# NotesTracker Guide

## Lotus Notes/Domino Document Usage Tracker
## Version 4.4

This guide was last updated on 25 July, 2006

**Asia/Pacific Computer Services**

http://asiapac.com.au

http://notestracker.com

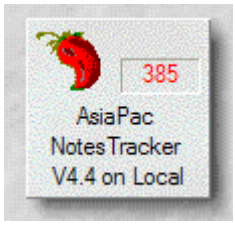
**Technical Support e-Mail**: NotesTrackerSupport@asiapac.com.au

We are very interested in your feedback about this guide.

Please report any technical inaccuracies, usability comments and missing or incomplete information to the above e-mail address.

# Introduction

## *The Purpose of NotesTracker*




The **NotesTracker Database**, herein alternatively referred to as the **NotesTracker Repository**:

- Acts as an **audit trail, or log, of the application activity** for your Notes databases -- a collection of Usage Log entries that are created whenever documents and views in your Lotus Notes databases are accessed. Items such as documents and views in Lotus Notes databases are sometimes referred to as “notes”, hence the name “NotesTracker”.

- Provides, via Notes views, a continuously-updated **analysis of document and view usage patterns** in your databases.

- Acts as a Software Development Kit (SDK) or **NotesTracker design repository.** Licenced users have open access to all NotesTracker source code, enabling their Notes developers to set up usage tracking for some or all of their Notes databases.

## *How NotesTracker Works*

1. The **NotesTracker code** distributed in this database is used by your Notes developer to quickly modify the design of the Notes databases for which you want usage tracking to be performed.

2. A **configuration document** (plus an associated configuration view) is also added to each such database. By editing each individual database’s NotesTracker Configuration document, the assigned NotesTracker Administrator for the database can at any time activate or deactivate usage tracking as a whole for that particular database, and for individual events within the database (Create documents, Read documents, Update them, Delete them, open a view, etc).

3. For each database that has had usage tracking activated at some point of time: whenever one of these events occurs NotesTracker writes a **Usage Log document** to the repository database that you designate. The usage log entries build up over days, weeks and months to give you an easy-to-understand picture of each application database’s document and view usage.

4. Whenever desired, old usage log entries can be archived for historical reporting purposes (in exactly the same easy and convenient way that you archive your old Notes Mail entries).

5. NotesTracker is designed to measure database activity carried out by users interacting with your Notes databases in the foreground, via a Lotus Notes Client or via a Web browser. However your developers should be able to extend NotesTracker's usage logging capabilities to some other environments, such as the tracking of database activities performed by LotusScript agents – but not Java or Formula Language agents -- running in the background (on a Domino server or Notes workstation).

## *How you see Usage Metrics generated by NotesTracker*

At any time thereafter, you can examine the various views in this database to analyse and examine the document and view usage patterns and trends. NotesTracker is distributed with a set of views that present the document and view usage in a number of interesting ways. If the supplied views are not sufficient for your analytic purposes, your Notes developer can easily build additional Notes views. Alternatively, you might use external reporting and charting software to provide the sort of usage analyses that you desire.

There is a brief discussion of this matter at the very end of the guide, under the heading “Extended Analysis and Reporting”.

## *Software Usage Terms and Conditions*

**Our aim:** to provide you with a capable. low-risk solution backed by professional support, while retaining our precious intellectual property rights invested in NotesTracker and receiving an adequate return in order to be here to support you in the long run!

On the following pages is a copy of the NotesTracker terms and conditions of use as at 08 June 2005.

Please note that the terms are varied from time to time and you should always refer to the following web page for the latest version of the terms and conditions, which apply at the time of purchase:

http://www.asiapac.com.au/Pricing/Usage_Tracker_pricing.htm

# SOFTWARE USAGE TERMS AND CONDITIONS

## as at 08 June 2005

**Asia/Pacific Computer Services ("APCS") reserves the right to make modifications to these terms and conditions from time to time. The latest terms and conditions are available on the APCS web site for viewing prior to the downloading and installation of the Software and will take precedence over any other form of the terms and conditions. Such modifications shall be effective immediately upon posting of the modified agreement at this Web site. You agree to review the agreement periodically to be aware of such modifications and your continued use and/or downloading of software available to you hereunder shall indicate your acceptance of the modified agreement.**

### COPYRIGHT AND LICENCE

The "NotesTracker" or "AsiaPac Document Usage Tracker" Software is owned by APCS and is copyrighted and licenced not sold. The term "Software" means the original program and all whole or partial copies of it. A Program consists of machine-readable instructions, its components, data, audio-visual content (such as images, text, recordings, or pictures), and related licenced materials.

The entity that purchases the license ("you") and all its officers, employees, contractors and other associated persons will be subject to this licence agreement.

The **evaluation form** of the Software is provided with hidden design and you shall not modify, translate, disassemble, decompile, or reverse engineer the Software. The **licenced form** of the Software is provided with unhidden design so that its design elements may be incorporated into the designs of your Notes databases either without modification or modified and adapted by you to meet your specific usage metrics requirements. For both forms you may not remove, alter or deface any proprietary notices on the Software.

You acknowledge that APCS owns all right, title and interest in and to the Software or portions thereof, including without limitation all Intellectual Property Rights. "Intellectual Property Rights" means any and all rights existing from time to time under patent law, copyright law, trade secret law, trademark law, unfair competition law, and any and all other proprietary rights, and any and all applications, renewals, extensions and restorations thereof, now or hereafter in force and effect worldwide.

The Software and documentation are proprietary to APCS. By installing and using the Software, you agree to comply with these terms and acknowledge that the Software and documentation contain valuable trade secrets and other proprietary information belonging to APCS. You also agree to not remove, obscure, or alter APCS's copyright notice, trademarks, or other proprietary rights notices affixed to or contained within the Software.

You may copy the Software only for purposes of backup including multiple archive backup copies or incorporation into the design of your Lotus Notes databases provided any and every copy must contain all of the original Software's proprietary notices. You may not distribute (for free or for sale) or sublicence the Software. You agree to hold the Software in confidence and undertake not to pass copies of it whole or in part to another party outside your organisation. If you distribute, rent or sell Lotus Notes applications to other organisations you will not include the Software or any derivative or adaptation of it in these Notes databases until such time that the receiving organisations have purchased the appropriate NotesTracker licence from APCS.

You will ensure that anyone who uses your copy of the Software does so only in compliance with the terms of the licence. Unlicenced evaluation software may not be used for production purposes. You are responsible for communicating the terms of this agreement to your employees and contractors and for ensuring their compliance with the terms of this agreement and any company policies and procedures you might have surrounding use of the Software. You must report to us, as soon as possible after you notice it, any suspected misuse arising from your possession of the Software including counterfeiting, piracy, disclosure to non-licencees or other copyright infringement in the Software.

When you purchase the Software a Licence Document will be sent to you and this will contain a Licence Number applicable to the Version of the Software as of the date of purchase, being the date that

payment funds are received by us. You may use the Software for evaluation purposes but not in production until the licence has been issued. The licence will apply to the latest Version of the Software at the time of purchase and to any sub-releases of that Version but not to any subsequent Version if an Upgrade Price applies to the later Version. Your license is void without registration of required information or registration with incorrect information.

The Software is priced on a tiered basis according to the number of Lotus Domino servers or standalone Lotus Notes Client workstations upon which the Software is installed in one or more Lotus Notes databases. If you purchase the Software at one tier level and later increase the number of installations to a higher tier level then you agree to immediately notify us and forward to us the incremental purchase price between the tier levels.

The Software is licenced as stated above. The licence does not constitute ownership of the Software, only the right to use it. Licences are not transferable.

If you do not agree to or cannot comply with any of the terms and conditions, do not attempt to access or use the Software. By installing the Software you agree to these terms and conditions. If you decide not to purchase the Software, or on expiry of a lease to the Software, you agree to destroy all copies including backup copies. APCS may terminate your licence if you fail to comply with the licensing terms and conditions. You agree that you will not continue to use and will delete the design and code of the Software or any derivative or adaptation of it in any of your Lotus Notes databases or elsewhere as soon as practicable after the licence expires.

**LIMITED WARRANTY**

This licence entitles the user to hold the source code of the Software in escrow and to run the Software, but not to disclose or sell it to other parties either in its entirety or any individual components of its design and code.

The cost or guarantee of support is not included in the licence. APCS will make reasonable effort to fix problems reported, and enhance functionality where requested. There will be no charge for fixing reported problems but APCS reserves the right to charge for enhancements and new versions or maintenance of the Software when such problems are caused by your modification of the delivered design and/or code of the Software. You are free to modify the Software and adapt it as you see fit for incorporation into your organisation's own Lotus Notes databases but any such modifications or adaptations and the effects pursuant to them are not supported by APCS under the warranty.

Subject to any statutory warranties which cannot be excluded, APCS shall not be liable for material, equipment, data or time loss, caused directly or indirectly by proper or improper use of the Software or for the effects of any modifications that you make to the Software. In cases of loss, destruction, or corruption of data, APCS shall not be liable. APCS does not take any other responsibility. APCS makes no other warranty.

# Overview of NotesTracker

The distributed NotesTracker Database (or "NotesTracker Repository") acts as:

1. The **NotesTracker Toolkit** or SDK (Software Development Kit), a container for NotesTracker design elements – forms, views, subforms, agents, etc -- that you build into the designs your Notes application databases.

2. A **repository for NotesTracker entries (or "usage log" entries)** that are generated during logging of document-related activities as users interact with one or more of your Notes/Domino applications.

**Usage tracking can be implemented for any Notes database, as long as you have designer access to it.** Naturally this means that the database designs cannot be hidden, as are the designs of some third-party Notes applications (which will prevent you from implementing usage tracking for them). Only simple designer skills are needed, unless you want to make any changes to the supplied NotesTracker code in which case a greater or lesser degree of familiarity with LotusScript is required.

In practice you it would be of little value to implement usage tracking in each and every one of your Notes databases, but only in those of them where the ability to track usage delivers an **adequate return on investment** or provides some **worthwhile operational payback** (typically, control and privacy reasons such as the monitoring of updates to critical fields and the tracking of document deletions). See http://asiapac.com.au/UsageMetrics.htm for some ideas about why and how you might use NotesTracker.

## *Usage Tracking for the Lotus Notes Client*

Usage Log documents can be added to the NotesTracker database every time that one of the following events occurs in any of your tracked databases:

- **A DOCUMENT IS READ** (opened in Read mode -- but never changed to edit mode and saved).

- **A DOCUMENT IS UPDATED** (opened in edit mode or opened in Read mode then changed to Edit mode, then saved).

  In most cases you will want to know more than merely that a given document was updated. NotesTracker offers **a generic, all-purpose FIELD AUDIT TRAIL capability**. When you switch on tracking of Updates, you can then also optionally switch on the tracking of **CHANGES TO THE CONTENTS OF THE DATA FIELDS IN DOCUMENTS**. When this option is activated for a database, the field "**before images**" (field contents before the Update) and "**after images**" (field contents after the Update) are placed into the Usage Log document for each document Update. When you open a Usage Log document, you'll be easily able to compare the before images with the after images, in both a "Side by Side" arrangement and an "Over and Under" arrangement, illustrated later in this guide under the heading "Tracking Changed Field Contents".

  As a refinement, you can opt to **log all fields** (both changed and unchanged) or to **log only the changed fields**. Since there's little use in logging the unchanged fields, the latter option eliminates "clutter" in the Usage Log and makes it far easier to hone in on the changed fields. This is especially true when there are many fields in a document, such as in Server Documents stored in the Domino Directory (Public Address Book) database.

- **A DOCUMENT IS CREATED** (composed - created as a new document and then saved for the first time).

- **A DOCUMENT IS DELETED** (removed permanently from the database -- more precisely, what the Lotus documentation refers to "marked for deletion", see discussion below).

- **A VIEW IS OPENED.**

## *Usage Tracking for Web Browsers*

A major feature was added to NotesTracker for Version 4, namely:

- **Tracking of WEB BROWSER accesses.**

  With a few simple additions to a database's design, NotesTracker enables the creation, updating and reading of Notes documents via a web browser to be logged in the same fashion as was done for the Lotus Notes Client in previous versions of NotesTracker.

  **This provides you with a different, more incisive way to track and analyse your Domino (web-based) document activity than is provided by other Domino web tracking products.** These others all rely on what is written to the DomLog.NSF database for their statistics, and they can't offer the same sort of detail that NotesTracker does – such as comparing before/after contents of all fields in a document, to name one.

  NotesTracker has a more **application-centric** approach, cantering on "CRUD" – document Creates, Reads, Updates (including field content changes), and Deletes.

## *About database accesses via a Notes Client*

When you create or edit a document via a Lotus Notes Client, a usage log entry is written only once -- when the document is closed -- regardless of how many times the document is saved during the editing process.

This is an intentional feature of NotesTracker, designed to report only the **net result** of the editing process. (If you really wanted to, it would be a simple matter for you to alter the NotesTracker routines to capture what is changed for each and every Save.) Another major benefit of this approach is the significant reduction in the number of Usage Log entries written. This not only conserves disk space (plus processor cycles and network traffic) but also facilitates your metrics analysis by eliminating the clutter that logging of multiple Saves would cause.

The field "before images" (field contents prior to change) are truly those that the user saw when she/he opened the document for editing.

## *About database accesses via a Web browser*

**F**or NotesTracker Version 4.0, only the "before images" (field contents prior to update) were logged when documents were updated via a web browser.

The nature of the HTTP protocol is for web pages to be sent out by the HTTP server (Lotus Domino, or any other), via a POST operation. This is "set and forget" or "stateless" style of operation. The server may receive the page back from the browser within a second or two of the POST, within some short or long period of minutes or hours, or may never receive the page back at all. Only when the user clicks a SUBMIT button in the browser page does the web server (via a GET operation) obtain incoming field contents. The HTTP protocol has no mechanism that automatically relates the fields in the page that was sent out by the server to the fields in the page that was just returned to the server. This means that there's no easy way to compare the page's field contents before and after they are updated.

With NotesTracker Version 4.0 it was decided not to attempt some sort of complex field change tracking solution, such as setting browser "cookies" to temporarily store the page's initial field values so that they could be compared with updated field values. Even if such a method was implemented, an individual browser user can disallow use of cookies, preventing such a scheme from working for that user anyhow!

However, in NotesTracker Version 4.1 a different approach was adopted. At the time that the browser page is submitted back to the Domino server, a copy of the so-called "back-end" document is retrieved from the database on the Domino server and the fields from this freshly-retrieved document are used as the "before images".

**Note:** It is important to be aware that the logged contents of such fields may possibly not be the same as the contents of the fields that initially were sent out to the browser page. There is always the chance – perhaps only slight -- that some other user(s) might have updated the back-end document in the period between the POST and GET operations. It's hard to come up with a foolproof solution for this issue, which in essence is caused by the "statelessness" of web browser sessions. (In terms of field content changes, this is analogous to the generation of Save Conflicts, caused when multiple users simultaneously update a document via a Notes Client.)

## The Need to "Sign" the NotesTracker Web Agents

Refer to the Administrator Topics section below for more details, but it should be pointed out at this early stage that, for security reasons, in the Domino server environment web agents need to be appropriately "signed". If the NotesTracker web agents are not signed so as to be acceptable for your Domino server then they will fail to execute, which means that NotesTracker will not log any Web browser interactions. (This is a normal Domino security issue and not a NotesTracker limitation.)

## *Tracking actions carried out other than via a Notes Client or a Web Browser*

As distributed, NotesTracker tracks actions performed on Notes documents via the so-called "front end" (the graphical user interface provided by a Lotus Notes Client or a Web browser). However, the NotesTracker routines are structured in a modular fashion that makes it easy to adapt them for tracking such actions when they are carried out in agents running in the background.

This could be important in some databases, for completeness of metrics, where you have the need to track all field changes or all document deletions no matter how they are performed (via the front end, or via Notes or Domino agents running in the "back end").

You could adapt the NotesTracker code to run in any agents written in LotusScript, since the NotesTracker routines were developed in this language. (LotusScript was chosen for NotesTracker since this language has all the features needed to perform the fairly complex tasks involved in usage tracking, in both the front end and the back end. The Notes Formula Language does not have the programmability, and the Java language only works in the back end.

## *What to Track? Individual Notes Databases or Sets of Databases?*

You have considerable flexibility on the way that you deploy NotesTracker to gather application usage metrics about your various databases. NotesTracker can be used to track activity in individual Notes databases, or in related sets of Notes databases – as few or as many as you desire.

One user even justified purchasing a corporate NotesTracker licence purely to track changes being made to a single database (which happened to be the Domino Directory, a.k.a. the Public Address Book).

You do not have to track activity in each and every database, and the degree of tracking can vary from database to database. Indeed, you will probably only wish to gather usage metrics for a limited number of databases – or even just a single database -- where you see definite value and payback.

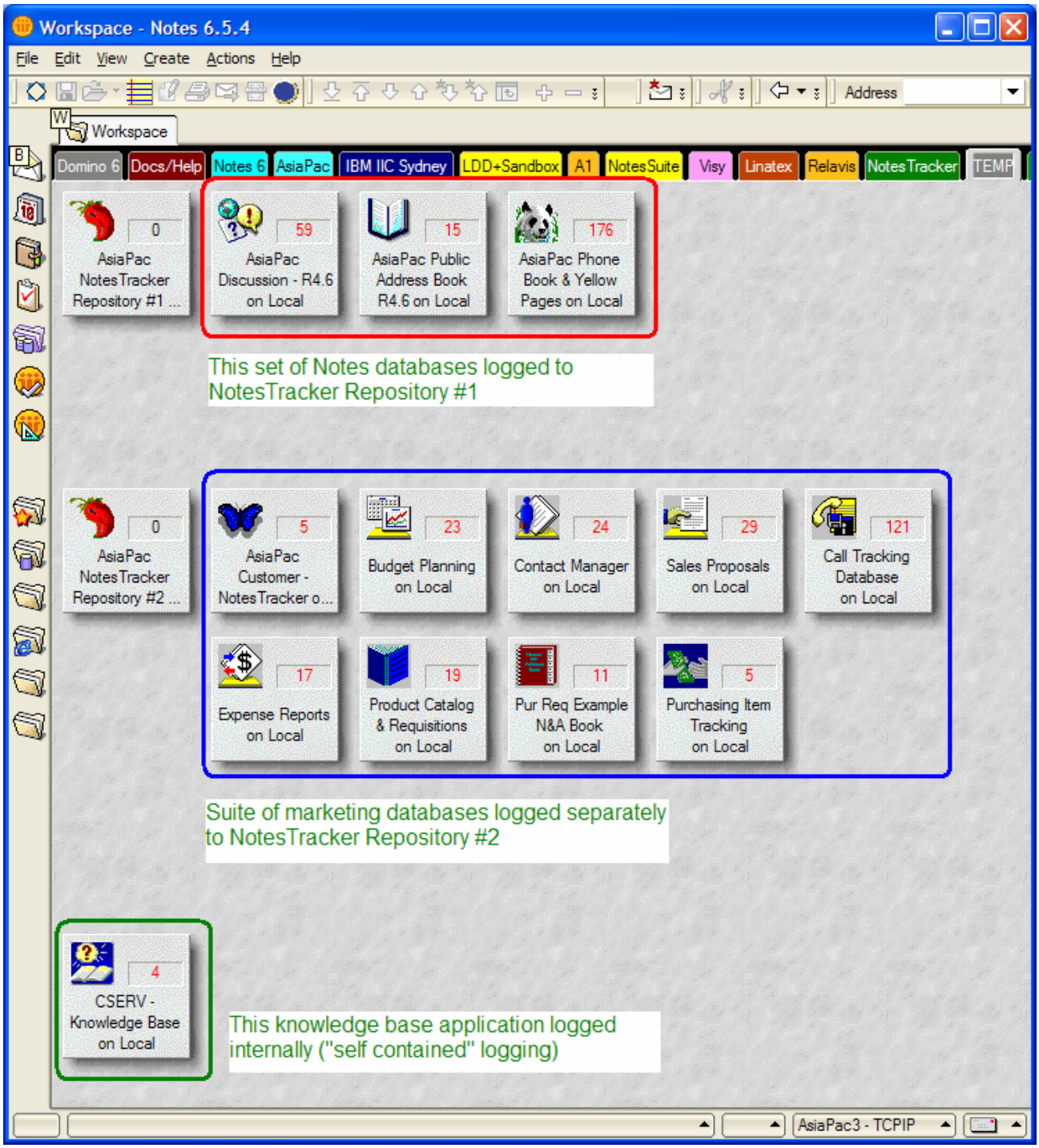
Furthermore, you can easily vary the nature or degree of tracking in a given database if your tracking requirements for the database vary over time. For example, you may want always to track document Deletions and Creates/Updates in a certain business-critical database, but only track document Reads in that database for periods of a day or two every now and then (just to get a feel for how overall use of the database is trending).

How is this achieved? As fully explained later in the Administration and Development Topics sections, each database that you wish to track via NotesTracker must have its own NotesTracker Configuration Document. It is easy – without the need for programmer intervention – to modify the various control values in this document and thereby changing the nature and extent of usage tracking (for that individual database) from that point of time onwards.

What is more, it is a simple matter to have a given database is tracked in isolation or as part of a related set of databases. This is determined merely by a setting in each database's **NotesTracker Configuration Document** which controls where the usage log entries are written (either to an isolated NotesTracker Repository or to a shared NotesTracker Repository). The decision is entirely yours, and you can easily adjust the various control settings as and when your usage tracking requirements change.

If it makes sense to track a database as part of a set (or "suite") of related databases, you merely specify that a single NotesTracker Database is to act as the repository of the usage log documents for all of the databases in that set. Different sets of Notes databases can have their usage log documents written to different NotesTracker Repository databases. This gives you great flexibility in configuring database usage logging for your Notes/Domino applications, and in the metrics analyses that you perform subsequently.

It does not end there. If it makes more sense for whatever reason to track a particular database in isolation, then you can specify a NotesTracker Database that is to act as a repository of the usage log documents just for that single Notes database. Going one step further, there is even an option to specify that "**internal tracking**" should occur, in which case the NotesTracker usage log documents are written to the database itself (rather than to an external NotesTracker Repository database). This enables usage tracking to be "**self-contained**" within that particular Notes database – the database acts as its own repository of usage log entries – and this may be quite desirable in some situations.




Detailed NotesTracker set-up considerations, together with the advantages and disadvantages of the various logging options, are discussed in depth later in this guide (in the Administration Topics and Developer Topics sections).

In summary, NotesTracker affords you unique flexibility in deciding how databases are grouped for usage tracking. What is more, the arrangement is quite easily modified as your grouping requirements change: you simply change settings in the NotesTracker Configurations document in the affected database(s), and then just wait a little while for the NotesTracker Configuration document(s) to replicate around your Domino network for the changes to take effect globally.

## *Usage "Reports"*

There are numerous views in this database that present the usage statistics in meaningful ways. The views are selected using the buttons in the navigator on the left-hand side of the open database's window. It is a simple matter for a designer or "power user" to add further views that present information in other meaningful ways.

Executive sponsors, content managers, knowledge managers, database administrators, auditors and others will find the NotesTracker log information invaluable for many reasons, such as:

- To determine the **most popular documents** (or the least popular ones)
- To discover the **rarely-used documents** that are good candidates for being purged
- To analyse the **time-dependency of documents** (e.g., whether a "hot" document is accessed soon after it is created)
- To discover who are the **regular users** of the databases (and, by implication, the occasional users)
- To recognise the **contributors** (document creators and editors)
- **To understand who deleted which documents, and when they did it** - sometimes a contentious issue!
- **To understand the pattern of View Opens** - because excessive opening of views can cause a severe drain on Domino server resources, new in Version 2.3 are usage metrics on view opens that can provide an extremely useful insight into which views in which databases are contributing to sever load. Views in databases holding many documents can also consume significant amounts of disk space. The view open metrics provided by NotesTracker offer you a definitive way to **determine which views are used infrequently and thus are good candidates for deletion from a database's design**.
- To set up a **"Breaking News" view** that is suitable for incorporating in a pane on your corporate or departmental Portal Page or Welcome Page. (An example implementation of such a view was added in Version 2.3.1)
- **To easily understand who changed the contents of which document fields.** This can be extremely useful in all sorts of ways, a few examples being:
    - a Domino Administrator knowing who changed a critical security or replication field in a Server Document
    - a manager finding out who updated vital information about an employee or a customer
    - an auditor quickly determining where, when and by whom changes were made to certain monetary fields
    - a lawyer better understanding where (and when) changes were made to parts of a legal document
    - a sales manager knowing exactly what quota changes were effected for his sales representatives

It is worth stressing that the usage tracking function (and sub-functions) can be turned on or off at any time, on an individual database-by-database basis. People with the appropriate authority do this simply by editing the NotesTracker Configuration Document in a given database. This is described fully in the Database Administration Topics section below.

**Note:** in the remainder of this document, the term "NotesTracker" may be used interchangeably with the earlier names "Usage Tracker" or "Document Usage Tracker". The name "NotesTracker" was adopted for Version 3 of the software, and some references to the original names might remain – but it's more likely you'll come across them embedded deep in the NotesTracker code, rather than in this user guide.

# Administration Topics

## *Selecting the NotesTracker Target Repository for each Database*

The usage tracking code is added to a Notes databases and is activated by events such as opening Notes documents or views, deleting documents, and saving documents with new fields or updated field contents.

