enables activity logging for all databases (thereby overriding your de-selection) and gathers usage data from the databases in the data directory on the server (data is stored in the *Domino Server Log* database). You can prevent StatLog from enabling the **Record activity** flag for all databases on the server by setting the `NO_FORCE_ACTIVITY_LOGGING=1` variable in the server's `NOTES.INI`. Usage data continues to be logged in the *Domino Server Log* database—it just won't be visible in the User Activity dialog box.

To limit who can read the logged activity to only database Managers, select the Activity is confidential option.

Note that there is a lag in the time it takes for the data to be written, but it is eventually displayed.

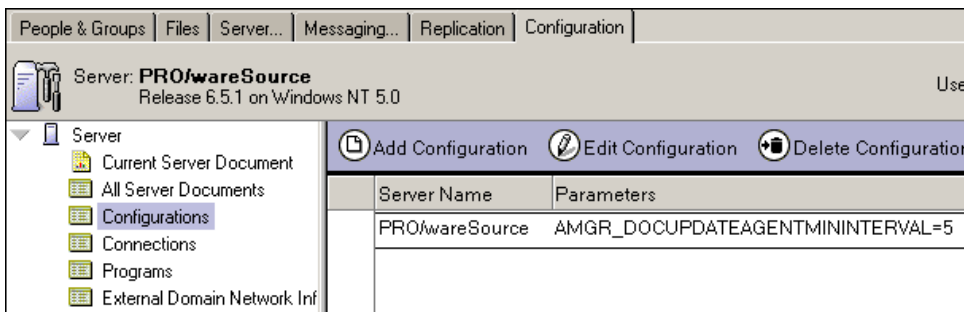
User activity data is periodically collected from all the databases and summarized in the Domino Server Log (`log.nsf`) database. To view the user activity data from Domino Administrator, select the server and then open the Server\Analysis function tab and expand the Domino Server Log to the Usage\By User view:

User	Date	Time	Minutes	Reads	Writes	KBytes	Transactions
▼ Kent Kurchak\wareSou			11769	7983	6922	1122343	420784
▼ 03/01			2228	997	1606	184856	68519
		07:43 AM	517	505	385	12742	1780
		07:43 AM	0	1	1	5	13
		07:44 AM	0	0	0	2	7
		07:44 AM	10	0	23	966	168
		07:45 AM	187	0	344	35992	22632
		07:53 AM	0	0	0	3	9
		07:58 AM	586	0	0	252	344
		08:03 AM	0	0	0	2	6

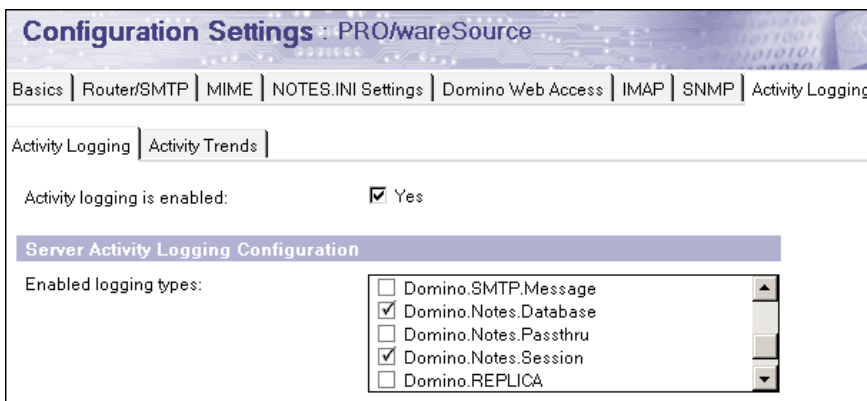
Open the desired document to see the detail data of the database user access:

Session Activity					
Server:	PRO/wareSource				
Time:	01/26 01:36 PM - 01/26 01:36 PM				
User:	Bob Barker/wareSource				
Elapsed Time:	0.1 minutes				
Documents Read:	1				
Documents Written:	0				
Network Port:	TCPIP				
Network Usage:	17 KBytes transferred				
Transactions:	29				
Database(s)	Documents Read	Documents Written	Time Open (Secs)	Bytes Read	Byte Written
Articles\useractivity.nsf	1	0	9	14636	1678

For even more detailed activity logging and analysis, the Domino administrator can enable Activity Logging (this is different than database user activity shown above). To enable Activity Logging from Domino Administrator, select the server and then open the Configuration function tab and expand the Server\Configurations view.



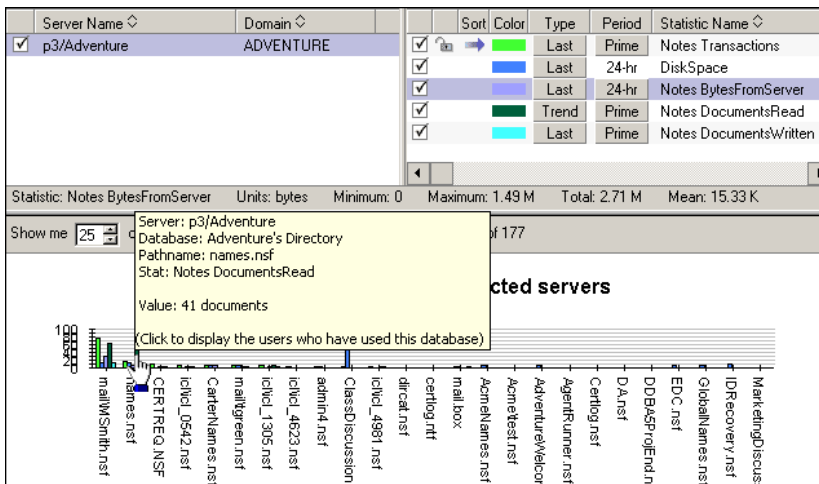
Add a new configuration or open the one that applies to the server. Switch to the Activity Logging tab:



