NotesTracker Guide

The Comprehensive Usage Tracker for your IBM Lotus Notes and Domino Applications

Version 5.0

This guide was last updated on 09 April, 2007

Asia/Pacific Computer Services

http://asiapac.com.au

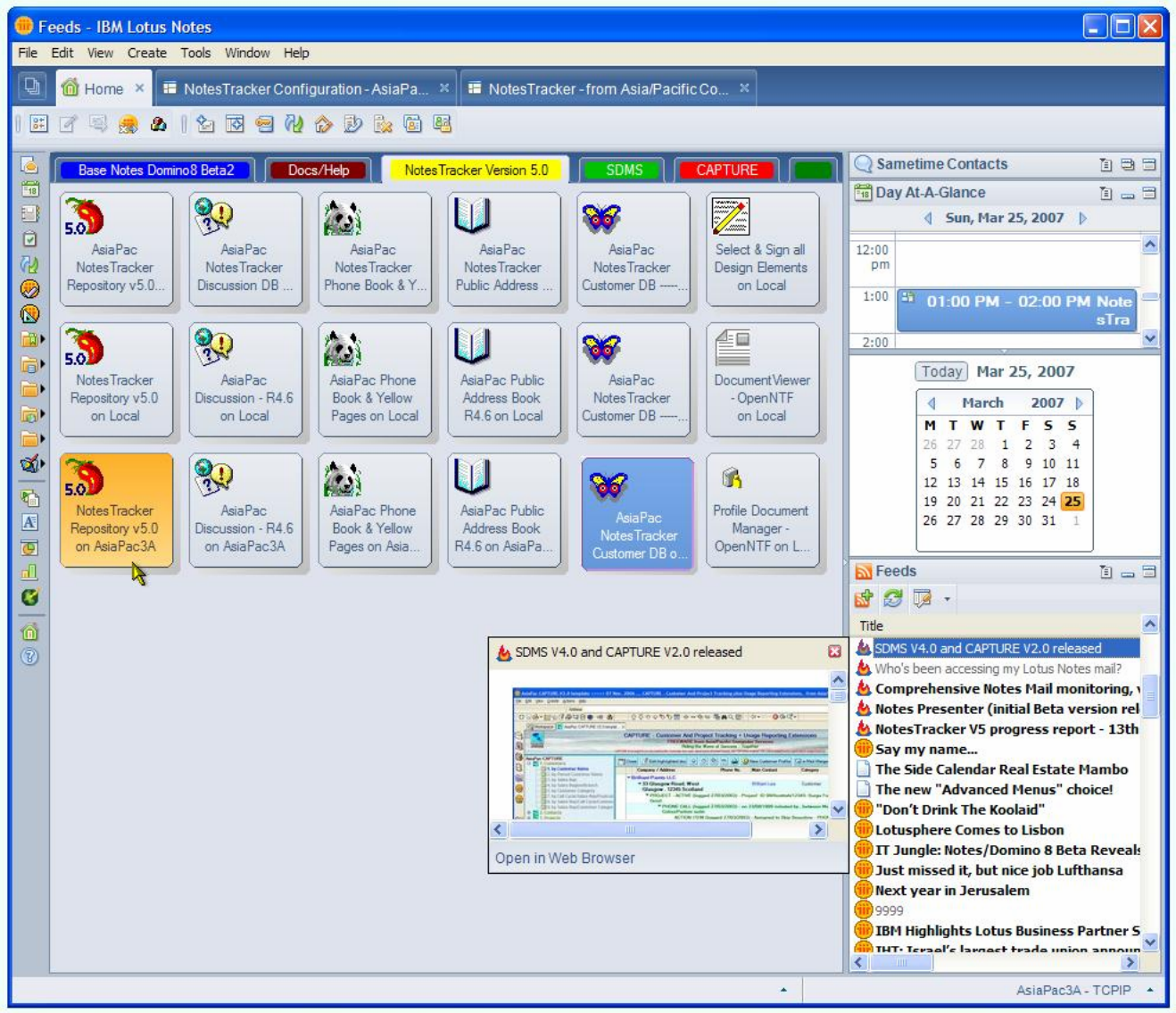
http://notestracker.com

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

We welcome your feedback about any technical inaccuracies plus your comments about this guide's clarity, usability, accuracy, and missing or incomplete information. (Send them to the above e-mail address)`

KEEPING ABREAST OF LOTUS NOTES / DOMINO PRODUCT EVOLUTION

NotesTracker V5.0 being tested successfully
against IBM Lotus Notes and Domino 8 (Beta 2, March 2007) …

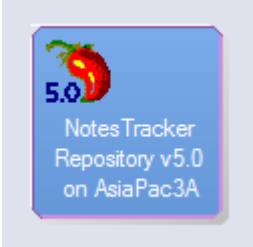


Into the future with Notes and Domino!

Introduction

The Purpose of NotesTracker

NotesTracker, from Asia/Pacific Computer Services, is a comprehensive developer toolkit crafted to make it easy and quick for you to extend your IBM Lotus Notes and Domino database applications in order that you can rapidly detect and report on a wide range of activities being carried out on those databases.



NotesTracker comes in the form of a **Repository database --** alternatively referred to just as "**the Repository**" (although you can deploy multiple repositories if you wish) –and it delivers several capabilities:

- A repository holds an **audit trail, or log, of the application activity** for your Notes databases. A collection of Usage Log entries builds up over time, as documents in your Notes databases are being accessed and other tracked events (such as database or view opens, document deletions) occur.

- The NotesTracker Repository database provides, via Notes views, a continuously-updated **analysis of document and view usage patterns** in your own application databases.

  You may have one or more NotesTracker repositories located throughout your Notes/Domino network, and they will behave and replicate just like any other ordinary Notes database. This is important, because it means that no special repository or reporting infrastructure needs to be learned about, installed and maintained – that is, with NotesTracker it's a simple case of "business as usual."

  The administration topics section of this guide explains how you would decide about and set up a single repository or multiple NotesTracker repositories, each repository used for logging the activities occurring in just a single database (if it's an important one) or in groups of related databases.

  The developer topics section of the guide discusses the advantages (and disadvantages) of a further option: modifying the design of a given database so that NotesTracker's usage logging entries are stored within the database itself, rather than getting sent to a central NotesTracker repository.

- The distributed NotesTracker repository also acts as the container for the NotesTracker Software Development Kit (SDK), that is, as the **NotesTracker design repository.**

  As a **licensed user** of NotesTracker you get a version with open access to all NotesTracker source code and other design elements (forms, views, agents, etc). This enables your Notes developers and administrators to adapt your Notes databases for usage tracking, either with or without modifying the distributed NotesTracker behavior.

  If you are using an **evaluation** version of NotesTracker, all of the functions work without limitations and can be fully exercised, but do not get access to the underlying design elements.

Also distributed with the Repository are several example Lotus Notes databases to give potential users, administrators and developers some NotesTracker-ready applications to test and experiment with.

The example databases in the NotesTracker evaluation version are fully functional (but their designs are hidden). Our expectation is that with a little familiarization and testing of these you will see the potential for using NotesTracker in your organization and be able to make informed purchase decisions.

The example databases in the NotesTracker purchased (licensed) version do not have their designs hidden. As a licensed user, your Notes/Domino developers can examine and use the various design techniques in these databases as a kick-off point to quickly learn, using this guide plus tutorial materials obtainable from the Web, how to adapt your own Lotus Notes applications and take full advantage of NotesTracker's capabilities.

Derivation of the Name "NotesTracker"

Items such as documents and views in Lotus Notes databases are sometimes referred to as "design notes" or just "notes", hence the name "NotesTracker" because it is the user actions being performed against them that are being tracked. NotesTracker's emphasis is on *application content and activity against it* (rather than on what some other tools track, such as Domino server performance and disk usage, or networking traffic loads).

NotesTracker Trademark, and Fair Use

How this Guide is Arranged

After some general introductory sections, the bulk of this guide is devoted to an **Administration Topics section** followed by a **Developer Topics section.**

The intention is for Notes developers to read right through and become familiar with administration matters before they delve into the database design topics in the Developer Topics section. On the other hand, Notes administrators will gain some value from perusing the Developer Topics section but do not have to become deeply familiar with all of its contents.

NotesTracker Usage Terms and Conditions

Our aim: to provide you with a reliable, resilient, capable and 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!

As a convenience, on the following pages is a snapshot of the NotesTracker terms and conditions of use that applied at the time that this guide was last updated.

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 since it is these which apply to your use of NotesTracker:

http://asiapac.com.au/Pricing/Usage_Tracker_pricing.htm
Or
http://notestracker.com/Pricing/Usage_Tracker_pricing.htm

SOFTWARE USAGE TERMS AND CONDITIONS

Last updated: 13 August, 2006

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 LICENSE

The "NotesTracker" or "AsiaPac Document Usage Tracker" Software is owned by APCS and is copyrighted and licensed 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 licensed materials.

The entity that purchases the license ("you") and all its officers, employees, contractors and other associated persons will be subject to this license 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 **licensed 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 that each and every copy must contain all of the original Software's proprietary notices. You may not distribute (for free or for sale) or sublicense 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 organization. If you distribute, rent or sell Lotus Notes applications to other organizations you will not include the Software or any derivative or adaptation of it in these Notes databases until such time that the receiving organizations have purchased the appropriate NotesTracker license from APCS.

You will ensure that anyone who uses your copy of the Software does so only for your authorized use and in compliance with the terms of the license. Unlicensed 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-licensees or other copyright infringement in the Software.

APCS grants you a nonexclusive license to use the Software when you lawfully acquire it. When you purchase the Software a License Document will be sent to you by APCS as a Proof of Entitlement and this will contain a License 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 license has been issued. The license 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. You may use the Software up to the level of use purchased. 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.

APCS cannot and does not give credits or refunds for charges already due or paid. Prior to purchasing the Software you are expected to make every effort to establish that the Software meets your requirements. APCS will provide you with information and guidance to aid your efforts but the final decision is yours.
The Software is licensed as stated above. The license does not constitute ownership of the Software, only the right to use it. Licenses 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 in electronic or any other form. APCS may terminate your license 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 license expires.

LIMITED WARRANTY

This license 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 including internal documentation.

The cost or guarantee of support is not included in the license except that APCS will make reasonable effort to fix any problem in the original Software and there will be no charge for fixing any reported problems. However APCS reserves the right to charge for enhancements or new versions and for all repair or maintenance of the Software arising from your modification of the original design or code of the Software. The Software is designed to be applied without change to the design of your organization's Lotus Notes databases but you are free to modify the Software and adapt it as you see fit for incorporation into your databases however any such modifications or adaptations and the effects pursuant to them are not part of free support 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.

Implementing NotesTracker in Your Environment

The **NotesTracker code** distributed in this database is used by your Notes developer to extend the design of the Notes databases for which you want usage tracking to be performed. How quickly and easily this can be done will vary from database to database, depending on the database's design complexity (and how familiar your developers are with its design). Many databases will be adapted in just minutes, or tens of minutes, but some might present more of a challenge.

A **configuration document** (plus an associated configuration view) is also added to each tracked database. By editing each individual database's NotesTracker Configuration document, the assigned NotesTracker Administrator for the database can for that particular database, at any time activate or deactivate usage tracking as a whole and for individual events within the database (Create documents, Read documents, Update them, Delete them, open a view, etc). The updated NotesTracker configuration document will, of course, be replicated across your Domino network just like any other document, and as soon as the updated document reaches each remote Notes database the new NotesTracker options will come into effect there.

For each database that has had usage tracking activated at some point of time: whenever one of the specified types of events occurs (create, read, update, delete¸ paste¸ etc) up 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.

Whenever you decide, a supplied agent can be run to archive old Usage Log entries for historical analysis and reporting purposes (in exactly the same easy and convenient way that you go about archiving your old Notes Mail entries).

NotesTracker has been targeted at measuring database activity initiated by users directly interacting with your Notes databases, via a Lotus Notes Client or via a Web browser. NotesTracker's key Your developers should be able to adapt and 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).

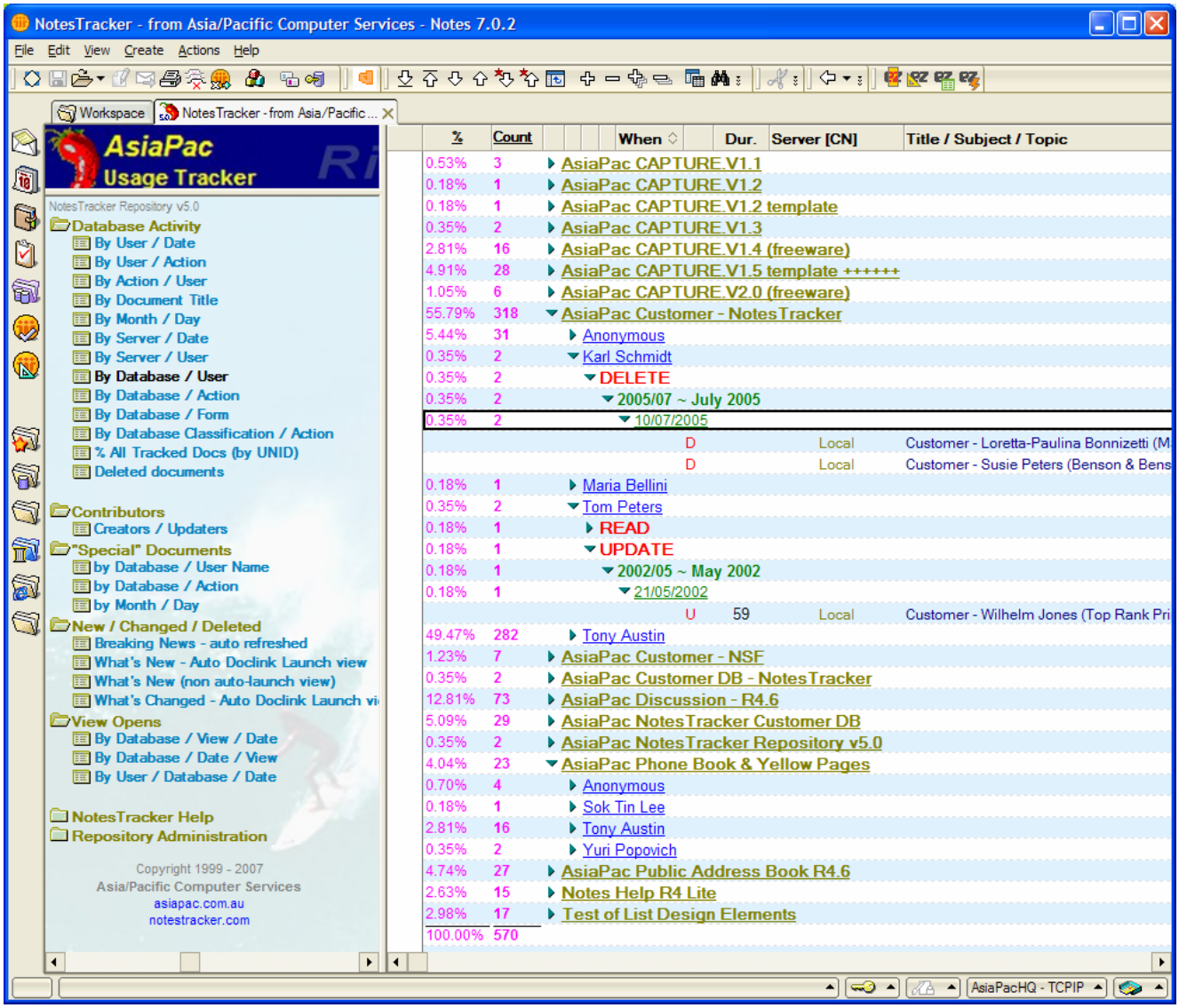
The key NotesTracker functions are all written in LotusScript; while the remained of NotesTracker uses nothing else than regular simple design elements (forms, views, outlines, pages, framesets, agents, etc). This is important, because it enables a single NotesTracker installation package to run in any workstation or server platform that supports Notes and Domino. The C API and C++ APIs were not used for NotesTracker, and neither was Java, because these languages are considerably more complex to write and deploy across all platforms. An additional disbarring consideration for Java is that it does not support the "front end" User Interface (UI) – only the "back end" server interface.

Even though LotusScript is used for the key NotesTracker functions, this does not mean that your developers necessarily need LotusScript skills, since it is possible in many instances to deploy NotesTracker with making any changes at all to the provided code. And even if you do want to make some changes, it is likely that a lot of them will only require elementary or perhaps modest skills LotusScript skills.

How you View and Analyze Usage Metrics generated by NotesTracker

As soon as Usage Log documents are written to the NotesTracker Repository database, you can examine them via Notes views provided in the repository. This is a simple yet powerful way to examine and analyze the document (and view) usage patterns and trends in your applications.

The various views distributed with NotesTracker were devised to present the database usage in a number of useful and interesting ways. For example, here is what the "**By Database / User**" view would look like:

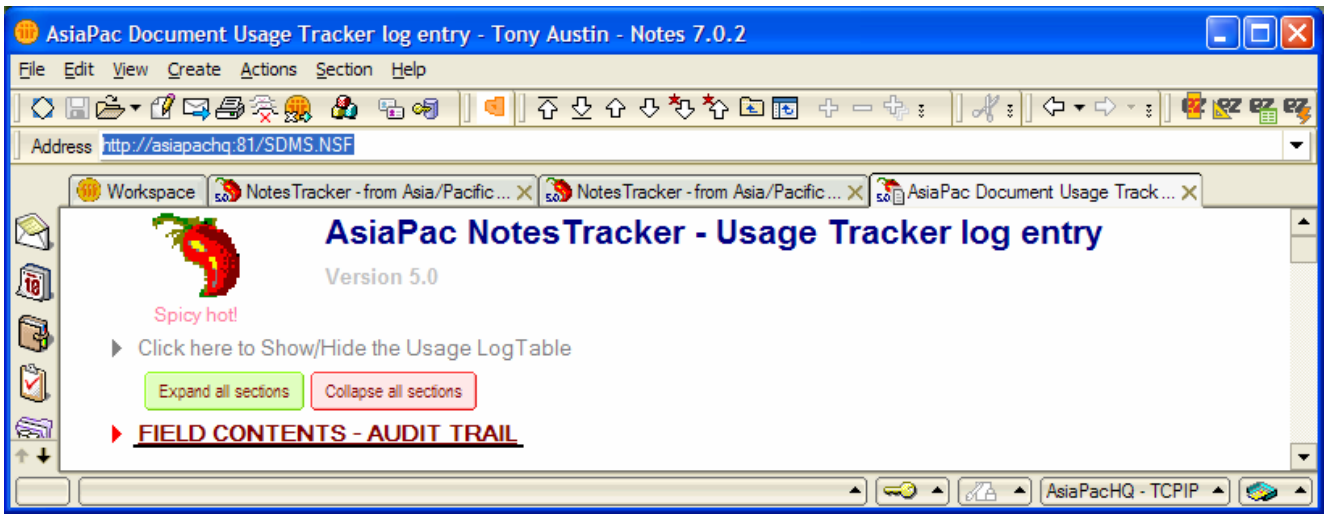


If the supplied views are not sufficient for your analytic or forensic purposes, it will be an easy task for your Notes developers (or even "power users") to build additional views or adapt the provided views so as to meet your needs.