For each such event, the code writes (appends) a new Notes document that we call a “Usage Log record” into a NotesTracker target database or “repository”. This means of course that NotesTracker must have a means of specifying, for each database being tracked, the repository to which such Usage Log documents are to be written.

The NotesTracker administrator and/or Notes Administrator will be involved in determining where on the Domino server (or Notes Client workstation, in the case of local usage tracking) each NotesTracker Repository is located.

A quite simple mechanism is used for this. You must set up in advance, in each Notes database being tracked, a **NotesTracker Configuration Document** that specifies the name and location of the NotesTracker Repository for that particular database. If the NotesTracker Repository is subsequently relocated, naturally the NotesTracker Configuration Document will have to be correspondingly updated to reflect the new repository location.

There are **three alternatives for specifying the locations of the NotesTracker Repository Database:**

- Via its Replica Identifier, normally shortened to "Replica ID" (independently of the operating system's directory structure)
- Via its Server, Path and Filename (that is, via the operating system's directory structure relative to the root Domino data directory)
- Internally (in the current Notes database)

There are certain advantages and disadvantages of using each alternative, which we will now consider.

## *Opening the Database by Replica ID*

The prime advantage of opening a database via its Replica ID is "flexibility" in locating the database. Notes/Domino will perform a search in order to locate the database -- you only have to specify its Replica ID value, rather than specifying some fixed location in the operating system's directory structure.

Every time that a usage log document is to be written (that is, whenever a document is closed that is based on a form nominated for usage tracking), the NotesTracker Database has to be accessed (located) by means of its Replica ID. If there is no replica of the NotesTracker Database on the local server, then Notes has to begin a search for the nearest server containing such a replica. This could take anything from a few seconds to minutes, dependent on network topology and network traffic load at the time.

The prime disadvantage of opening by Replica ID is that sometimes the search for the replica may take a long time (in the worst case tens of seconds or even minutes, if the network being searched is far-flung with a complicated topology, or if network traffic is heavy at the time). Users will not be impressed by the consequent long wait times that ensue!

**HINT**: Place the NotesTracker in the root Notes data directory (such as C:\Notes\Data).

**GUIDE: You should place a replica copy of the NotesTracker Database on EACH server that holds an application database that is being tracked.** Otherwise, there almost certainly will be problems with usage tracking (logging will fail because the Notes/Domino will not be able to open the NotesTracker Database so as to add usage log entries to it).

A secondary disadvantage of opening by Replica ID is that sometimes the database that gets opened (following the replica search) is not the one you might expect. For example, it may not be on what appears to be the "nearest" server, or it may turn out to be some other replica copy such as a test or backup copy rather than the "production" replica.

**HINT:** In the NotesTracker environment this may lead you to conclude that there's some failure in usage tracking, when in fact the NotesTracker is working fine and all that's happening is that unexpectedly the log documents are being written to the duplicate replica database. When you open the expected NotesTracker Database, you hunt in vain for new log entries (and think that usage tracking has failed) without realising that logging did occur successfully but to some other NotesTracker Database replica. This is the first thing you should investigate if usage tracking stops working (perhaps due to the appearance on the scene of a new replica copy somewhere in Domino's replica search path).

## Placement of Replicas of the NotesTracker Database

For best response times, it would probably be best to place the NotesTracker Database's replica in the root data directory on each Domino server, rather than in a subdirectory (or worse, on another system).Of course, contrary to this is the reasonable approach that you should keep the root Domino directory as free of application databases as possible.

What about the case of having the NotesTracker Database on a user's Notes Client workstation or notebook computer? By doing this, usage can be tracked even for users without "live" connection to the Domino server network. In such cases, the above rules still apply, but it may prove difficult to stop the user from moving the database to a different path and filename (or even from creating a non-replica copy).

If the Notes Administrator (or perhaps the Notes "power user") sets up one-way selective replication from the local NotesTracker replica to the server-based replica, this will keep down both the local database size and replication traffic over the network, while enabling Usage Log documents to be sent in even from "mobile" users, so as ultimately to be centrally consolidated and analysed for an overall analysis of usage.

## *Opening the Database by Path and Filename*

In this method of opening a database, you specify a **server name** plus a **path and filename**, as is illustrated not far below.

The main advantage of opening a database via Path and Filename is that it's usually a very fast operation, so there are few if any problems with the long open times (that sometimes may occur when opening by Replica ID, as just discussed) causing user dissatisfaction.

The main disadvantage is that opening by Path and Filename is inflexible, compared with opening via Replica ID, since it will only succeed when you specify the exact path and filename. Also, administrators or users may (for various reasons) move a database from its original path on a server (or local workstation) -- possibly to a new drive that has more space, and sometimes even with a changed filename -- which causes the dependent database-opening code in NotesTracker to fail.

**Note:** since there is only a single NotesTracker Configuration document in each of your databases, you must follow the rule that the NotesTracker Database has **the same path and filename** on each Domino server or Notes Client workstation. This is the first thing that you should verify if you encounter problems opening the NotesTracker Database using this method.

If the Path and Filename value is changed to point to a different path location in a NotesTracker Configuration Document on one server, the changed path value will replicate to other servers and this will probably cause the database open to fail on other servers (unless the database is moved to the same directory on all the other servers).

## Note regarding clustered Domino servers

You should be aware NotesTracker as distributed uses simple database file open operations, rather than “Open with Failover”. In the event of a failover, it is reasonably likely that the use of a specific server name and file path will cause the database open to fail. Therefore, in a clustered Domino server environment, opening by Replica ID may be the preferred option.

Otherwise, your Notes developer could decide to modify the distributed NotesTracker code so as to use the “Open with Failover” method to handle the clustered server situation. Since there is only a single field in the NotesTracker Configuration Document for the Path and Filename, you would have to be careful to deploy the database using the same Path and Filename on each server. (It would be possible to modify the NotesTracker Configuration Document’s form design and tracking routines to cater for different paths and filenames on individual clustered servers, but probably not worth the effort – not to mention that the increased complexity could cause administration and/or operational problems.)

## *Internal Logging – Logging to the Current Database*

For a particular Notes database, instead of directing Usage Log entries to an external database your NotesTracker coordinator and/or Notes developer may specify that they will be written to the "current" database. This makes usage tracking “self-contained” to that database.

One advantage is that if you select this option, the Usage Log documents may be written more rapidly than if they are directed to an external database.

On the other hand, the current database's size will grow – perhaps rapidly -- due to the additional documents being added. Also, the view indices will be built for the views you add to display the usage metrics. The impact on extra disk space occupied by the database and extra processor cycles must be taken into account when managing Domino servers and Notes Client workstations that hold the database.

## *Security, Access Control and Privacy*

The distributed NotesTracker Database has the following Access Control List (ACL) settings:

- **Default access should be "Author"** (or perhaps you can try “Depositor”)**.**

  **NOTE: this is important.** It is the basic mechanism that allows your Notes database users to create Usage Log documents in the designated NotesTracker target database.

  Usage tracking will not operate unless users can add log documents. Users do so implicitly (without being aware of it), every time that they perform a tracked action (document Create/Update/Read/Delete or View open) in a database which has usage tracking switched on.

  An alternative would be to create a person group -- say, one named "**NotesTracker Authors**" -- and populate it with names of users appropriate to your operating environment.

Also, you might create and use these other person groups:

- "**NotesTracker Readers**" group, with Reader access -- not allowed to delete documents -- for people who need to be able to see the usage tracking views.
- "**NotesTracker Editors**" group, with Editor access -- and optionally allowed to delete documents, if you want them to have the ability to "clean" the NotesTracker Database by manually deleting documents from it (there may be situations where unwanted log documents appear in the database).
- "**NotesTracker Managers**" group, with full Manager access (for the usual reasons).

**IMPORTANT NOTE: only database managers have the ability to create and modify the NotesTracker Configuration document**. This is explained in the Developer Topics section below.

Another consideration: replica copies of your Notes databases will often be deployed not just on Domino servers but also "locally" (that is, on Notes Client Workstations and notebook computers). If a database's ACL is not set to "**Enforce a consistent Access Control List across all replicas of this database**" then it is quite possible (or even likely) that the Notes user -- being by default the Manager of the locally-deployed database -- may change some of the usage tracking control settings in the NotesTracker Configuration document. In the worst case, the user may even completely switch off usage tracking. Furthermore, any such change to the NotesTracker Configuration will probably replicate throughout your Domino server network and disrupt usage tracking on a wide scale!

These are only suggested ACL settings. They are NOT mandated by NotesTracker’s usage tracking code, and you can set up your own alternatively-named groups or use existing ones with whatever access rights you deem appropriate.

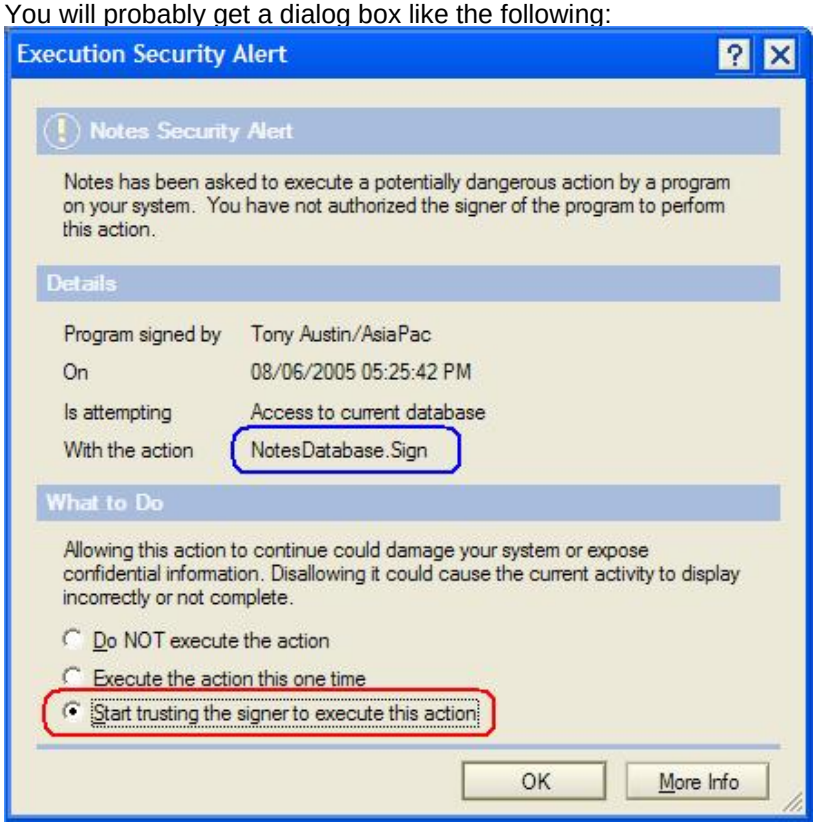
## *Authorizing the NotesTracker Web Agents by Signing Them*

If you want to track document usage activity against a database that is performed via a Web browser, then (as later described in the Developer Topics section) it is necessary for two Web agents to be included in the database. The two agents are **NotesTrackerWebQueryOpen** which tracks documents being opened by the browser, and **NotesTrackerWebQuerySave** which tracks documents being created and fields being updated.

It is beyond the scope of this document to go into details about the need to **sign** a Web agent so that it has sufficient rights to execute on a domino server. Please refer to the Lotus Domino Administrator Help for an explanation of the procedure for signing agents, as well as how to decide which ID file has appropriate execution rights and so can successfully be used to carry out the signing.

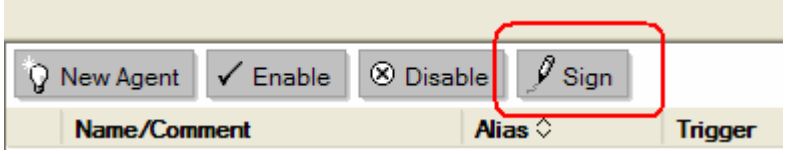
To save you the trouble of going through the signing process manually -- via the Lotus Domino Administrator interface -- there is an agent (added to NotesTracker in Version 4.4) that will sign these two agents automatically. Simply log on to Lotus Notes (not the Domino Administrator) **with the appropriately authorized ID file**, and execute the agent named **Sign the NotesTracker web agents** (which will sign just the two above-named Web agents and not any other design elements in the database).

You will probably get a dialog box like the following:




Select "**Start trusting the signer to execute this action**" and click the **OK** button.

Alternatively, if you are using Domino Designer 6 or later you can sign each agent using the button in the Agent List design view, as follows:




**Note**: you must sign the agents with an **appropriately authorized ID file** (an ID with sufficient authority to allow the two agents to run on the Domino server).

## *General Security and Privacy Considerations for Document Tracking*

**IMPORTANT**

**Security & Privacy Considerations for Administrators/Developers**

When implementing usage logging in your databases, **be very careful not to allow confidential or sensitive information to be logged**. Logged information might not be appropriate for general consumption.

Even the document titles may give clues to confidential, sensitive or personal information. Consider the disastrous implications of just the mere mention in a Usage Log document's title of such things as a proposed merger or acquisition, legal action, an employee's possible termination, or many other such matters!

**You should thoroughly test the logging activity before deployment to ensure that confidentiality and privacy are maintained appropriately.**

**It may even be that certain databases, or at least aspects of them, should not be tracked.**

**Developers and administrators must always keep this in mind.**

## Sample Privacy Notice to Users

The database contains a form which you can use as a starting point or guideline for inclusion in each tracked database as a privacy/awareness notice to your users.

**SAMPLE PRIVACY NOTICE TO USERS**

**Important information regarding Usage Tracking**

Usage Tracking may be activated continuously or from time to time so as to record activity against the documents in this database. A snapshot is taken of information about all users (such as user name, date/time and server) plus such things as document "Read", "Create", "Update" and "Delete" operations, and field changes..

**Why are we tracking usage?**

Tracking Read access lets us see what parts of the database are getting the most "hits", enabling better knowledge management of the database (e.g., what is useful and should stay in, and what is rarely used and could be dropped out).

Tracking lets us proactively manage potential opportunities or issues. For instance, we may find that a particular database or server is being heavily used, or that certain documents are noticeably more popular than others. We can see from the usage pattern how to improve the information provided via the database or perhaps that we need to improve the design of the database so that you can use it more efficiently and effectively.

Usage tracking can enable a "Latest News" or "What's New" function to be added for the database. This would identify all new or recently-updated updated documents that could (for example) be displayed on the home page and so keep you better informed.

Each time that a contributor to the database creates or updates a document, a record of the change is filed by the Usage Tracker and a list of these documents could be displayed in descending date order (from today) in "Latest News". This enables you to identify new or modified documents at a glance, rather than having to scour the containing database (or databases) for any changes.

The information collected by the Usage Tracker will not be used for any purpose other than those outlined above.

# *The NotesTracker Configuration Document*

A **single** NotesTracker Configuration Document is needed in each database that is being tracked.

If there is no configuration document in a database, then (in order to not interfere with user access to the database) the NotesTracker routines detect this and no tracking occurs for that database until the configuration document is subsequently added.

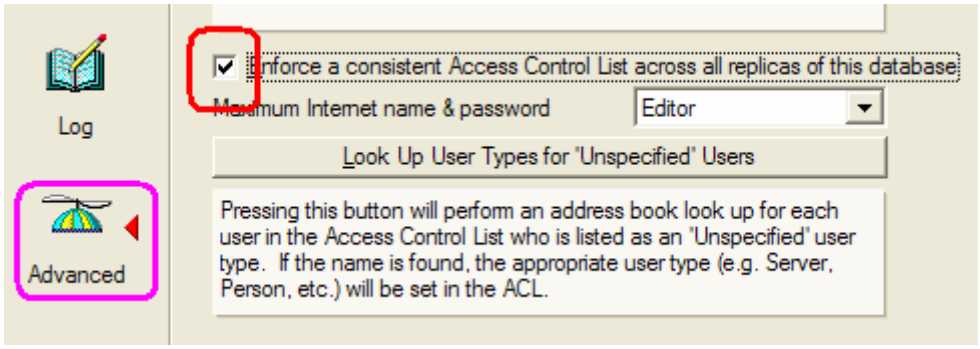
You are prevented from manually creating more than one such configuration document per database. However, experience has shown that multiple configuration documents can sometimes be present, presumably due to replication faults. It is safe (and probably wise) to delete superfluous configuration documents, leaving only one per database.

## Access Level Required for Editing the Configuration Document

Because the NotesTracker Configuration Document is the key means to control whether usage logging is active, and which tracking options are in effect for a database, it is most important that the ability to edit this document be restricted.

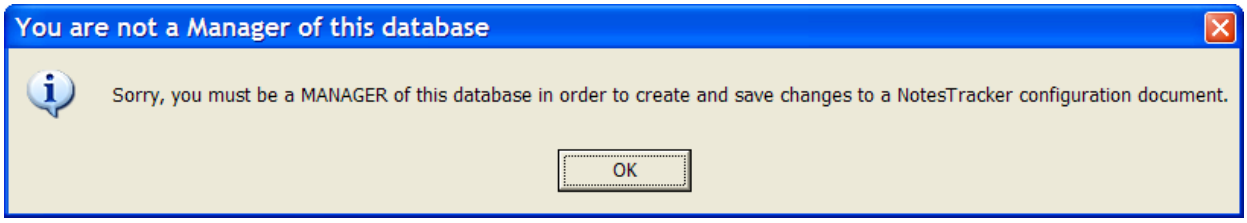
Therefore, only those with Manager level access in a database's ACL are allowed to edit the NotesTracker Configuration Document. (This is controlled by code in the Querysave event of the NotesTracker Configuration form.)

**Note**: following normal Notes behaviour, the checking of access level will only effective if the database is opened from a Domino server or if the Access Control List for the database is opened locally and the database has the setting (under "Advanced") to enforce a consistent ACL, as follows:



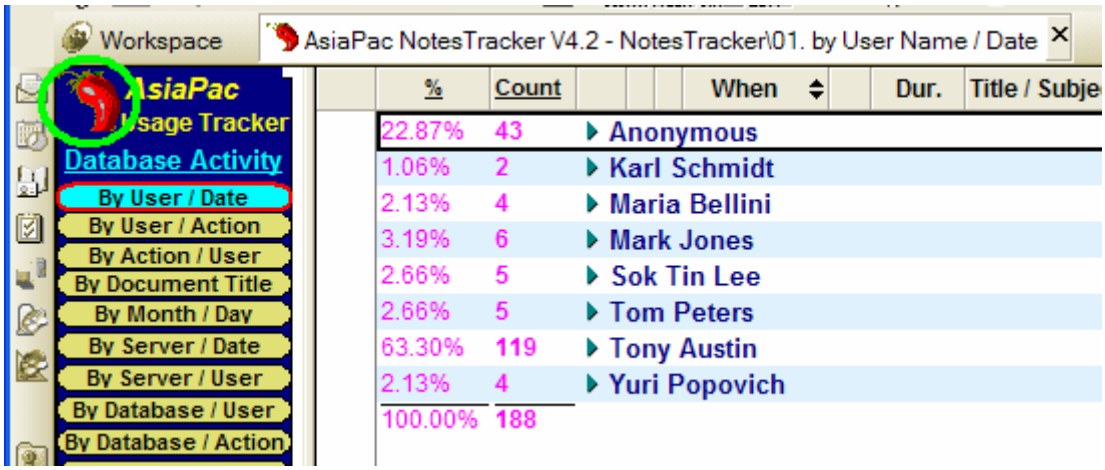

If the database is opened locally without the enforcement of a consistent ACL, then a user doesn't need Manager access level to edit the configuration document. Any changes they make would most likely be replicated around the network to other replicas of the database, which would not be a good situation at all for controlling usage tracking of the database.

If you don't have the required Manager rights, you will see the following warning when you attempt to edit the NotesTracker Configuration Document:



## NotesTracker Configuration View

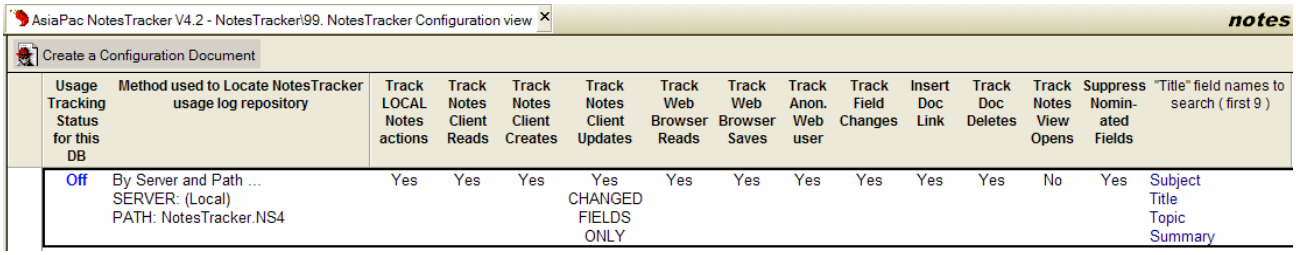
In the NotesTracker database an easy way to switch to the NotesTracker Configuration view is by clicking on the red chilli image at the top of the navigator (circled in green in the following illustration):



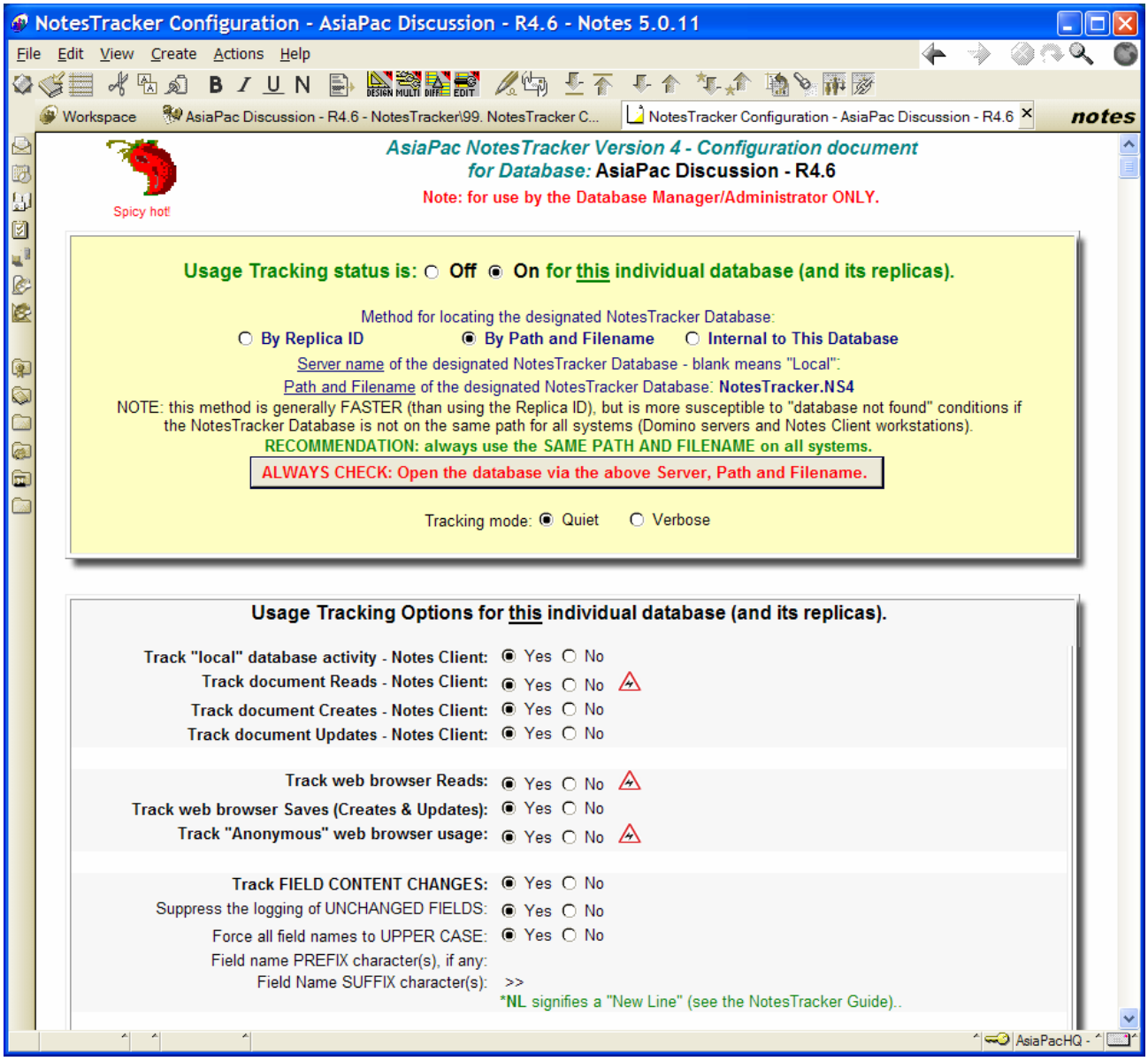

You should ensure that usage tracking if **off** for the NotesTracker database itself. There is no point at all in tracking usage in this database.