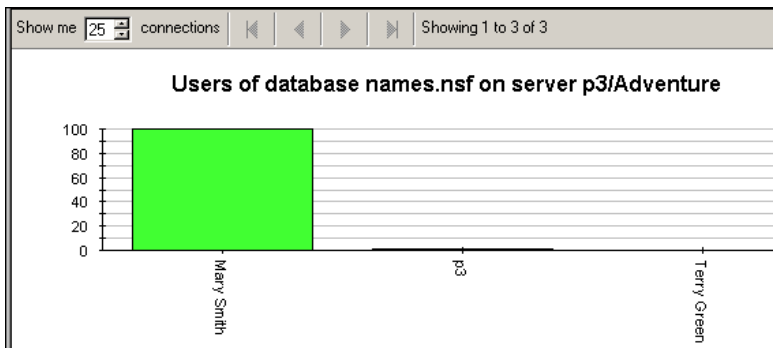
The activity logging data is collected and stored in the Activity Log database (LOG4A.NSF by default). The first collection takes 24 hours. You must use the Activity tool on the Server\Analysis function tab in Domino Administrator to extract activity data from the Activity Log database, shown here:

Organization	Server	User	Database	Timestamp	Bytes Sent
Adventure					7930000
	Rock/Adventure				7930000
		Mary Smith/Adventure			7916860
			activity.nsf		12834
				03/08/2002	22834
					1146
					156
					10786
					156
			admin4.nsf		590
				03/08/2002	226
					3532
					820
					874

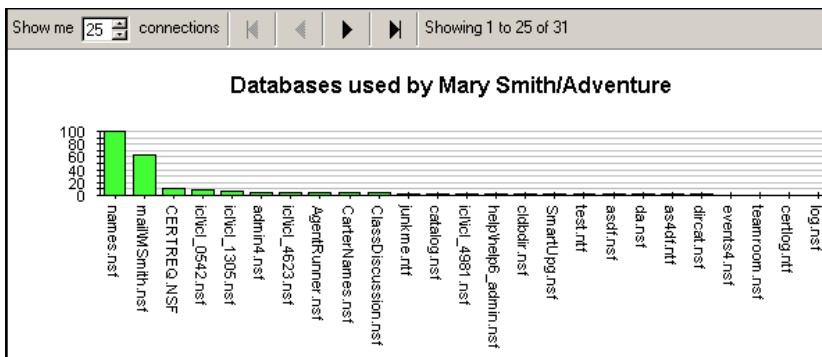
For even greater flexibility and power to understand user activity, you can license the Tivoli Advanced Monitoring add-in. Advanced Monitoring collects and stores server, database, users, and database connections statistics as historical trends using data from the Domino Server Log and the Activity Log databases. Most charts displayed by Advanced Monitoring allow you to drill down to the data behind the graph. For example, while looking at a chart of the statistic DocumentsRead, hovering over a database provides help:



The detail behind the chart shows that Mary Smith is the most frequent user of the `names.nsf` database:



Hover over Mary Smith to see what data is behind this bar. In this case, clicking the bar displays all databases used by Mary Smith:



Advantages of coding your own user activity logging

While the various types of user activity logging built into Domino is very helpful to administrators for trend analysis and resource balancing of servers, it is not particularly useful for individual database managers or for compliance officers because the:

- User activity is not collected in real time; it is all historical information.
- The Documents Read counter is not specific as to *which* documents have been read.
- There is no application-specific use for the activity logging records in the Domino Server Log or Activity Log databases.

But you can easily add a database-specific user activity tracking feature to your Notes applications that records when users first and last accessed the database. This tracking information is stored in the database as a `UserActivity` document (one for each user), and from the `UserVisits` view, you can quickly tell who is currently using the database.

Beyond tracking who opens the database, you can also use the `UserActivity` document to:

- Store display preferences, field defaults, or as we demonstrate, a list of document unique IDs (UNIDs) users have opened for reading (compliance officers will like this feature)
- Compare the total number of active users with some predetermined number, such as to enforce a license agreement or to cap the number of users in a database

We'll also show you how to maintain an audit trail of who has edited documents.

Install the example database

The code shown in this article is from the *User Activity Tracking* database (`useractivitytracking.nsf`), which you can download from the Sandbox. To get the application ready for demonstration, your server administrator must perform these steps:

- Copy the database to the data directory on a non-production server.
- Assign you Designer level access or greater and assign the [DBAdmin] role to you.
- Assign the server (and the group of servers) Manager access and assign the [DBAdmin] role.
- Assign the server administrator (or the group of administrators) Manager access.
- Sign the database elements with a Notes ID file from your organization (use the Files tab in Domino Administrator).

Database open tracking in practice

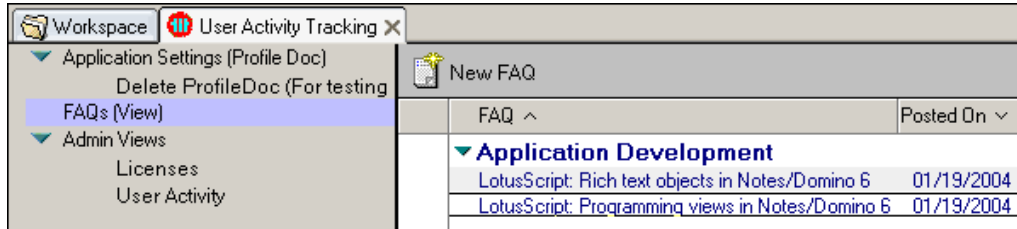
Before explaining the code behind the *User Activity Tracking* application, let's see how user activity is tracked. It is assumed that the database has been installed on the server as described above.