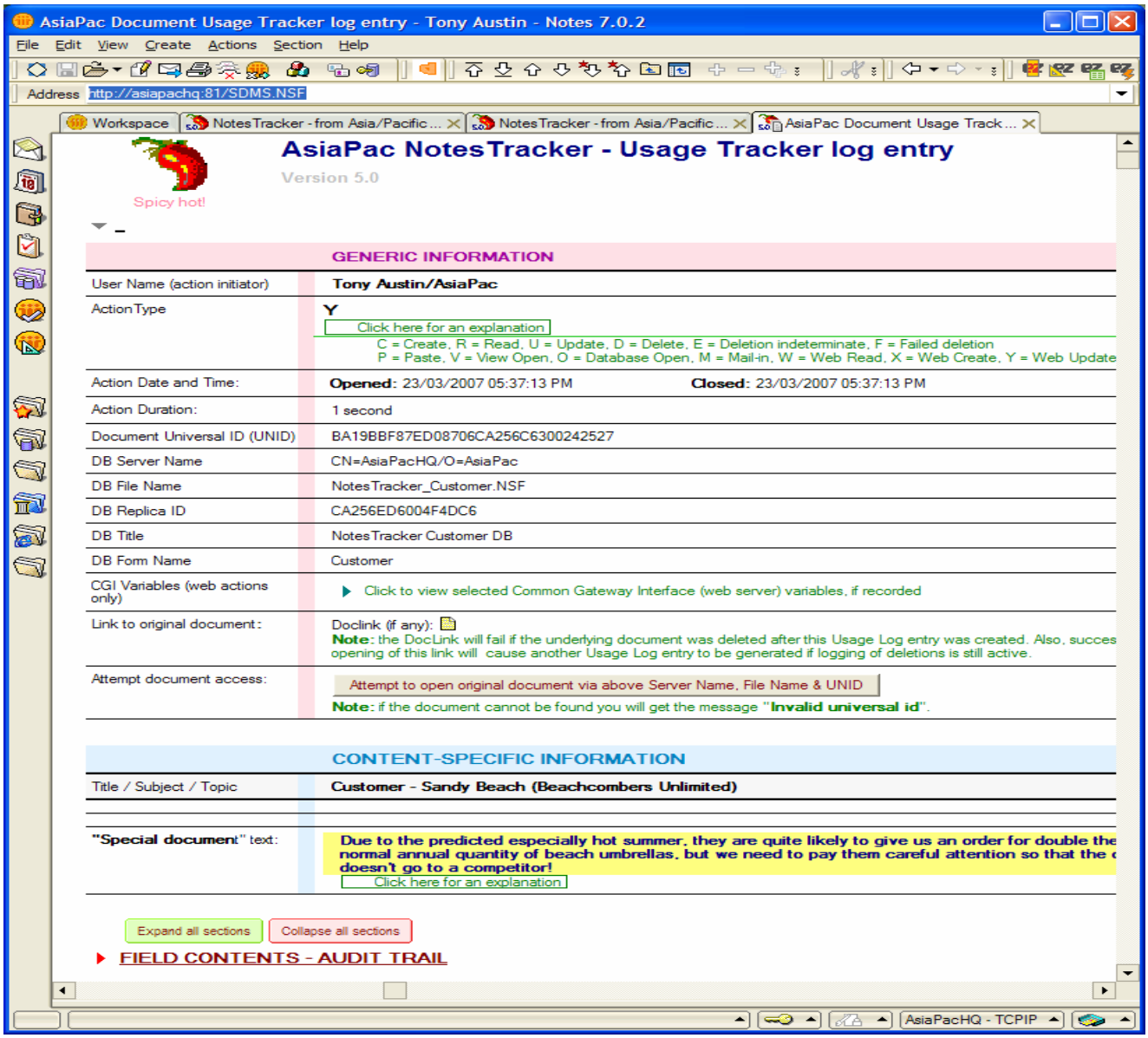
Alternatively, you could use third-party reporting or charting software to provide more extensive usage analyses. There is a brief discussion of this possibility at the very end of the guide, under the heading "Extended Analysis and Reporting".

Displaying the Usage Log Document

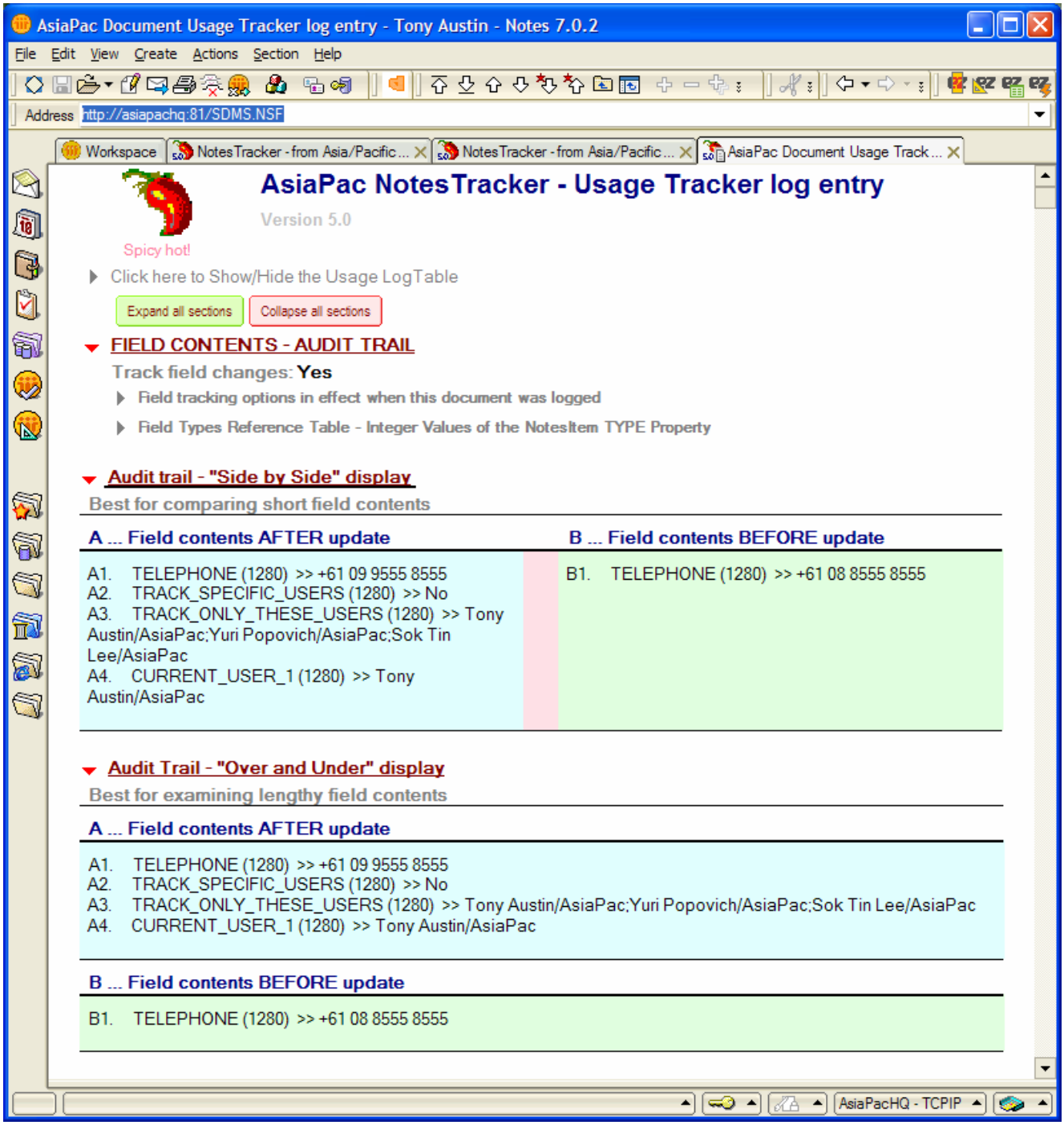
In its most condensed form, with all sections collapsed, a Usage Log document would look like this:



The ability to collapse the top section of the form is a convenience aid added in NotesTracker Version 5.0, and when you click on the top twisty the section is displayed:



If you are examining the fields that were updated, you now may collapse the top section so that there is more room to display the field “Before” and “After” values:



Usage Tracking and Reporting with NotesTracker

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

- 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.

- An optional **repository for NotesTracker entries (or "usage log" entries)** that are generated during the logging of database-related activities when users directly 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 or http://notestracker.com/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** (or "composed" to use Lotus Notes terminology – that is, created as a new document and then saved for the very first time).

- **A DOCUMENT IS DELETED** (removed permanently from the database), or **A DELETION REQUEST IS MADE** (which may fail for various reasons, typically if the user does not possess deletion rights to that database). A single document or a set of documents may be deleted in a single operation. NotesTracker will log the result for each individual document.

- **A VIEW IS OPENED** (in the Lotus Notes client) – intended to be used for once-off or occasional analysis of view usage.

- **A DATABASE IS OPENED** (new in NotesTracker Version 5.0).

- **A DOCUMENT IS PASTED** (new in NotesTracker Version 5.0). A single document or a set of documents may be pasted in a single operation. NotesTracker will log the result for each individual document.

- **A DOCUMENT IS ADDED VIA MAIL-IN** (new in NotesTracker Version 5.0).

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.

Usage Tracking for Web Browsers

The ability to track Web browser accesses was one of the major features added to NotesTracker in Version 4.o (prior to which only Notes Client accesses were tracked):

With a few simple additions to a database's design, NotesTracker will track the creation, updating and reading of Notes documents via a web browser. The actions are recorded in same general format as in the Usage Log repository 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 yet simple way to track and analyze your Domino (web based) document activity than is provided by some 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. Furthermore, the conventional Web Logs are usually overloaded with a maze of trivial, uninteresting information (such as the names of the myriad insignificant image files which form part of a typical Web page).

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 Web browser

For 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 consideration and not a NotesTracker limitation.)

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

NotesTracker was conceived primarily to track actions performed by a real person acting directly on Notes documents via the so-called "front end" -- the graphical user interface or "UI" provided by a Lotus Notes Client (or perhaps a Web browser).

As at NotesTracker Version 5.0, NotesTracker also tracks some types of actions that are **database-level events** rather than document-editing type of events:

- **Document Deletions** -– commonly initiated at the front end, but can be carried out by back-end processes, and normally are not executed on a document that has been opened. They are recorded as database-level events.
- **Document Pastes** -– carried out at the front end, but not involved with the opening of a document (followed by its closing). They also are recorded as a database-level event.
- **Document Mail-ins** – operate in the back end, adding one document at a time to the database. These too are recorded as a database-level event, but are quite unlikely ever to seriously affect performance.

Note: a single deletion or paste request may involve either a single document or multiple documents, and can be performance intensive since NotesTracker will write one Usage Log document per document deleted or pasted. Be aware that both deletes and pastes are **requests** that might fail (if the user has insufficient rights to the database). All the same, NotesTracker will faithfully log each such request even if it fails. A paste operation occurs synchronously and its result is immediately recordable. On the other hand, we have discovered that deletions occur asynchronously -- even though they might *appear* to be synchronous -- and that the result of a deletion request is impossible to log with absolute certainty. For this reason you will see some deletions logged as "indeterminate" (rather than successful or failed). This seems the best that NotesTracker can accomplish, since it operates at the LotusScript event level. Nevertheless, NotesTracker's logging of deletion events is till useful.

For types of events other than all the above, the fact that NotesTracker routines are structured in a modular fashion could make it easy for you to adapt them for tracking other types of actions, such as button clicks or agents running in the background, on the Notes Client or Domino server. This might be important for you to do in some databases, for completeness of usage metrics where you have the need to track other classes of events (or document actions even when they are not performed via the front 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 license 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).

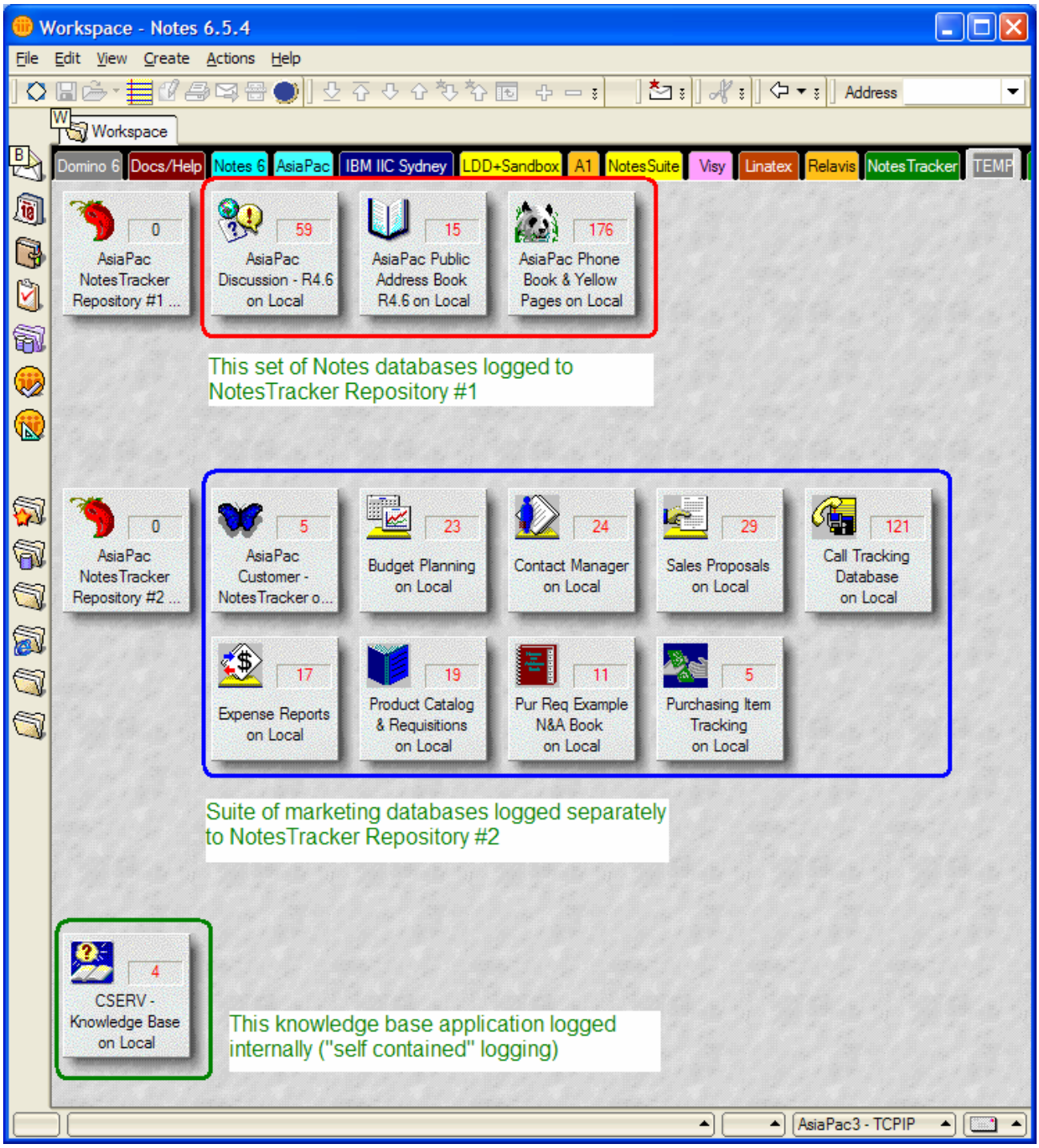
How is this achieved? As fully explained later in the Administration and Development Topics sections, each database that you wish to track via NotesTracker must have its own **NotesTracker Configuration Document**. It is very easy, without any developer intervention needed, to edit the various control values in this document thereby changing the nature and extent of usage tracking (for that individual database) from that point of time onwards. And, as soon as the updated configuration document is replicated across your Domino server and Notes Client network, the updated logging control options take universal effect.

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.

The following diagram illustrates this quite clearly:



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.

Note: a feature added in NotesTracker Version 5.0 is the ability to specify a database's application "classifiers." You would classify a database using terms such as Marketing, Support, Administration, Finance, Human Resources, CRM, or whatever is relevant. You can specify one or more classifiers per database. There is an associated new view in the NotesTracker Repository that enables you to **examine all of the actions performed against your databases categorized by classification of application**. See page 69 for more details about database classifiers.

NotesTracker Usage "Reports"

There are numerous views supplied in the NotesTracker Repository database that present the usage statistics in meaningful ways. They a conventional or “plain vanilla” Notes views, so it will be a simple matter for your Notes developer -- or even a "power user" -- to add further views that present information in other meaningful ways (or to remove unwanted views to reduce Domino server overhead).

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 analyze 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 recognize 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. (This is discussed later, under Administrator Topics and especially Developer Topics.)
- **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. However, it’s much more likely that only your Notes developer will come across them, occasionally, embedded deep within the NotesTracker code.

Administration Topics

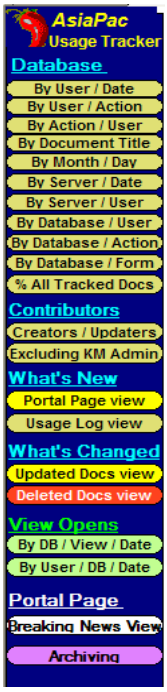
Note: It is important for the Notes developer to be quite familiar with all NotesTracker administration options, too.

NotesTracker Navigator Options

In version 4.4 a new view was added ("View Opens by Database / Date / View name"), and a superfluous view was deleted ("Excluding all KM [admin]").. Also in version 4.4 a LEGEND capability was added to provide some additional online guidance about the NotesTracker Repository views.

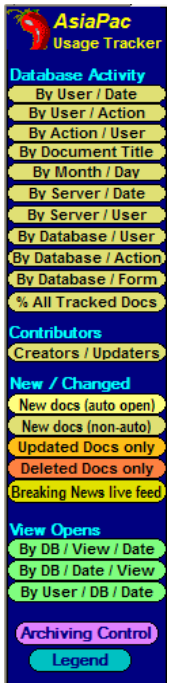
The navigator structure used up to and including NotesTracker version 4.4 was starting to become space-limited and inflexible when additional views and functions needed to be added, therefore a new approach has been adopted for subsequent versions. Starting with Version 5.0 there are alterative ways to set up navigation in the NotesTracker Repository database. These use two alternative (and fully resizable) framesets, are shown on the right in the following illustration. The "Main frameset" has a blue bar right across the op, and the "Main frameset – alternate" has a narrower blue bar.