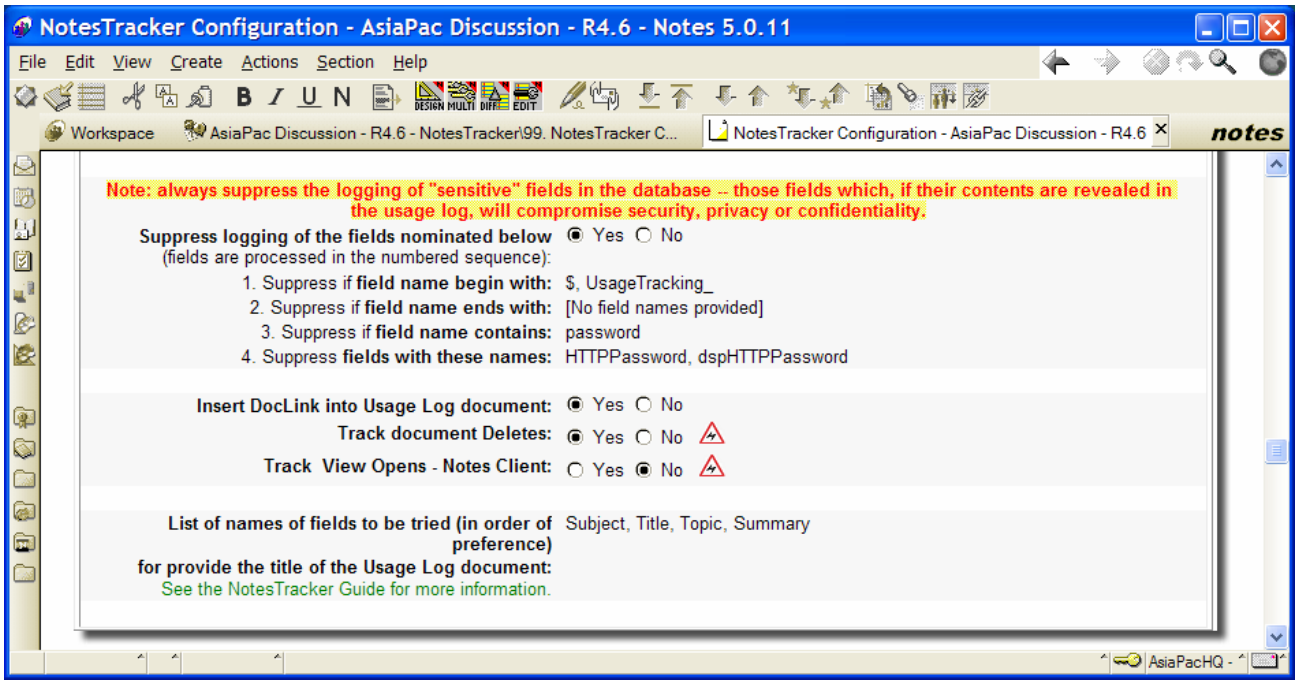
In other databases, how you get to the NotesTracker Configuration view is entirely dependent on how your Notes developer has set up the view (discussed in the Developer Topics section later in this guide).

This view shows the selected NotesTracker option settings at a glance:

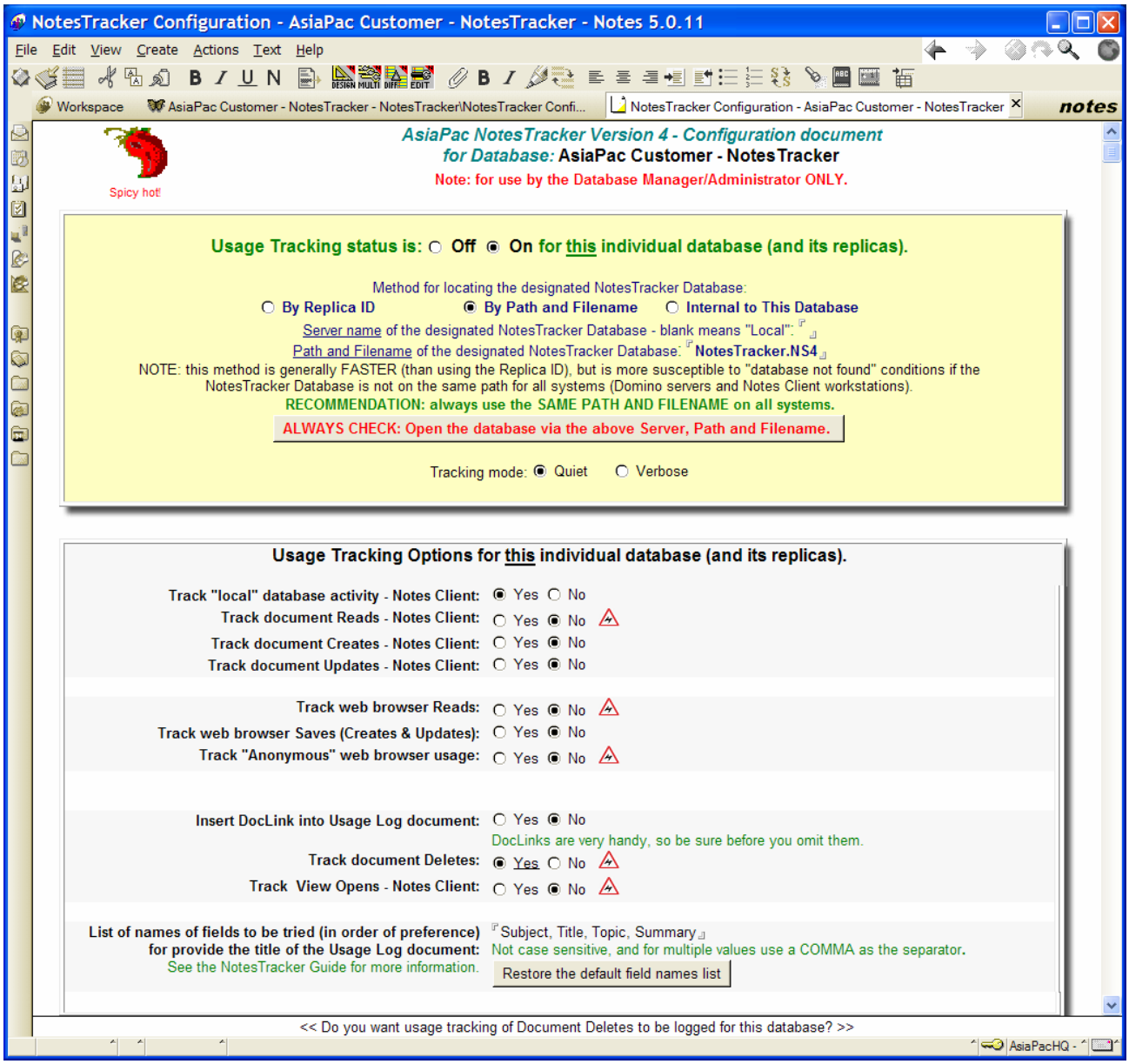


If you decide to access the NotesTracker Database **via Replica ID**, an example (with all options activated) is:




NotesTracker Configuration - AsiaPac Discussion - R4.6 - Notes 5.0.11
File Edit View Create Actions Section Help
Workspace
AsiaPac Discussion - R4.6 - NotesTracker\99. NotesTracker C...
NotesTracker Configuration - AsiaPac Discussion - R4.6
notes
Note: always suppress the logging of "sensitive" fields in the database -- those fields which, if their contents are revealed in the usage log, will compromise security, privacy or confidentiality.
Suppress logging of the fields nominated below (fields are processed in the numbered sequence): Yes No
1. Suppress if field name begin with: $, UsageTracking_
2. Suppress if field name ends with: [No field names provided]
3. Suppress if field name contains: password
4. Suppress fields with these names: HTTPPassword, dspHTTPPassword
Insert DocLink into Usage Log document: Yes No
Track document Deletes: Yes No
Track View Opens - Notes Client: Yes No
List of names of fields to be tried (in order of preference) for provide the title of the Usage Log document: Subject, Title, Topic, Summary
See the NotesTracker Guide for more information.
AsiaPacHQ

If you decide to access the NotesTracker Database **via Path and Filename**, an example (tracking deletions only) is:



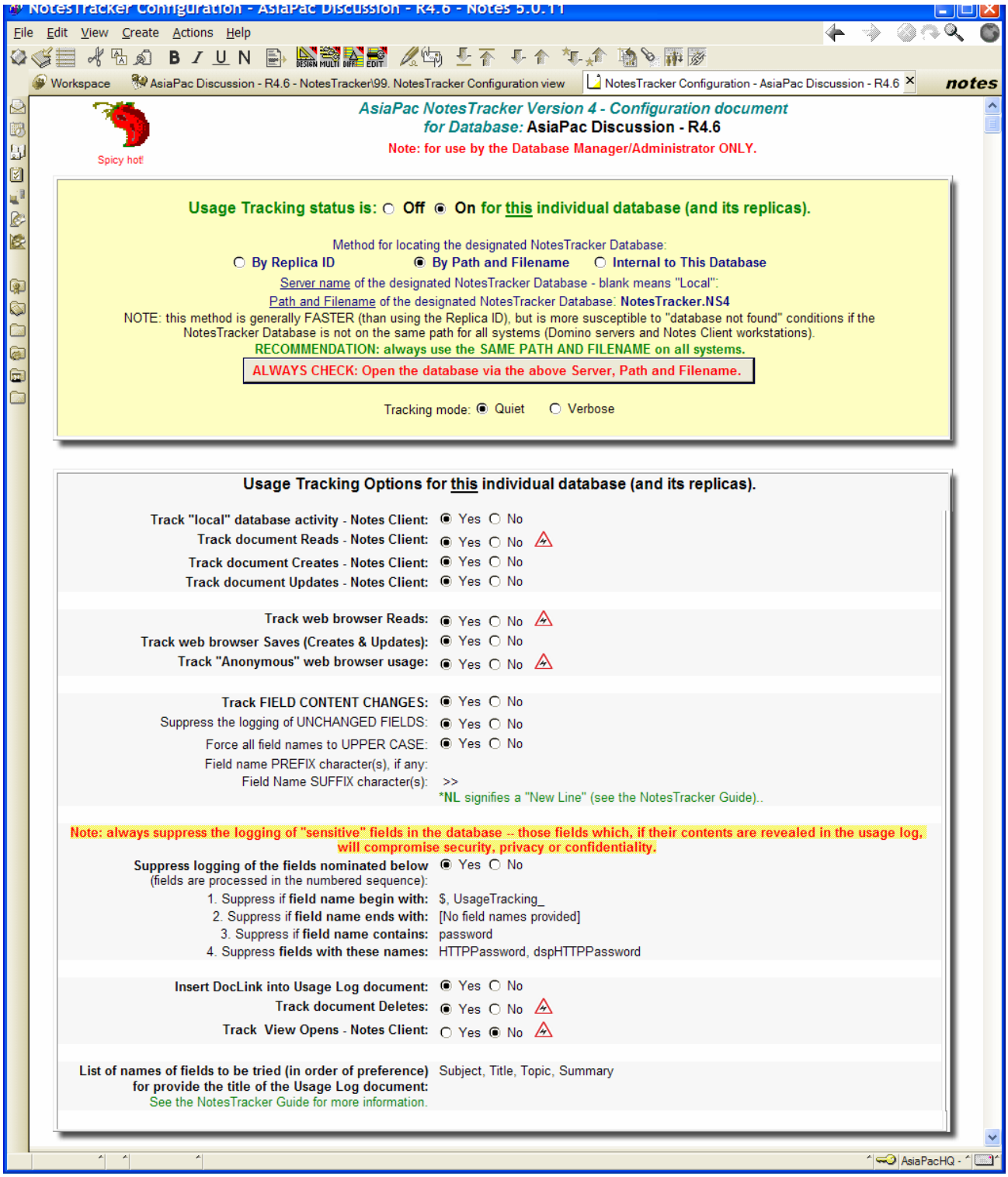

You should supply the **server name** (new in Version 3.3), leaving the field blank to represent a "Local" database, plus the database's **Path and Filename** relative to the default Notes data directory, not in the operating system's path and filename format).

**Note:** in the above illustration, it does not matter that the setting “Insert DocLink into Usage Log document” has been left on (set to “Yes”), since DocLinks are never created for deleted documents anyway.

## Self-Contained Usage Tracking – “Internal to This Database”

New in Version 3.1 is an option to write the NotesTracker log documents into the same database which provides a "self-contained" usage tracking capability for a particular database. You could always have done this in previous NotesTracker releases, simply by using the database's own Replica ID value, but also new in Version 3.1 is the button ;labelled "**For logging to the current database itself - Set this database's own Replica ID**" which is provided to save you the trouble of determining it from the database's design synopsis.

However, a third NotesTracker Database location method was added in Version 3.1 to dispense with the need to enter the database's own Replica ID. This option is activated by clicking the Radio Button labelled "**Internal to This Database**", as the following example shows:



This also illustrates using the special value ***NL** to specify that, in the NotesTracker log, each updated field's content is to be displayed starting on the line immediately below the field's name (using the New Line character as a separator). You should find that this makes it easier to decipher the field Before/After audit trail section of the Usage Log document, compared with using some other separator between each field's name and its contents.

An alternative way of achieving the same result is to select the "By Replica ID" radio button and then to supply the database's own replica ID. To guarantee entry of the correct replica ID value, simply click the button provided for just this purpose:


There's no advantage in doing it this way (as against selecting the "Internal to This Database" option), but the facility is provided for compatibility with early releases of NotesTracker.

## Special Considerations for Internal (Self-Contained) Usage Tracking

In some cases, it may be advantageous to write usage log documents "internally" in a database. For one thing, it avoids the need to deploy a separate NotesTracker Database. (There's noting special about NotesTracker here, just all the usual issues that tend to arise when you go about deploying a new database across your Domino network.)

Such "self-contained usage tracking" may be a reasonable solution when one of your databases is not related in any way to other databases, and it would not make sense to write log documents for this database to the same NotesTracker Database as for others of your databases.

However, you must **allow for the increase in size of the database** caused by the writing of NotesTracker log documents into it, especially if the database is a popular one that has many operations performed on it (many document accesses -- Reads, Creates, Updates, etc).

Also -- as discussed in the Developer Topics section below -- at least one new view must to be added to the database design or else you'll never be able to examine the log documents, and all of the existing views must be checked and modified if necessary to ignore the NotesTracker log documents.

## *Usage Tracking Options*

Usage tracking is controlled, for each tracked database, by the entries in the NotesTracker Configuration Document that is accessed at the instant that a database activity is performed. When you make changes to a configuration document on one Domino server the changes do mot take effect until they are replicated around your network to other replica copes of the database. Naturally such configuration changes do not take global effect until the replication cycle is finished, which is entirely dependent on the usual replication control settings (at both the server/network level and the individual database level).

You can turn tracking **on** or **entirely off** for each database.

When tracking is turned on, you can **separately control tracking** for individual document "action" types - **Read, Create, Update,** and **Delete**, and for other database events such as the opening of views.

You can specify an external database into which usage log entries (documents) are to be written via **Replica** ID, or via **Server plus Path and Filename. Alternatively,** you can specify that the usage log entries be written into the current database itself rather than to an external database.

You can **suppress logging of "local" database activity**. That is, you can switch off the tracking of databases that opened on a desktop system's or notebook PC's local hard disk, so that tracking is carried out only if the database is opened from a Domino server. This may or may not be a good thing to do. You must make a policy decision, it's entirely up to you -- but should be discussed with the appropriate Knowledge Manager or other person who is sponsoring or managing the deployment of a particular database.

You also can **specify tracking of web browser Reads, and of web browser Saves (Creates and Updates).** As a means of reducing logging overheads in a high-transaction-rate Internet situation, you can **suppress the logging of "Anonymous" web user activities.**

**NOTE:** always use the "test" button (which automatically appears) to **verify that an external database can be opened via the Replica ID value (or Server plus Path and Filename values)** that you have entered.

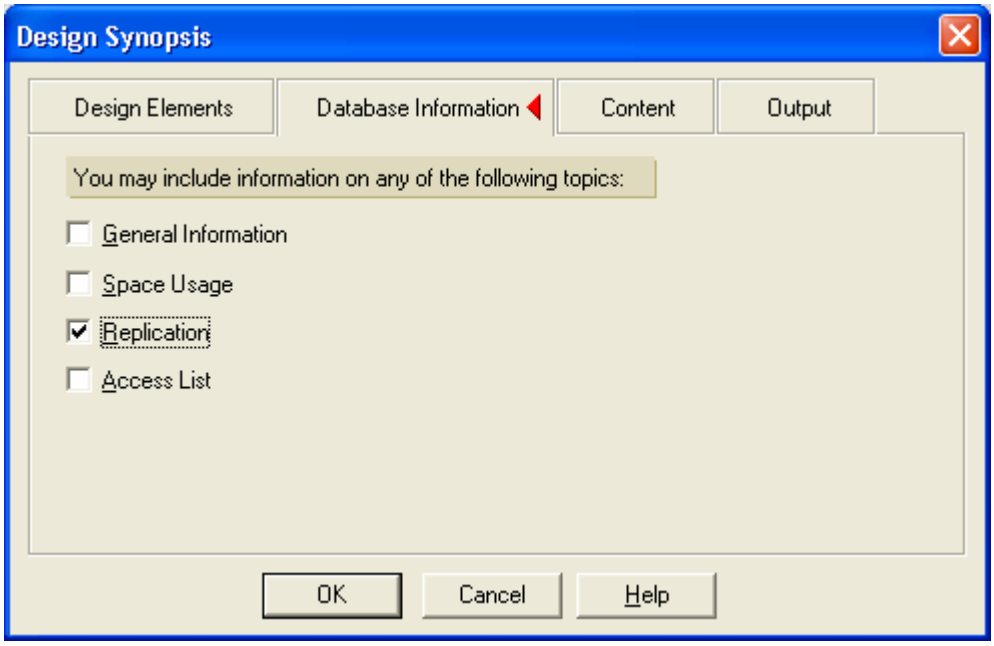

Or


Usefully, this also gives you the opportunity to gauge the difference (if any) in opening speed using these alternative approaches. As discussed earlier, opening a database via Replica ID can be a lengthy operation, taking seconds (or even minutes in the worst cases, if many Domino servers have to be touched to locate the replica).

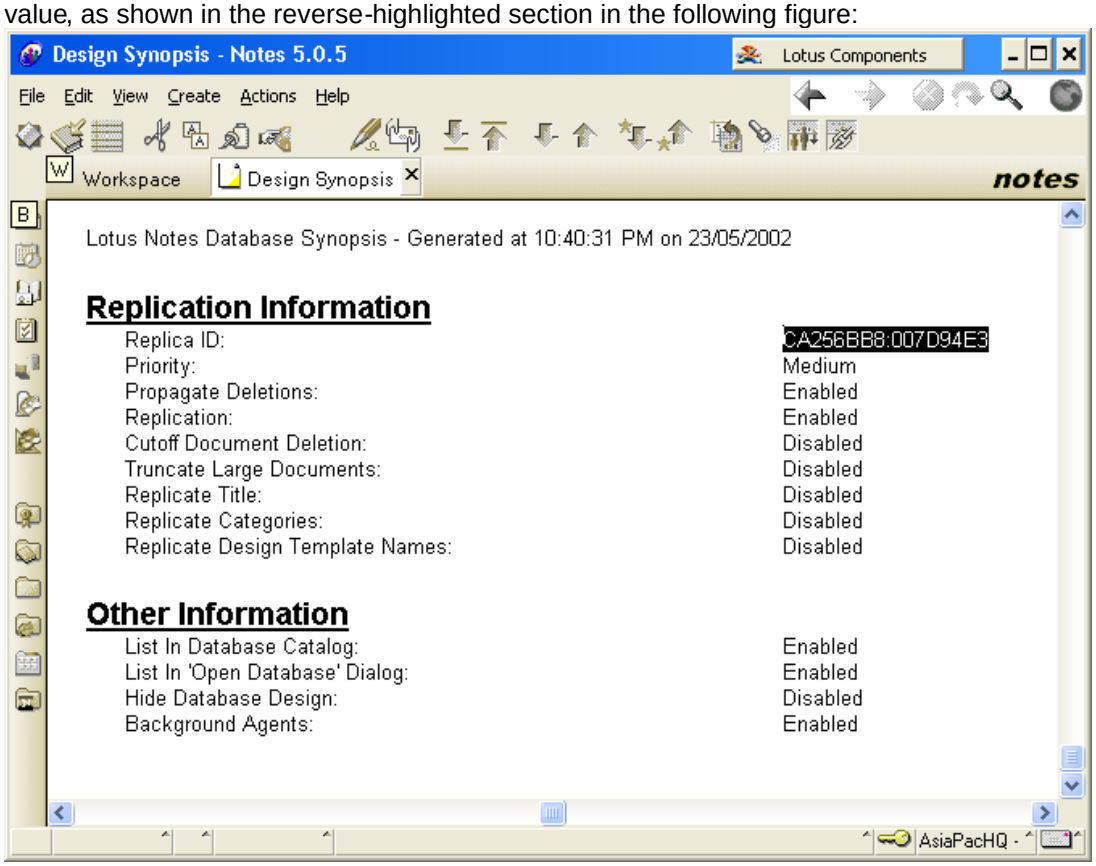
As previously discussed, opening the NotesTracker Database via its Path and Filename has the potential of being faster (and so yielding a better user experience). However, it means that each database and its replicas must all be stored in the **same Path and Filename on all systems** otherwise the usage Tracking code will fail to open the NotesTracker Database on at least some system(s).

Using the button can be a good troubleshooting aid for problems relating to opening of the target NotesTracker Database into which you want usage log entries to be written.

**Note 1:**
The Replica ID can be obtained via the **File - Database - Design Synopsis** menu option, like this (for Notes Release 5):




It is highly recommended that you minimise transcription errors by using Copy-and-Paste of the Replica ID value, as shown in the reverse-highlighted section in the following figure:




**Note 2:**
**T**he design of the NotesTracker Configuration form (via a hidden field) prevents a configuration document from being saved if there is already one in the NotesTracker Database. Experience has shown that the

presence of multiple configuration documents in the database can cause problems, and such multiple documents probably exist due to replication between servers (they cannot occur from the user interface).

**The database administrator should delete all but one of any such multiple configuration documents.** If multiple NotesTracker Configuration documents inadvertently finish up in a database -- for example, by replication error, or by a document copy-and-paste operation -- you should be aware that only the contents of the first one will be accessed by the NotesTracker code. (Although duplicate NotesTracker Configuration documents are ignored by the NotesTracker code, it's a good idea to purge any duplicates, leaving only a single NotesTracker Configuration document)

## Verbose and Quiet Tracking Modes, and Some Typical Tracking Issues

This section applies only for database accesses via a Notes Client.

Messages are sent only to the **Notes Client status line** (at the bottom of the Notes Client window). If you find that usage tracking is not occurring at all for a given database, or that some but not all of the selected usage tracking options seem to be working, you should examine the messages that appear on the status line.

A small number of additional messages pop up when you click on the status line. Since the status line has a limited buffer (only about ten or fifteen messages) you'll need to be quick to catch any relevant messages.

To be non-disruptive to the normal operation of your Notes applications that are being tracked, NotesTracker does not display any dialog boxes. This is the "Quiet" mode, when only an occasional advisory message appears on the status line.

There is a "Tracking Mode" setting in the NotesTracker Configuration Document. Its purpose is to provide an intermediate level of troubleshooting and problem solving or each database. If you switch to "Verbose" mode, then a range of additional advisory or error messages should appear in the status line.

Careful monitoring of these additional messages should help you to diagnose and rectify some NotesTracker problems.

Be sure to switch the setting back to "Quiet" mode when you've finished your troubleshooting.

If the status line messages do not help you to clear up the tracking problem, then it's time to call in your Notes developer (the one who's supporting NotesTracker) to carry out some in-depth problem analysis for usage tracking on that particular database.

**STATUS LINE MESSAGES RELATED TO DEVELOPMENT ISSUES:**
The messages might indicate that the developer has not set up NotesTracker correctly for the database, such as forgetting to add a required design element (like a form or view). Problems like this should have been eliminated by careful testing prior to deployment, and really shouldn't occur in the production environment!

**STATUS LINE MESSAGES RELATED TO OPERATIONAL ISSUES:**
Problems that might occur in the production environment are generally due to operational issues. These are generally related to incorrect settings in the NotesTracker Configuration Document, such as:

- Incorrectly specifying the NotesTracker repository database's Replica ID or Server and Path names
- The NotesTracker Configuration Document being inadvertently deleted from the database.
- The NotesTracker Configuration suffering from corruption (rare, but possible; and probably any corruption in the database will cause other operational problems that are far more troubling than the effects of corruption on usage tracking).
- The NotesTracker Configuration Document undergoing replication problems, leading to multiple configurations documents. Only the settings in the first document in the configuration view will be honoured. (The superfluous configuration documents should be removed, with care, from the database and the settings in the remaining configuration document should be reviewed for accuracy.)

# *Tracking Changed Field Contents*

## Tracking Changed Field Contents – Lotus Notes Client Accesses

Added in NotesTracker Version 3 was the ability to track, for the Lotus Notes Client, not just that a document had been updated but also what value each field in the document contained before and after the update -- let's call these the field "Before Images" and "After Images".

If the option “Track document Updates” is set to "Yes" then as shown in the following figure shows you can switch on the tracking of field content changes.

In turn, you can suppress the logging of the contents of unchanged fields (that is, where a field's After Image is the same as its Before image).

If you prefer, you can also force each field name to be converted to upper case in the Usage Log documents, which can help the field names to stand out from the field contents.