1. Open the database using Notes 6.x.

Note: While reading this article, you will likely have both Notes 6.x and Domino Designer 6.x open to the example database at the same time. Be aware that if you have the database open in Designer, the Database Postopen event won't run in Notes when you still have the database open in Designer. To get the Postopen event to run, simply close all the design elements in Designer before testing the application in Notes.

2. You are warned that this is a demonstration version of the application. By the way, this warning only displays for users in the [DBAdmin] role. Click OK to continue.

3. If you get a version of the database with the profile document deleted, you will be taken to the Application Settings form. For now, just click the Save & Close action button to save the profile document with the default application settings.
4. A page with an embedded outline control is in the left frame and a page with an embedded view element is on the right:



The Outline control entries all have Hide When formulas. For example, who can edit the profile document is limited to one or more "database administrators," defined in the ACL as members of the "[DBAdmin]" role. Because you are in the [DBAdmin] role, you can also see the administration views.

5. Click Application Settings entry in the outline to open the Application Settings profile document. We define all the options later. For now, enter this Registration Code (case sensitive):

wareSource

This removes the demo version dialog box. (We show you how this works later.)

Click the Save & Close action button to save the other default settings.

6. Close the database and reopen it. Now you are warned that the number of users has exceeded your number of valid licenses. Again, this only opens for users in the [DBAdmin] role. Click OK to continue.
7. Click Licenses under the Admin Views. Click the Add License Pack action button. Add five licenses and click the Save & Close action button.
8. Close the database and reopen it. Because you have entered the valid Registration Code and added enough licenses, no dialog boxes display.

9. Click User Activity under the Admin Views. What's this? There is a document with your name on it! Open the document to display the information recorded when you opened the database:

User Activity	Bob Barker/wareSource
First Visit:	01/28/2004 09:00 AM
Most Recent Visit:	01/28/2004 09:03 AM on PRD/wareSource
Total Visits:	2
Notes Version:	Release 6.5 September 26, 2003
Notes Build Version:	194
Platform:	Windows/32

Your first and last visits, total number of visits, and information about your version of Notes is recorded. When done, close the document.

Local database uniform access test

You installed the database on a Domino server for a reason. If you install the database locally from the download, roles aren't enforced, and the whole security model of the application fails. This is not good.

Roles work, however, if you open the Access Control List dialog box and set the "Enforce a consistent Access Control List across all replicas" option on the Advanced tab.

To make sure this property is set, we inserted code in the Database Postopen event to check for this property and to alert the user. To see this code, open the database in Domino Designer 6.x. Expand Other in the object tree and click Database Resources. Open the Database Resources to the Postopen event; this event fires after a user opens the database.

```
15 Dim acl As NotesACL
16 Set acl = db.ACL
17 If ( db.Server = "" ) And Not acl.UniformAccess Then
18 msg1= "This database is running locally, and can NOT run properly
    unless the ACL has uniform access enabled."
19 msg2 = "Please see your Domino Administrator about correcting this
    problem."
20 Messagebox msg1 & Chr(10) & Chr(10) & msg2 , MB_ICONSTOP, "Access Control
    List Error"
21 End If
```

The application runs fine as a local replica copy if you make sure the "Enforce a consistent Access Control List across all replicas" property is set first before you create a local replica from the first one on the server.

Database open tracking code

The first really interesting part of the Postopen event code starts on Line 38. The GetActivityDoc function is called (from the UserActivity LotusScript Library) to find the current user's UserActivity document from the UserVisits view and return a NotesDocument object.

```
38 Set ActivityDoc = GetActivityDoc(s, db, s.UserName)
```

If a UserActivity document matching the user's name cannot be found, a new document is created, and in any case, the ActivityDoc object reference variable is set to point to a valid UserActivity document that can be updated in the following lines with the latest user activity data:

```
39 ActivityDoc.TotalVisits = ActivityDoc.TotalVisits(0) + 1
40 activitydoc.LastVisit = Now
41   If ( db.Server = "" ) Then
42       svr = "Local"
43   Else
44       svr = db.Server
45   End If
46   activitydoc.LastServer = svr
47   activitydoc.NotesVersion = s.NotesVersion
48   activitydoc.NotesBuildVersion = s.NotesBuildVersion
49   activitydoc.Platform = s.Platform
50 Call ActivityDoc.Save(True, False, True)
```

Don't worry at this point if the user has gone into different replica copies of the database on different servers or even a local replica. There is a cleanup routine that merges replication conflicts and duplicate documents, retaining the earliest first visit time/date and the latest last visit time/date. We show you the routine later in the article.