Old-style Navigator Prior to V5

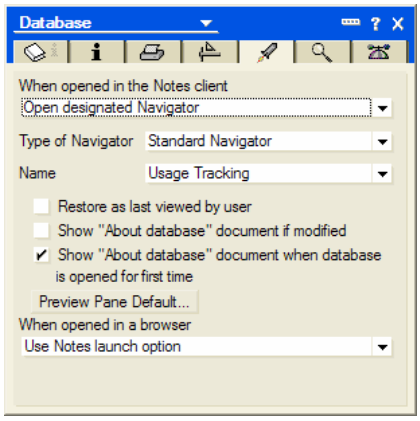


Navigator name: **Usage Tracking**

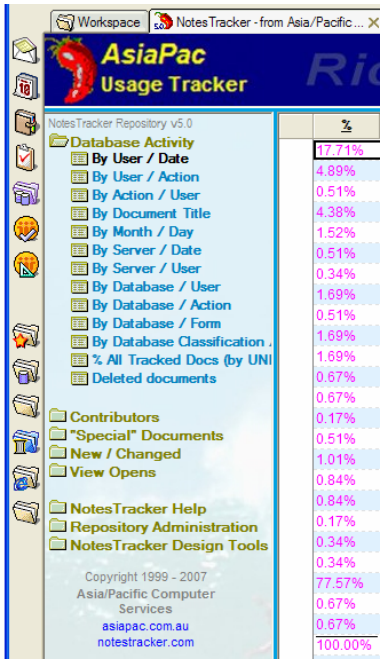
Old-style Navigator V5 onwards



Navigator name: **Usage Tracking**

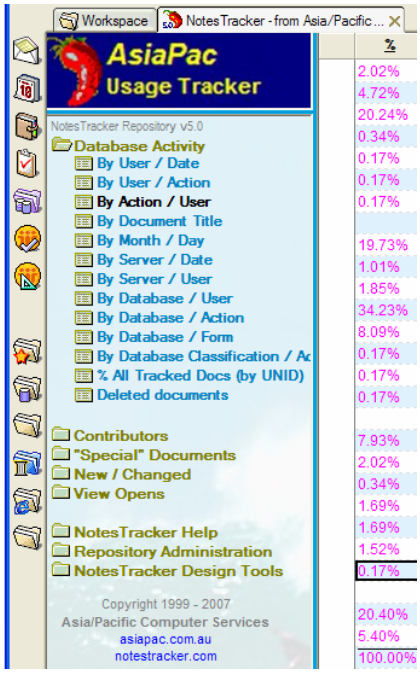


New-style Frameset V5 onwards

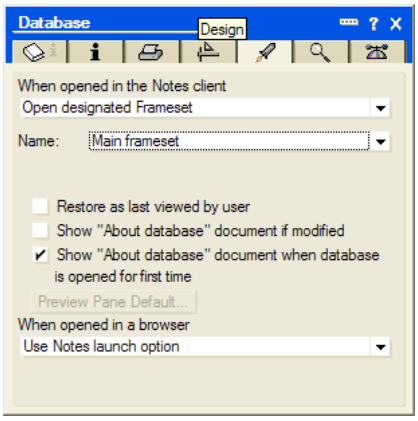


Frameset name: **Main frameset**

New-style Frameset V5 onwards



Frameset name: **Main frameset - alternate**



Starting with NotesTracker Version 5.0 the old views are optionally presented using framesets, enabling additional functions to be accommodated very easily.

The old-style navigator was updated for Version 5.0 but ran into vertical space limitations so that some of the new Version 5 functions could not be accommodated. For this reason, not only is the Therefore the old navigator will not be updated in any future releases or sub-releases of NotesTracker, but also the it should be regarded as being limited (compared with the framesets).

Not only do the new-style framesets look nicer than the old-style navigator, but also they offer better design flexibility (and reduced designer time) to modify than does the old graphics-based navigator. (The latter is very "fiddly" to adjust because of its dependency on precise positioning of the many graphic elements.)

There are two three-pane framesets to choose from, named "**Main frameset**" and "**Main frameset – alternate**". They are identical in function, the only difference between them being the layout of the "AsiaPac Usage Tracker" logo at the top of the screen.

The panes/frames are fully resizable, enabling you to move them around in order to maximize the space available for displaying the Usage Log views.

This frameset-based approach offers better control over screen layout (or "real estate") than the old-style navigators provide. Also, the framesets are more web-compatible so that web browser users will see almost the same as do Notes Client users.

How to Select the NotesTracker Target Repository for a Database Being Tracked

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:**

1. Via its Replica Identifier, normally shortened to "**Replica ID**" (independently of the operating system's directory structure)

2. Via its **Server, Path and Filename** (that is, via the operating system's directory structure relative to the root Domino data directory)

3. **Internally** (inside the current Notes database itself)

There are certain advantages and disadvantages of each alternative, which we will shortly consider in turn.

Using NotesTracker with Clustered Domino servers

You should be aware that, as distributed, NotesTracker 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 would 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.)

Opening the Repository 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: It may be easiest if you place the NotesTracker Repository database in the server's root Notes data directory (such as C:\Program Files\Lotus\Domino\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 realizing 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 analyzed for an overall analysis of usage.

Opening the Repository 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.

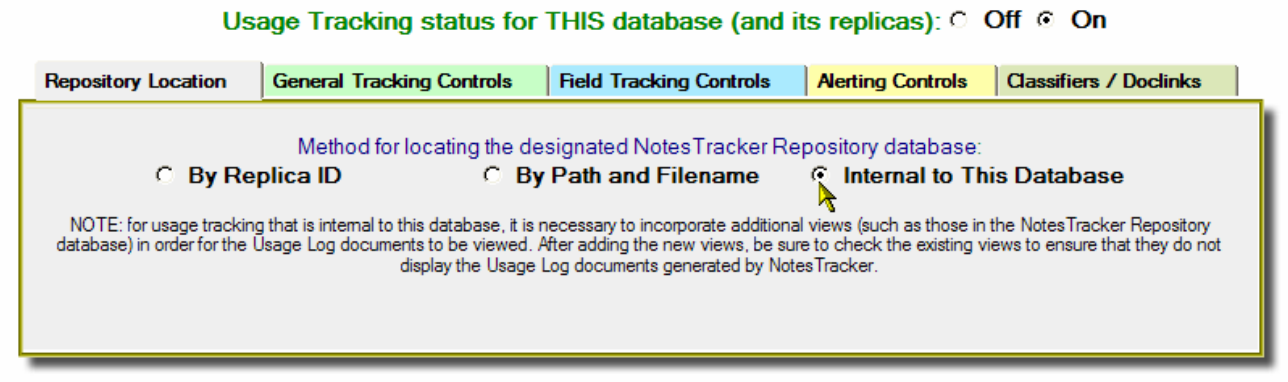
The main disadvantage is that opening by Path and Filename is inflexible, compared with opening via Replica ID, since it will only succeed when you specify the exact path and filename. Also, administrators or users may (for various reasons) move a database from its original path on a server (or local 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).

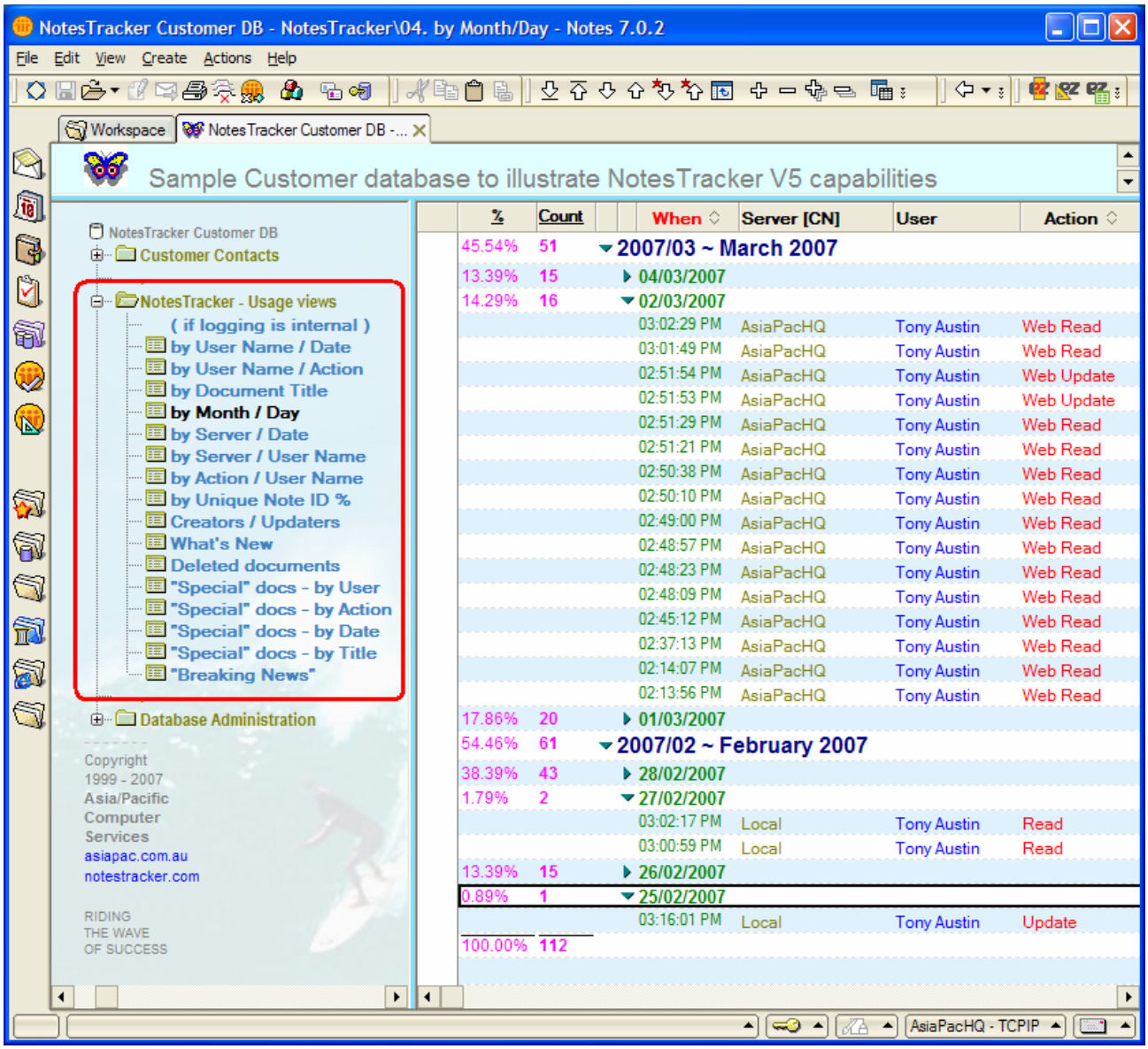
Internal (or "Self Contained") Logging to the Current Database

For a particular Notes database, instead of directing Usage Log entries to an external NotesTracker repository it might be better to quickly modify that database's design so that if you specify that logging is internal you will be able to view the log entries that NotesTracker writes to the database. This makes usage tracking "self-contained" to that database, and in some cases this is very convenient (See, for example, the SDMS and CAPTURE free database applications that are downloadable from http://asiapac.com.au/ or http://notestracker.com/)



Modifying the design will be quite easy. Essentially, it will be just a matter of adding a few Usage Log views (maybe even just a single view) and updating the database's navigator to include the added views.

Here's an example of a database with Usage Log views added and navigator updated (as circled in red color): it's the **NotesTracker Customer DB** that is part of the NotesTracker Version 5.0 distribution package:



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.

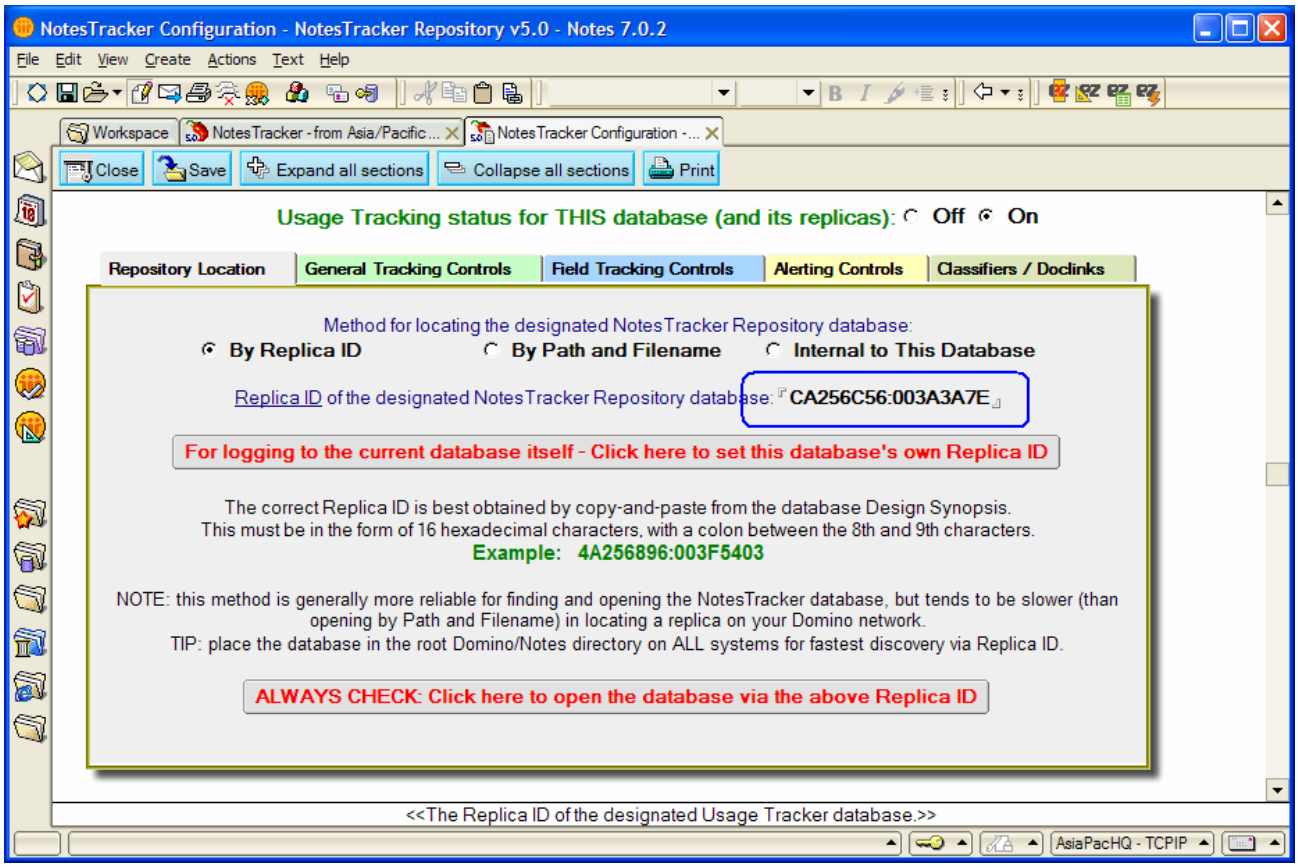
On the other hand, the current database's size will grow – perhaps rapidly -- due to the additional documents being added. Also, the view indices will be built for the views you add to display the usage metrics. The impact on extra disk space occupied by the database and extra processor cycles must be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

Using Multiple NotesTracker Repositories, for Different Sets of Applications or Different 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.

Security, Access Control and Privacy Considerations

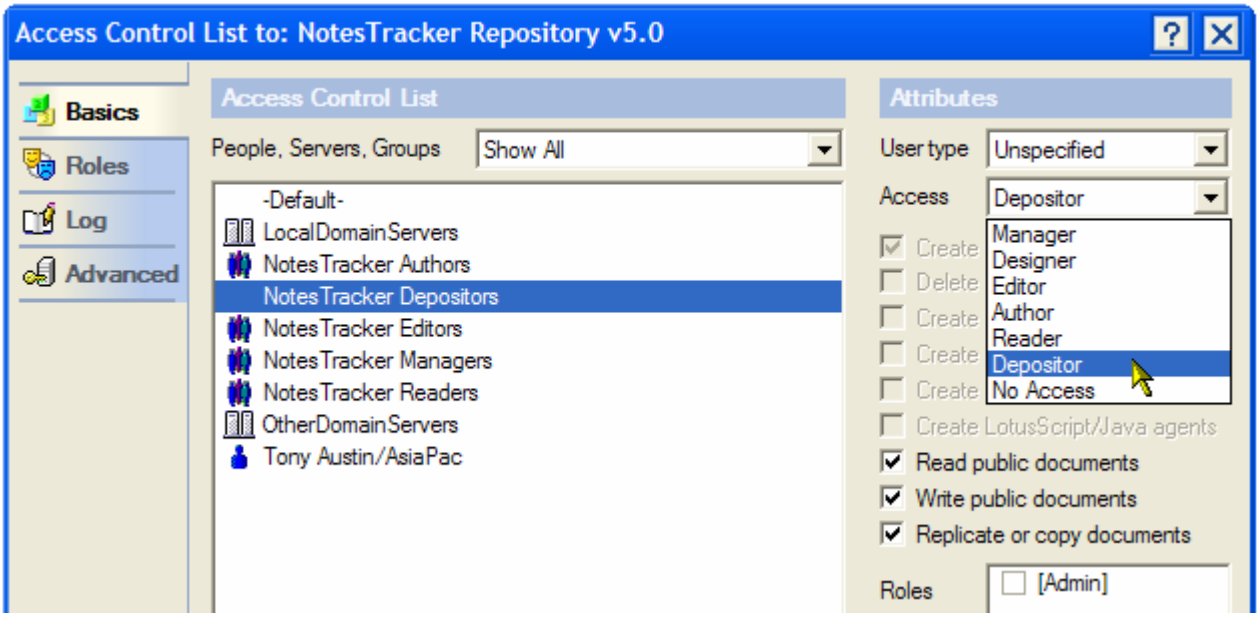
The distributed NotesTracker Repository database has the following Access Control List (ACL) settings:

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

  Usage tracking will not operate unless Usage Log documents can be added to the NotesTracker Repository database. This is what happens implicitly, without the user being aware of it, for every tracked action (Create, Read, Update, Delete, Paste or Mail-in of a document, and opening of a View or Database).

  For NotesTracker logging to succeed, a basic requirement is that each user (or background agent) involved must have the authorization level needed to **create documents** into the repository. An alternative would be to give them the authorization to **deposit documents** into the database

  You might decided to create a person groups for this -- say, a group named "**NotesTracker Authors**" (with Create authorization) or “**NotesTracker Depositors**” (with deposit authorization). Such groups might be used against all NotesTracker repositories, but you might need more fine-grained control and therefore set up groups specific to each particular repository.



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 or those with the [Admin] role 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.

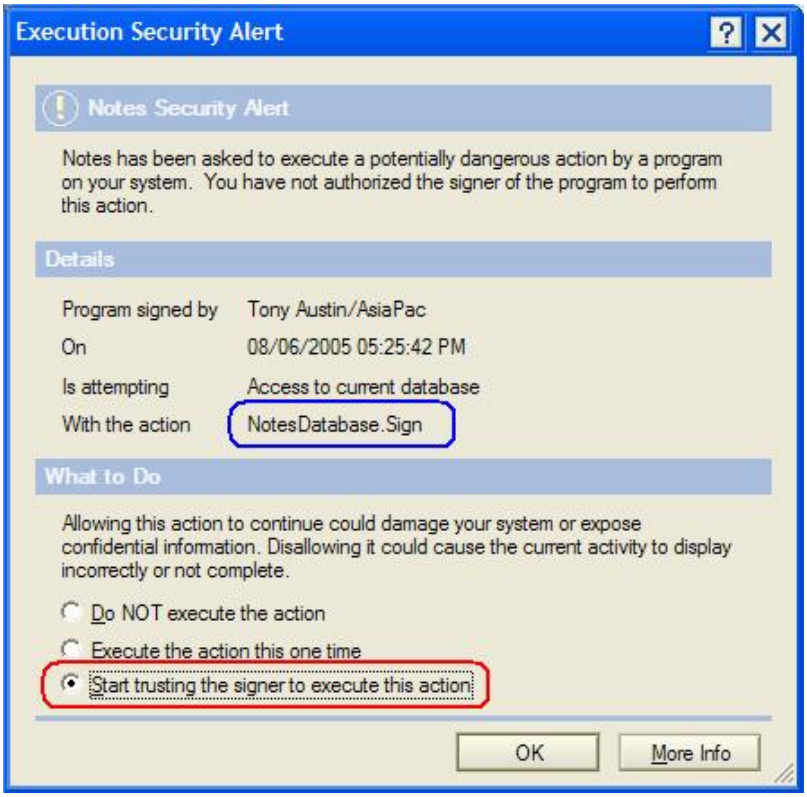
Authorizing the NotesTracker Agents by Signing Them

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. (The same authorization considerations may apply to other agents used by NotesTracker, such as the **Paste** and **Mail-In** agents, even though these do not operate in a Web environment.)