Additionally, you can provide a string of Prefix characters, which can be put to any use but might be handy for making the field names to stand out even more, or to put some sort of "eye catcher" in front of the field names (such as a Run Number, project identifier, or other meaningful text string). For example:

**A ... Field contents AFTER update**

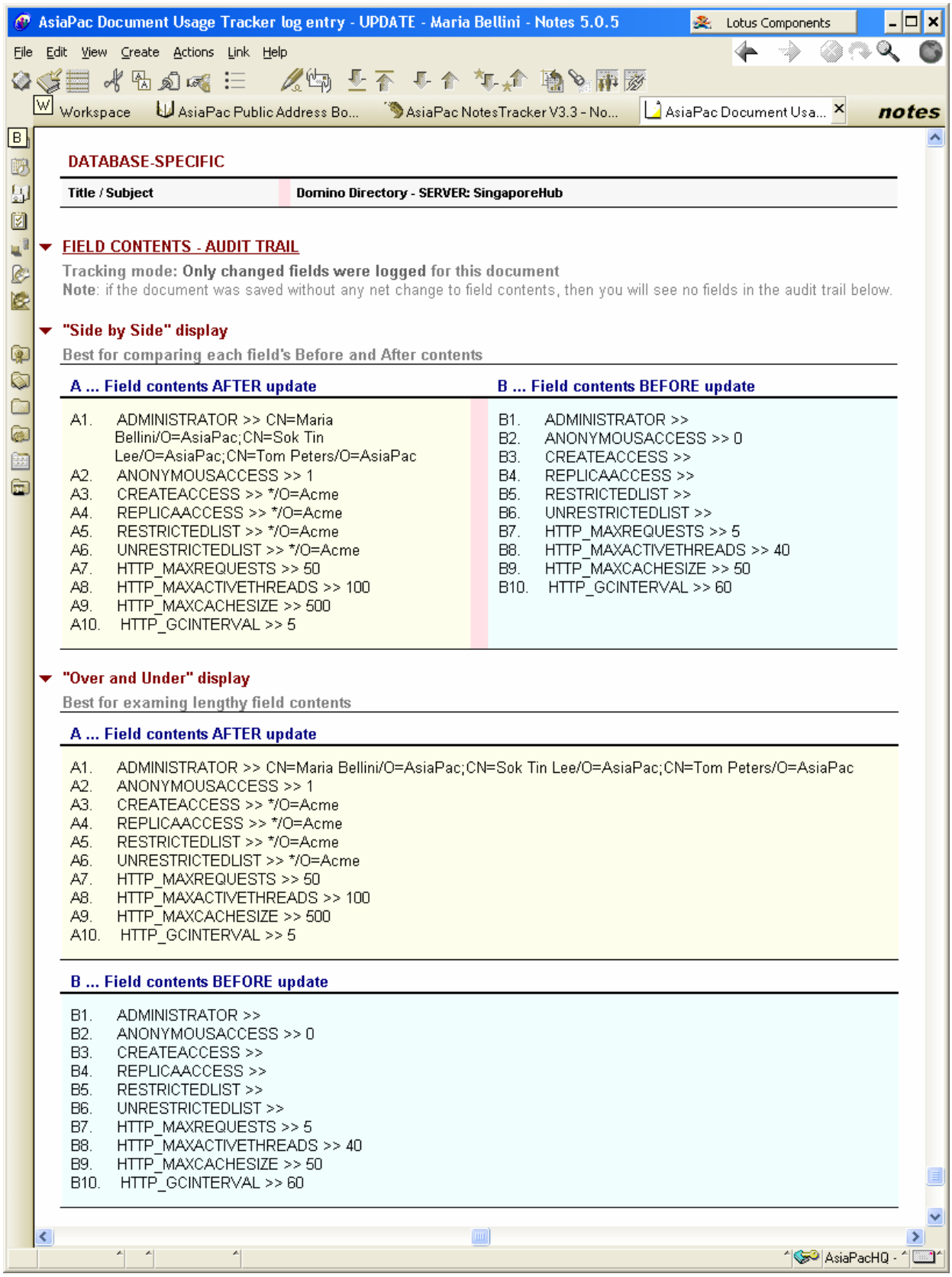
A1. COMPANYNAME >> Marietta's Pizza and Pasta
A2. TELEPHONE >> 5555 7777
A3. SALESPOTENTIAL >> 35

Finally, you also specify a Suffix string for the field name in the Usage Log document. This may not be a null character, but can be a single blank character. Another option is to specify *NL (standing for “New Line”) as the suffix, which displays the field name on one line and the field contents beginning on the next line, for example:

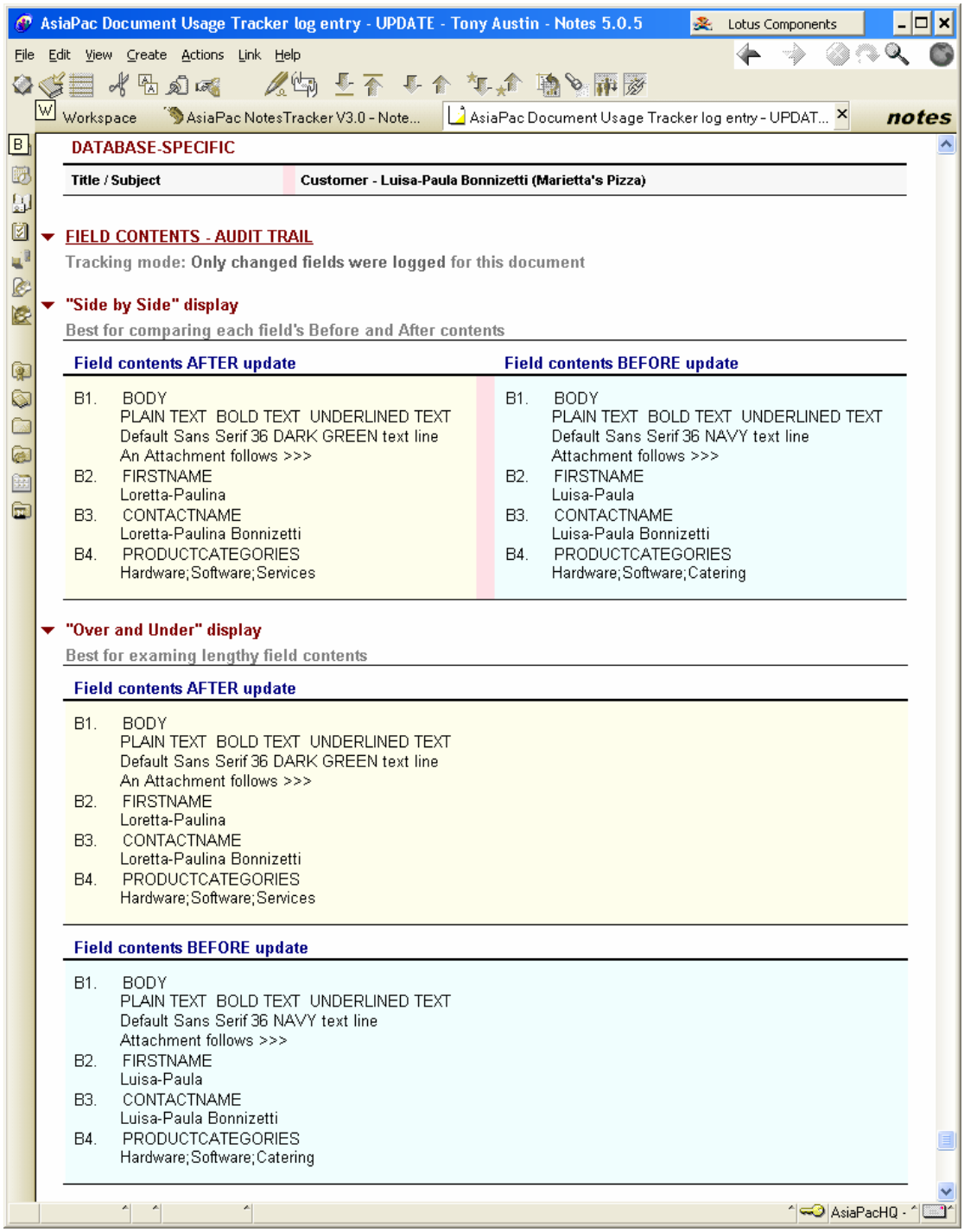
**A ... Field contents AFTER update**

A1. FIRSTNAME
Wilhelmina
A2. CONTACTNAME
Wilhelmina Jones
A3. TELEPHONE
1234 6789
A4. EMAIL
willie_jones@betterprinting.com.au
A5. PRODUCTCATEGORIES
Hardware;Software;Consulting

Let's see these settings at work on a larger scale. The default suffix is the chevron string " >> " (a space followed by two Greater Than symbols followed by another space). This will make the field contents stand out from the filed name, for example:



If you specify the special value of *NL (an asterisk followed by an uppercase N and an uppercase L), then a New Line character is inserted between the field name and the field contents. Using this value in conjunction with "Force all field names to UPPER CASE" can lead to better legibility in the Usage Log document. An example is:



## Tracking Changed Field Contents – for Web Browser accesses

The major new feature in NotesTracker Version 4.0 was the option to track accesses made to your Notes databases via a **web browser**.

You can separately specify whether you want tracking of **web browser Reads** and **web browser Saves**.

A web browser **Read** is logged every time that a web page is posted out from your Domino server to a browser user.

A web browser **Save** is logged every time that a web page is returned from the browser to your Domino server. Web saves include both the **Creation** and the **Update** of Notes documents. This is in distinct contrast with the Notes Client environment, where document Creates and Updates can be detected separately.

The Lotus Notes Client environment is "stateful" or "state-aware". A long-running session is established between the Domino server and the Notes Client. The session lasts from sign on to sign off, and the Domino server is able to associate each read/write operation with a specific Notes Client user session.

In contrast, the web browser environment is "stateless" -- the Domino server POSTs (sends) a web page to browsers, and GETs (receives) web pages back from browsers. There is no simple way to associate the field values in an outgoing web page with the field values that are received back some time later from the browser window.

A limited degree of field change tracking might have been possible via browser "cookies" (sometimes called "session cookies" because they are used to simulate session awareness that is missing from the HTTP protocol). However, this would have involved intricate programming for NotesTracker itself, and changes to many of your database forms) which was felt not to be justified for NotesTracker. More importantly, using cookies for such a purpose is not a reliable mechanism for tracking field changes, because an individual browser user can disallow the use of cookies (this being a personal decision, or maybe a corporate requirement).

Starting with NotesTracker Version 4.0, field “After Images” are logged every time that a user clicks the SUBMIT button in the browser window so sending the page back to the Domino server.

Starting with NotesTracker Version 4.1, field "Before images" are logged too. In distinct contrast to the Notes Client environment, the before images are obtained by retrieving their values from the so-called “back end” Notes document (that is, by retrieving the document from the database).

It is important to note that the Notes document so retrieved might gave been updated in the intervening period -- seconds, minutes or even hours – since the page was originally posted to the user’s browser window, (In the Notes Client environment, no matter how many times you save the document it is only when the document is finally closed that the field after images are recorded. This is possible because the session between the Notes Client and the Domino server is “stateful” or “state aware”.)

It is wise to keep **performance considerations** in mind at all times. This is particularly so for Internet accesses to your databases, which are outside your control and could well place very heavy loads on your Domino server. The great bulk of web browser accesses to your Notes databases almost certainly will be performed by non-authenticated users, who go under the user name **"Anonymous"**. You can significantly reduce Usage Log overheads (Domino server CPU load, memory and disk space) by specifying that you do not want anonymous user accesses s tracked -- although naturally this will reduce the subsequent analytical value of the usage logs. Or maybe you will decide (for any given Notes database) to retain the logging of anonymous accesses but only for Saves (Creates and Updates) and not for Reads.

The new section of the NotesTracker Configuration Document for Version 4.0 that gives you this degree of control is as follows:


**Note:** remember that these settings apply to individual Notes databases, giving you a fine degree of control on a database-by-database basis. This contrasts with use of the regular Domino Web Server Log database (DomLog.NSF) used by other web tracking solutions.

With NotesTracker you have the option of logging all of the field contents ("After images") associated with each new or updated Notes document, but DomLog.NSF does not record this level of detail.

It is our opinion that DomLog.NSF and NotesTracker provide complementary web tracking mechanisms. As far as we are aware, only NotesTracker provides a generic Notes Client activity tracking mechanism.

## *Limitations for the Logging of Rich Text Fields*

As the above illustration indicates, Rich Text fields (such as the commonly-occurring "Body" field) are not fully rendered by NotesTracker.

Only a simplified plain-text rendition of Rich Text field contents is written to the Usage Log document. Attachments are not captured by NotesTracker, and text styles (such as bold, underlined and italic) are not retained.

This is a deliberate NotesTracker design decision. It was felt to be unnecessary to retain the quite complex structure of Rich Text fields as being beyond the purposes and scope of NotesTracker.

Firstly, it was regarded as undesirable to copy attachments (or embedded objects) as this would dramatically increase the disk space requirements for the NotesTracker repository database.

Secondly, in many cases it anyhow will be apparent from the simple text rendition that the contents of a Rich Text field have changed.

## *Setting up a Meaningful "Title" Field for each Usage Log Entry*

This is one of those topics that are harder to explain than they are to work with!

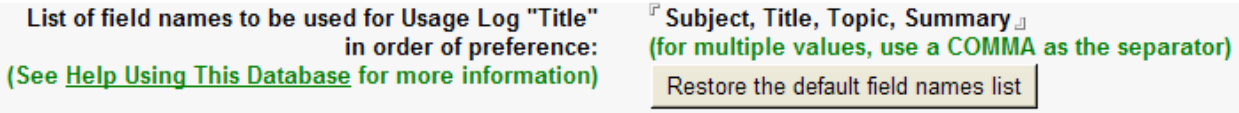
NotesTracker produces "usage log" documents, or entries -- one for each operation (Create, Read, Update, Delete, or view switch) in every one of your databases being tracked. These entries are your means of "usage metric analysis", and the entries are presented as various categorized views in the NotesTracker repository database.

Each row in a view has a particular column that displays a "title" for the document that was logged. The question arises: how do you ensure that there is a meaningful title to display for any given usage log document, so that when you traverse a view it is immediately fairly apparent to you (without your having to open the usage log document) which underlying database document it refers to?

For each individual database being tracked (and its replicas), a list of field names can be set up that will be searched for in the current document being logged and then used as the title of its Usage Log entry.

The default list of field names is: "Subject", "Title", "Topic" and "Summary"

You may alter this list of field names to suit each individual database application.




For example, you can alter the order in which the field names are listed. Or you can insert/append other field names (such as "Abstract" or "Heading") that are pertinent to the particular database. This gives a good degree of control over the Usage Log entry's title, without the need for any programming.

**Note**: the field names in each document in a database being tracked are searched for in the order specified in this list. The first occurrence in the current document of a field name from this list determines which field's contents is used as the usage log entry's "title" for that document.

Yo can reset the list of names to the default values by clicking the "Restore default field names" button.

## The "UsageTracking_Title" Special Field

Normally, as described just above, NotesTracker works its way down the list of field names ("Subject", "Title", "Topic" and "Summary" -- or whatever you supply), and looks in the document being tracked for a matching field name. Upon encountering a match, NotesTracker uses the contents of the matched field as the "title" for the Usage Log document.

But what do you do if a document being tracked has no such suitable title field (in its form design)?

The NotesTracker solution is for your Notes developer to add a field called "**UsageTracking_Title**" to the form design, and populate it with a character string that is appropriate for the document. The contents of the **UsageTracking_Title** field are used as the usage log entry's "title" for that document – taking precedence over the contents of any of the fields in the field names list.

There is a detailed discussion of this matter in the Developer Section below.

## *Suppressing the Logging of Document Links*

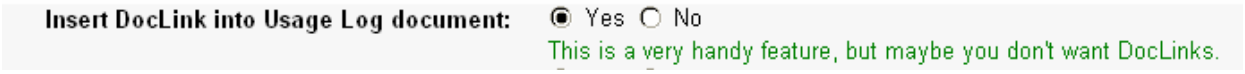
NotesTracker will store a Document Link -- usually abbreviated to “DocLink” -- in the Usage Log document that points to current Notes document for each Read, Create or Update.

**Note**: storing of a DocLink never occurs for document Deletes (no document remains for such a link to be made), and is not applicable for View Opens.

The DocLink makes it easy for you to open the original document when you are examining a particular Usage Log entry.

Also, the existence of a DocLink is essential if the special view labelled "41. What's New - Auto Doclink Launch view" is to work. This variant of the regular "What's New" view (view number 40) is designed to be embedded in a corporate portal page. It causes the originating Notes document to be opened automatically when you double-click on a row in the portal page’s What’s New or Breaking News view. This is a usability or convenience feature, a "one-step" approach. Rather than your having first to open a user-unfriendly Usage Log document -- which may not be of much interest to you -- and then having to click on the embedded DocLink so as to open the original Notes document, you go directly to the original Notes document

In previous versions of NotesTracker the DocLink was always stored in the Usage Log entry, but with NotesTracker Version 4.0 you can elect to not have the DocLink stored:




This maybe will save a little in the way of system resources -- processor time, disk space -- but probably not much, so it's probably wise not to select this option unless you're quite sure of the benefits and that you don't lose the convenience and portal page functionality described just above.

## *Suppressing the Logging of Changes to Particular Fields*

There are at least two good reasons why you should not -- or might not -- want to log the before/after contents of all the fields that are changed in a document:

- The contents may need to be suppressed for reasons of **security, confidentiality or privacy.**

  Examples of fields in this category are **password fields** (such as the field "HTTPPassword" in a Person document in the Domino Directory database), personal information (salary, etc), and such things as company strategic/sales data.

- There are probably many fields of little or no interest for logging purposes.

  One category is fields used internally by Notes, such as the so-called "dollar fields" like $UpdatedBy, $Ref, and $Revisions. As well, there will be many fields in your databases that are not of any significance for content tracking and KM (Knowledge Management) purposes -- hidden fields, work fields, computed-for-display fields, and the like.

An important enhancement in NotesTracker Version 4.2 is the ability to specify which field or fields in a database should not be tracked (that is, the logging of field content changes should be suppressed).

Since there may be dozens or even hundreds of fields in a document, to reduce the burden you can also use three types of **wild-card specifiers** for field name matching. Otherwise, you must list the field names explicitly.

You specify the field names as simple string (not surrounded by quote characters), and the names are not case-sensitive (since the names are all converted to upper case during comparisons). The search for a matching field name proceeds in the following order (stopping after the first match, if any):

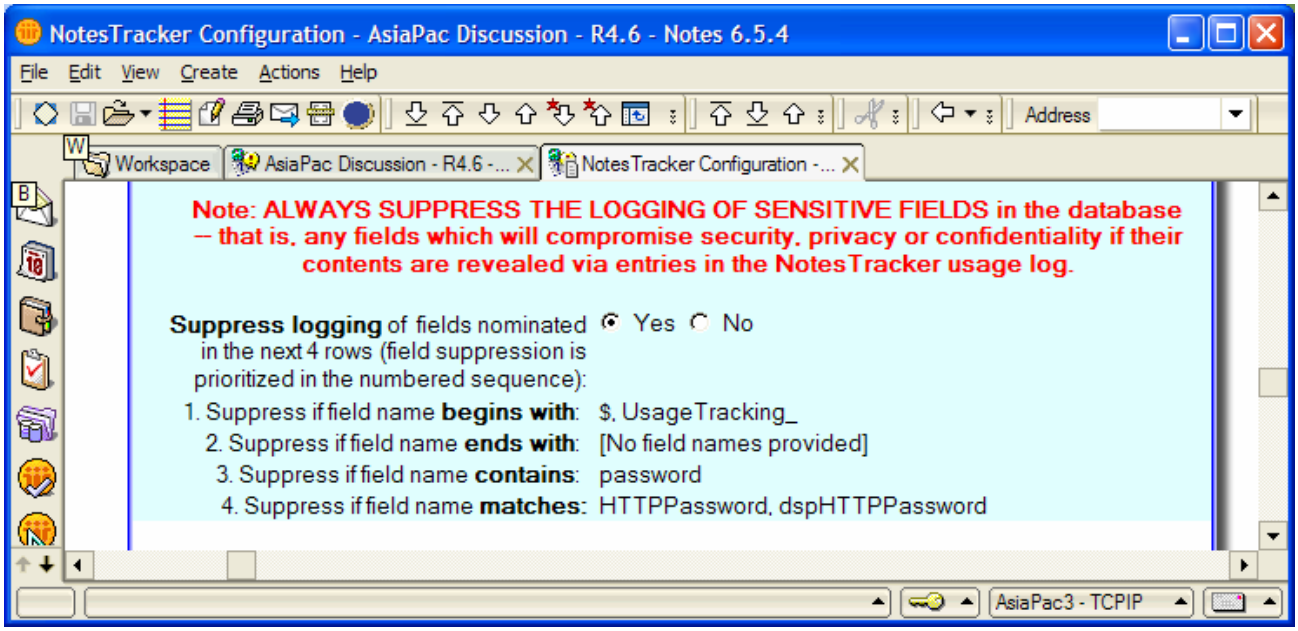
1. **Starting string** – such as **$**, **UsageTracking** or **UsageTracking_** (for fields generated by NotesTracker itself), **thread** (in, say, a discussion database), and **temp** or **work** (which you might use to signify temporary work fields).

2. **Ending string** – such as **disp** or **_disp** (which you might use as a convention for indicating computed-for-display fields).

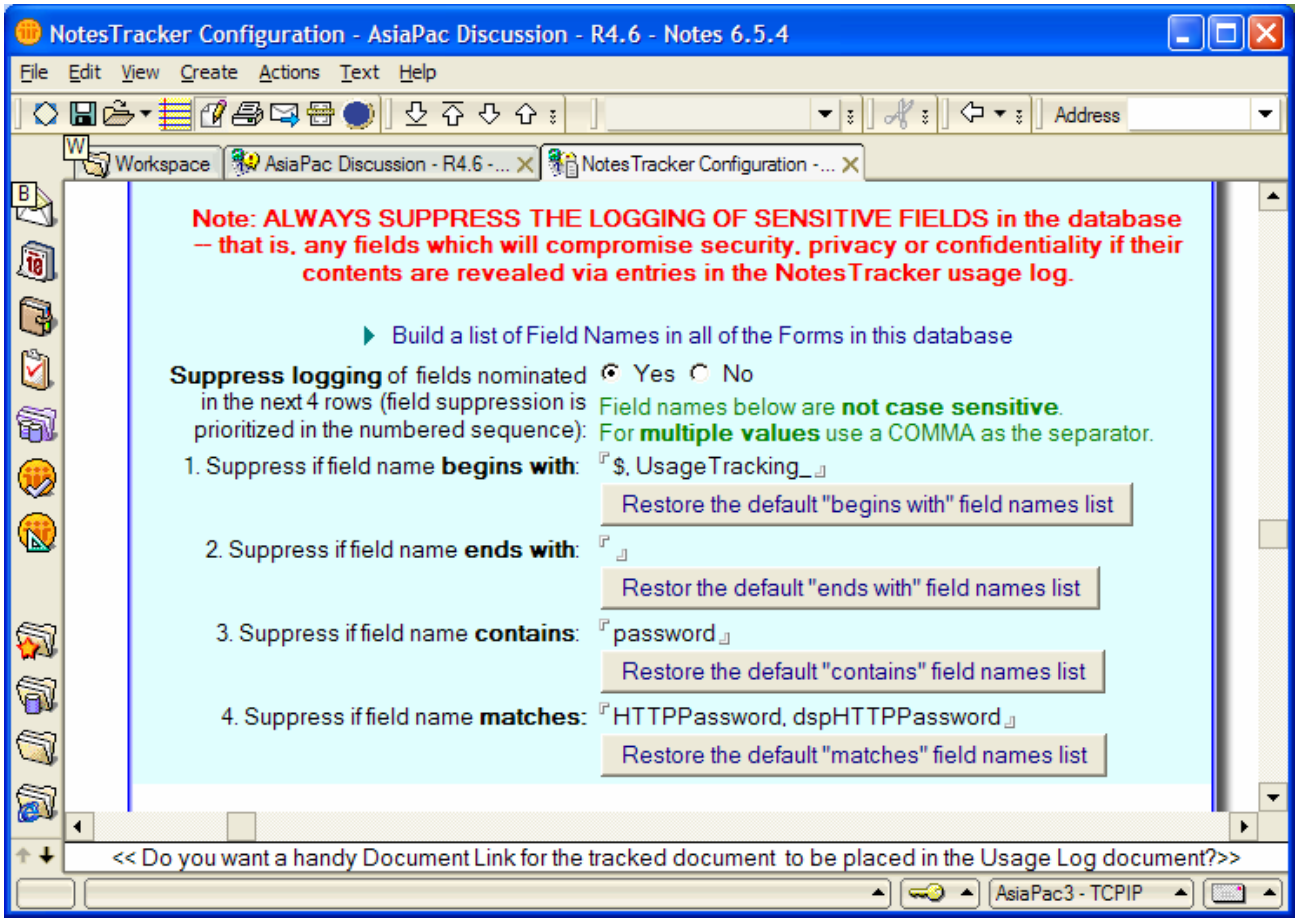
3. **Embedded string** – such as **password** or **salary** or **hidden** (a match occurs if such a string is found anywhere in the field name).

4. **Field names list** - Examples would be: **form**, **author**, **httppassword** or **HTTPPassword** (case makes no difference, except perhaps in legibility).