Database license enforcement code

Picking up with the Postopen event code, the next task is the license checking that you saw when you first opened the application. Earlier in the code, we determined whether or not the user is in the [DBAdmin] role. This is easier to do with the formula language and Evaluate than it is with the equivalent LotusScript.

```
25 dbadmin = Evaluate(|@IsMember("[DBAdmin]"; @UserRoles)|)
```

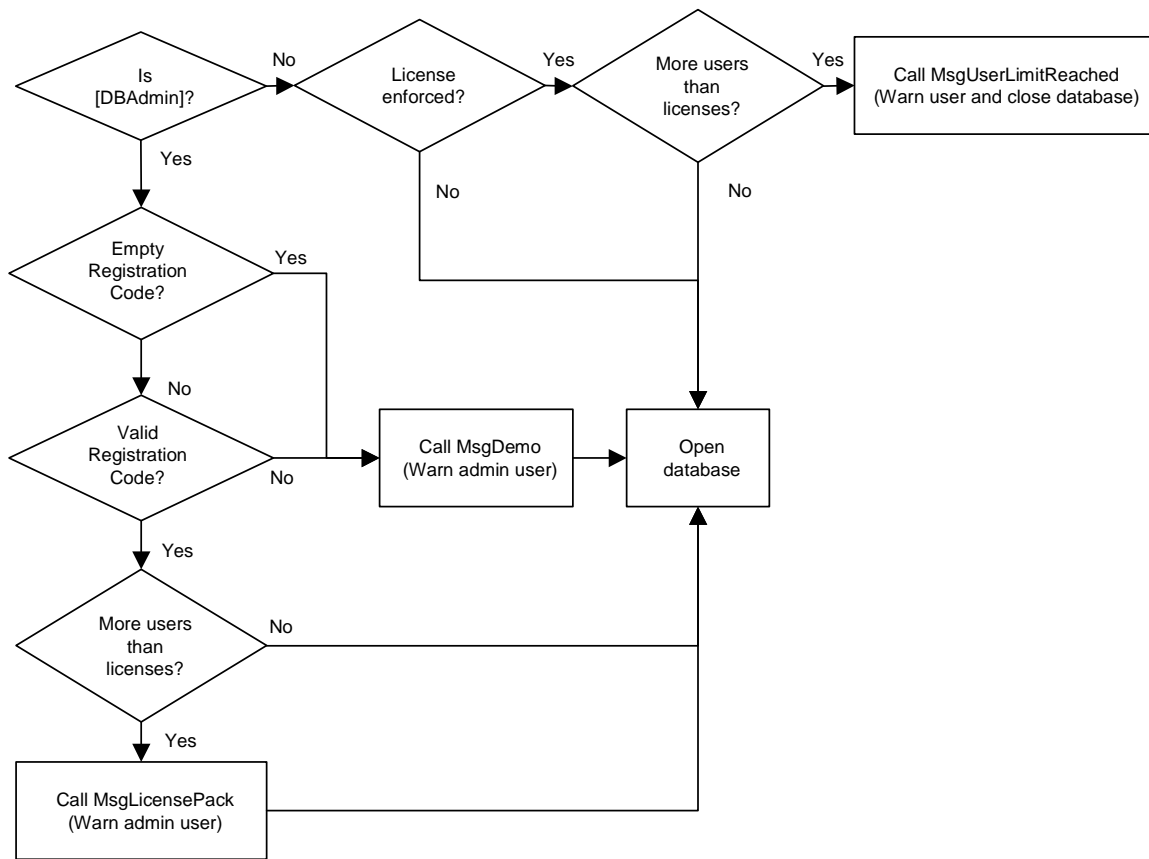
It makes sense for only the DBAdmins to see the warning dialog boxes concerning the Registration Code and license problems. In spite of the warnings, the database still opens. The exception to this pattern is only for users and only if you have elected to enforce the license limit in the Application Settings:

Application Settings		Last Modifier: Kent Kurchak/wareSource Last Modified On: 03:11 PM Today
Application Title:	<input type="text" value="User Activity Tracking"/>	
Database Open Document Read Document Edit		
Registration Code:	<input type="text" value="wareSource"/>	
Current License Packs:	0	
Active Users:	1	
Compliant?	No	
Enforce License limit:	<input checked="" type="radio"/> Yes <input type="radio"/> No	
<input type="button" value="Activate DailyUserActivity Agent"/>		The DailyUserActivity agent runs daily and is responsible for collecting user activity that occurs over multiple server or local replicas into a single User Activity document. This agent cleans up the replication conflicts seen in the User Activity view.

In this case, users are shut out of the application if the number of active users exceeds the number of valid licenses. DBAdmins can always open the database, albeit with warnings if missing a valid Registration Code or adequate licenses.

Warning: Don't be tempted to change the code so that the database is closed when DBAdmins open the database and the Registration Code is invalid or the number of users exceeds the number of licenses! This locks out the database such that the DBAdmin cannot enter a valid Registration Code or add License Packs (unless the code is changed back using Domino Designer).

Here is the logic of Lines 57 through 80 in the Database Postopen event that performs various license-related checks:



There are a couple of things worth pointing out in the code that performs the license checking:

- We start with the test to see if the user is in the [DBAdmin] role by testing the false condition: `If dbadmin(0) = False Then`. This may seem strange, but this is a workaround for a key difference between True in LotusScript (-1) and @True in the formula language (1). If you test for True in LotusScript based on @True generated from the formula language, your test won't work properly. Because False (LotusScript) and @False (Formula language) are 0 in both languages, you can reliably test for the false condition. In this example, we set the value of dbadmin with `dbadmin = Evaluate(|@IsMember("[DBAdmin]"; @UserRoles)|)`. If the user is in fact in the [DBAdmin] role, @True is returned. Since the test is performed in LotusScript, `dbadmin(0) = True Then` won't ever come true. Hence the need to test for `dbadmin(0) = False`.
- If any of the license conditions are not met, the code jumps out to one of three subs: Sub MsgDemo, Sub MsgLicensePack, or Sub MsgUserLimitReached(Source As NotesUIDatabase).

- You may notice the calls to the ActiveUsers and LicenseCount functions, for example, in Line 59: `If ActiveUsers(db) > LicenseCount(db) Then Goto UserLimitReached`. These two functions can also be found in the UserActivity LotusScript Library. They go to the respective views (using NotesViewEntryCollection and NotesViewEntry) to determine the counts.