It is beyond the scope of this document to go into details about the need to **sign** an 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.

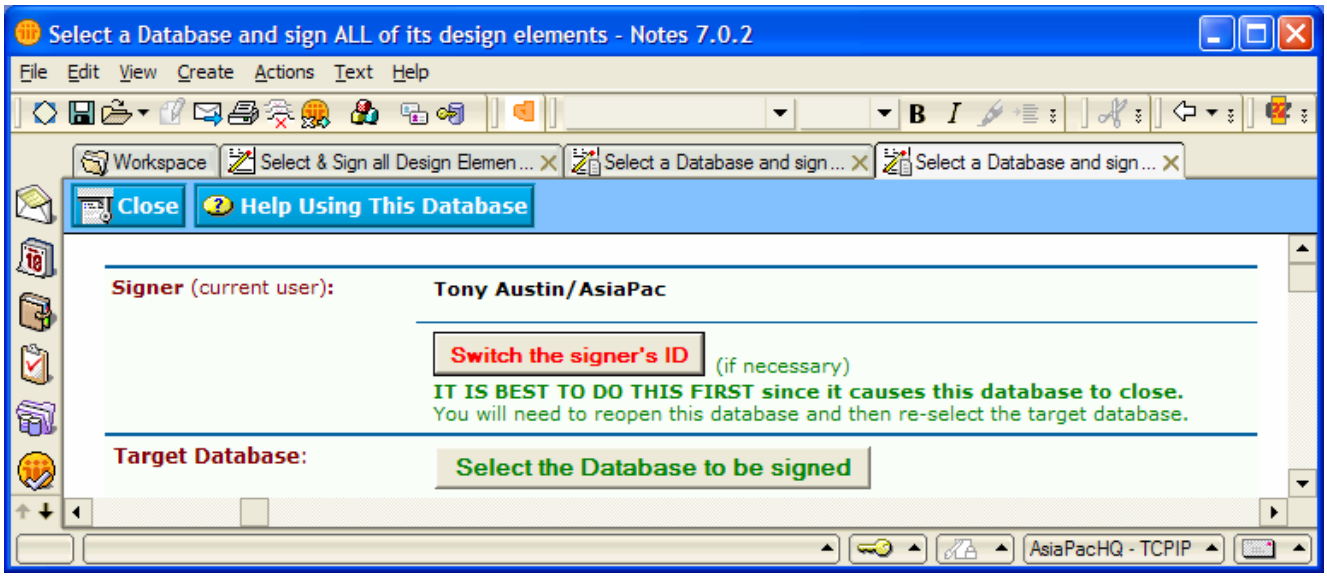
Alternatively, if you are using Domino Designer 6 or later you can sign each agent using the button in the Agent List design view, as follows:

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).

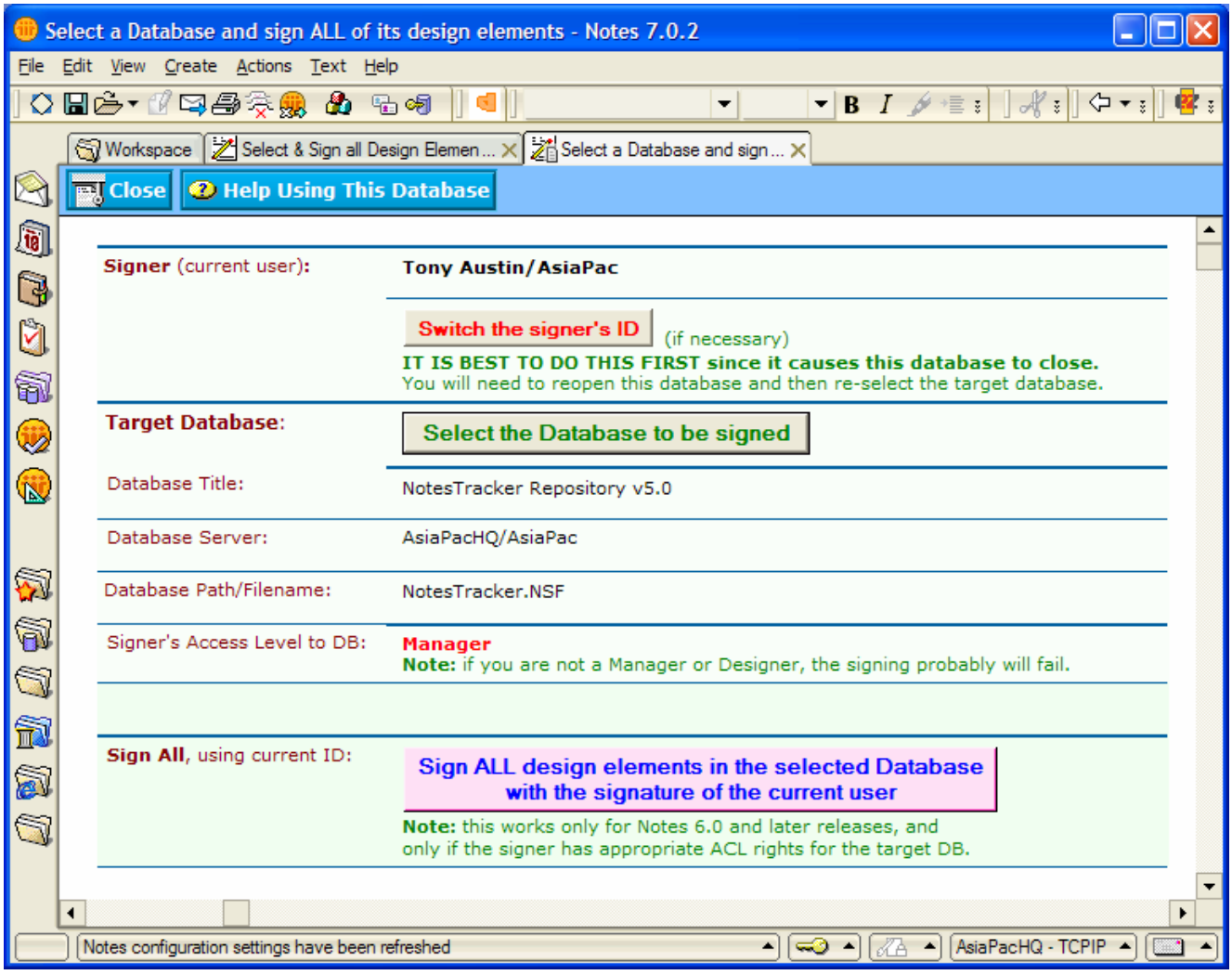
The "Simple Signer" Database Signing Tool (Free)

An alternative solution was released by Asia/Pacific Computer Services in April 2006. It is an easy-to-use utility designed purely for signing Notes databases. You can download this free "**Simple Signer**" tool from either http://asiapac.com.au/Simple_Signer_Download.htm or http://notestracker.com/Simple_Signer_Download.htm

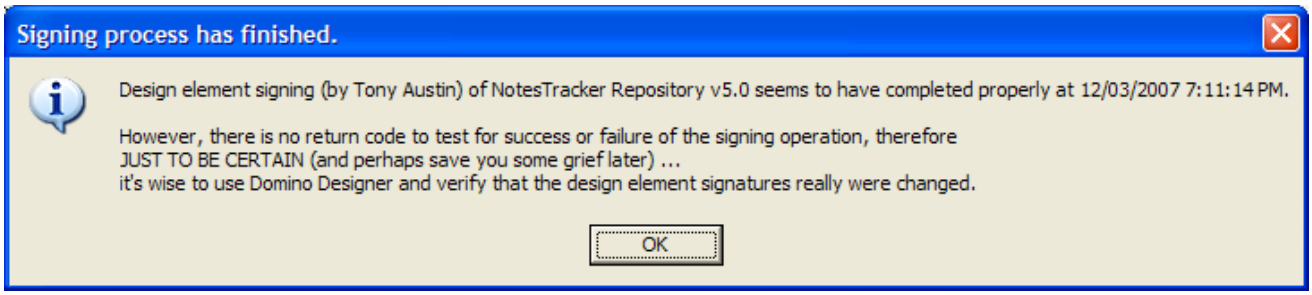
It is a Notes database that you merely open, whereupon you are presented with a simple dialog for selecting the ID of the signer as well as the target database (the Notes database that is to be signed). You then only have to click a button to carry out the signing process. In action, the Simple Signer will look something like this:



Click on the "**Select the Database to be signed**" to get something like this:



After clicking on the purple “Sign ALL design elements in the selected Database…” button, the signing will take place (but only if you have the necessary authority to do so). After that, you should see a dialog box like this:



There is more information about using the Simple Signer database in its “Help Using This Database” document.

Note: The simple Signer database’s design is not hidden. Therefore if you do not wish to sign **all** of the design elements with the selected Notes ID, you should either use one the previous methods or else modify the LotusScript code in the Simple Signer to sign whichever design elements that you want.

Security and Privacy Considerations for Administrators and Developers

IMPORTANT

NotesTracker 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 for Tracked Database 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.

PRIVACY NOTICE (SAMPLE)

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 allows us to 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.

NotesTracker Configuration – Starting Off

The NotesTracker Configuration Document

A **single** NotesTracker Configuration Document is needed in each database that is being tracked. It is used by each and every NotesTracker routine (whether running on a Notes Client, a Web browser, or as Domino server-based agent).

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.

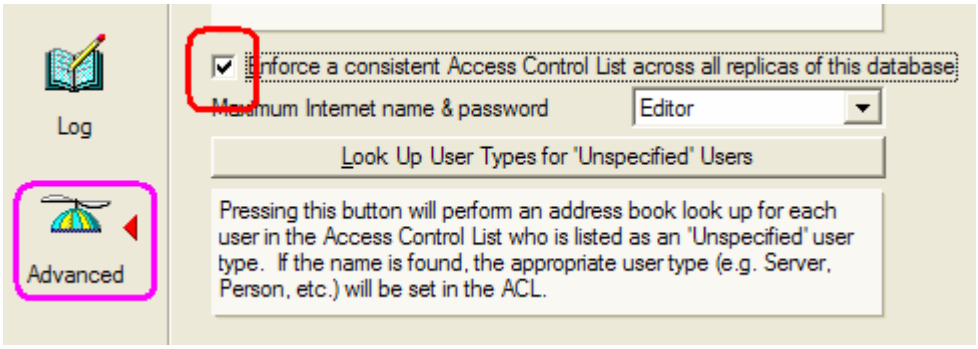
You are prevented from manually creating more than one such configuration document per database. However, experience has shown that multiple configuration documents can sometimes be present, presumably due to replication faults. It is safe (and probably wise) to delete superfluous configuration documents,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

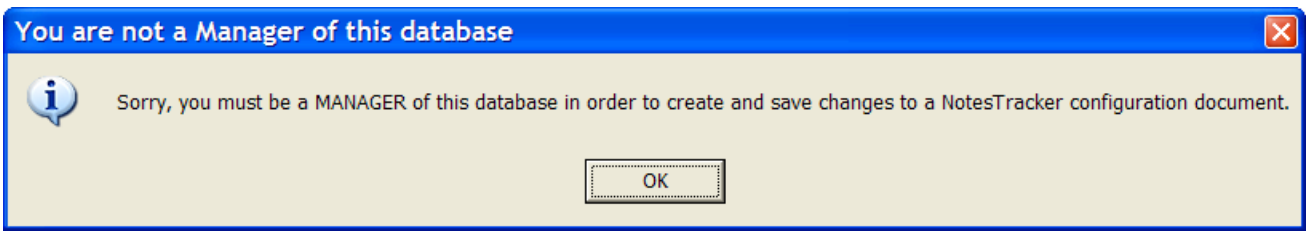
Therefore, only those with Manager level access in a database's ACL are allowed to edit the NotesTracker Configuration Document. (This is controlled by code in the Querysave event of the NotesTracker Configuration form.)

Note: following normal Notes behavior, 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:



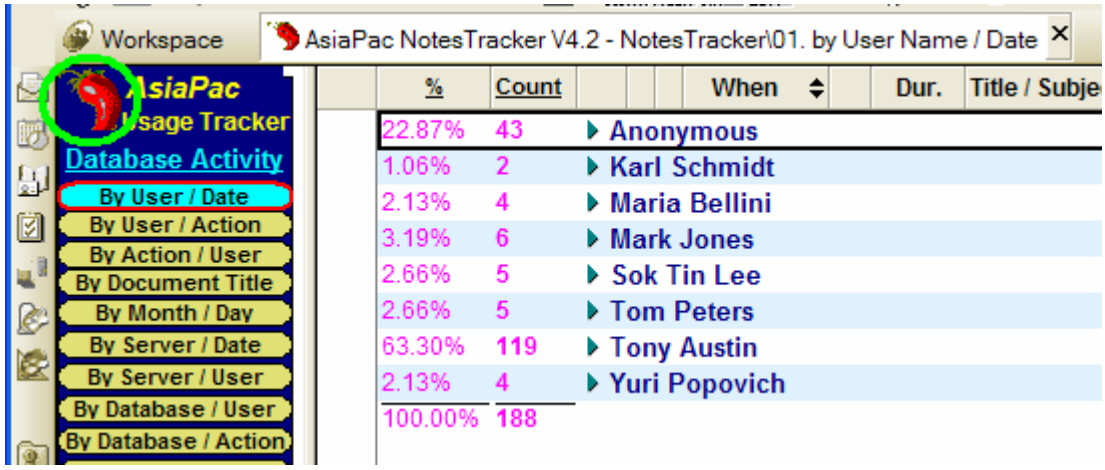
If the database is opened locally without the enforcement of a consistent ACL, then a user doesn't need Manager access level to edit the configuration document. Any changes they make would most likely be replicated around the network to other replicas of the database, which would not be a good situation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

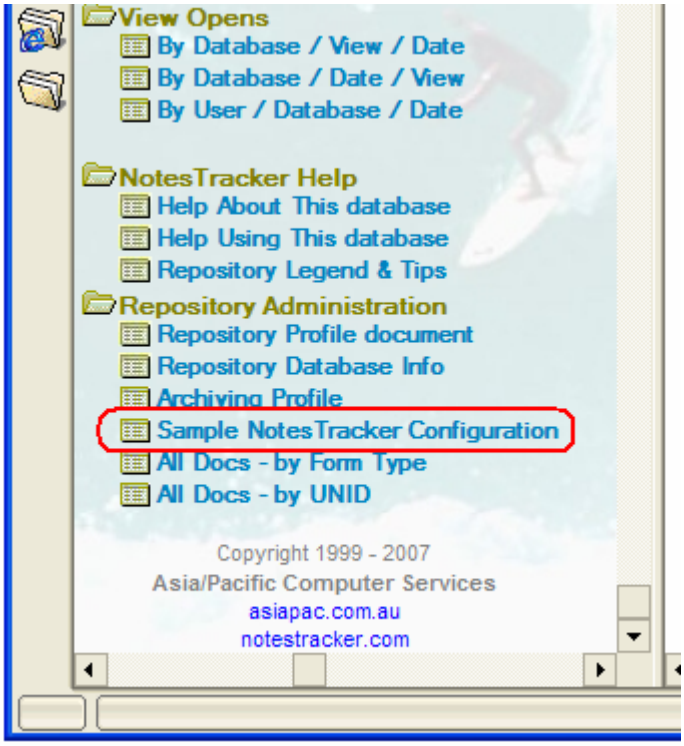


NotesTracker Configuration View

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



Starting with NotesTracker Version 5, using the frameset approach explained above you will find a visible selector for switching to the NotesTracker Configuration view (as circled in red in the following illustration):



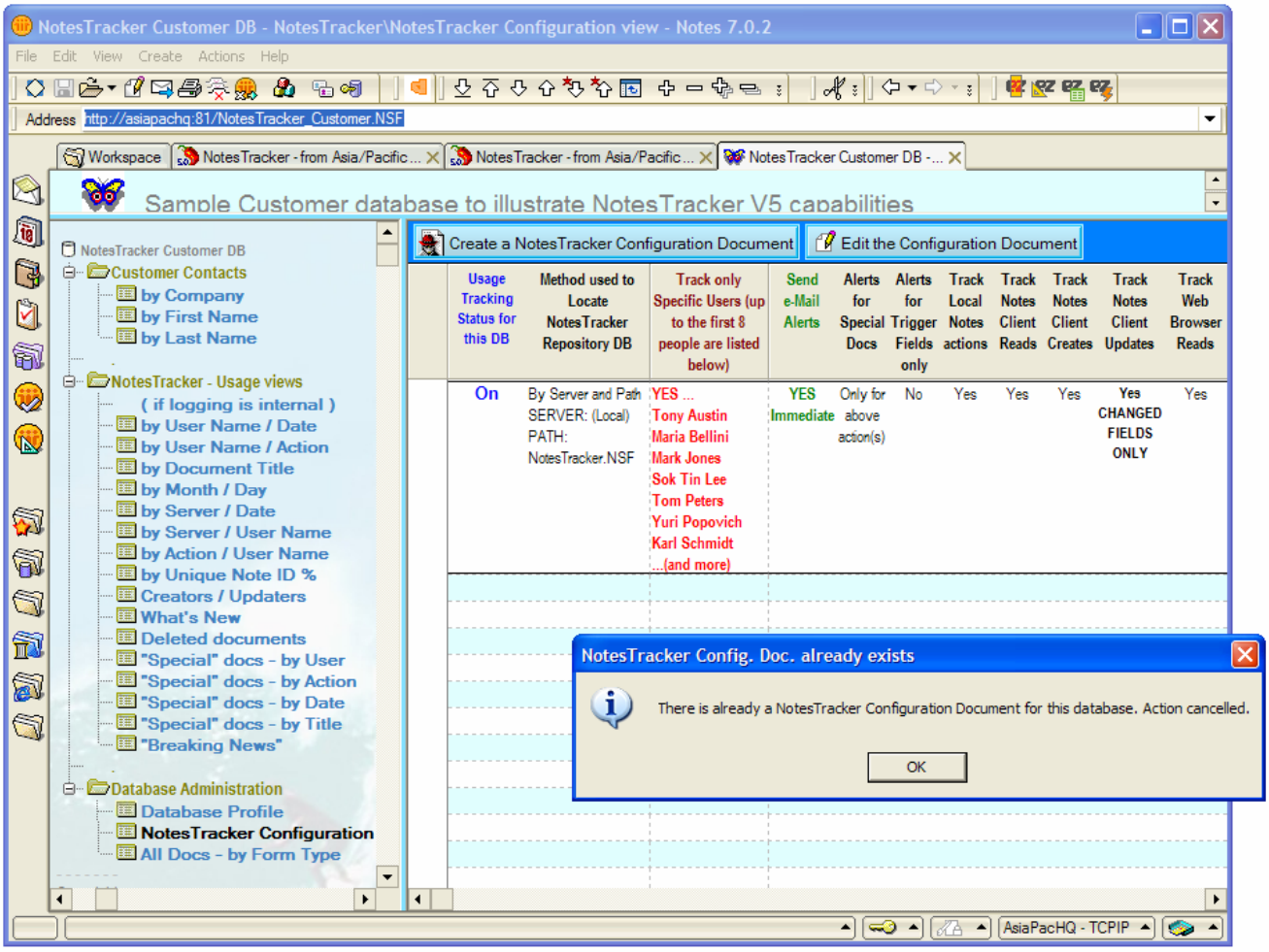
Tip: you should ensure that usage tracking is kept switched off for each and every NotesTracker Repository database. There is no point whatsoever in tracking usage in a NotesTracker repository. In fact, it will produce unwanted system overheads when you do such things as deleting multiple Usage Log entries (say, during an archiving operation) with tracking of document deletions active for the repository.