These strings are specified in the following part of the NotesTracker Configuration Document:



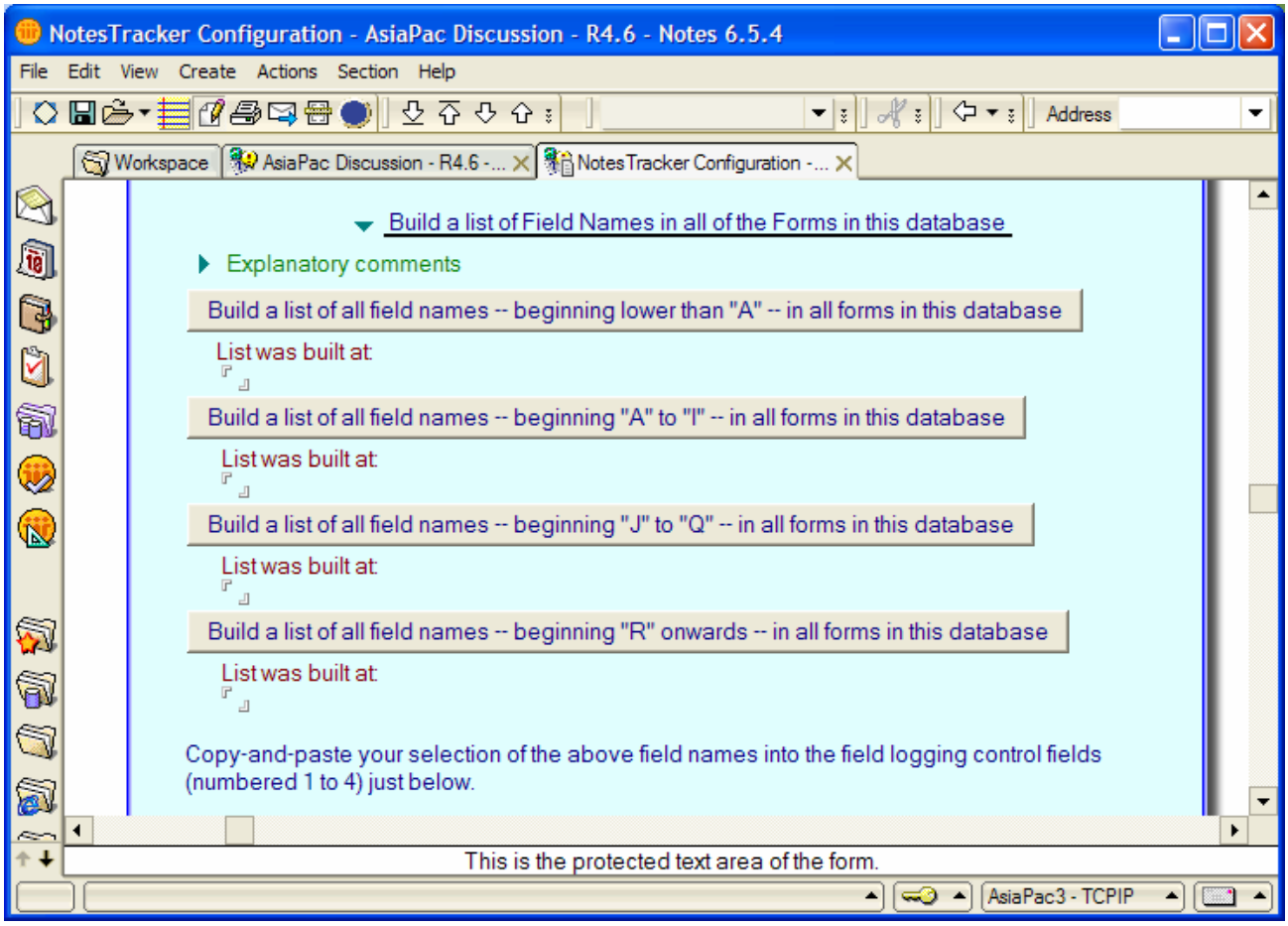

When you switch to edit mode, this section of the form looks like:




As the green Help text indicates, you enter multiple values using a comma as the separator character. The buttons provide a convenient way to restore default values or clear (erase) entered values. Your Notes developer might want to tailor the names affected by these buttons to better meet the requirements of a given database or your Notes installation as a whole. We will welcome your suggestions for commonly-used field names that we might use behind these buttons in future releases of NotesTracker.

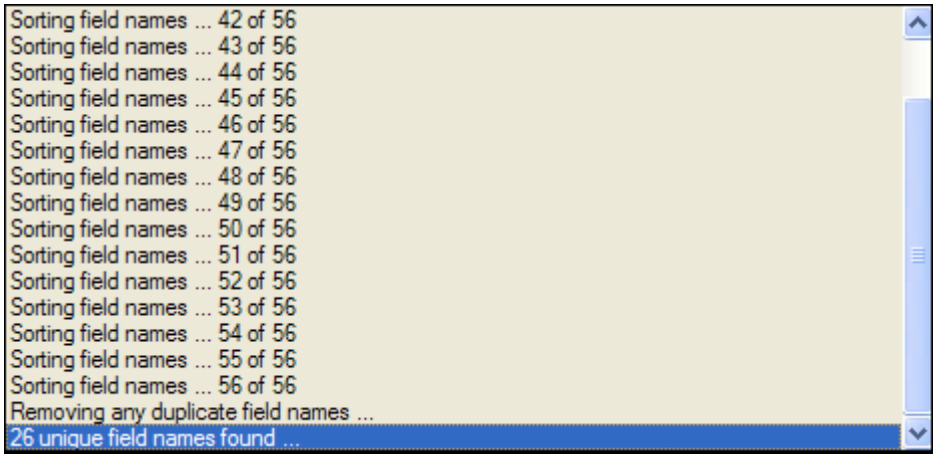
In NotesTracker Version 4.3 a button was added that will list the names of all the fields **in all of the forms** in the database. In NotesTracker Version 4.4 this single button was replaced with four buttons that break down the field name lists alphabetically into smaller groupings, in order to overcome a known Notes limitation that crops up when the database has many hundreds of fields.

Click on the twisty labelled "**Build a list of Field Names in all of the Forms in this database**" and the four buttons will be displayed, thus:

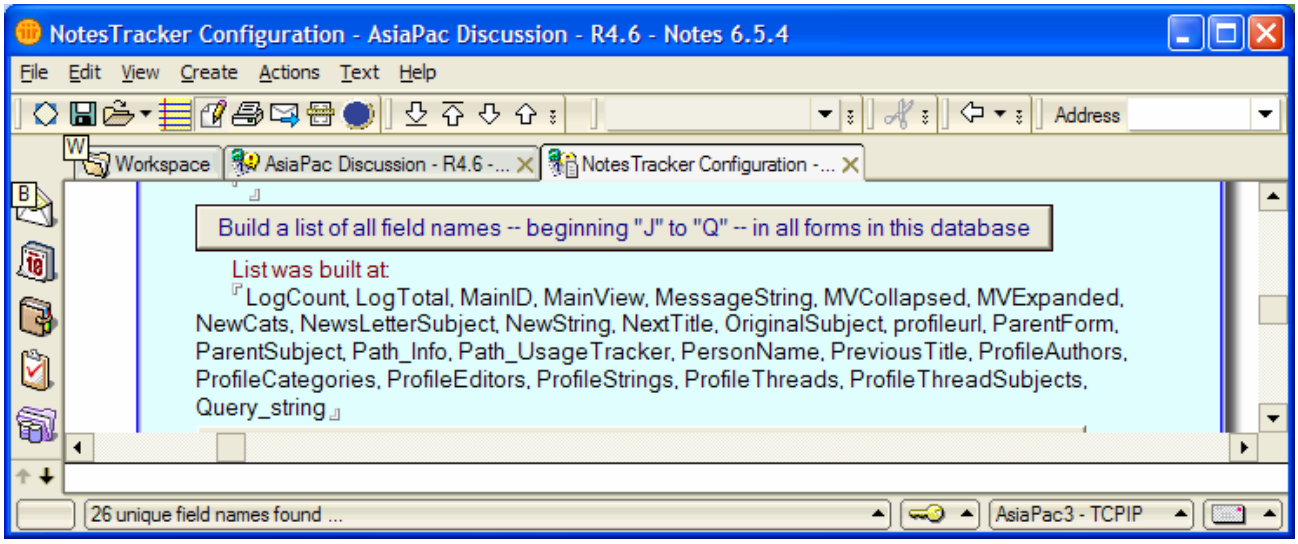



The first button is designed to display the names of "special fields" in the database, typically Notes reserved fields whose names begin with a dollar ($) sign.

The process usually takes a matter of seconds, but will take longer if there are lots of fields in the database. You can watch progress via the Notes status line, thus:



A sample Field Names list would look like this:




The date/time that each list was built is displayed. This is useful to know, in case you save these field names (there is no benefit in saving them, but you can if you wish to). When you later edit the NotesTracker Configuration Document again, this is a reminder that the Field Names list was computed earlier on and may no longer be accurate.

You can conveniently copy-and-paste desired Field Names from these lists into the **Suppress Logging** control fields (numbered 1 to 4) a few lines below them.

## *Setting Up Multiple NotesTracker Databases for Different Sets of Applications or Tracking Periods*

If you wish to track different groups or suites of Notes database applications separately, this is quite simple to do!

All that it's necessary to do is to create a separate non-replica copy of the NotesTracker Database for each such set or suite of application databases. Then you direct the Usage Tracking (logging) activity to the separate Usage Tracked Database copies merely by storing the appropriate Replica ID in the configuration document of each database being tracked -- just use a unique Replica ID for each unique set of application databases.

Similarly, you could use different replicas of the NotesTracker Database to track activity in separate time periods. For example, you could switch over to a different NotesTracker Database for each year, each quarter, or (if you have large amounts of activity) each month.

# *NotesTracker Database Size, Views, and Performance Considerations*

## The Build-up of Usage Log documents, and View Overheads

As a very rough approximation, the database size increases at 1.5KB to 2KB per log document. The growth rate needs to be monitored, and you should devise an appropriate archive-and-purge strategy if disk space is a worry. How frequently you purge log documents should primarily be determined by the length of time -- typically a number of months (or even years) -- for which you wish to retain usage metrics.

**Note:** in NotesTracker Version 4.2 an **archive agent** was added. This can be run on an as-required or scheduled basis, giving you the control you need over database size. This agent is discussed in the next section.

Keep in mind that the number of indexed views will have a major impact on database growth, rather than the relatively small amount of data stored in the log documents. To reduce Notes Client view opening overheads (and Domino server workload needed to maintain the view indexes), the number of sorted view columns has been kept reasonably low. However, you may wish to alter the view designs to decrease the number of sorted view columns even further, or to make other changes that balance view opening times against indexing overheads to your satisfaction.

## Web Browser Usage Tracking Performance Overheads

Web browser usage tracking was introduced in NotesTracker Version 4.0, and this brings some new performance matters to consider. With a web application -- for Domino or any other web server -- there is an extra processing burden placed on the web server by *each and every browser interaction* with the server. This applies to HTTP GET/POST operations, running of CGI programs on the server, running or Java servlets, and so on. You need to be cognisant of the fact that whenever you have NotesTracker web tracking active for a given database, this will add somewhat to the Domino server load, as will the additional NotesTracker Usage Log entries (which are ordinary Notes documents in the NotesTracker repository database). Such things should be kept in mind and become part of your Domino capacity planning and performance monitoring operations.

Some other related performance considerations are discussed several pages above, in the section on Web Browser usage tracking.

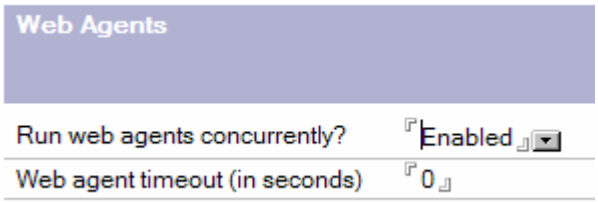
For some general background on Domino web server performance, refer to the Lotus product documentation and to IBM developerWorks articles such as:

- The Architecture of the Domino Web Server – Part 1
- The Architecture of the Domino Web Server – Part 2
- Introduction to Domino performance tuning

## Tip - Allowing Web Agents To Run Concurrently

Web agents run serially (consecutively, one at a time) unless you change the default setting, which can cause requests to queue up and might lead to poor response times

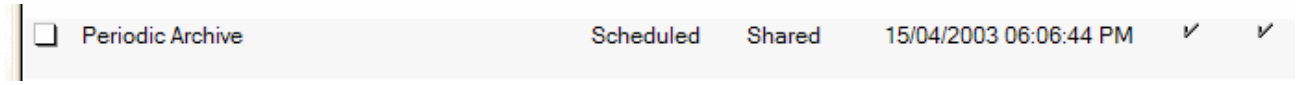
You can experiment with this by editing the **Internet Protocols** section of the Server Document:



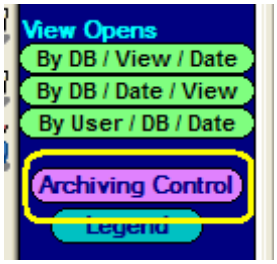

Allowing web agents to run concurrently might improve the user's perception of Domino responsiveness, but equally might lead to server workload problems. Monitor the effect (on both user response times and server workload) for a suitable period, and turn this feature back off if the net results are not satisfactory.

# *Managing the Usage Log – the NotesTracker Archiving Agent*

An important feature of NotesTracker (added in Version 4.2) is an archiving agent.




## Archiving View




Associated with the agent is a NotesTracker Archiving view, which is displayed by clicking the purple button at the bottom of the NotesTracker navigator (circled in yellow in the figure to the right).

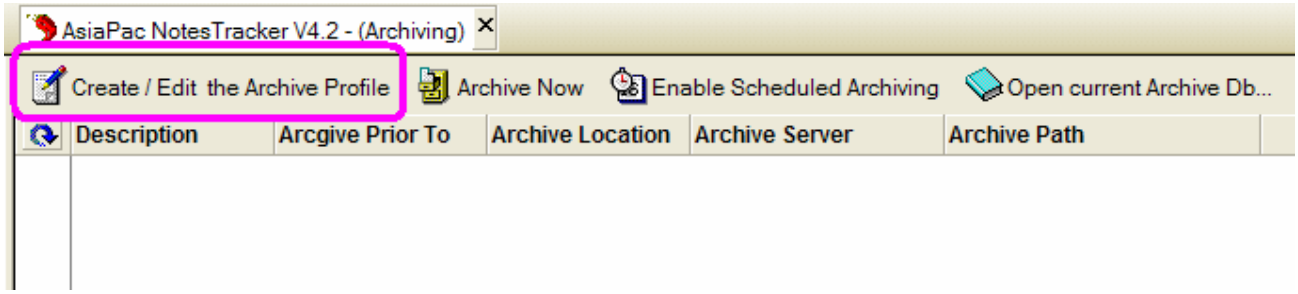
The agent is designed to work similarly to the archiving agent in your Notes Mail database.

The view is designed to hold a single Archive Profile document, and you must create and populate this profile before the agent will run.

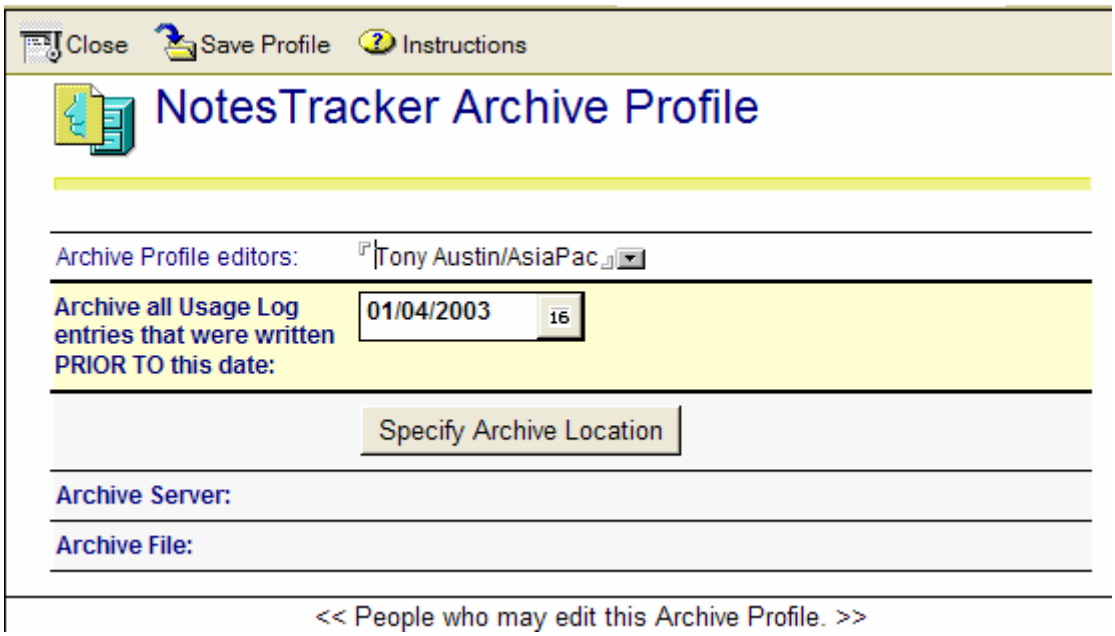
## Setting Up NotesTracker Archiving

**You** must have **Manager** or **Designer** access level rights to be able to view the action buttons described below for setting up and running the archiving agent.

When you initially click the purple Archiving button, the archiving view will be empty, like this:



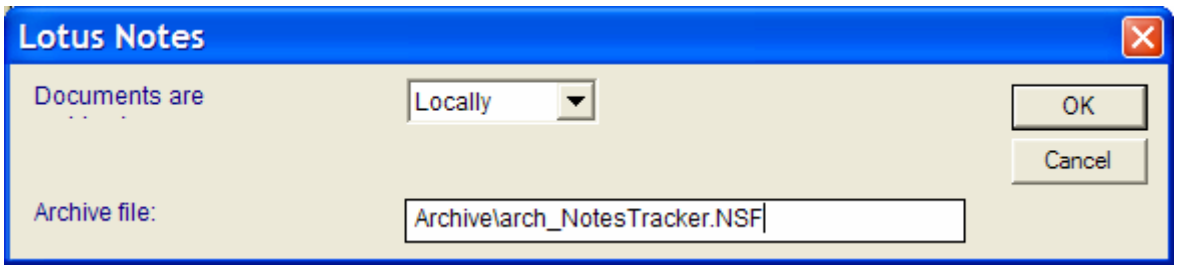

Click on the “Create / Edit the Archive Profile” action button (circled in purple) to set up the profile document. When being edited, the NotesTracker Archive Profile document will look like this:




You need to specify three things:

- The person(s) allowed to edit the NotesTracker Archive Profile document.
- The date **prior to which** any Usage Log documents in the NotesTracker database are to be archived. The default is the first day of the current month. It is suggested that your archive be made on a monthly basis, but this is entirely up to you.
- The **location** of the archive database. This is made up of a **server** component and a **file** component.

  You click on the **Specify Archive Location** button to provide the names of these components, and should see the following dialog box:




  Documents can be archived either “Locally” or on a Domino server.

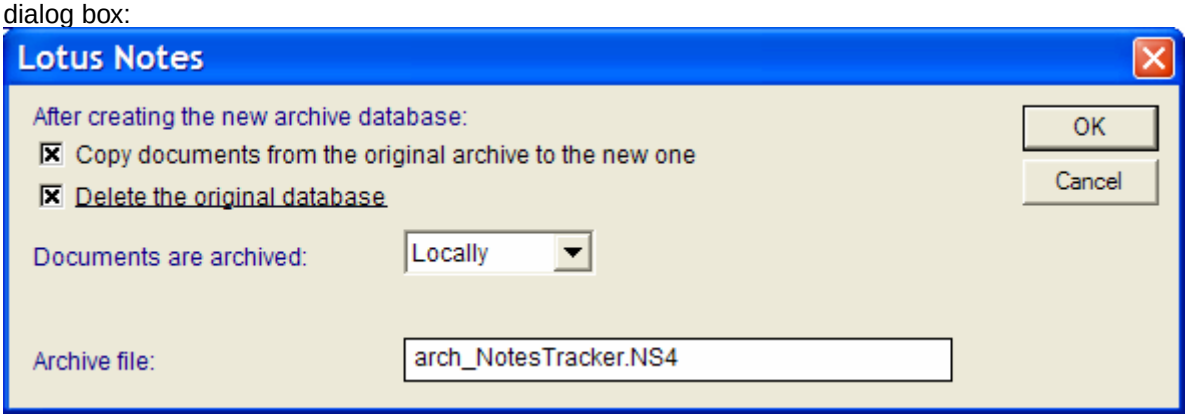
  The archive file name is the path to the Notes database which will receive the archived Usage Log documents. The file name can be preceded, in the normal fashion, by a directory path (in the proper fashion for the operating system, and relative to the Domino data directory).

  Examples are: **arch_NotesTracker.NSF** and **archive\2003\ arch_NotesTracker_June.NSF**

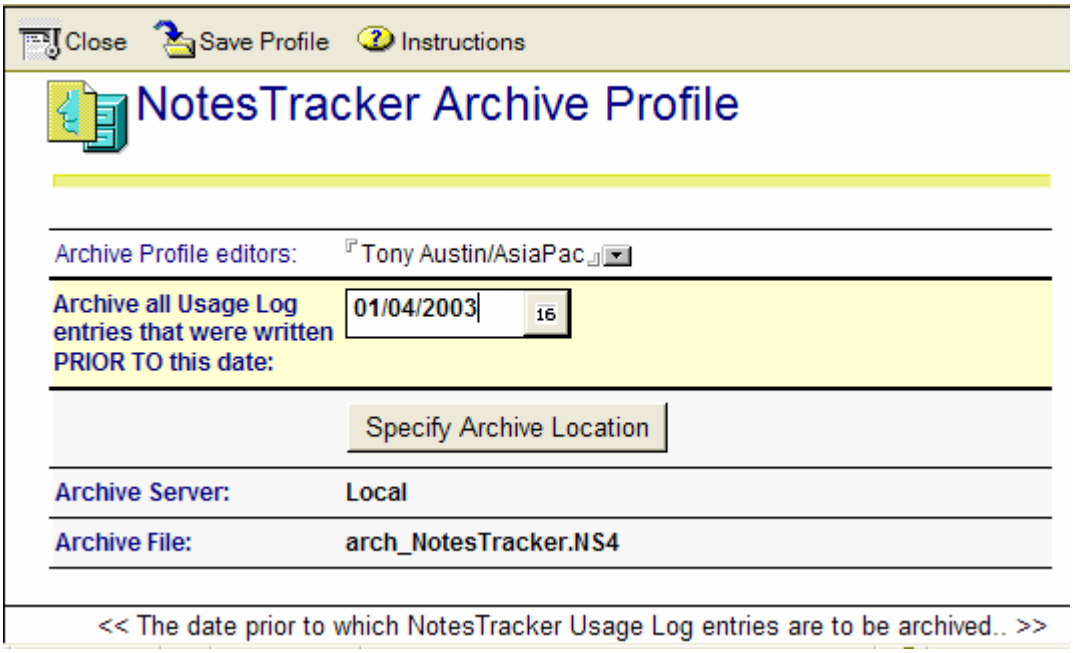
  **Note**: the archive database can be any database to which you have rights to add documents. However, it will make life easier for you if the archive database is a copy of the NotesTracker Database (which is the default), since this will contain all the regular NotesTracker views. This enables you to view the archived documents without having to add suitably designed views to the archive database.

  When you click the OK button, the archive database is created if it does not already exist. (This takes a few seconds.)

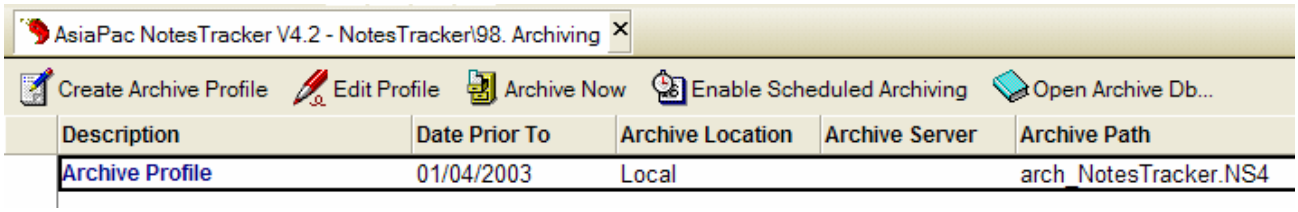
If you click the button after the archive database has been created, you will see a different form of the dialog box:



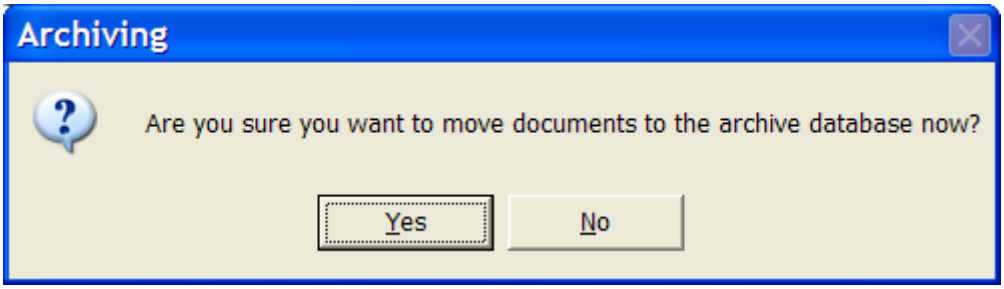
A completed Archive Profile looks something like this:




When you return to the NotesTracker Archiving view, you should see something like this:

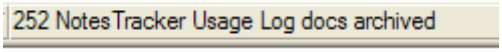



At this stage, you can click the **"Archive Now"** action bar button to cause the archiving agent to run immediately. You are asked for confirmation:

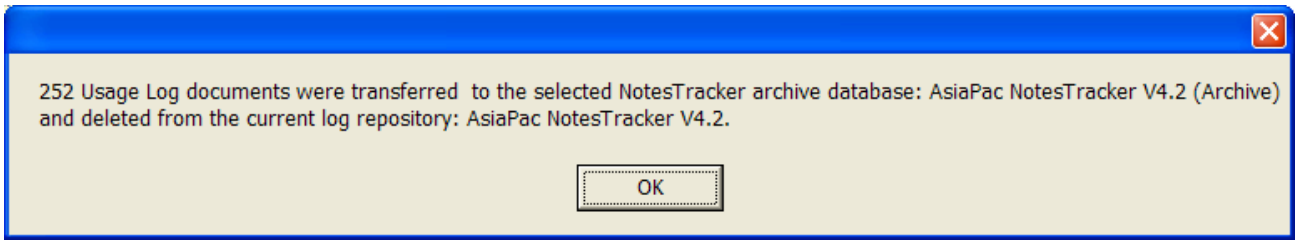



If you confirm by clicking the "Yes" button, then the agent runs on your Notes Client workstation.