Database Open tab

Let's go back and take a look at how to configure the application. Click the Application Settings outline entry to open the ProfileDoc form. The Database Open tab of the ProfileDoc form configures what happens when a user opens the database.

Application Settings	Last Modifier: Kent Kurchak/wareSource Last Modified On: 03:11 PM Today
Application Title: <input type="text" value="User Activity Tracking"/>	
Database Open Document Read Document Edit	
Registration Code: <input type="text" value="wareSource"/>	
Current License Packs: 0	
Active Users: 1	
Compliant? No	
Enforce License limit: <input checked="" type="radio"/> Yes <input type="radio"/> No	
<input type="button" value="Activate DailyUserActivity Agent"/>	
The DailyUserActivity agent runs daily and is responsible for collecting user activity that occurs over multiple server or local replicas into a single User Activity document. This agent cleans up the replication conflicts seen in the User Activity view.	

The Application Title field is used to feed the title for the HomeFrameset frameset:

Frameset	
HTML	
Name	<input type="text" value="HomeFrameset"/>
Alias	<input type="text" value="HomeFrameset"/>
Comment	<input type="text"/>
<input type="checkbox"/> Available to public access users	
Title	Formula Window
<pre>@ReplaceSubstring(@GetProfileField("ProfileDoc"; "AppTitle"); @NewLine; " ")</pre>	

The Registration Code field requires a password that matches the hashed value of a "secret" password (in this case "wareSource") hard coded in the Database Postopen event code:

```

70 makePW = Evaluate(|@Password(" | & getReg & |")| )
71 pw= "(02319F0AEE6508C2D9C03FF4F531E5DF)"
72 If pw<>makePW(0) Then
73     Call MsgDemo()
74     Exit Sub
75 End If

```

How did we create the hashed value? This is an old Notes trick. Enter the following formula in any text field (for example, the SendTo field of a Memo):

```
@password("wareSource")
```

Then press Shift+F9. Notes interprets the formula and returns(02319F0AEE6508C2D9C03FF4F531E5DF). Copy the result into the code.

There are more secure ways to ensure license compliance, but we are only interested in a reasonable effort. To beef this up a little (without major architectural work), we could create a database template from the database and then create a production version of the database (the one we give to customers). Then replace the design of the production version of the database, choosing the Hide formulas and LotusScript option to hide the code. This hides the code, and therefore, the ability to change the password.

By the way, the Current License Packs, Active Users, and Compliant fields are simply informational. If the number of active users exceeds the number of licenses, the Compliant field changes to No.

UserVisit conflicts

Before we talk about enabling the DailyUserActivity agent with the Activate DailyUserActivity Agent button, we must tell you what happens when a user has opened different replica copies of the database on different servers or even a local replica. Each replica of the application records and updates its own UserActivity documents. Following replication, the impact to the UserVisits view is unfortunate, but unavoidable given the distributed nature of the Notes/Domino environment:

Merge Visits					
User v	Visitor	First Visit	Last Visit v	Total Visits	
Bob Barker/wareSource	1	01/25/2004	01/26/2004	5	
Kent Kurchak/wareSource	2	01/27/2004	01/27/2004	8	
Kent Kurchak/wareSource	3	01/27/2004	01/27/2004	1	
◆	[Replication or Save Conflict]				
◆	[Replication or Save Conflict]				
				21	

Duplicate documents and replication conflict documents abound. If the number of active users is pushing up against the number of licenses, the needless duplicates could easily push you over the limit and restrict users from the database. There are three ways to resolve this problem:

- Manually delete the duplicate documents. This is the least attractive option for any busy database manager!
- Enable the DailyUserActivity agent, which calls a cleanup routine to merge conflict documents and to retain the earliest first visit time/date and the latest last visit time/date. This routine, ProcessActivityView, is also in the UserActivity LotusScript Library. As the routine iterates the NotesViewEntryCollection of the UserVisits view, it calls the MergeConflicts sub when it finds that the user name of the current ViewEntry is the same as the next one.

MergeConflicts is actually responsible for meshing the two documents into one so that the two time/date values are the earliest and latest and the number of total visits is the higher of the two. When the agent finishes, the duplicates are removed and order is restored to the UserVisits view:

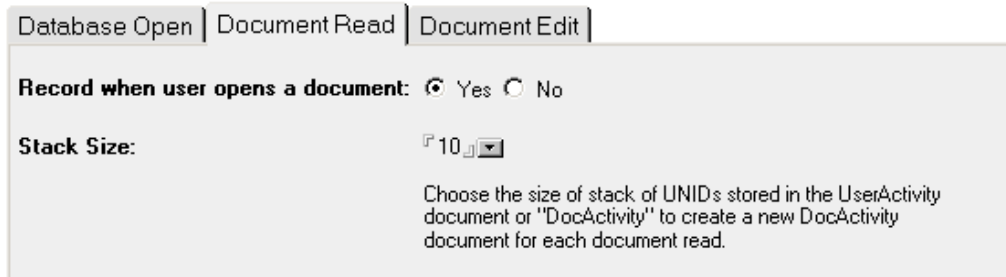
Merge Visits					
	User ▾	Visitor	First Visit	Last Visit ▾	Total Visits
	Bob Barker/wareSource	1	01/25/2004	01/26/2004	5
	Kent Kurchak/wareSource	2	01/27/2004	01/27/2004	8
					<u>13</u>

You can, of course, change the schedule of the agent from the Agent list so that it runs more or less frequently, depending on your needs.