How you get to the NotesTracker Configuration view in each of your other Notes database applications is entirely dependent on how your Notes developer has set up the view in that database (discussed later, in the Developer Topics section).

Creating the NotesTracker Configuration Document

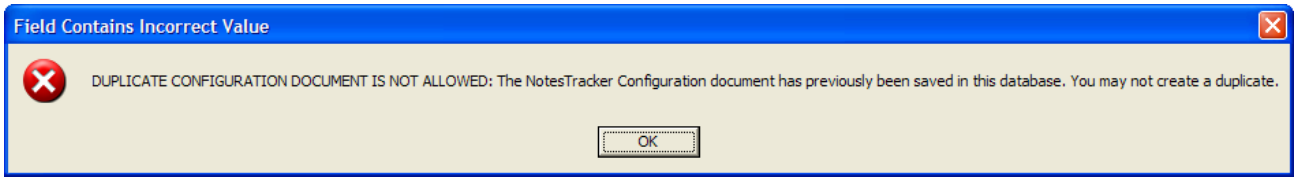
You click the **Create a NotesTracker Configuration Document** button to start the procedure:

If a configuration document already exists, a dialog box appears which advises you that you may not create another one and (when you click the OK button) cancels the action:



Note: it still is possible for duplicate configuration documents to appear later. Most likely this will be caused by replication errors, which might occur occasionally. (Duplicates might also arise from save conflicts, or from document copy-and-paste operations.)

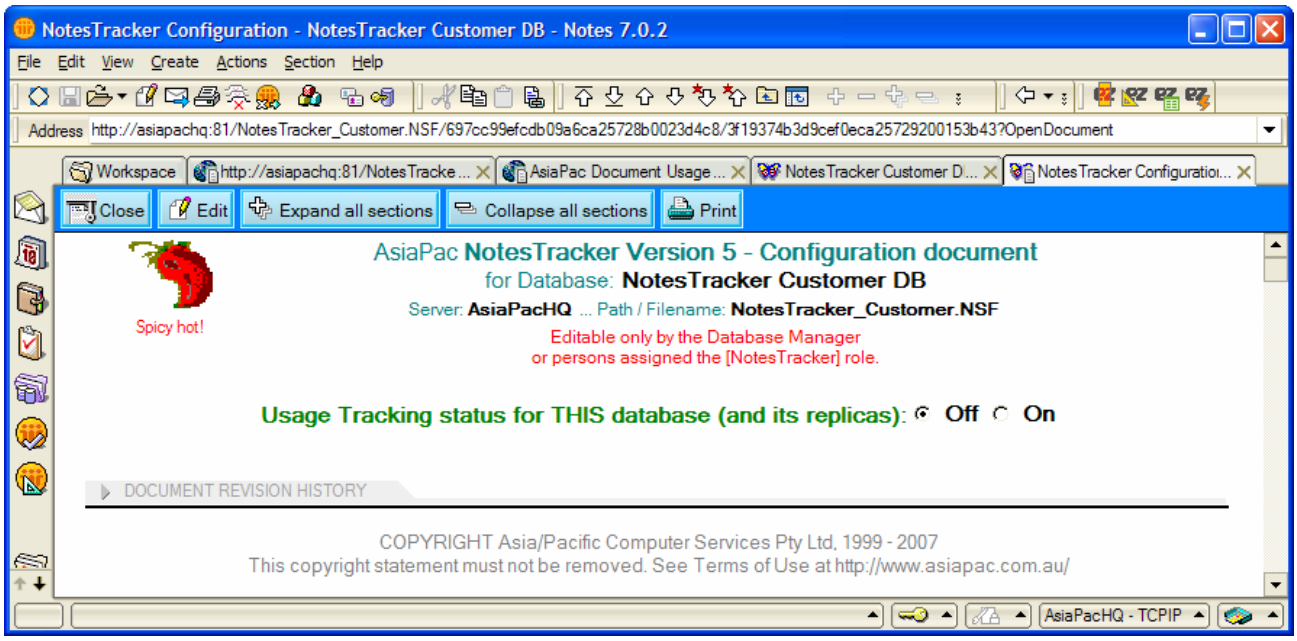
Under certain circumstances you might get the following error message immediately upon attempting to save a configuration document:



If this arises, merely delete the superfluous configuration document(s), and then check the settings in the remaining one for completeness and accuracy. The NotesTracker code will only retrieve the “first document” in the configuration view, and if there are duplicates this might not be the one you expect.

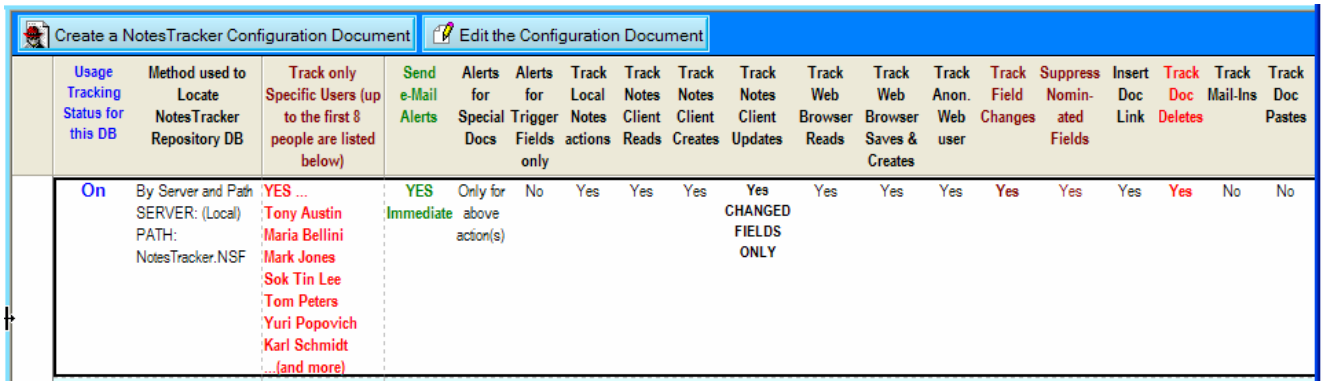
Initial State of the NotesTracker Configuration Document

When you initially create the configuration document, it should look like this:



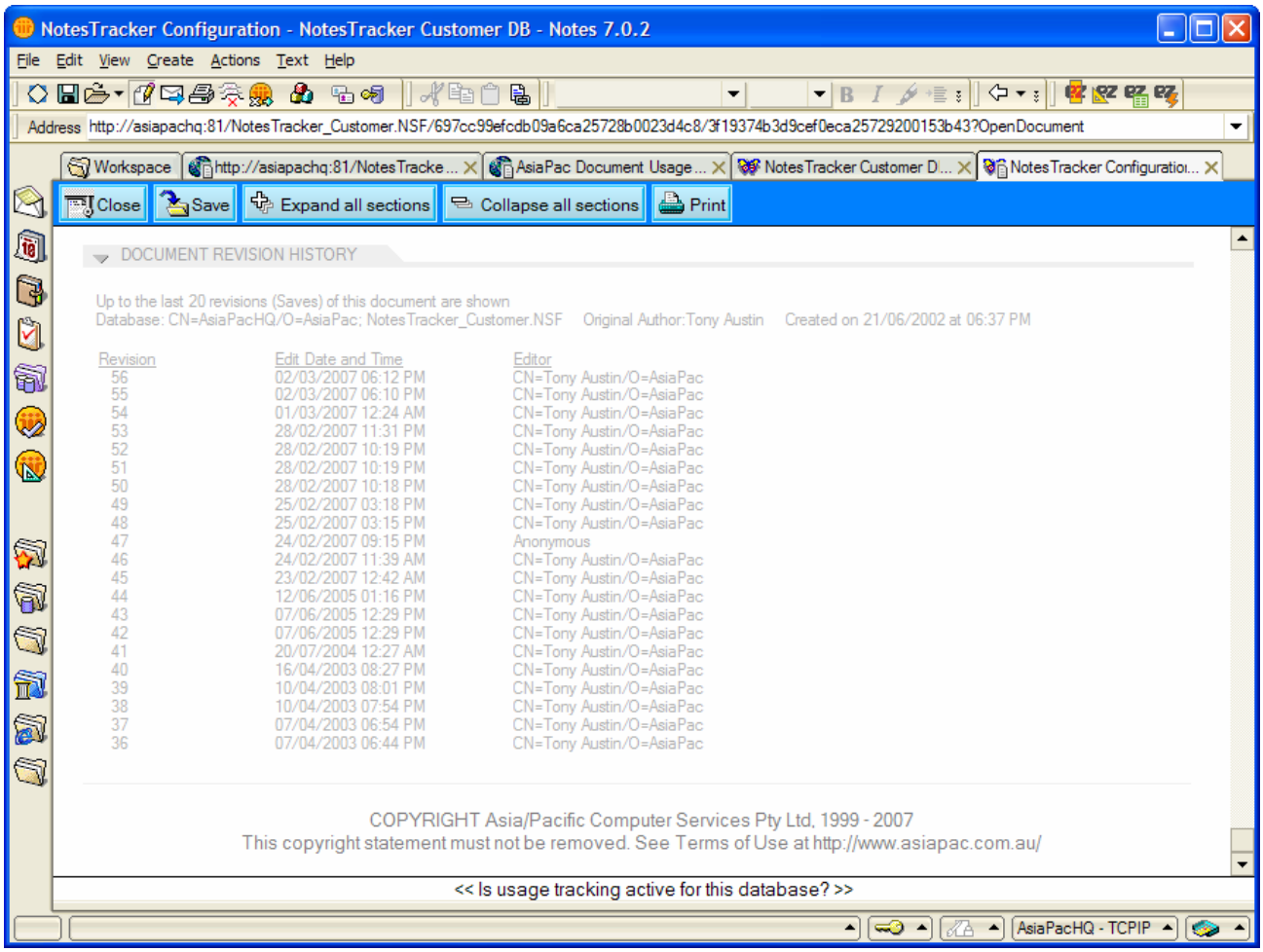
The NotesTracker Configuration View provides a Snapshot of the Configuration Settings

When you change the tracking status from Off to On, there are quite a few options to be set. The NotesTracker configuration view shows at a glance a subset of these options:



Simple Edit History of the NotesTracker Configuration Document

After you have been making edits to the configuration document, you can see a simple summary of up to the last 20 configuration edits by clicking on the “twisty” at the bottom of the form, thus:



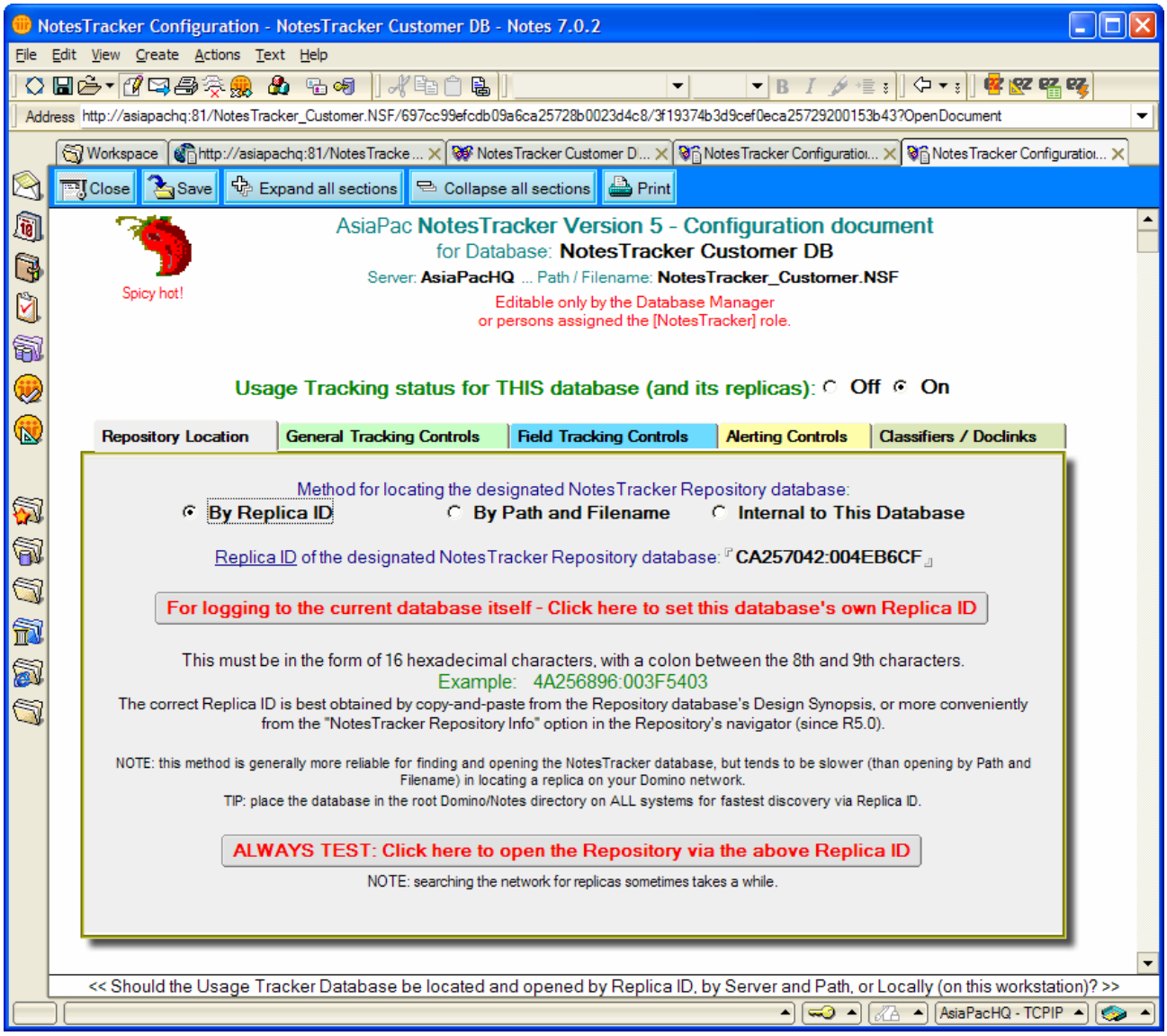
The edit history was deliberately designed with pale grey text in order to remove “screen clutter”.

Note: if you want to, it is a simple matter to modify the design of the edit history footer by modifying the subform called **Simple Edit Tracking Footer**. As an example, a change that you might consider is to alter the value stored in the field “EditHistory_ListLength from 20 to some other value, which represents the number of entries in the edit history list.

If this simple edit history does not meet your requirements, it’s a simple matter to use NotesTracker to provide precise tracking of changes to its own Configuration document! (It should only take your Notes developer a few minutes to enhance the design of the configuration form.)

Specifying the NotesTracker Repository Location by Its Replica ID

You start the configuration process by changing the top radio button from "**Off**" to "**On**" which automatically refreshes the window and displays the following:

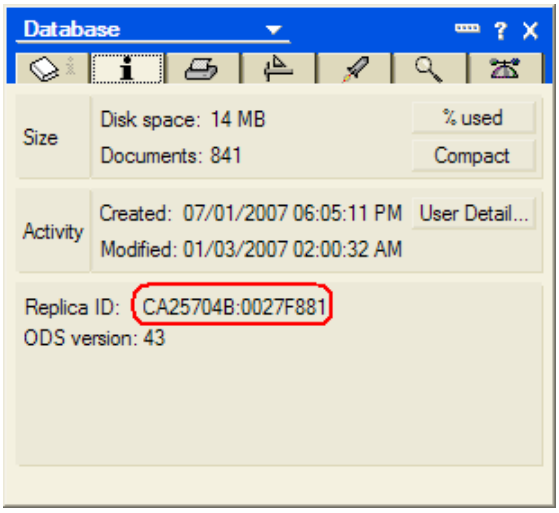


Pragmatically, this seems to be the most reliable way to specify the location of an external NotesTracker repository database.

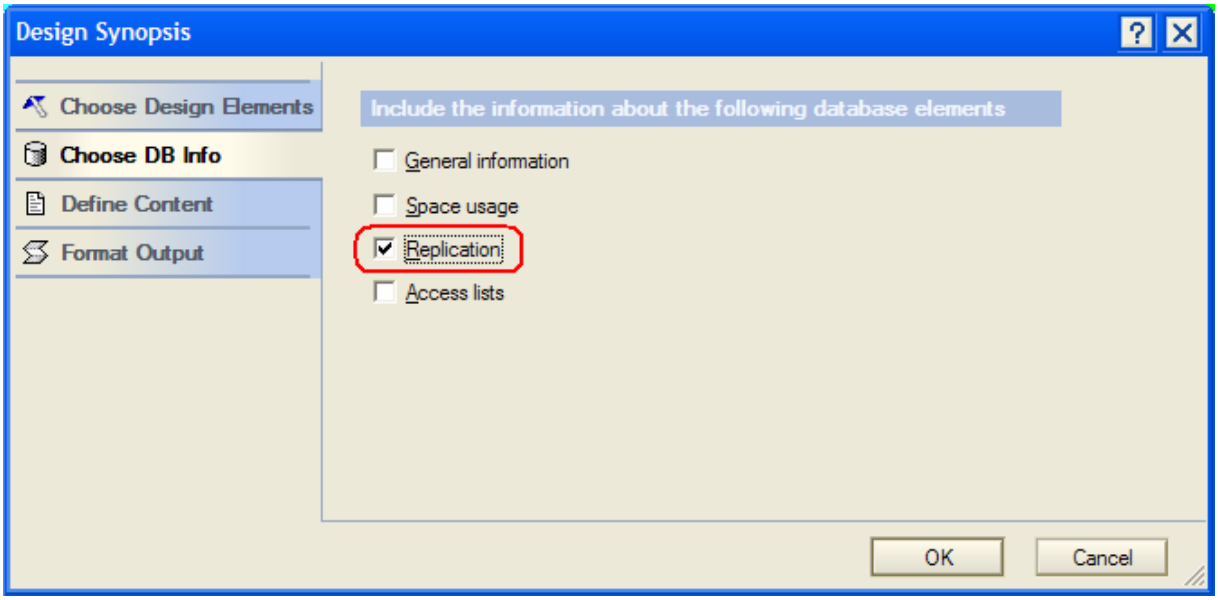
Note: the Replica ID value must be stored in the format expected by @Formula language commands, namely, with eight hexadecimal characters followed by a central colon followed by another eight hexadecimal characters.