Watch the status line to follow the agent's progress. This might only take a few seconds or a few minutes (depending, of course, on how many documents are to be archived), but you will see something like:



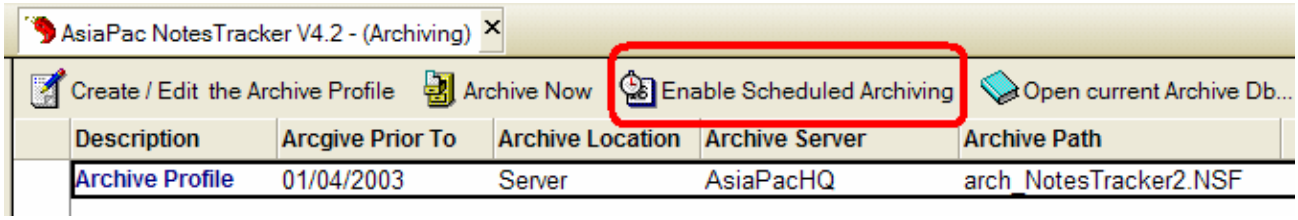

Upon successful completion you will see a dialog box like the following:



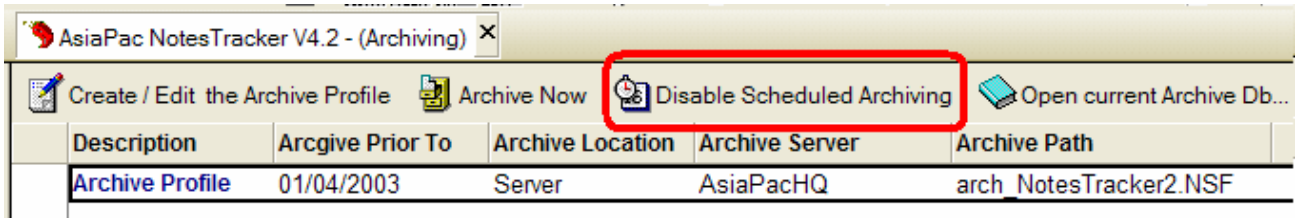

Alternatively, you can **schedule the Archiving Agent to run later**. You do so using normal techniques for setting up the agent's schedule (daily, weekly, monthly, etc) and for activating the agent... It is beyond the scope of this NotesTracker Guide to explain this any further.

If you don't run the agent immediately (via the "Archive Now" button), then you can schedule it to run automatically at some designated periodic interval or date/time.

Click on the "Enable Scheduled Archiving" or "Disable Scheduled Archiving" button (circled in red) to enable or disable, respectively, the schedule for the Archiving Agent. (You set the schedule period and date/time in the normal fashion, as described in the Lotus documentation.)




The above is what you see if the Archive Agent is not scheduled. Once the agent has been scheduled, the view will look like this:




## Factors Determining the Frequency of Archiving

How often you run the Archiving Agent is up to you, and will depend on such factors as:

- How long you wish to keep NotesTracker Usage Log documents in the repository database so as to analyse usage patterns and trends, etc;
- What restrictions you have on disk storage space on your Domino server(s); and
- How rapidly Usage Log entries are being added to the repository database.

Such factors will vary from organisation to organisation. from application to application, and from time to time (there will be peaks and troughs in usage). This means that the frequency of archiving for any given usage tracking environment might be days, weeks or months.

- Click on the "Open current Archive Db" button ti open the database specified in the Archive Profile document.

## Viewing the Archived Usage Log Documents

To view archived Usage Log documents, as a convenience you can click on the “Open current Archive Db” button to open the archive database that is specified in the Archive Profile document.

# Developer Topics

## *Introduction to the NotesTracker Developer Toolkit*

The AsiaPac NotesTracker (Domino Document Usage Tracker) comes in the form of a developer toolkit (or SDK - Software Development Kit).

It was designed to track activity performed through the Notes Client UI (User Interface, also called the "front end"). That is, it tracks interactions between a real person and the Notes Client as she or he performs normal activities, such as editing a Notes document, switching from one view to another, or deleting a Notes document.

In its distributed form, it is NOT set up catch actions performed via such things as button click events and "back end" agents. However there is the opportunity for you to adapt the NotesTracker LotusScript code so that it works when a button is clicked, in agents, and so on. (If you don't have the available developer skills or resources to perform such adaptations, Asia/Pacific Computer Services could provide offsite services to do this for you.)

## NotesTracker Design Philosophy

The design of Lotus Notes applications can vary tremendously from database to database, therefore it is not possible to provide a "shrink-wrapped" usage tracking solution to accommodate all users' requirements that requires no programming.

In the design, development and testing of NotesTracker every effort has been made to keep down to a **simple, repeatable procedure** the process of implementing usage tracking in your Notes databases.

Our aim was to minimise developer workload and enable rapid, dependable development and deployment. Therefore, the design was kept as "neutral" as possible, so that you can integrate the NotesTracker code into your existing Notes database designs with a minimum of fuss.

## General Security and Privacy Considerations for Document Tracking

**IMPORTANT**

**Security & Privacy Considerations for Administrators/Developers**

When implementing usage logging in your databases, **be very careful not to allow confidential or sensitive information to be logged**. Logged information might not be appropriate for general consumption.

Even the document titles may give clues to confidential, sensitive or personal information, Consider the disastrous implications of just the mere mention in a Usage Log document's title of such things as a proposed merger or acquisition, legal action, an employee's possible termination, or many other such matters!

**You should thoroughly test the logging activity before deployment to ensure that confidentiality and privacy are maintained appropriately.**

**It may even be that certain databases, or at least aspects of them, should not be tracked.**

**Developers and administrators must always keep this in mind.**

You will find a sample form design that, as provided or with rewording that your organisation deems appropriate, might be suitable for a "privacy disclosure" that can be displayed to users. (It was designed to be displayed automatically, or when the presses a button, by the execution of a document "Compose" action. Note that it contains a SaveOptions field that is set to zero, so that the displayed document can never be saved - only displayed.)

## *Step-by-Step: Adapting a Notes Database's Design for Usage Tracking*

You only need to follow a few simple steps, for each database that is to be tracked:

**REMEMBER: With just a little practice, it should take you only minutes or seconds per form or view to implement usage tracking in each of your databases!**

## **** THE PLANNING STAGE ****

Firstly, it is important that you meet with the executive sponsor, knowledge manager or database owner for each database and draw up a **list of the forms and views within the database** that are to be tracked (and the "document title" to be stored for each document that is tracked, as explained elsewhere). You may choose to track all forms and views in the database, but usually not all of them are worth tracking. For example, some forms and views may be for presenting trivial content that is not at all important for "knowledge metrics", you may know in advance that they will be rarely used, or they might be hidden forms and views that are never apparent at the user interface and so do not merit being modified for usage tracking.

## **** GAINING FAMILIARITY ****

It's an extremely good idea to start by selecting one or two test databases that have very simple designs – just a few forms and views. Gain experience in following the numbered steps below, before attempting to implement NotesTracker in your other databases that have complex designs (with large numbers of forms and views and/or complex structures , say, like the Lotus Domino Directory and the Notes Mail template). Experiment in the simple databases with the various NotesTracker Configuration Documents settings, and test that usage logging occurs properly in both the Notes Client and the web browser environments.

## **** DESIGN MODIFICATION - THE GOLDEN RULE ****

**Users should not notice any difference in the apparent behaviour of any application after NotesTracker functionality has been incorporated!**

How often have you seen software "enhancements" or "fixes" that produce noticeably different behaviour in an application? Features that used to work one way now work differently, or they are "broken" in the new version. New and unexpected advisory messages and/or warnings appear to disturb the end user. The application might become unfriendly or even unusable, to the user's way of thinking. We don't want any such thing happening when we add usage tracking to our Notes databases, do we?

The code in NotesTracker has been crafted so that any error situations that arise -- such as the NotesTracker Repository database being inaccessible or a database's NotesTracker Configuration Document not being present -- are not obvious or apparent to end users. NotesTracker's error-handling does not send out any "in-your-face" dialog boxes, but deliberately skips any further usage tracking code. (Subtle warning messages are displayed on the Status Line at the bottom of the Notes Client window, In the Internet environment, no messages at all are posted to the browser window.) The end users should be blithely unaware that NotesTracker has encountered problems, and should notice no difference at all in the way that the database application behaves. Only you as the responsible Notes administrator/developer need to be aware that usage tracking is not operating as expected.

Let's now carefully go through the simple design modification steps that you need to follow in order to add usage tracking to a database. Some steps are mandatory while others are needed only to achieve optional effects (such as the tracking of View Opens in a Notes Client environment).

## **** STEP 1A ****
(Once per database)

Copy the **CGI Variables Subform** from the NotesTracker Database into the database being tracked.

| Name/Comment | Alias |
|---|---|
| CGI Variables<br>AsiaPac NotesTracker | CGI |

**Note:** the CGI Variables Subform is built into the Usage Tracker Subform described in the next step, so this step **must** be carried out before the Usage Tracker Subform is copied in

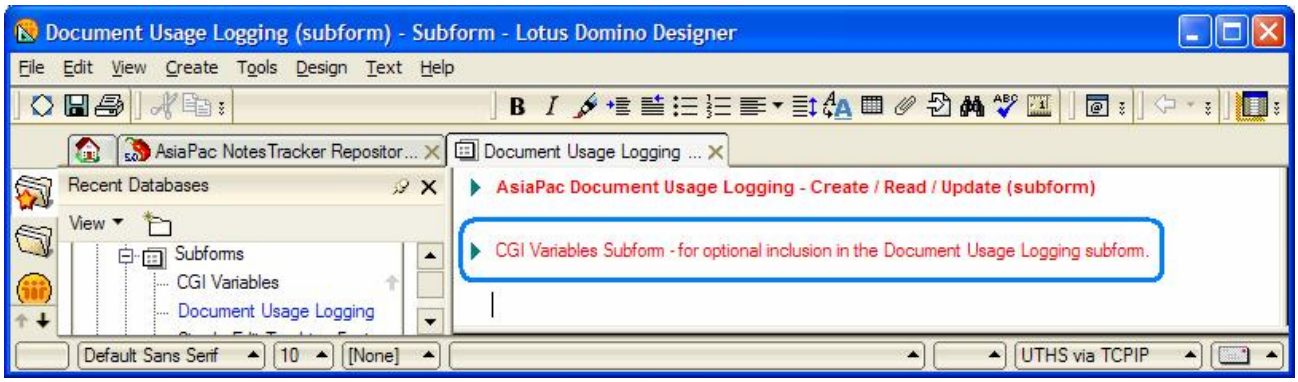
## **** STEP 1B ****
(Once per database)

Copy the **Document Usage Tracking subform** from the NotesTracker Repository database into the database being tracked:

| Name/Comment | Alias |
|---|---|
| Document Usage Logging (subform)<br>AsiaPac NotesTracker | |

**Note:** In Step 1A, you should copy the CGI Variables subform by itself. Experience has shown that if you copy this subform together with the Document Usage Tracking subform, it can cause the CGI Variables subform not to be included in the Document Usage Tracking subform.

Be sure to check that the Document Usage Tracking subform includes the CGI Variables subform, otherwise tracking of CGI variables (for Web browser accesses to the database) will not succeed. The Document Usage Tracking subform should finish up looking like this (the CGI Variables subform is circled in blue in the illustration):




Strictly speaking, the CGI Variables subform is only needed if you select the logging of CGI variables, but it is probably simpler to *always* include the CGI Variables subform in the design.

## **** STEP 1C ****
(Once per form that is used for documents being tracked)

Insert the Usage Tracker Subform **into each form** that will be used to create, read or edit any document that you want to track.

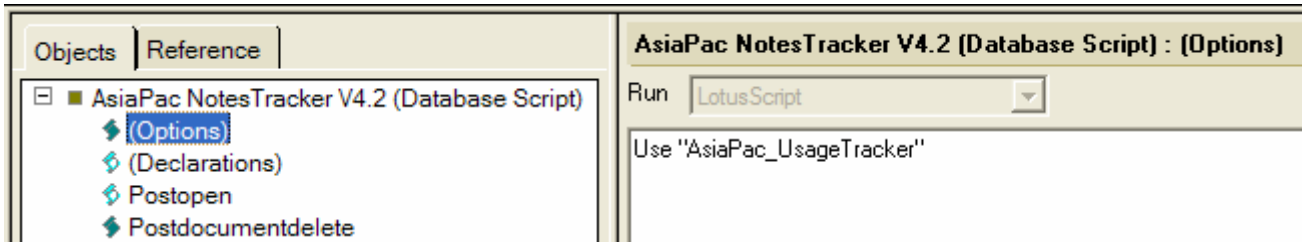
## **** STEP 2A ****
(Once per database)

Copy the script library **AsiaPac_UsageTracker** from the NotesTracker Repository into the database being tracked.


## **** STEP 2B ****
(Once per database)

Then copy-and-paste, into the (Options) for the Database Script of the database being tracked, the following statement:

**Use "AsiaPac_UsageTracker"**




**POSSIBLE TRAP**: If you copy-and-paste the above statement ( **Use "AsiaPac_UsageTracker" )** from this guide into the Database Script, it is possible to get unexpected error messages when you try to save the Database Script, such as:

Not a constant: "ASIAPAC_USAGETRACKER"

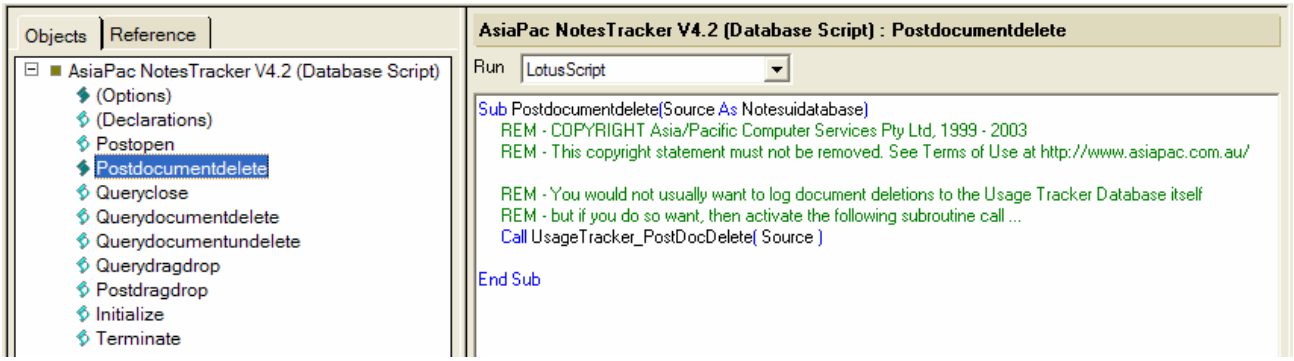
This can happen if there are invalid characters pasted into the statement. The statement appears valid, but in fact contains character(s) that prevent the script from compiling, and it will not be at all obvious why not!

**TIP:** directly type this statement into the Database Script (rather than pasting it in from this guide).

## **** STEP 3 ****

(Only if document deletions are to be tracked)

If you want to **track document deletions** go to the Database Script of the NotesTracker Database, open the Postdocumentdelete subroutine, then copy-and-paste its entire contents into the **Postdocumentdelete** subroutine of the database being tracked:




Be sure that the following Call statement in the subroutine is not commented out, otherwise document deletions will not be tracked:

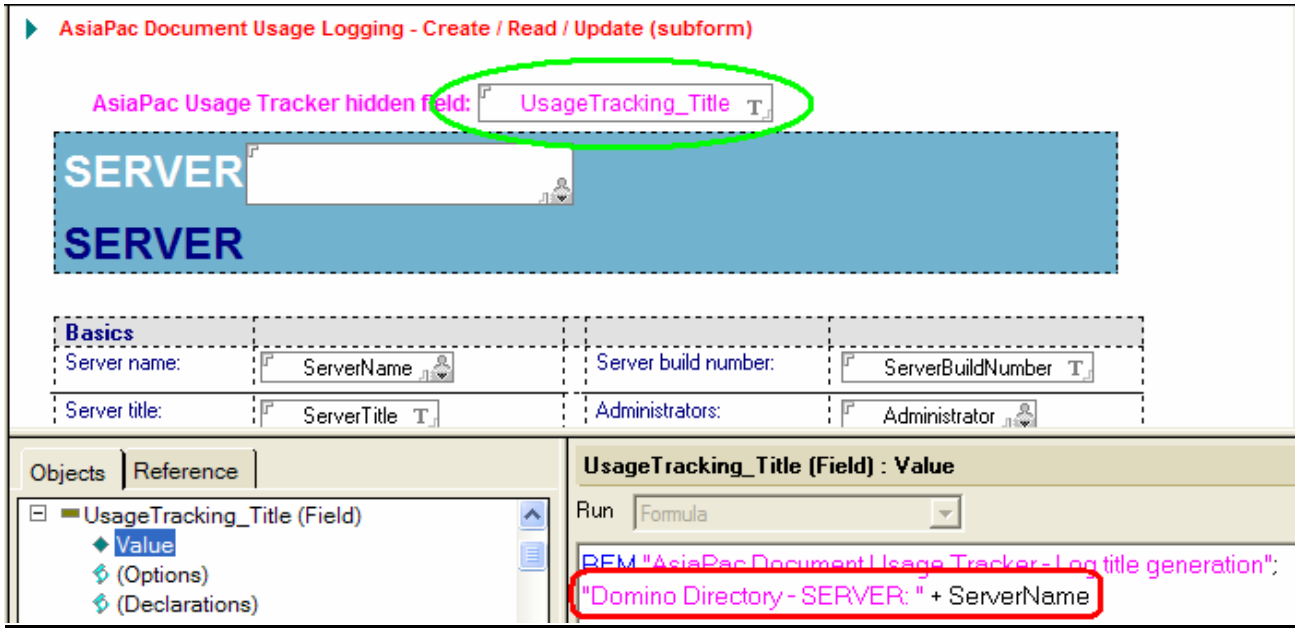
**Call UsageTracker_PostDocDelete( Source )**

**Note:** Do not carry out step 3 prior to step 2, since step 3 depends on the prior existence of the script library subroutine **UsageTracker_PostDocDelete.**

## **** STEP 4 ****

(Might be required in all, some or none of the database's forms)

**Setting the Title Field Value in the Usage Log Document**




For each form, determine if there is already a field called "**Title**" which contains a suitable character string that adequately identifies each document. (This string appears in the various views of the NotesTracker Repository database.) If there is no such string, or in the fairly common situation that the "Title" field holds a personal salutation (such as "Mr", "Mrs", "Ms", "Dr", "Prof") -- which would be quite meaningless in the NotesTracker Repository views -- then add a **hidden computed-for-display field** called "**UsageTracking_Title**" and populate that field with a suitable string value. For example, you might move the contents of a field called "Subject" or "Topic" into this hidden field, or perhaps set its value via some appropriate formula (say, by concatenating a "Category" field with the date). It's entirely up to you the developer, in conjunction with the database's knowledge manager or sponsor/owner, to determine how best to populate this field with a meaningful text string. Examples of this field are included in the various sample databases distributed with NotesTracker.

**Note:** in NotesTracker versions prior to Version 3 the special computed-for-display field was named "UsageTracker_Title" and was renamed "UsageTracking_Title" for consistency with other associated NotesTracker fields. Therefore you would have to make simple adjustments to some of your forms to accommodate this field name change when upgrading to NotesTracker Version 3 from earlier versions.

## **** STEP 5A ****
(Once per database)

Add the **NotesTracker configuration view** to each database being tracked.

| Name/Comment | Alias |
|---|---|
| NotesTracker\NotesTracker Configuration view | vwNotesTrackerConfig |

## **** STEP 5B ****
(Once per database)

Add the **NotesTracker Configuration form** to each database being tracked.

| Name/Comment | Alias |
|---|---|
| AsiaPac NotesTracker Configuration | UsageTrackerConfig |

## **** STEP 5C ****
(Once per database)

Create a **NotesTracker Configuration** document, and edit its tracking control settings (as explained in detail in the Database Administration Topics section).

Create a Configuration Document

| Usage Tracking Status for this DB | Method used to Locate NotesTracker usage log repository | Track LOCAL Notes actions | Track Notes Client Reads | Track Notes Client Creates | Track Notes Client Updates | Track Web Browser Reads | Tr W Bro Sa |
|---|---|---|---|---|---|---|---|
| On | By Server and Path ...<br>SERVER: (Local)<br>PATH: NotesTracker.NS4 | Yes | Yes | Yes | Yes<br>CHANGED FIELDS ONLY | Yes | Y |

**Note:** Each database that is being tracked must contain **a single NotesTracker Configuration document** (as also explained in the Database Administration Topics section). You will see by examining the early part of the Queryclose event that the NotesTracker code ignores any but the first configuration document in a database. (Occasionally more than one configuration document can be present in a database due, say, to replication errors.)

The hidden field called "**Ensure_Config_Uniqueness**" is intended to prevent multiple configuration documents from being saved through the front end (Notes client user interface). However, as pointed out in the previous section, multiple documents in practice occasionally can arise due to replication/save errors or other indeterminate causes, and it important to keep an eye out for any such superfluous documents and to manually delete them. The NotesTracker code checks only the first configuration document found in the configuration view, and any such superfluous configuration document might be accessed instead of the intended one.

## **** STEP 6 ****
(Only if desired)

This step is only required if you decide to implement "self-contained" or "internal" usage tracking, introduced earlier in the Overview section and the Administration Topics section.

In order for Usage Log documents to be written "internally" – directly into an application database, rather to an external NotesTracker Repository database -- several additional matters have to be considered. At the very least:

- You must add one or more of the views supplied with the NotesTracker Repository Database (or your own variations of such views) into your application database. Otherwise the Usage Log documents will just build up inside your database and there will be no way for you to see them so as to analyse document usage.

- You must **check the View Selection Formula of each and every existing view** in the application database, checking and (if necessary) modifying the View Selection formula to ensure that the Usage Log documents are filtered out of these existing views.

- The internally-stored Usage Log documents will build up over time. The NotesTracker Database contains a **Usage Log Archiving** capability, which you can reproduce in your own database. Refer to the operational considerations for archiving in the Administrator Topics section of this guide.

  You should copy-and-paste into your database (from the NotesTracker Repository) the agent named "**Archive Selected Documents**". You will also need to copy the **NotesTracker Archive Profile** and **Archive Log** forms, and set up a hidden **Archiving** view in your database, which also may be copied from the NotesTracker Database.

  Examine this agent, make any modifications needed to suit your database, and schedule it as you see fit in order to periodically archive old Usage Log documents.

These additional steps are not onerous, but can be a little time consuming for databases with many views. They must be carried out **carefully** so as not to disturb the pre-existing behaviour and "user experience" of the database.