- Open the UserVisits view and click the Merge Visits action button. This calls the same routine that the DailyUserActivity agent calls. You can use this button when testing the application across replica databases.

Document Read tab

The Document Read tab of the ProfileDoc form enables the application to record the UNID of every document opened by the user.



Database Open | Document Read | Document Edit

Record when user opens a document: Yes No

Stack Size:

Choose the size of stack of UNIDs stored in the UserActivity document or "DocActivity" to create a new DocActivity document for each document read.

The nice thing about already having a UserActivity document for every user means that you now have an application-generated place to store all sorts of user-specific data. And that is exactly where we want to store the list of UNIDs of documents the user has opened.

The Stack Size field lets you store up to 500 UNIDs in the UserActivity document. With a little bit of testing, it seems like the multi-value field that holds the UNIDs can store something like 900 or so entries, but we capped it at 500. The list of UNIDs is kept at the stack size in FIFO order. (There is another Stack Size option, DocActivity, which we talk about later.)

By the way, using documents such as a UserActivity document to store user-specific data often works much better than creating a profile document for each user. This is because you have granular control over how the conflicts generated from multiple replica databases are merged; profile document conflicts use total data note replacement (newest profile document wins).

To track which documents users open, add the DocumentHistory subform to the forms of documents you want to monitor. This subform contains the code in the Postopen event to record the user opening the document:

```
12 If profiledoc.EnableRecordDocOpen(0) = "1" Then
13   Print "Recording opened document in " & ProfileDoc.TrimAppTitle(0)
14   Call FindDocID(s, db, Source.Document, profiledoc)
15   Print ""
16 End If
```

The tracking work is done by the FindDocID sub in the UserActivity LotusScript Library. This sub also calls the GetActivityDoc sub that you saw above. But instead of updating the database-related tracking information, it appends the current document UNID to a multi-value list of UNIDs in the OpenedDocs field, for example:

User Activity	Bob Barker/wareSource
First Visit:	01/28/2004 09:00 AM
Most Recent Visit:	01/28/2004 09:03 AM on PRD/wareSource
Total Visits:	2
Notes Version:	Release 6.5 September 26, 2003
Notes Build Version:	194
Platform:	Windows/32
Opened Documents:	7F906602718F00E685256E2500769F58 49862418EE0EECA85256E2500769F57

When the UNID is stored in the UserActivity document, you only track the UNID of the documents users open, remove any duplicates from the list, and then reduce the number of UNIDs to match the stack size.

Caution: Be sure of your legal standing to track the documents users open for reading. In some places, this is an illegal infringement of privacy; in others, it is legal as long as you inform users you doing so. Even if legal, it may be against company policy to record this information. But in today's security and compliance-oriented environment, this may be exactly the kind of information you need to track. To be even stealthier about the tracking, comment out Line 13 in the subform Postopen event code so that users do not know they are being monitored. (We talk later about how to prevent users from reading their own UserActivity document.)


Track specific documents

If you only want to track when *particular* documents are open (instead of all documents), copy the EnableRecordDocOpen field from the ProfileDoc form to the DocumentHistory subform and rename it to something like EnableRecordThisDoc. Then change the following line in the Postopen event to test for that field on the current document, for example:

```
12 If profiledoc.EnableRecordDocOpen(0)="1" And  
    Source.Document.EnableRecordThisDoc(0) = "1" Then
```

DocActivity Stack Size option

When you select DocActivity in the Stack Size field on the Document Read tab in the profile document, no document UNIDs are stored in the UserActivity document. Instead, a DocActivity document is created every time a user opens an InfoDoc document:

Document Activity	Kent Kurchak/wareSource 
Document: LotusScript: Programming views in Notes/Domino 6	
Opened: 01/30/2004 02:55 PM on PRQ/wareSource	

The advantages of this method of tracking which documents are opened are:

- There is no limit as to the number of UNIDs you can store.
- You can track *when* users open a document as well as the number of times they open it.
- It is easier to create views that total the number of times a document has been read by all users.

By the way, the hotspot to the right of the user name opens the user's UserActivity document.

How to display who has read documents

We didn't put much work into what to do with the tracking data after we collected it. But the data is there to use in a way that makes sense to you and that fits your requirements. To demonstrate what is possible, we designed two simple views to give you some ideas of how to display the tracking data:

- *Opened Docs.* This view selects InfoDoc, UserActivity, and DocActivity documents, sorting them by document UNIDs so that you can see the readers for every document.

FAQ	Readers
LotusScript: Rich text objects in Notes/Domino 6	Bob Barker/wareSource Kent Kurchak/wareSource
LotusScript: Programming views in Notes/Domino 6	Bob Barker/wareSource

The bolded Subject column is actually from the InfoDoc document, whereas the Readers column is from either the UserActivity or DocActivity documents.

- Doc Activity selects just the DocActivity documents.

Reader ^	Date	Total Visits
LotusScript: Rich text objects in Notes/Domino 6		2
Kent Kurchak/wareSource	01/30/2004 02:57 PM	
Bob Barker/wareSource	01/30/2004 02:58 PM	
LotusScript: Programming views in Notes/Domino 6		1
Bob Barker/wareSource	01/30/2004 02:58 PM	
		3

We like this view a little better than the OpenedDocs view because it totals the number of times people have read the documents and also displays multiple times a user opens a document.

Caution: We didn't code for the possibility that an InfoDoc document for which there are DocActivity documents will be deleted. Even if a document is deleted, you may still have a reason to know that someone read it. But if you change your mind, it is a simple matter to create a cleanup agent that loops through the DocActivity documents, reads the DocUNID field, and attempts to find the corresponding InfoDoc document. If not found, the agent would delete the DocActivity document.