You can obtain the correct value for the Replica ID in several ways:

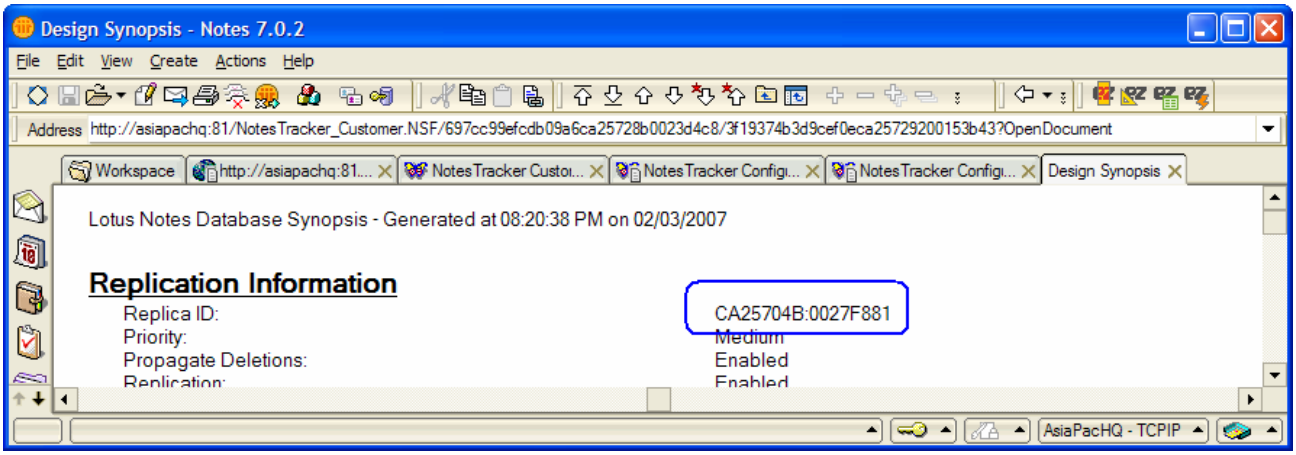
1. By opening (or just selecting) the external repository database and manually transcribing the Replica ID from the second tab of the database property dialog box:



2. By getting the Replica ID value from the database's Design Synopsis:

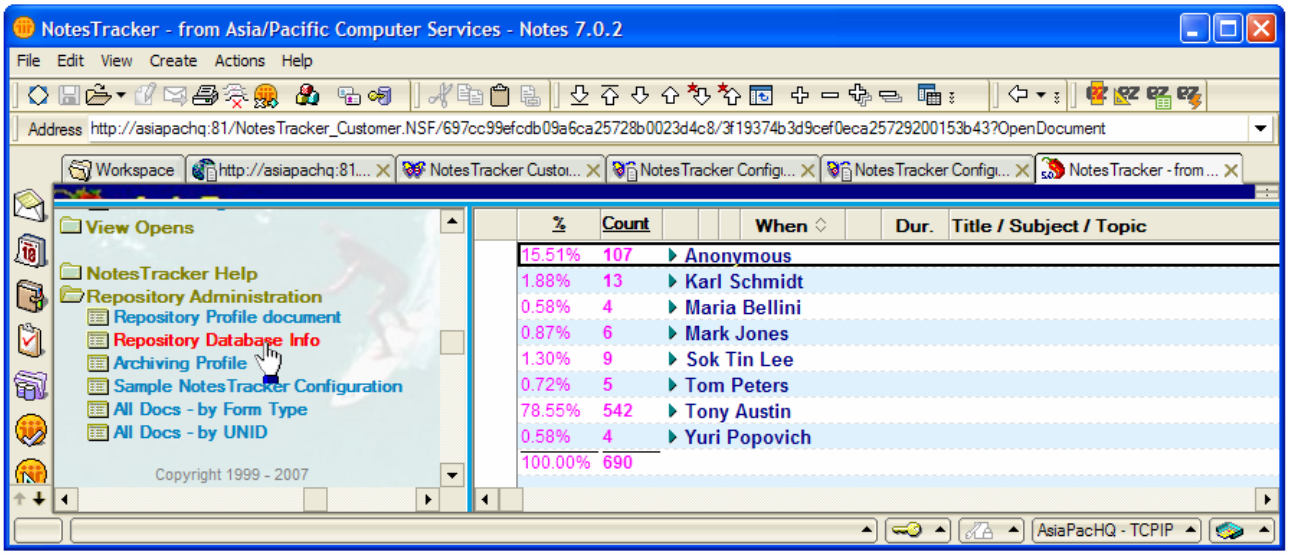


Click the OK button then copy-and-paste the value from the resulting display, like so::

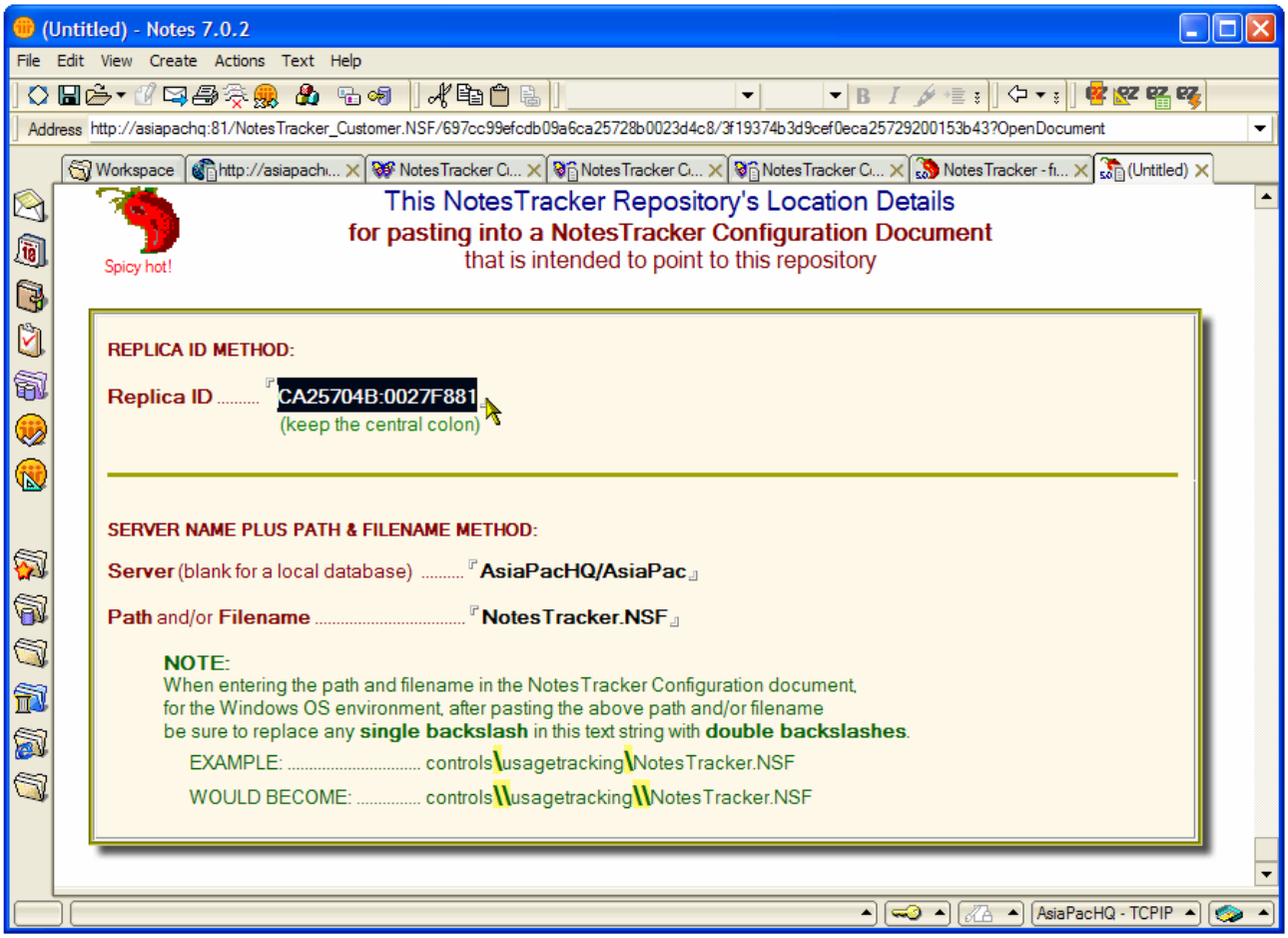


It is highly recommended that you minimize transcription errors by using Copy-and-Paste of the Replica ID value, as shown in the reverse-highlighted section in the following figure:

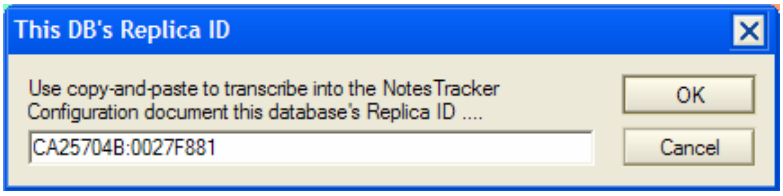
3. By opening the database, and clicking on the **Repository Database Info** option, as follows:



Copy-and-paste the repository's Replica ID from the resulting display (this includes the mandatory central colon):

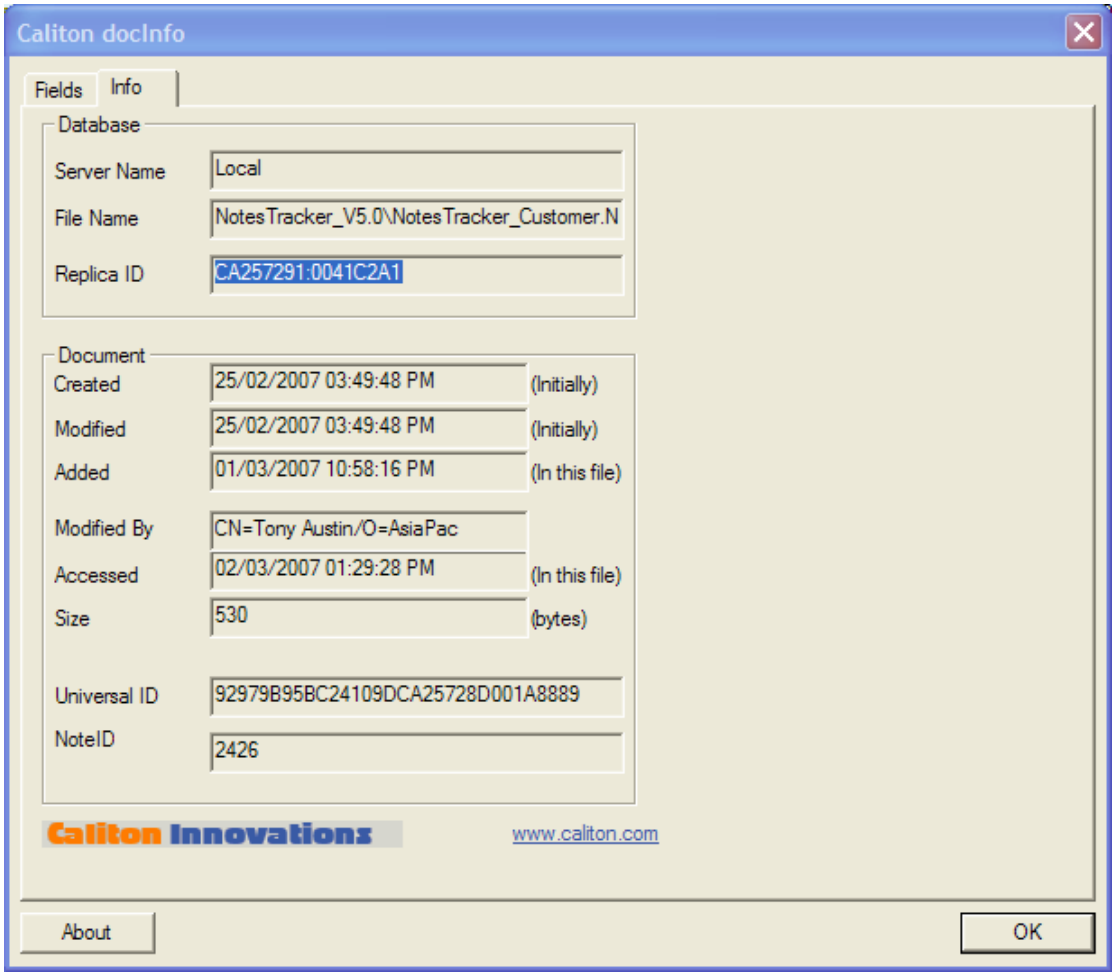


4. By using an agent in the NotesTracker Repository database that is designed purely to display its Replica ID. Simply go to the NotesTracker Repository and run the agent called "**Display this DB's Replica ID**" and you will be able to use the following convenient dialog box for copying purposes:



Copy it to the system's clipboard and paste it into the current NotesTracker Configuration document.

5. By using a third-party tool, such as the (free) **docInfo** from Caliton Innovations at http://www.caliton.com/Lotus-Notes/Lotus-Notes-Tools.nsf/Lotus-Notes/Lotus-Notes-Developer-Downloads



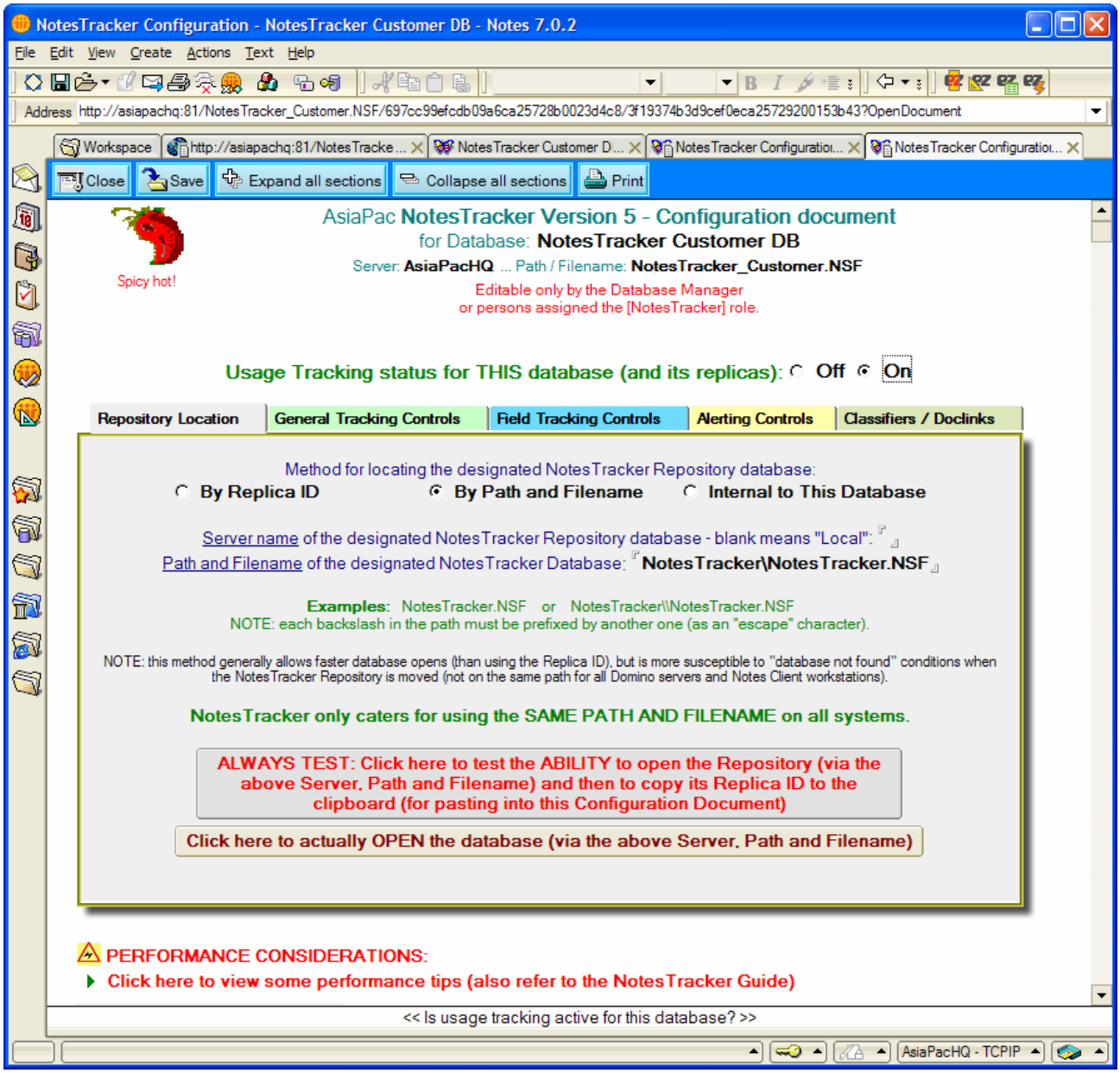
Specifying the NotesTracker Repository Location by its Path and Filename

You can define a database's location by providing its filename (typically in the form xxxxxxxx.NSF) together with the path to that file. The path value is blank or null if the file resides in the Notes workstation's or Domino server's root data directory (which for a Notes workstation is usually referred by the term "local").

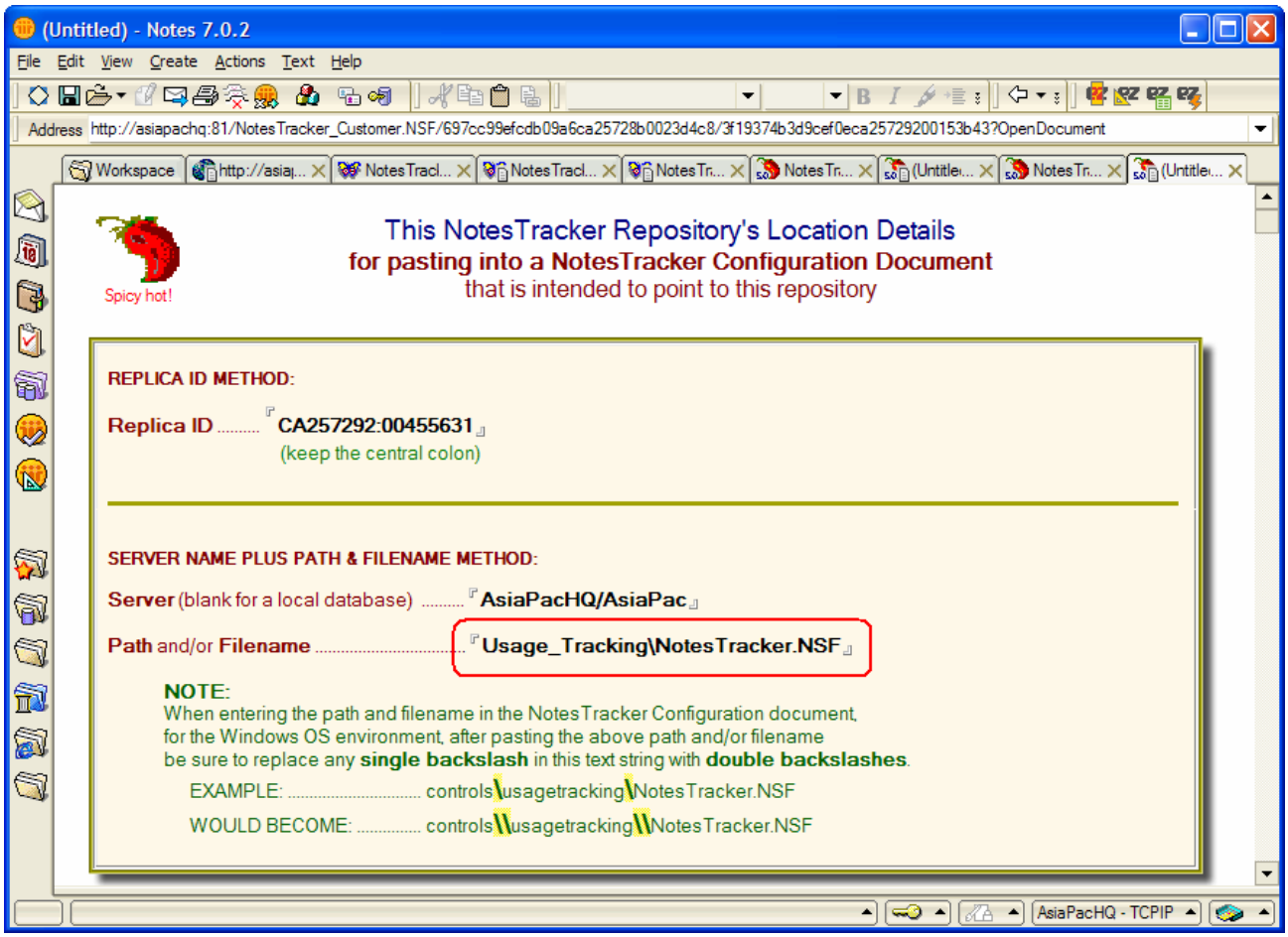
Issues with Specifying the Path and Filename of a Repository

Using the path and filename can be more problematical than using the database's Replica ID for several reasons. Firstly, if the path is not blank (the root directory), you may encounter issues with entering the separator between the various directory levels. Secondly, if the servers are running different operating systems (such as Windows on some and Linux on others) there are issues with whether the directory separator character is a forward slash or a backslash.