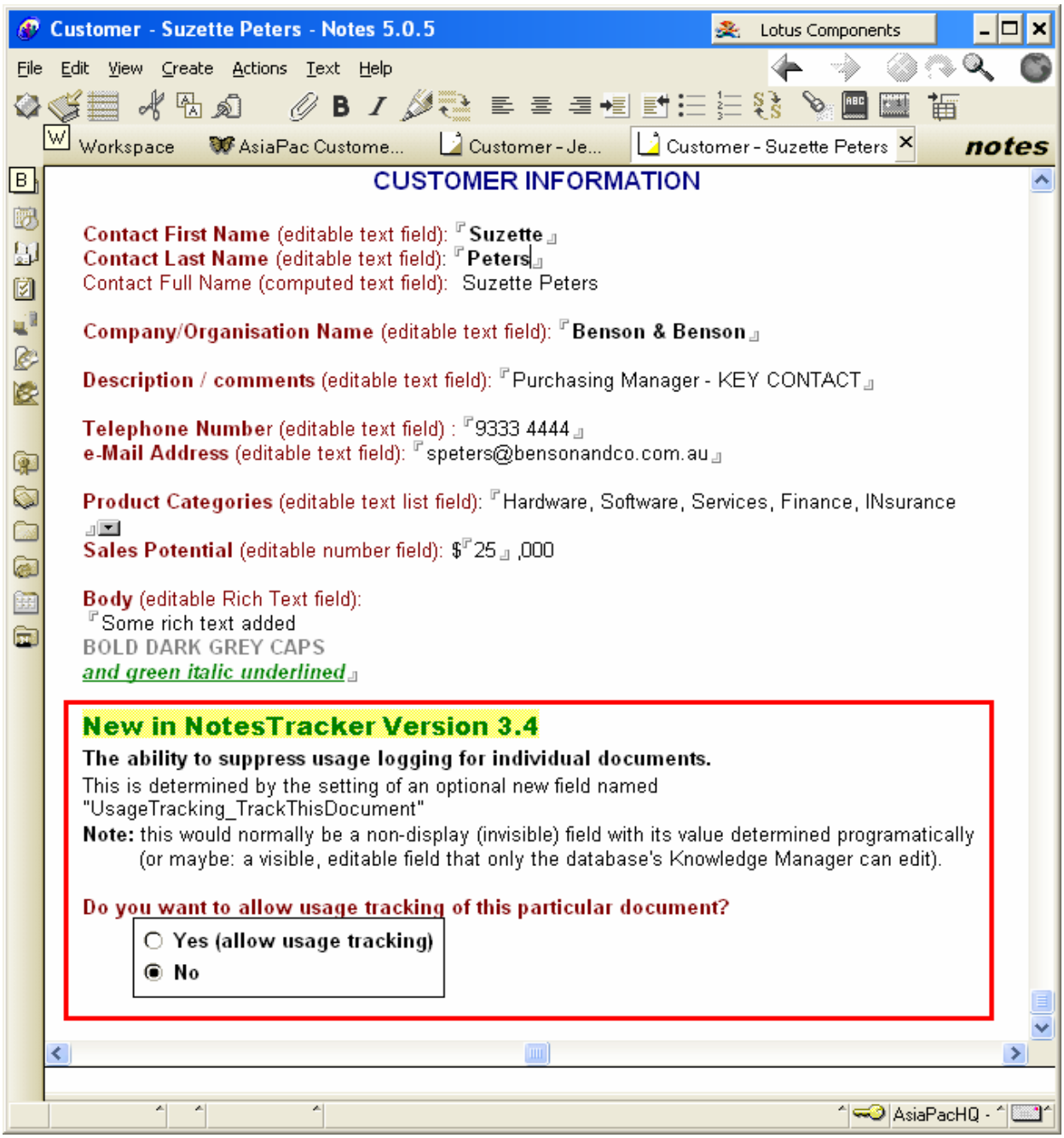
## **** STEP 7 ****
(Only if desired)

**Suppressing Usage Logging at the Individual Document Level**

What if you want to ignore or suppress the tracking of certain documents? These might be documents that are of no interest or value for tracking purposes, that would just ad ”noise” to your usage metrics. Or they might be documents that contain sensitive information, such as personnel data, salary data, or the like.

To do this, the Notes developer merely adds a text field called "**UsageTracking_TrackThisDocument**" to the pertinent form or forms in the database, and must populate the field with the string value “No” (or “NO”) if the document is not to be tracked. The NotesTracker Queryclose event checks for the existence of this field in the document. If this field exists, and if it contains the value "No" (or "NO"), then usage tracking is not performed for that particular document.

The example customer database (named "AsiaPac Customer - NotesTracker") has this field implemented as a Radio Button editable field so as to allow you to test this function by manually switching on/off the “No” value. Usually the field would be populated programmatically (as described in the preceding paragraph.)



As long as the value of this field contains (at Queryclose time) the value "No" or "NO" then the Queryclose event is immediately exited and a Usage Log document is not created. The test is carried out at the very beginning of the Queryclose event, in order to enable a quick exit keep usage tracking overhead to the bare minimum. **Note:** if the field's value is "Yes" (or anything else other than "No", including blank or null, or the field doesn't exist in the document), then the usage tracking routine continues to the next stage.

In practice, the field most likely will be implemented as a **non-display computed field**, and the developer will programmatically set its value according to some condition(s) based on other fields in the document, or according to the user's identity (such as being a member of a certain group such as the Knowledge Managers group), or some such thing.

# **** STEP 8 ****

## (Once per database - only if Web tracking is desired)

**To enable Web Browser Usage Tracking**

**Note:** this step is only required in a given Notes database if you wish to track Web browser reads/writes to that database.

There are several parts to this step:

A. Copy the two NotesTracker Web agents named **NotesTrackerWebQueryOpen** and **NotesTrackerWebQuerySave** from the NotesTracker Database into the database being tracked.

B. In each form that you wish to be tracked …

Go to the Webqueryopen event:
@Command([ToolsRunMacro]; "<Your agent goes here>")

Replace the string "**<Your agent goes here>**"
with "**(NotesTrackerWebQueryOpen)**"

Be sure to include the surrounding parentheses. The result should be:
**@Command( [ToolsRunMacro]; "(NotesTrackerWebQueryOpen)" )**

C. In each form that you wish to be tracked:

Go to the WebQuerySave event:
@Command([ToolsRunMacro]; "<Your agent goes here>")

Replace the string "**<Your agent goes here>**"
with "**(NotesTrackerWebQuerySave)**",

Be sure to include the surrounding parentheses. The result should be:
**@Command( [ToolsRunMacro]; "(NotesTrackerWebQuerySave)" )**

D. Copy in the NotesTracker hidden view named **“(All docs by Unique Note ID)”**

**Note 1:**: be careful not to delete the view alias “**vwUNID**” upon which the NotesTracker code relies. This view is critical for logging field Before Images during Web accesses.

**Note 2:** without this view in the database, whenever the Webquerysave event fires you can get a Domino server console error message “Object variable not set” (although the NotesTracker code attempts to mask out this error).

E. **Ensure that these two agents are appropriately signed**, as described earlier in the Administrator Topics section of this guide.

**Note:** if this signing is not done, it is unlikely that the Web agents will be allowed to execute on the Domino server. (Error messages – possibly a flood of them -- will appear on the Domino console.)

Generally, NotesTracker is designed to trap any errors arising from omission or misapplication of the above design elements, and merely to quietly exit without performing usage logging. So if usage logging is not taking place as expected, you should review your code for the presence and correct installation of all of these design elements.

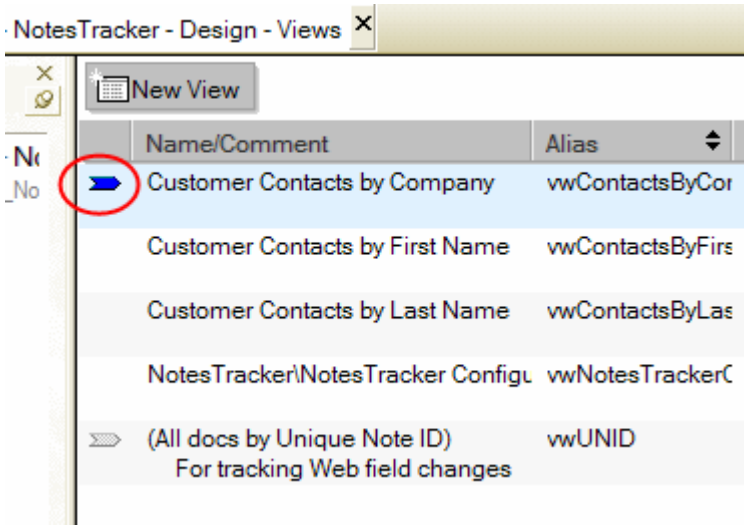
It would have been nice to incorporate these agent calls (that is, parts 2 and 3) into the existing NotesTracker Document Usage Logging subform that is used for Notes Client usage tracking. However, Webqueryopen and Webquerysave events are not available in subforms, so you must follow parts 2 and 3 for each and every form that is used by Domino to serve out web pages.

**Note:** you will need to become familiar with Domino web agents (if you already aren't), since it can be tricky to get Domino web agents to execute correctly. There are aspects such as to appropriately adjust the database's ACL so that Anonymous or authenticated users will be able to cause the web agent to run. This NotesTracker user guide is not meant to be a tutorial on Domino web agents, so you should consult the on-line Domino Help databases and the Lotus/IBM web site (and other Domino-related web sites or books) for guidance. You might look at such articles as "**Webifying an existing Notes application**" in LDD Today (part of the Lotus Developer Domain), from which the **webify.PDF** document is available:

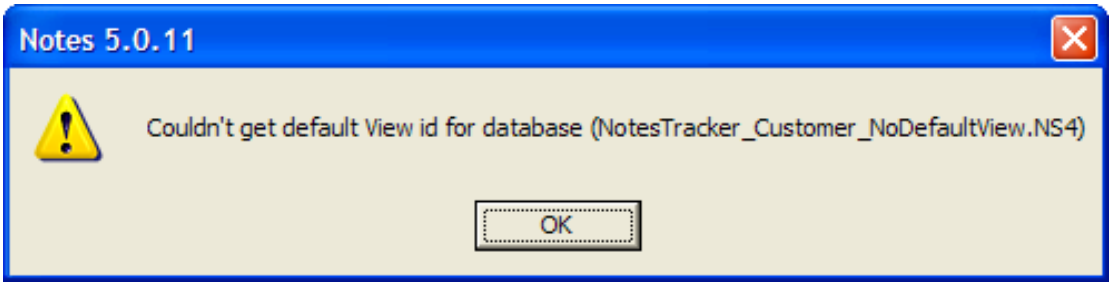
## **** STEP 9 ****
(Once per database)

**Ensure that the Database has a Default View**

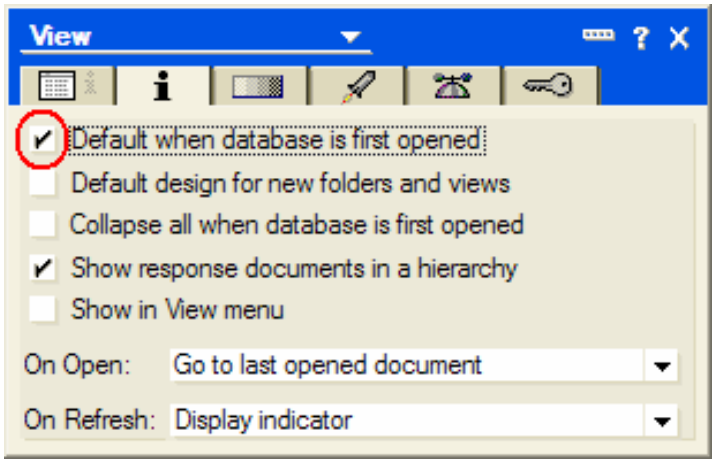
Every Notes database should have a default view. Indeed one is present in each database when it is first created. Look for a blue arrow that indicates the presence of a default view, for example:




It is quite possible for the default view to be deleted, although thankfully this tends not to happen too often. The absence of a default view may not have much impact on normal use of the database, but causes NotesTracker to throw out a "**Couldn't get default View for database**" error message like the following when a trackable document is being closed:




To eliminate this error, it is essential to specify the database's default view, and it's quite simple to do so. Simply open the appropriate view's design and set the "**Default when database is first opened**" property, as follows:



# **** STEP 10 ****

## (Once per non-hidden view, if View Opens are to be tracked)

**Tracking of View Opens (only applies to the Notes Client)**

It has long been accepted by Domino performance specialists that frequently opening and switching between views and (especially in databases containing many documents) having a large number of indexed views can both be a major contributor to server slowdown due to processor and memory loading and to significant server disk space consumption.

For this reason -- and others, such as scarcity of developer resource to build and maintain view designs in the first place -- you might need and want to track view opens. You would do this so as to find out which are the popular views, and whether there are any infrequently-used views that are not very popular but perhaps still are consuming significant server resources, dependent on their view index maintenance attributes, the latter being prime candidates for being dropped from the database's design.

There is in the NotesTracker Configuration document for each tracked database a field (described earlier on in this guide) that is used to switch on/off the tracking of View Opens for that database.

**Note:** View switching by users tends to be rather frequent. **It is important to understand that you should only judiciously conduct view tracking on a database -- definitely do not leave it switched on all the time -- and for just as long as needed to establish a typical view usage/switching pattern for the database.** It should take only hours, or at most a day or two, for you to establish the pattern. Otherwise, the tracking of view opens could add significantly to the size of your NotesTracker Repository Database, because there tends to be a lot of view switching this could well have something of a slowdown effect on user response times and your Domino server performance.

**The steps to add View Open tracking to a view** in a database are:

1. Copy the **AsiaPac_UsageTracker** script library from the NotesTracker Database into the target database.

2. In each view for which you want to track View Opens, simply do the following:
   (a) Paste into the view's **Globals - Options** the line:
   **Use "Asiapac_UsageTracker"**
   (b) For Notes R5 and later, paste into the view's **Postopen event** the line:
   **Call UsageTracker_ViewOpen( "" )**

A sample such view (hidden) is stored in the NotesTracker Database. Look for the view named:
**(UT_Track_ViewOpenEvent_Code**

If you do elect to switch on the tracking of View Opens, you should be aware that in Notes prior to Release 5 it is not possible to determine the name of a view programmatically. Therefore instead of the unobtainable (null) view name, the string "(View name not obtainable in Release 4)" will be stored in the Usage Log document.

As from NotesTracker Version 3, the Postopen event has been enhanced so that instead of the using a null parameter as shown just above in step 2(b) you can insert the actual view name. For example, you might code:

**Call UsageTracker_ViewOpen( "Customers by Amount Outstanding" )**

By adopting this enhanced approach, instead of the view name being logged as "(View name not obtainable in Release 4)" you will see a meaningful view name in the Usage Log document. This is a trivial amount of work for a much better outcome.

For Notes R4 Clients, there is *some* possibility that the following error can occur if tracking of View Opens is switched on (in which case you may decide to switch off the tracking of View Opens):


This problem probably will not arise, as the tracking code should intercept and gracefully handle this condition.

## *Tip – An Efficient Way to Debug Domino Web Agents*

During our development of NotesTracker's ability to Web activities (database document Reads and Updates), we had to code and debug the NotesTrackerWebQueryOpen and NotesTrackerWebQuerySave agents discussed just above.

Unlike the Notes Client environment where you can go into the nice and easy "Debug LotusScript" mode, Domino web agents can be quite difficult to debug, particularly under R4 and R5 where the Remote Debugging capability added in Domino 6 is not available (and even this can be tricky to use). Refer to the following IBM developerWorks articles for some excellent guidance:

- http://www-10.lotus.com/ldd/today.nsf/lookup/DebuggingLotusScript_1
- http://www-10.lotus.com/ldd/today.nsf/lookup/DebugLS2
- http://www-10.lotus.com/ldd/today.nsf/lookup/ND6NewAgentFeatures

We were seeking **a quick-and-easy way to carry out Domino web agent debugging**. We weren't quite satisfied with other methods (like as those discussed in the developerWorks articles), good as they are. We came up with an uncomplicated Domino web agent debugging methodology that works in any release of Notes/Domino. We found that this methodology enabled us to rapidly work our way through some fairly complex LotusScript web agent code. It is also quite useful for debugging non-Web Domino agents.

Its key points are:

1. Develop **a single, simple LotusScript statement** that will cause the agent to terminate abruptly with a specific, well-known error message being thrown at the Domino console. (This, of course, will be identified with your agent's name, and this will distinguish it from any other sources of console messages.)

2. Place that statement at some strategic point in the agent's code stream and watch for a specific termination error message at the console. We then know that the web agent has successfully reached exactly that point in the code.

3. Quickly cut-and-paste that statement to another strategic point further on in the agent's code stream, and then go back to Step 2, running the web agent again.

4. Continue doing this until eventually a point is reached in the agent's code stream where either (a) there is an error message to the Domino console that is caused by something other than the above LotusScript statement, or (b) the agent terminates without that error message. In the latter case you know that the faulty code lies between that statement and the end of the code stream.

**What is a "strategic point" in the code stream?** We suggest adopting a "binary search" technique to determine such points firstly in the agent's mainstream so as to find out if the error lies in the mainstream itself. After that, place the statement at the very start of each successive subroutine (or function), and move on in a "binary search" fashion within the subroutine until you're sure that the fault does not lie in that subroutine. This methodical approach is far more efficient than randomly picking points to place the statement.

We decided to deliberately force a **Zerodivide error** to generate the Domino Console errors. In a mathematical operation, there cannot be an attempt to divide by zero, so the LotusScript compiler disallows statements that it can predict will cause division by zero, such as: **z% = 1 / 0** or **z% = 1 / ( 1 – 1 )**

To get around this we used a statement of this form: **zerodivide% = 1 / ( zerodivide% - zerodivide% )** and to save typing this becomes: **zd% = 1 / (zd% - zd% )** or simply: **z% = 1 / ( z% - z% )**

We generally used the following statement:

```
z% = 1/( z% - z% )  ' ################################################
```

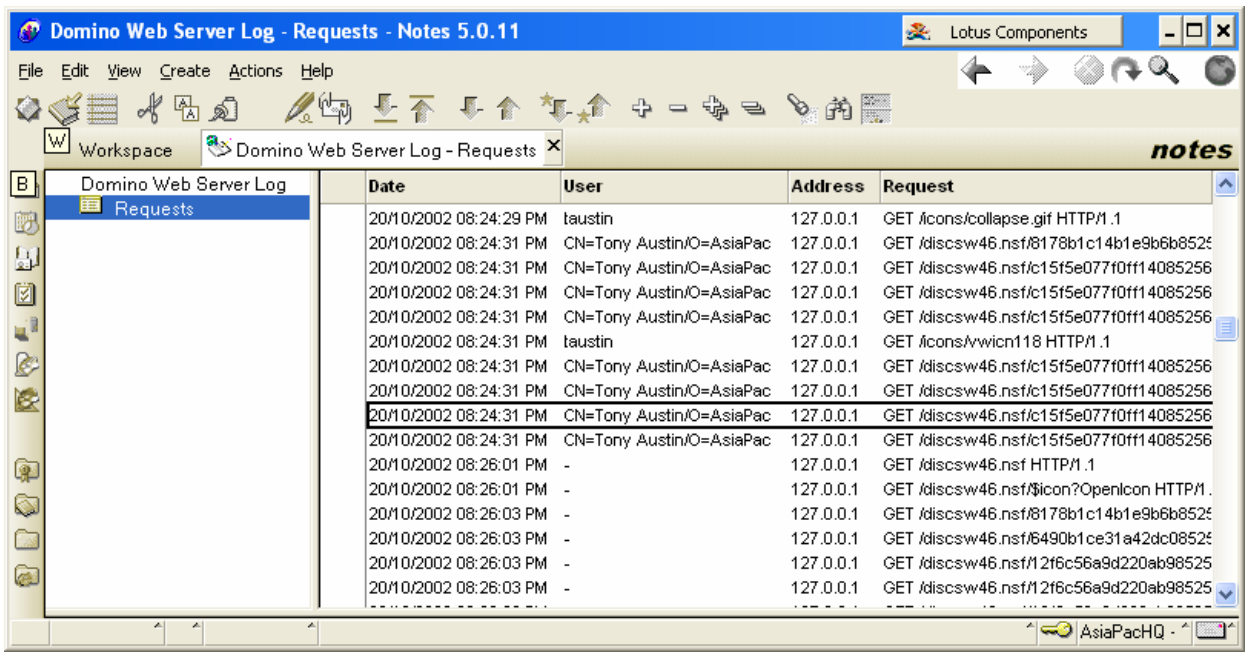
The string of hash symbols (#) makes the statement stand out. This makes it much easier to locate it rather than overlook it when you have finished debugging the agent, and when you want to cut-and-paste it elsewhere in the code stream.

For maximum efficiency and convenience, we ran our Domino test server on the same workstation that we were using for the Domino Designer so that we could detect the Zerodivide error message immediately that it appeared on the Domino Console, an instant after we performed the error-causing action in the web browser (closing a web page opened for reading, or clicking the Submit button to save changes made on a page being edited). A remote Domino Console on your workstation is the next best thing, or a Domino server system close by. This arrangement can substantially improve your debugging turnaround time.

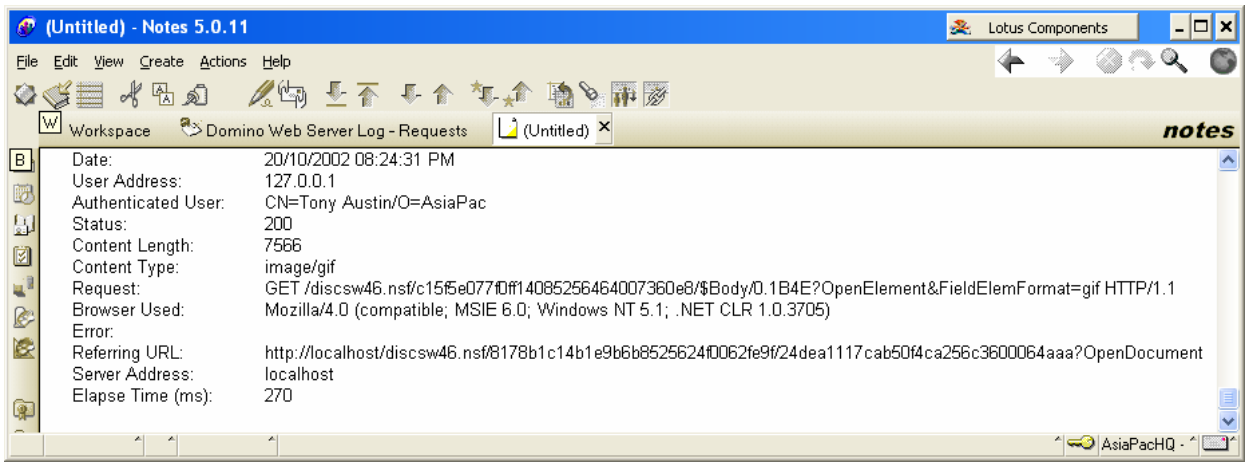
## *Why the Tracking of Web Browser Accesses was added to NotesTracker*

NotesTracker was originally designed to measure purely Notes Client database accesses. The browser tracking capability was added in Version 4.0 and was in recognition of the fact that an ever-increasing proportion of Notes applications are being designed for web browser deployment.

As discussed earlier, in the Administration Topics section, the Domino server can deposit web usage statistics in the Domino Web Server Log database (DomLog.NSF). The data so deposited tends to be rather “raw and formidable" – very many log entries can be generated in a short time, and the sheer volume of data tends to be an overwhelming problem.




Furthermore, content of any one log entry may not be of great significance – maybe only showing that a certain graphic JPEG or GIF image file was loaded into a web page. But even if the log entry relates to a significant web event, such as a page GET or page POST, you may not be able to readily relate the data in that DomLog.NSF entry with anything much more than that something in a given database was accessed by the event -- such as the R4.6 Discussion database (discsw46.nsf) in the following example:




The only way for you to cope with this bombardment of information is to develop your own web log analysis routines, or purchase one of the commercially available packages (of which a few are designed specially for Domino web server log analysis while others are generic web log analysers).

In contrast, NotesTracker Version 4:

- Writes web browser Usage Log documents that are in the same **easy-to-understand format** as for earlier NotesTracker versions

- Presents them via the same **useful Notes views as for earlier** NotesTracker versions. You don't have to wait for a batch analysis run to finish. The usage information **is immediately available** via these views. And the results from remote servers are available for review as soon as the NotesTracker Usage Log documents have replicated across your Domino server network.

- Logging can easily be configured so that different sets or groups of related databases – such as all marketing databases, or financial databases – are logged to different NotesTracker repository databases. This gives you the option to break down the usage collection and analysis in smaller, more manageable, application-oriented "chunks" (compared with being forced to manage and analyse one large central repository).

**Note:** It must be realized that NotesTracker makes use of the WebQueryOpen and WebQuerySave events running on a Lotus Domino server to track document activities. If your database uses Browser-only techniques to affect what appears on the Browser without any interaction between Browser and Domino server, then NotesTracker will not track them while they are taking place. An example of such a technique is described in the IBM developerWorks article: *Using AJAX to manipulate Lotus Notes documents.* Nevertheless, once the document changes are sent back to the Domino server, the WebQuerySave event will detect them and the changes will be recorded by NotesTracker. This is equivalent to the way that NotesTracker only records changes made to a document via a Notes Client once the document is finally closed, not every single time that the document is saved before closure.)

## *Maintainability - Field Names used & Coding Conventions Followed*

NotesTracker was designed not only to be as easy as possible to incorporate into your databases, but also – quite importantly – to cause minimal, if any, disruption to or interference with the existing code in the databases.

One aspect of this is the naming conventions used by NotesTracker for such things as field names and subroutine names.

For instance, when you're debugging your code with the LotusScript debugger you will encounter:

- Subroutine and function names like "UsageTracker_OpenUsageDB" and "UsageTracker_SetLogFields" and "UsageTracker_FieldsAfterUpdate" and "UsageTracker_SetTitle".
- Database field names like "UsageTracking_Status" and "UsageTracking_TrackUpdates" and "UsageTracking_Title" and "Path_UsageTracker".

The presence of the string "UsageTracker" in these names should make it easy for you to discriminate between NotesTracker code and all the other code in your databases. (The product name "Domino Document Usage Tracker" was used for the early releases of NotesTracker.)