There are two things to point out about the Opened Docs and Doc Activity views:

- You won't see their entries in the Outline control until you enable document open tracking on the Document Read tab in the profile document and then close/reopen the database.
- The two views are restricted to users in [DBAdmin] role so users not in this role won't be able to see who is opening documents.

Other uses for tracking data

What you want to do with the information will determine how you further code this application to display who has read documents. We can envision an agent that builds a report of documents that have been read, in order of total number of readers, listing the reader names and last read dates. We can also envision an agent configured to scan for a specific document having been read, sending the report of who read the document.

If we were to code this, we would set it up like Interest Profiles in the Discussion template such that (a restricted group of) users could set a "watch" on a document, and then periodically the agent would loop through the watch documents and then with the document UNID in each watch document, loop through the UserActivity documents looking for a match to find readers.

The type of reporting you probably want to do would be much easier if Notes databases were relational, but that thought will have to wait for a future version of Notes. In the meantime, you may have good results with Crystal Reports or even Lotus Approach to join the FAQs and UserVisits views. With Lotus Approach, use the Notes Normalizer to make pseudo-records from the multi-value OpenedDocs field in the UserActivity form and then join the tables before creating reports.

If you use the DocActivity stack size option, you have more flexibility in how you work with the tracking data because the DocActivity documents store the single UNID of the InfoDoc document (we would dare say this is the foreign key). It would be very easy to display the document readers right in the InfoDoc document:

- Embed a single category view (categorize the view on the DocUNID field stored in the DocActivity document)
- Add a Computed for display field using @DBLookup in the InfoDoc form and list all the document readers.

Be sure to hide the list of readers from non-DBAdmins!

Security and privacy of *UserActivity* documents

While you may want to read all database and document tracking information, you may not want users to read each other's tracking information stored in the *UserActivity* documents. Even though users are updating *UserActivity* documents via backend LotusScript methods, they still need to have Author access in the database ACL (and have the ability to create new documents). So if you do nothing else, everyone can read everyone's *UserActivity* documents.

To prevent this from happening, we set a Readers field to the user's own name and to the [DBAdmin] role so only these two names can read the document. We do this when a new *UserActivity* document is created using the GetActivityDoc function (from the *UserActivity* LotusScript Library):

```
20 Dim ReaderArray (1) As String
21 ReaderArray(0) = getUser
22 ReaderArray(1) = "[DBAdmin]"
23 Dim item1 As New NotesItem( doc, "Owner", ReaderArray, AUTHORS )
24 Dim item2 As New NotesItem( doc, "Readers", ReaderArray, READERS )
```

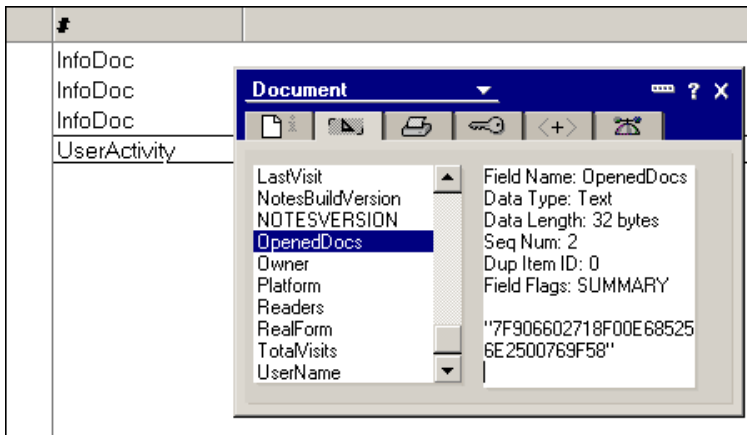
We also set the Owner field to the same value so that the user can update the backend document. But we want to take this security/privacy measure a few steps further. We don't even want users to be able to read their own *UserActivity* documents!

Already you have seen that the Outline control to open the UserVisits view is hidden from everyone except DBAdmins. But this doesn't prevent users from manually opening the UserVisits view, using the Shift+Ctrl keys. To prevent this from happening, we added some Queryopen event code to warn the users when they open the UserVisits view and dump them out by setting Continue = False:

```
2 dbadmin = Evaluate(|@IsMember("[DBAdmin]"; @UserRoles)|)
3 If dbadmin(0) = False Then
4   MsgBox "You can NOT open the " & Source.view.name & " view.
   Only [DBAdmin] users can open this view.", MB_ICONSTOP, "Open Canceled"
5   Continue = False
6   Exit Sub
7 End If
```

This works well, but smart users can get around this pseudo-security device by creating a private view. Although they will only be able to see their own *UserActivity* document in the private view (because of the Readers field), we don't want them to open their document. To stop this, we put the same Queryopen code in the Postopen event of the DocumentHistory subform that we put into the UserVisits view.

Of course, the savviest users can open the Document properties from their private view to see the item values, including the record of opened document UNIDs:



If this ability to read one's own data is still a problem, you can introduce these final security countermeasures:

- Prevent any access by the user to the UserActivity document by coding a two-phase document update scheme. Rather than updating the UserActivity document directly from the Postopen event of the DocumentHistory subform (which requires the user to have Author access to his own UserActivity document), you could instead spin off another type of (temporary) document. The temporary document goes into a queue to be processed by an agent. The agent updates the UserActivity document (instead of the user) and then deletes the temporary document. This adds too much complexity to the application for only a small benefit and has the added downside of filling the database with potentially thousands of document deletion stubs every day (the bane of Domino administrators everywhere).