As distributed, NotesTracker does not cater for this scenario, because for any one database there is only a single NotesTracker configuration document for all of its replica copies (across workstations and servers). And thirdly, since as just described you have to deploy the database file in the same directory on all systems, you would encounter difficulties if you run out of capacity on a particular drive on a workstation or server and need to move the database to another drive on that system. This is why specifying the database's location by Replica ID is usually the best way.



The bottom section of the form displayed by clicking on the **Repository Database Info** option, as mentioned a little earlier, contains helpful information for this step. For example:



Replication Considerations for Local NotesTracker Repositories

When you leave the server field blank, NotesTracker attempts to use a repository (with the given path and filename) on the user's Lotus Notes workstation (desktop or notebook system). Of course, by definition there is no concept of "local" when you are using a Web browser to access a database that is located on a Domino server.

If the repository database does not exist on the user's local system, NotesTracker will put out a message on the Notes status line and then exit quietly. If the repository database can be located and opened on the user's local system, and assuming the user has the necessary Author or Depositor rights to it. Then NotesTracker will start logging to that local repository.

The same would happen for all users of a given application database, meaning all copies of the database having the same Replica ID and therefore the same NotesTracker Configuration document. This means that the Usage Log documents generated by NotesTracker will be distributed across the multiple workstation systems (of the multiple users of this application).

If you want centralized storage and analysis of Usage Log documents, you will need to control the proper set-up and maintenance of all users' replication settings for their local repository database in order for the dispersed Usage Log documents to be replicated to a central NotesTracker Repository.

This is just a reminder that for "local" repositories you will need to consider the administration of their replication settings. This includes such matters as scheduling of replication, sending Usage Log documents to the Domino server, whether or not you want to receive Usage Log documents from the server, preventing the users from changing the replication settings of their local repositories, and so on. Since all of this is "business as usual" for Notes/Domino administration, it is not the intention of this guide to go into detail about the matter.

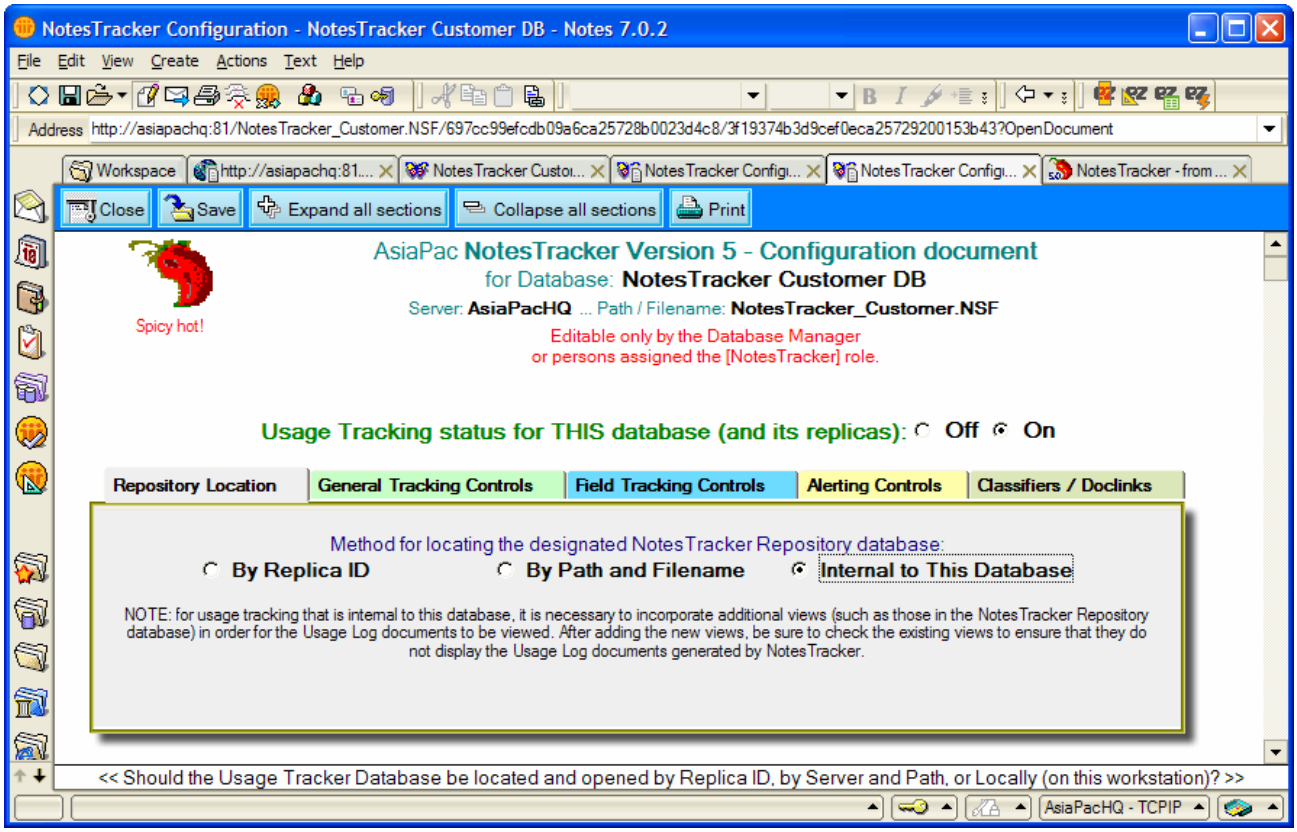
Specifying the NotesTracker Repository Location as "Self-Contained" – Internal to the Database

There is an interesting 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 can always do this just by entering the database's own Replica ID value, and then clicking the button labeled "**For logging to the current database itself – Click here to set this database's own Replica ID**" as follows:

This button is provided to save you the trouble of having to determine the Replica ID by one of the several methods described above.

However, an easier repository location method (added in NotesTracker Version 3.1) dispenses with the need to enter the database's own Replica ID. This option is activated by clicking the Radio Button labeled "**Internal to This Database**", as the following example shows:



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 may avoid the need to deploy a separate NotesTracker Repository database, if you haven't already done so.

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.

As described later on, in the Developer Topics section, since this option causes the Usage Log entries to be written to the database itself, your developer has to add a view (or multiple views) to the database's design in order for the Usage Log entries to be seen. It is pointless to select this option without any such Usage Log views. And then all of the existing views must be checked and modified if necessary to omit the NotesTracker usage log documents.

Additionally, users who do not have authority to create documents in the database will not cause Usage Log entries to be added internally.

Nevertheless, if these design issues are resolved, it can be extremely handy to have a completely self-contained application, where the NotesTracker log entries are carried in the same database as the application documents.

For example, SDMS and CAPTURE are two popular free applications downloadable from the Asia/Pacific Computer Services web sites (http://asiapac.com.au/ or http://notestracker.com/) are perfect examples of how internal usage logging can be of considerable value.

However, the designs of SDMS and CAPTURE are hidden, so it is recommended that your developers should try out the sample Customer database distributed in the NotesTracker package. This has internal logging implemented, and (in the licensed package) your developers can examine its design to see how it's done.

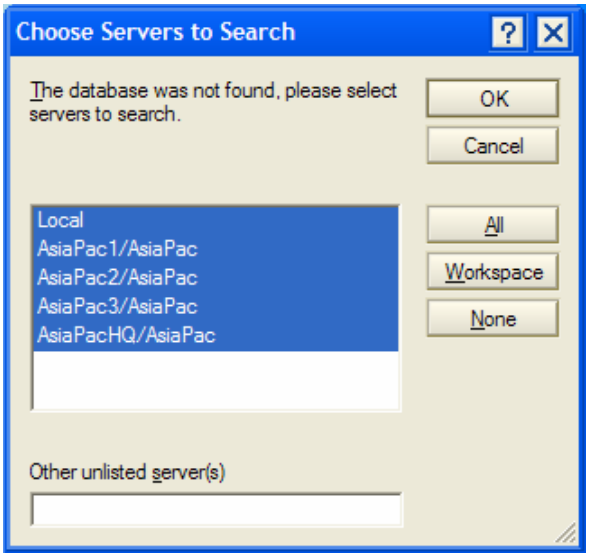
Testing the NotesTracker Repository Location

For each tracked database, Usage tracking is controlled by the settings in the database's NotesTracker Configuration document. (The configuration document is retrieved each and every time that a database activity is performed.) When you make changes to a database's configuration document on a given Domino server it should be obvious that the changes will not take effect on other servers (and the Notes Client workstations or Web browsers serviced by them) until the document gets replicated around your network to other replica copes of that database. This is "business as usual" for replication. The NotesTracker configuration changes do not take global effect until the replication cycle is finished to all servers and Notes workstations (entirely dependent on the various replication control settings).

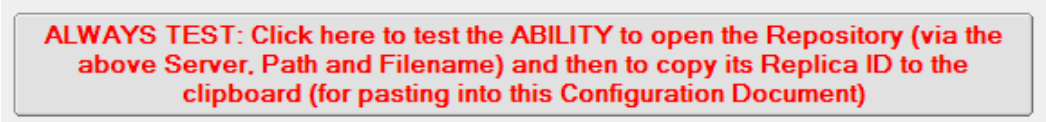
ESSENTIAL TIP: Once you have saved the configuration settings, always use the provided test button to **verify that the external Repository database can be opened, via the Replica ID value or via the Server plus Path and Filename values** that you have entered.

In the case of opening the external repository via Replica ID, you would click the following test button:

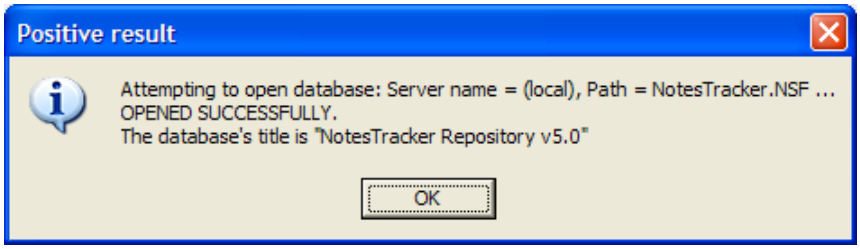
If Notes has difficulty in locating a database having the Replica ID that you provided, you will be presented with the standard "Choose Servers to Search" dialog box, similar to this:



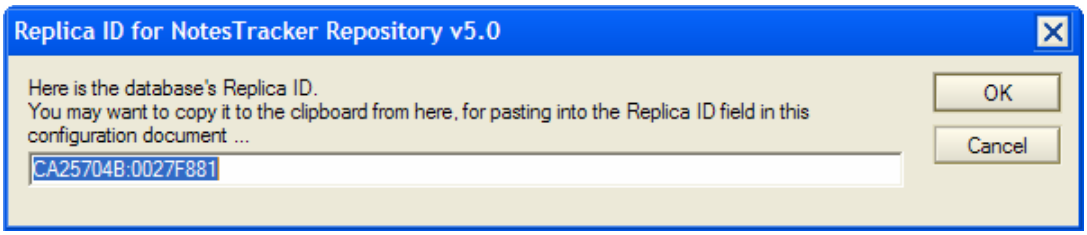
In the case of opening the repository via its path and filename, you would click this button:



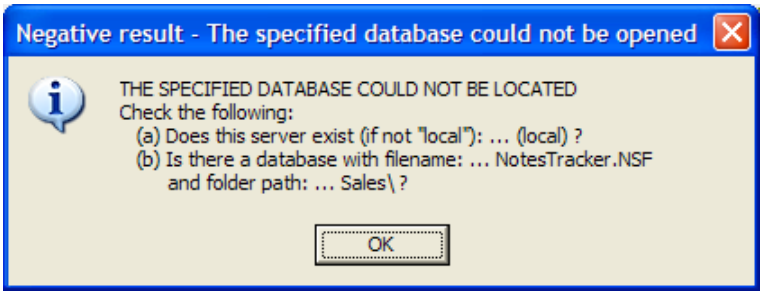
When you click the button, if the database can be opened you will see a dialog box like this:



Then when you click the OK button, you will be presented with a second dialog box containing the current database's Replica ID like the following (just in case you want to use it for "internal" logging):



If for some reason the database cannot be located, you are prompted with a few suggestions as to how you might correct the situation, like this:



To actually open the external repository database (highly recommended as final proof), click on the following button:

Speed versus Reliability? – Opening by Replica ID versus Path plus Filename

Trying both of these methods (Replica ID, and Path plus Filename) should give you the opportunity to gauge the difference in database opening speed, if any, using these alternative approaches.

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). However, once the Replica ID has been resolved and cached, subsequent opens via Replica ID are much faster.

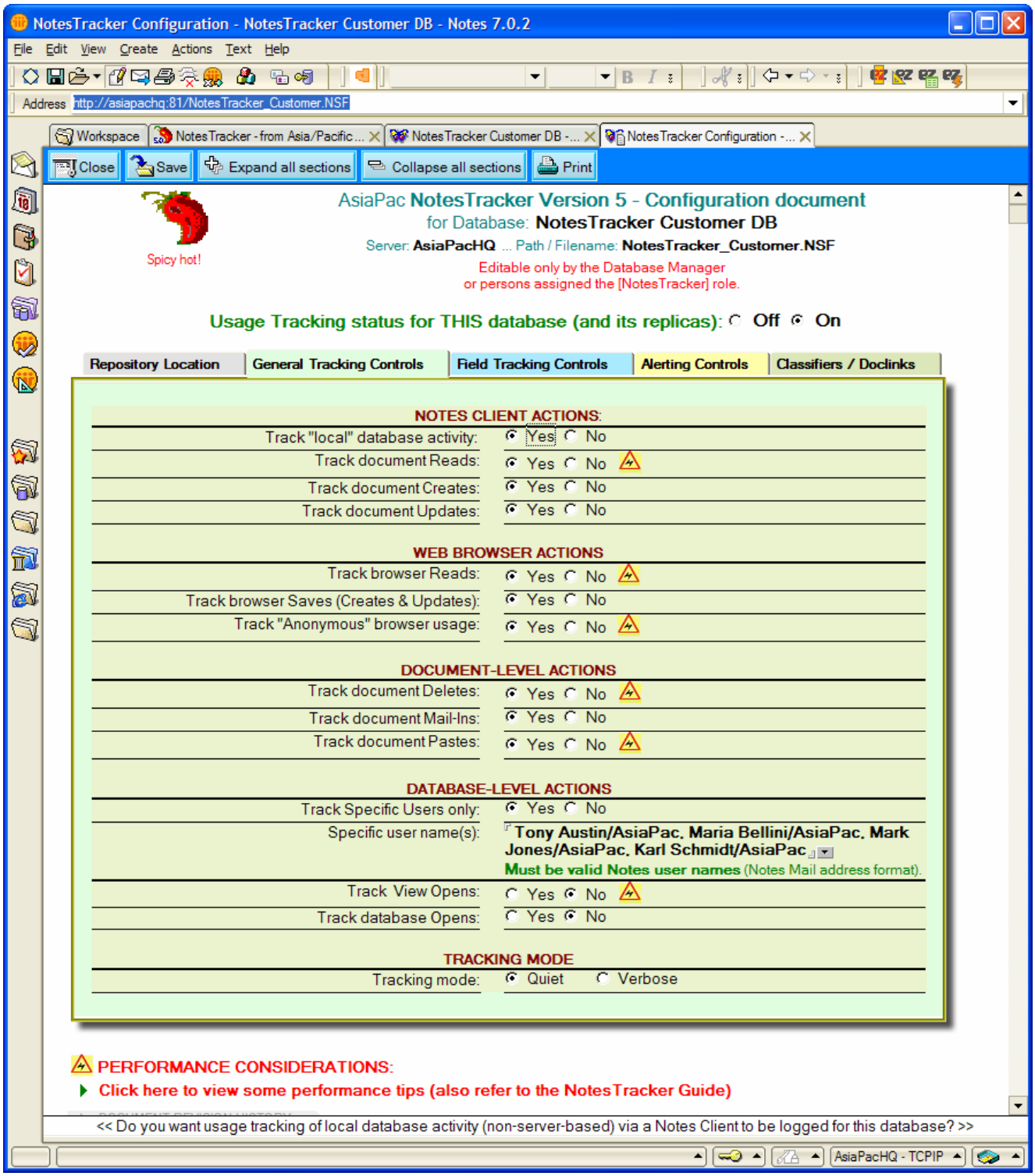
Opening the NotesTracker Database via its Path and Filename has the *potential* of being faster and thereby yielding a better user experience. Its big disadvantage is (because of the way that NotesTracker is designed) it means that each database and its replicas must all be stored in the **same Path and Filename on all systems**. Therefore it is possible that, without careful and consistent file management, the usage Tracking code will fail to open the external NotesTracker repository 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.

General Tracking Controls

Once you have defined where the NotesTracker usage log entries will be stored (external repository, or internal to the current database), you can switch to the other tabs in the configuration document and specify precisely how usage tracking should operate for the current database.

The first thing to do is to check the default settings of the “general tracking controls” fields and modify them as required:



Notes Client Actions

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 document pastes.

You can **suppress logging of "local" database activity**. That is, you can switch off the tracking of databases that are opened on a desktop system's or notebook PC's local hard disk, meaning 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. It's a policy decision that is entirely up to you, the administrator. However it certainly should be approved by the application's owners/sponsors, who would go by titles such as "Knowledge Manager" or "Business Unit Executive" or "Project Manager" or "Deployment Leader" or something like that. If usage log entries are generated locally, then presumably (but not necessarily) you will arrange the database's replication local settings so that local Usage Log entries are sent to a server-based NotesTracker repository so that overall usage of the database – workstation-based plus server-based -- can be consolidated and analyzed.