As well as this, the recommended convention is followed that:

- The database fields (permanently stored fields) have names that start with an uppercase letter.
- Work fields (temporary variables never stored in a database) have names that start with a lowercase letter. A few examples of work field names are "verbosity" and "replicaID_UsageDB" and "foundTitle" and "suppressUnchanged".

This makes it easier to identify places in the code where database field contents are changed, usually prior to the field being written out to the database (as part of a Notes document).

The field names were designed to be as meaningful as possible. Line spacing and comments were used wherever it was thought they would help you to understand and navigate the code more easily. Coding "tricks" were avoided in the interests of clarity and maintainability.

Design elements such as hidden fields and developer/administrator comments are generally in a purple color, this being chosen since it stands out quite well and generally purple is not used for visible design elements.

## *"Breakings News" or "What's New?" Views - a Unique and Generic Portal Page Capability*

We think that NotesTracker offers a **unique design capability**, and that it does so in an extremely painless, quick-and-easy fashion!

Three sample supplied views can be used for **tracking all new and changed (updated) documents** in the database, or set of databases, being tracked within a given NotesTracker Repository Database. Remember that different groups of your Notes databases can be tracked in different NotesTracker Repository Databases, determined solely on the repository's Replica ID stored in each database's NotesTracker Configuration Document.

The titles of these views are:

40. What's Changed - Auto Doclink Launch view
41. What's New - Auto Doclink Launch view
42. What's New (non auto-launch view)

The intent of these is to allow you to embed them as "What's New" and "What's Changed" dynamic views into a pane within your "portal" or "Welcome Page" designs.

You can easily build useful variations of the above views. Simply by changing the View Selection values to filter out different sets of Usage Log documents in a NotesTracker Repository, based on any of the field stored in a Usage Log document: the tracked database's name, user name (you can split out the Canonical Name components if this is useful), action type (Create, Update, Read, Delete), and so on.

Please contact us for advice if you are unclear how to go about adapting your portal page design. Send an e-mail explaining your portal page requirement to: NotesTrackerSupport@asiapac.com.au

Sample views 40 and 41 are "auto Doclink launch" views. They were designed with additional code that is tailored for placement in an embedded view in a frame on your portal page. When the user double clicks to open a document within the "What's New" or "What's Changed" frame, instead of opening the NotesTracker Log Document -- which would be meaningless to the end user -- a Form Formula ("UsageLogAutoLaunch") comes into effect. Sample view 42 is similar but does not have the additional code to automatically launch the Doclink.

The "UsageLogAutoLaunch" form is a variant of the regular NotesTracker Log Document form that has the form property "Auto Launch: -First Document Link". This is intended to automatically launch the Doclink and so cause the document in the original database to open (rather than just the Usage Log document itself). Naturally, if the original document has been deleted, the launch attempt will fail, but it is not possible to trap this error situation and handle it gracefully (a "Doclink cannot be located" message will occur)

See more related information in the section below labelled **"Breaking News" View for Portal Page**.

## *Tracking of Document Deletions and Deletion Attempts*

Going by its name, the **PostDocumentDelete** subroutine (found in the Database Script library) should allow you to track the deletion of documents, but bitter experience shows that this is not what it always does.

In fact, what we the event enables you to track more correctly can be termed the “attempted deletion” of documents. The Domino Designer Help indicates, somewhat misleadingly, that the PostDocumentDelete event “occurs just after a document is deleted (cleared or cut)” and then further on states “Unauthorized users can execute any script that responds to this event, since PostDocumentDelete occurs before Notes verifies user authorization. Do not enter a script that you want only authorized users to execute.”

Pragmatically, this amounts to the triggering of the PostDocumentDelete event not truly being “post” (after) the document has actually been removed from the database! You should think of this event as being a request for document deletion – and so perhaps it should be called “**PostDocumentDeleteRequest**”. It can eventuate that, if the user is not authorised to delete documents, the deletion request will fail.

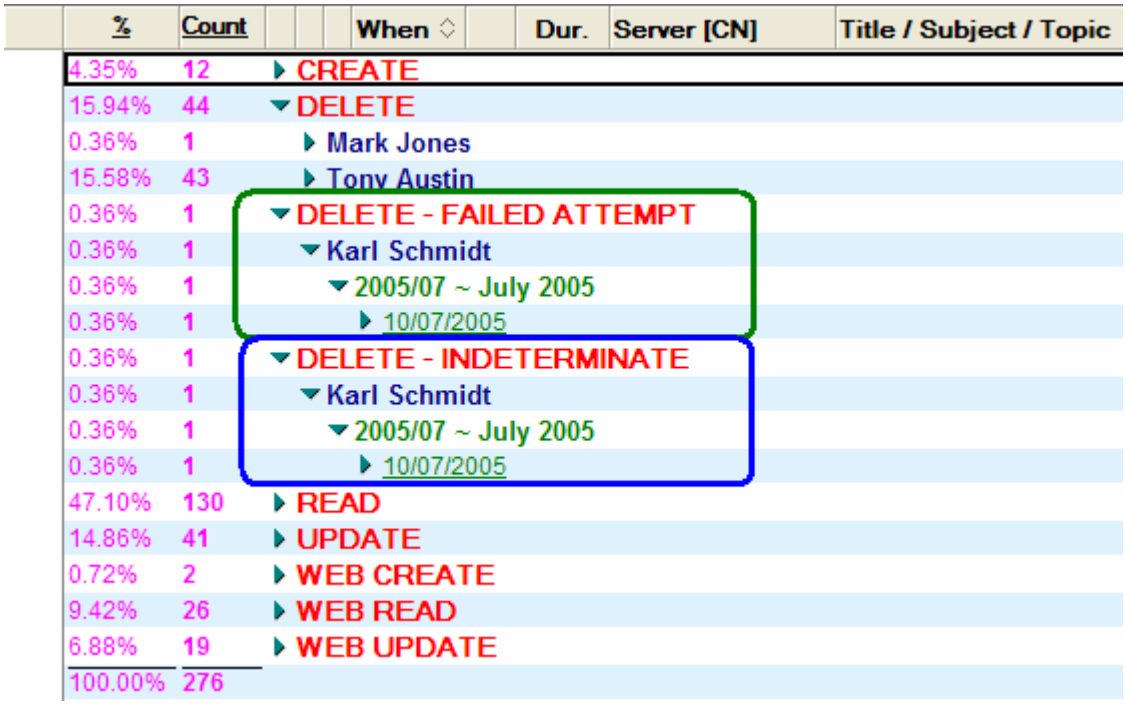
To our knowledge there is no capability to properly handle this misleading situation, due to inadequacy of features in the LotusScript language (at the time of writing, up to and including Release 6.5.4).

In NotesTracker Version 4.4 we modified the design of the deletion-tracking code in a “best efforts” attempt to provide more accurate or meaningful results. We added a database Queryopen routine that determines whether the user is authorized or not to delete documents in the tracked database and writes this into a Profile Document (specific for each user, so that users’ deletion rights are separately recorded).

Then in the PostDocumentDelete subroutine – as stated above, this is prior to the actual deletion which occurs asynchronously to the event -- we use this to set an Action Type of “D” for an authorized user and “F” for an unauthorized user (The “F” stands for “Failed deletion attempt”). Also, in some cases we cannot determine the user’s authority so we set the Action Type to “E” (meaning that the deletion attempt is “Indeterminate”).

Finally, in the database’s Queryclose event we do some housekeeping, trying to delete the no-longer-needed Profile Document.

This results in some new Action Type entries appearing in NotesTracker Version 4.4, looking like this:




Action Type “F” is translated to “DEELETE – FAILED ATTEMPT” as shown in the green rectangle in the illustration, and Action Type “E” is translated to “DELETE – INDETERMINATE” as shown in the blue rectangle.

## Steps to add Document Deletion tracking to a database

1. (New in Version 4,4) Copy the **AsiaPac_UsageTracker** script library from the NotesTracker Repository Database into the target database.

2. In the Database Script, add the following:
   (a) paste into Database Script's **Options** the line: **Use "Asiapac_UsageTracker"**
   (b) paste into the Database Script's **Postdocumentdelete event** the line:
   **Call UsageTracker_PostDocDelete(Source)**

3. (New in Version 4.4) Optionally: copy from the NotesTracker Repository Database the housekeeping agent named "**NotesTracker deletion profile docs clearout**" and you should run this agent occasionally to clean out any residual NotesTracker document deletion Profile Documents (that have not been deleted automatically via the database's Queryclose event in the AsiaPac_UsageTracker script library).

## Setting a Usage Tracking Title for Document Deletions

We discovered during development and empirical testing of this capability that the technique of using a **UsageTracking_Title** field (described in "Step 4" earlier in this Developer Topics section) does not work for document deletions. This technique sets the **UsageTracking_Title** field as a Computed for Display field in the LotusScript UIdocument environment (also referred to as the "front end" document).

The following discussion is a little abstruse, but here goes! When a user deletes a document (or multiple documents) from a Notes view environment, then naturally enough there is not any "front end" document open during the document deletion operation. This means that there is no way for the Querydocumentdelete subroutine to get a meaningful document "title" from such a Computed for Display field, so rather than recording nothing we simply record the value of the document's Form field in the Usage Log document.

If it is important for you to record in the Usage Log document a meaningful "title" for each deleted document , you must have the **UsageTracking_Title** field as a permanent (Computed) field in the document that is deleted. That is, it is only such a pre-existing permanent field that the Querydocumentdelete subroutine can retrieve and store in the Usage Log.

There is an issue regarding the nature of the **UsageTracking_Title** field when it comes to tracking document deletions. If you define this field as computed-for-display, this has the benefit that there is no stored value for this field in your documents. However, at the time a document is deleted at the view level the UI document will not be open and so this field does not exist, consequently a null value will be put into the Title field in the NotesTracker log document making it nigh impossible to later understand which particular document has been deleted.

## Sample Agent for Setting the Usage Tracking Title Field for Document Deletions

One way to overcome this would be to define the field as "Computed" (rather than computed-for-display), and ensure that it is has a stored value prior to the document deletion. The stored value would be created either by manually editing and saving each document, or by running an agent to refresh the field values. The following is an example of such an agent, designed to handle the forms in the Domino Directory. (For simplicity, only a subset of the many forms in this database's design is shown in the following example.)

```
rem "Agent name: Refresh UsageTracking_Title field";
rem "Sample code for Domino Directory (subset of forms shown below)";
rem "Target:  All documents in database";
SELECT @All;
@If(
Form = "Connection";
        @SetField( "UsageTracking_Title";
        "Domino Directory - SERVER Connection : " + @Text( Source ) + " to " + @Text( Destination ));
Form = "Domain";
        @SetField( "UsageTracking_Title";
        "Domino Directory - SERVER Domain : " + OtherDomainName);
Form = "CrossCertificate";
         @SetField( "UsageTracking_Title";
         "Domino Directory - CROSS CERTIFICATE: " + CertificateType);
Form = "Group";
         @SetField( "UsageTracking_Title";
         "Domino Directory - GROUP: " + @Name([Abbreviate];ListName)));
Form = "Location";
         @SetField( "UsageTracking_Title";
         "Domino Directory - LOCATION: " + @If(Name="";"";": "+@If(Type = "Server";ServerName; Name)));
Form = "Holiday";
        @SetField( "UsageTracking_Title";
        "Domino Directory - SERVER Holiday : " + Subject);
Form = "Server";
        @SetField( "UsageTracking_Title";
        "Domino Directory - SERVER Resource: " + @Text( ServerName ));
Form = "Person";
         @SetField( "UsageTracking_Title";
         "Domino Directory - PERSON: " + @Trim( FirstName + " " + MiddleInitial + " " + LastName ));
@Success
)
```

Another strategy would be not to use the UsageTracking_Title field in the form's design, and instead to add appropriate field name(s) to the Title Field Name Preference list described earlier.

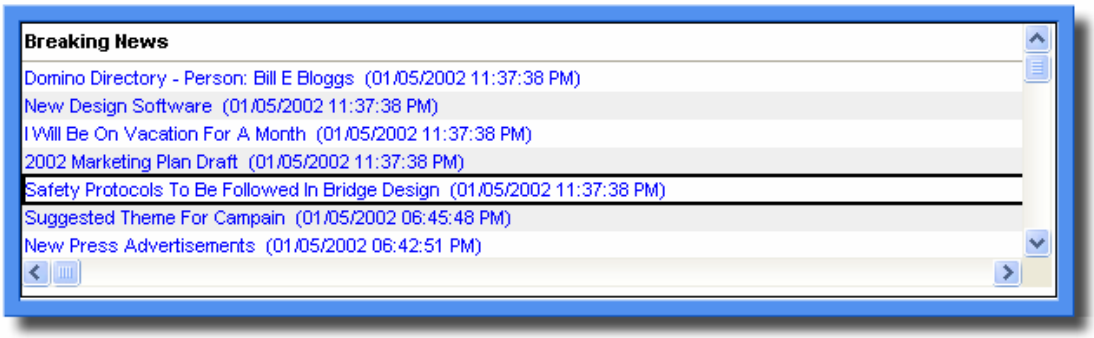
Otherwise, to meet your specific requirements you could modify the generic title-handling code of the "**UsageTracker_PostDocDelete**" and/or "**UsageTracker_SetTitle**" subroutines (found in the "AsiaPac_UsageTracker" script library).

## *"Breaking News" View for Portal Page*

In the navigator is a new button labelled "Breaking News live feed":

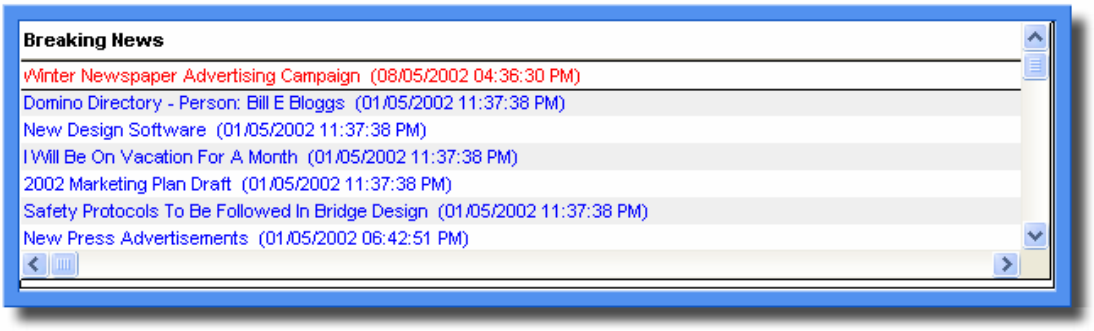

When clicked, it causes a document to be opened (in Create mode, with the SaveOptions field set to zero "0" so that the document can never be saved, since it is meant to act only as a vehicle for displaying the embedded view.)

For example, to see the latest news on "Safety Protocols To Be Followed In Bridge Design", you select the appropriate document




You simply double-click on it to open the document.

Once the refresh interval has expired, the view will be refreshed, it will include any newly-added documents, and the "selected document" pointer will have been moved back to the top of the view. Here's an example, showing a newly-arrived document (about a "Winter Newspaper Advertising Campaign", the red colour indicating that this document has not yet been opened in the Breaking News view):




The demonstration embedded Breaking News view has only two columns:

1. A hidden column, SORTED in descending date/time sequence (the date/time that each document Create was logged)

2. A text column containing the "Title" field from the Usage Log document, and with the column heading "Breaking News".

This embedded view has a **view timer event** triggered on Postopen, with a variable that you can easily change to alter the view refresh interval:

```
Dim timerTickSeconds As Integer  ' The refresh interval (in seconds)
timerTickSeconds = 30
```

It is set to such a low value for demonstration "special effects" and in production should probably be set somewhere between 600 (ten minutes) and 3600 (one hour). The interval really should be determined by the expected frequency of document Creates that are filtered out by the embedded view, and should be shorter if time-critical information is expected that needs to come to peoples' attention sooner rather than later.

Additionally, it should be noted that in the view's **refreshTimerHandler** subroutine the following lines cause the "selected document" pointer to be moved back to the top of the view (that is, to the first document in the view):

```
Set docTop = viewBackEnd.GetFirstDocument    ' Get first doc in back-end view
Call viewUI.SelectDocument( docTop )   ' Select first doc in view
```

The intent here is to move the "selected document" focus back to the most-recently-added document, which is at the top of the view. You should comment out or remove these two lines if you don't want this effect to occur.

## How would you make use of this "What's New" style of view?

Essentially, whatever may be built into a meaningful Notes view may appear in such a portal page view. They need not be just document Creates (as in the above example), but could be document Updates as well as or instead of Creates, and can be filtered by any sort of document Selection formula and sorted/arranged in any way that is able to be specified for a Notes view.

## Generating RSS Feeds automatically from NotesTracker

By building and running a suitable scheduled agent against your Breaking News views, you could generate the XML code needed for RSS Newsfeeds suitable for you intranet or Internet portal pages.

If you’re not familiar with such newsfeeds, which are rapidly becoming increasingly popular as an easy and highly efficient way to keep up with news, you really should do so!

We have some general information about RSS feeds in general on our web site. Refer to:

http://asiapac.com.au/Links/KM.htm#Newsfeeds_Webfeeds_RSS (Australian server)

or

http://notestracker.com/Links/KM.htm#Newsfeeds_Webfeeds_RSS (US server).

## *Controlling Changes to the NotesTracker Configuration Document*

What about control over changes to the NotesTracker Configuration Document itself?

If NotesTracker is being used to measure who accesses which documents in a given Notes database, and just how they accessed the documents (Read, Update, Delete, etc), then it's important to have control over the NotesTracker Configuration Document in that database so that some unauthorised person does not alter your NotesTracker settings or even completely switch off tracking for the database.

As an aid, a simple "audit trail" or "edit history" was added to the bottom of the configuration document, illustrated by the following:

EDIT HISTORY - Up to the last 20 revisions (Saves) of this document
Created by Tony Austin on 21/06/2002 at 06:37 PM

| Revision | Edit Date and Time | Editor |
|---|---|---|
| 1 | 21/06/2002 06:40 PM | CN=Tony Austin/O=AsiaPac |
| 0 | 21/06/2002 06:38 PM | CN=Tony Austin/O=AsiaPac |

As distributed, the last 20 revisions (document Saves) of the configuration document are tracked, but you can easily changes this to more (or fewer) merely by changing the value calculated by the "EditHistory_ListLength" field.

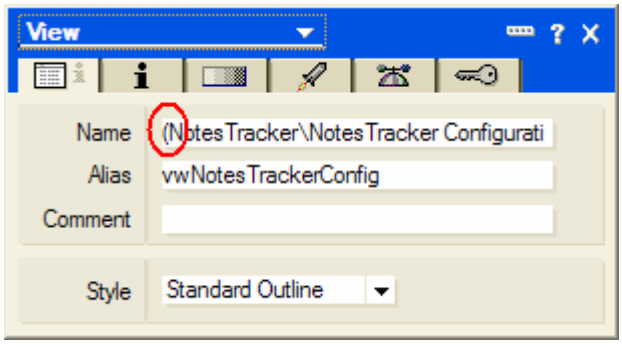
While this "edit history" tells you who made changes to the configuration document and when the changes were made, it does not control who can make changes.

**Note:** in NotesTracker version 3.2, the design of the Configuration form was changed so that **only those who have Manager access to the database may save a configuration document.** The Postopen event is used to warn those who open the Configuration document that they don not have the required access rights (Manager level), and the Querysave event prevents anyone not having Manager level access from saving a configuration document.

You should refer to the security section under NotesTracker Database Administrator Topics for a discussion of ACL considerations related to this important matter.

## Hiding the NotesTracker Configuration View

Another approach is to make it less likely that users will see the NotesTracker Configuration View. You can do this by hiding the view, done by simply surrounding the view name with **parentheses**, as indicated in the following illustration:



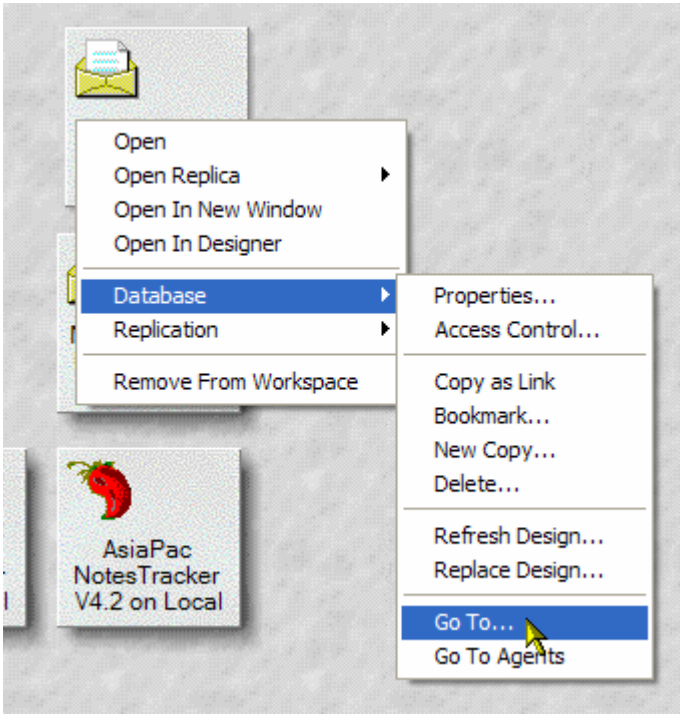

**Note:** be particularly sure not to alter (or remove) the View Alias name “vwNotesTrackerConfig” since this exact alias name is critical to the successful running of the NotesTracker code.

Since the view is now hidden, the question arises “How can the NotesTracker Configuration Document be viewed in order to be edited?”

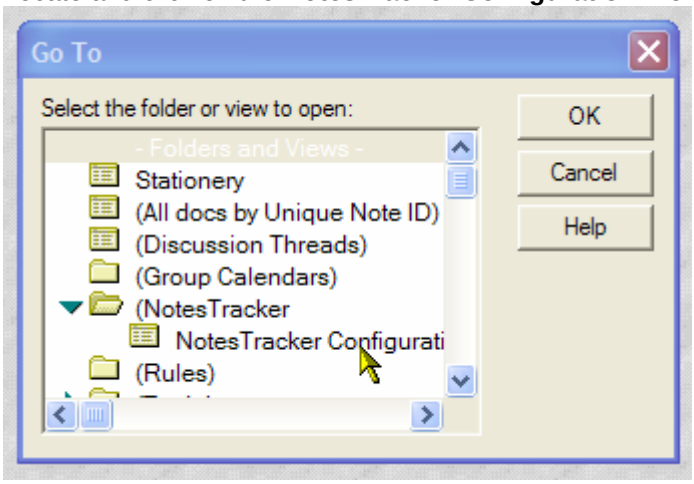
The answer is to use a “trick” that is not known by many Notes users (and for our purposes it is best that you don’t publicise this method).

The trick method is:

1. Go to the Notes workspace
2. Hold down simultaneously the **Ctr**l and **Alt** keys
3. Click on the database workspace icon using the Right mouse button
4. Click on the **Database** list item and from there the **Go To…** list item as shown on the following illustration:




5. Locate and click on the **NotesTracker Configuration View** item:




6. Click the OK button

The NotesTracker Configuration View appears, and from there you proceed as described elsewhere in this guide.

If you use this approach, document it and be sure to adequately train everyone who needs to be aware of it (such as Domino administrators).

**Note:** Since the view can be accessed by anybody who knows this trick, it cannot be considered a foolproof security approach, but it certainly is a way of decreasing the likelihood that many of your users will see the view and open the NotesTracker Configuration document.

A rather more secure approach would be to use a NotesTracker Profile Document (rather than the current approach of NotesTracker Configuration Document plus NotesTracker Configuration View). The form design for the Profile Document would probably need only one extra field, to hold the “key” value for the Profile Document. It would be considerably harder for general users to know the correct key value and somehow make use of it to open and edit the NotesTracker Configuration Document. Your Notes developer should be able to make the necessary design changes without too much difficulty.

# Extended Analysis and Reporting

## *Tailoring of Views, and Interfacing to External Analysis Tools*

In this NotesTracker Database itself, a navigator and a set of views have been put together in the expectation that they will meet your initial requirements for usage analysis.

After a while, you will probably want to add more views or modify the existing ones to better meet your usage metrics needs.

For example, the duration in seconds for which a document is open is recorded, but is not used as a main category in any of the supplied views. (Duration, if exceeding one second, added in Version 3.2 as a non-categorised, non-sorted column only. The "one second" durations are assumed to be trivial operations and are blanked out in the column so as to make the column appear less cluttered.). You could use this value to explore how long your users keep documents open, and so develop a new view that categorises the documents by duration (perhaps using categories like as "Less than 10 seconds", "11 to 30 seconds", "31 to 60 seconds", "between 1 and 5 minutes", "between 5 and 10 minutes" and "Over 10 minutes").

Useful as they are, there are definite limitations to the extent of analysis that can be performed using views. Therefore you might decide to write an agent (probably using LotusScript or Java language) that extracts selected Usage Log data to a file, which then becomes the input to reporting tools that offer more power and flexibility to generate just the sort of document usage reports that you need (tables, graphs, charts, trend analyses, etc).

To extract NotesTracker information, you might find value in a tool such as **Export-Wiz** from Kim Beros Consulting (Melbourne, Australia):

http://www.lotus-notes-export.com/ExportWiz.asp?FrontPageClickHere