(In days past, we have coded a tracking scheme that used two databases; when a user opened a document in one database, the tracking document was mailed to the second tracking database for processing and viewing. This is very easy to do because it follows the architecture of the Response.)

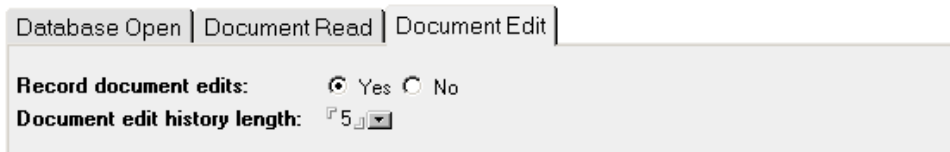
- Add a reserved field named \$KeepPrivate (Text, Computed when composed, set to "1"), which grays out the item names and values in the Document properties Fields tab, so users at least can't *copy* the item values to the clipboard. This also keeps them from copying the document out of the database into another database for inspection. This is a simple measure that can increase the security of the user's UserActivity document.
- Choose the Hide formulas and LotusScript option when replacing the design of the database from its database template. This has the added benefit of hiding the item names and values in the Document properties in the Fields tab. In combination with the other techniques, this measure effectively makes the UserActivity document inaccessible.

Security and privacy of DocActivity documents

When you choose to track document access using the DocActivity documents, the situation changes significantly. Users don't need to update them as they do with UserActivity documents. You can easily change the FindDocID sub code (in the UserActivity LotusScript Library) so that the Owner and Readers fields exclude the user so that all future access by the user is denied.

Document Edit tab

The Document Edit tab of the ProfileDoc form allows you to track who has edited a document and to set the length of the audit trail:



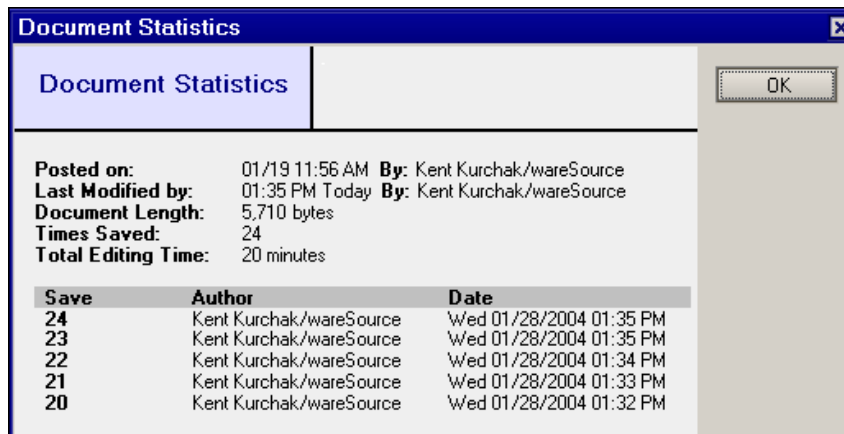
Database Open | Document Read | Document Edit

Record document edits: Yes No

Document edit history length: 5

The same DocumentHistory subform you added to the forms of documents to track document reading also is used to store the edit audit trail. This subform contains the code in the Postopen, Postmodechange, and Querysave events to capture the editing session metrics.

To view the audit trail, open a document in Read mode and click the Statistics action button. This particular document has been edited/saved 24 times for a total of 20 minutes of the document being open in Edit mode. The edit history length was set to five, so only the last five sessions are recorded:



Document Statistics

Document Statistics [OK]

Posted on: 01/19 11:56 AM **By:** Kent Kurchak/wareSource
Last Modified by: 01:35 PM Today **By:** Kent Kurchak/wareSource
Document Length: 5,710 bytes
Times Saved: 24
Total Editing Time: 20 minutes

Save	Author	Date
24	Kent Kurchak/wareSource	Wed 01/28/2004 01:35 PM
23	Kent Kurchak/wareSource	Wed 01/28/2004 01:35 PM
22	Kent Kurchak/wareSource	Wed 01/28/2004 01:34 PM
21	Kent Kurchak/wareSource	Wed 01/28/2004 01:33 PM
20	Kent Kurchak/wareSource	Wed 01/28/2004 01:32 PM

Summary of design elements

To use these features in your application, copy these design elements from the example database to your database:

- **Forms:** LicensePack, ProfileDoc, UserActivity, DocActivity
- **Subforms:** DocumentHistory (add to your content forms), DocStats
- **Views:** License, UserVisits, OpenedDocs (optional, needs to be changed to select and display your content documents).
- **Agent:** DailyUserActivity
- **Script Library:** UserActivity

You also need to manually copy code from the Database script Options and Postopen events to those same events in your database.

Finally, you also need to create your own replacement for the HomeFrameset frameset and HomeOutline outline control to give DBAdmins (only) the ability to open the profile document and hidden views.

Note: The InfoDoc form and FAQs views are not needed in your application. They are used here just for demonstration purposes to represent any ordinary user-accessible document and view. By the way, the example application contains a few FAQs with content copied from the [LDD Today](#) database.

Conclusion

This article described an application strategy that provides granular user activity tracking for database open, document read, and document edit events. This type of specific user activity tracking is of greater interest to database developers, managers, and compliance officers than the built-in activity logging performed by Domino. And because you are adding these features to a Notes application using Domino Designer, it takes very little effort to quickly realize significant benefits.

About the author

Kent Kurchak is the owner of [wareSource](#), an IBM Business Partner that provides robust, customizable, and economical training materials and applications for Lotus Notes and Domino users, developers, and administrators. wareSource courseware is used worldwide by IBM Business Partners, training companies, independent instructors/consultants, corporate training departments, and individuals.