Web Browser Actions

You also can **specify tracking of web browser actions: browser Reads, and browser Saves (Creates and Updates).** And as a means of reducing usage tracking overheads in a busy (high transaction rate) Internet situation, you can **suppress the logging of "Anonymous" web user activities** meaning that only the actions performed by authenticated browser users will be tracked (those users who can provide an acceptable user name and password when challenged and who most likely will be far fewer number than anonymous users).

Document-Level Actions

You can also track document-level actions, meaning the sort of actions that occur on a document as a whole without its having to be opened. Also, such documents might take place against a set of documents and not just against a single document at a time. These action types are **Deletes, Pastes and Mail-ins**. (The ability to track pastes and mail-ins was added in Version 5.0 of NotesTracker.)

Database-Level Actions

Finally, there are database-level actions: **View Opens** and **Database Opens**. These are unrelated to any individual documents at all.

The tracking of view opens is meant to be a very short-term activity that will gives your Notes developer or administrator some idea of the relative usage of views in a given database, so that there is a rational basis for changing the properties and/or designs of the views in order to give better view performance and/or reduce Domino server overheads (processor and view index space).

The tracking of database opens (new in Version 5.0 of NotesTracker) is of similar interest. It can give you an overall feel for the popularity of a database. For what it's worth, note that if you are also tracking view opens naturally enough there is some tracking overlap between the two, because the database's default view is opened then too.

Restricting Usage Tracking to Specific Users

This feature was added towards the very end of the new in NotesTracker Version 5.0 development cycle. In fact, it was going to be held off until a later NotesTracker release, but it was incorporated in Version 5.0 because it was felt to be significant enough to be brought forward.

If you select "**Yes**" for the "**Track Specific Users only**" option then you are required to enter at least one user's name. The name must resolve to a valid Notes name format, such as "John Smith/Acme" or "Jacques La Salle/Sales/France" (the so-called "abbreviated" name format). Be very careful to enter names in this format. Tracking will not occur if a user's Notes name is misspelled. To assist with this, an address dialog is provided so that you can select the name(s) from your address books(s).

A variation of this rule is that the name "**Anonymous**" may be used if, for some reason, you wish to track usage by unauthenticated Web browser users. (The name has to be *typed* into the "Track Specific Users only" field, since it is not available in the address dialog.)

Note: the test as to whether or not a user is specified to be tracked is made very early in each monitored event, to help minimize usage tracking overheads.

Tracking Mode

Quiet and Verbose tracking mode are explained a page or two later in this guide.

NotesTracker Performance Considerations

Naturally, any sort of usage tracking will have some performance overheads. In the design and coding of NotesTracker, every attempt has been made to minimize these overheads. The configuration document has some built-in brief tips and guidelines about the performance implications of the various settings, the most significant settings being highlighted with the icon, with a summary that you can read by clicking on the twisty at the bottom of the configuration document.

As a rule, the great bulk of operations against a database (80 percent or more in most cases) are document reads via Notes Clients or Web browsers. Therefore setting **Track document reads** and **Track browser reads** to "**No**" will have a great effect in reducing tracking overheads.

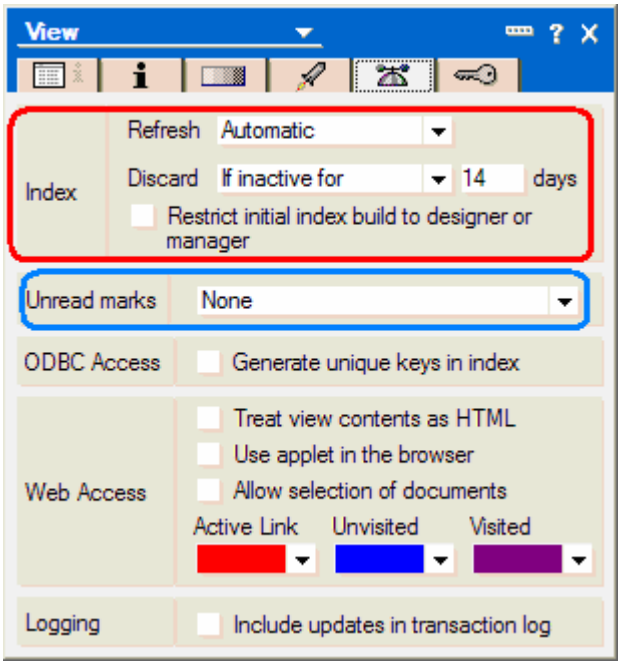
Similarly, setting **Track "Anonymous" browser reads** to "**No**" could significantly reduce the browser usage tracking overheads, especially if the database is getting a large number of hits across the Internet with most of the database accesses being carried out by unauthenticated browser users. This setting still causes tracking of authenticated browser users (where you have set the database's ACL to force the username/password challenge to appear when an attempt is made to open the database via a browser). Presumably you will want to track whatever operations the authenticated users are carrying out, so this setting is seen as providing a reasonable compromise between reducing performance overheads and obtaining useful tracking information.

The default settings presume that you will always want to track document deletions, but not always to track view opens, database opens, document mail-ins, and document pastes (the last three of these being options added in NotesTracker Version 5.0).

It is fairly common for document delete and paste operations to be performed against *groups of documents*, so if you are tracking deletes and pastes keep this in mind.

The ability to track **view opens** was added to an early release of NotesTracker to provide (we believe) a unique yet simple way for Domino administrators and developers to determine which views in a database are being most heavily used and which ones are rarely used. It should only take a few hours, or a few days at most, to determine this, because users tend to switch views very frequently. As soon as you have gathered sufficient view usage statistics, you definitely should switch off view tracking.

Given an insight into view usage you might decide to entirely eliminate some views or at least to consolidate some of them. As well, and you should review and if necessary change their properties which impact performance, such as index refresh and discard intervals, and whether to keep unread marks:



Of course, switching off some of the above tracking options would not be feasible if it eliminates exactly the type of usage tracking that you want to carry out against the database. In this case, you might consider adopting a "sampling approach" such as carrying out view tracking once off (or only occasionally) or for short periods of time, say one day per month.

Verbose and Quiet Tracking Modes, plus Some Typical Tracking Issues

As a licensed user with access to the source code, if you examine the various LotusScript routines you will find that wherever possible error situations have been foreseen and handled in the code. Our own research, together with feedback and suggestions from NotesTracker users worldwide, have gotten NotesTracker to the stage where most error conditions are handled gracefully – that is, no surprises!

The key design philosophy behind NotesTracker's handling of errors is that when an error occurs anywhere in the tracking routines the end user should not notice any difference whatsoever in the behavior of the application.

For example, if the NotesTracker configuration view or the configuration document is missing or if the specified external NotesTracker Repository database cannot be opened, the code detects this and handles it "quietly" in order to cause minimal or no disruption to the normal operation of an application.

Tracking mode

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

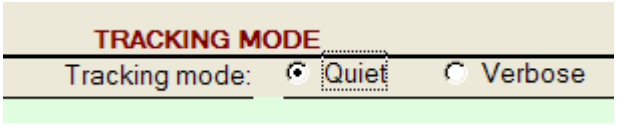
NotesTracker sends occasional status and all of its error messages only to the **Notes Client status line** (at the bottom of the Notes Client window), rather than displaying disruptive dialog boxes.

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 very modest range of additional advisory and error messages can appear in the Notes Client status line. Careful monitoring of these additional messages should help you to diagnose and rectify some NotesTracker problems. The messages are generated via Print statements (in the LotusScript code), which cause the messages to be sent to the status line

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 in your organization) to carry out some in-depth problem analysis for usage tracking on that particular database. If you see fit, you might drop in some additional Print statements at critical points in the NotesTracker code to provide you with a more "aggressive" style of error reporting. (A reminder: keep a record of your additional statements, and where you have placed them, so that you can re-enter them in the same spots following future releases of NotesTracker.)

Of course, you can always contact us at Asia/Pacific Computer Services for additional support.

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:

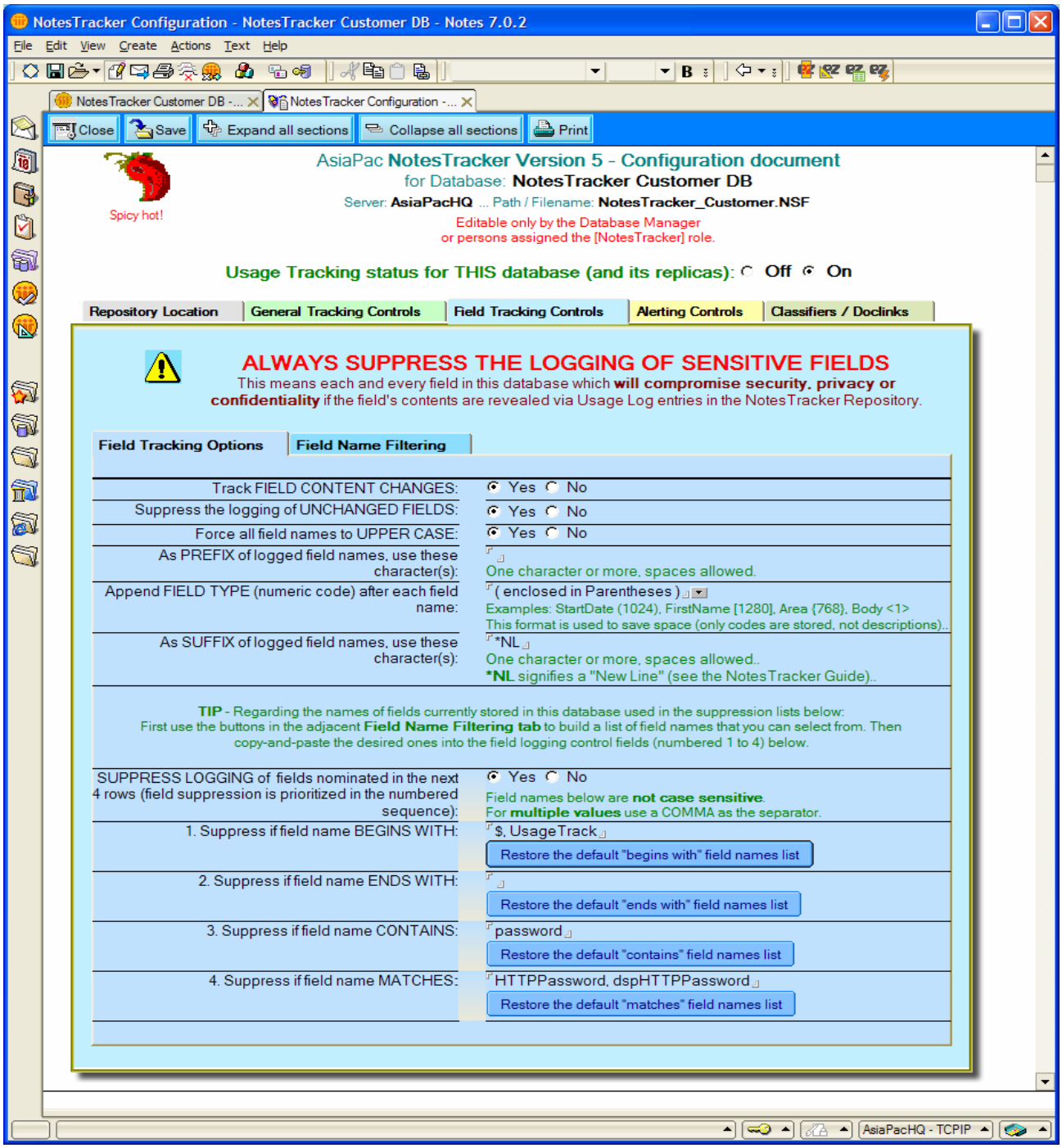
- Incorrectly specifying the NotesTracker repository database's Replica ID or Server and Path names
- The NotesTracker Configuration Document being inadvertently deleted from the database.
- The NotesTracker Configuration suffering from corruption (rare, but possible; and probably any corruption in the 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 configuration documents. Only the settings in the first document in the configuration view will be honored. (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 Document Updates – Field Tracking Controls

NotesTracker is all about tracking what happens to the content of your database applications. A key component of the content is the array of **fields** stored within documents, and this section of the configuration document enables you to specify in considerable detail how fields are to be tracked.

Field Tracking Options

If the option **Track document Updates** in the General Tracking Controls tab is set to "**Yes**" then (as shown in the following figure) under the **Field Tracking Controls** tab you can switch on the tracking of fields by setting the option "**Track FIELD CONTENT CHANGES"** to "**Yes**" (the default value):



The default value of "**Yes**" for the second option "**Suppress the logging of UNCHANGED FIELDS**" causes NotesTracker to 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 alter this option to "**No**" then NotesTracker stores the values of all of the fields in a document, not just of the changed fields (defined as the new, updated and deleted fields).

Many Notes databases after some time at the hands of a range of developers finish up with a higgledy-piggledy collection of fields some with field names in upper case, some in lower case, and some in mixed case! In order for a Usage Log document to present the field names uniformly, there is a default value of "**Yes**" for the option "**Force all field names to UPPER CASE**". This forces each field name to be converted to upper case in the Usage Log documents, which (as well as uniformity of presentation) can help each field name to stand out from the field's contents.

Improving the Presentation of Usage Log "Before" and "After" Field Values

Also for better presentation purposes, the next option "**As PREFIX of logged field names, use these character(s)**" allows you to provide a string of "eye catcher" characters as a prefix to each field name. This prefix could be put to any use -- say, putting a Run Number, project identifier, or other meaningful text string in front of each field name. The default prefix is blank.

Via the option "As **SUFFIX of logged field names, use these character(s)**" you should specify a suffix string to be placed after each field name in the Usage Log document. The intention is to cause field names in the Usage Log to stand out from the field contents:

A ... Field contents AFTER update

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

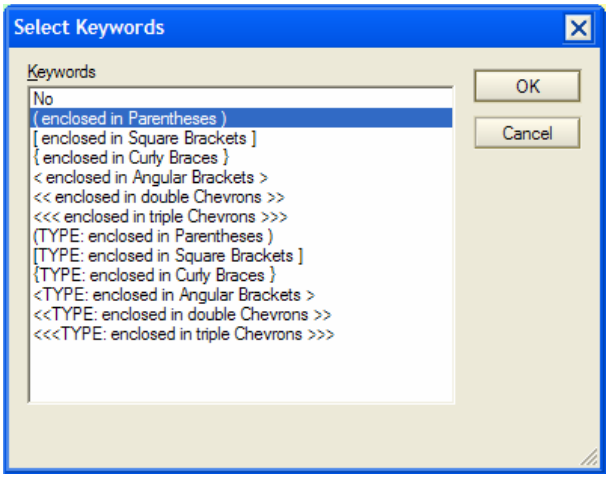
The default suffix is a chevron pattern: " **>>** " (notice the leading and trailing single space characters). The suffix could be a null character (an empty value), but should be at least a single blank character.

A special option for the suffix is to specify ***NL** which stands for "New Line" (do not omit the leading asterisk). This causes the field name to display on one line and the field contents to display starting at the beginning of the next line. For example:

A ... Field contents AFTER update

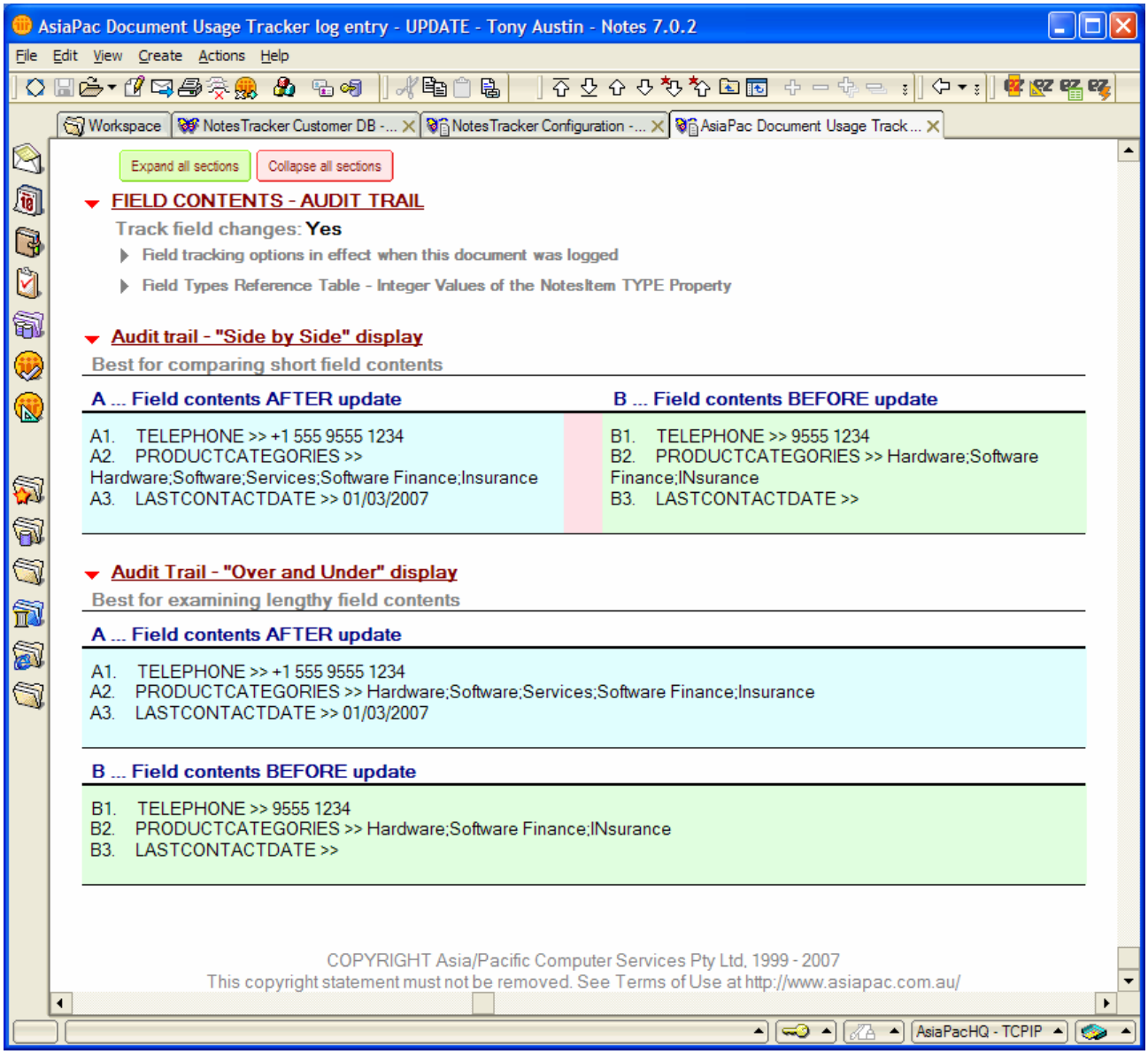
A1. FIRSTNAME
Wilhelmina
A2. CONTACTNAME
Wilhelmina Jones
A3. TELEPHONE
1234 6789
A4. EMAIL
willie_jones@betterprinting.com.au
A5. PRODUCTCATEGORIES
Hardware;Software;Consulting

A new option in NotesTracker Version 5.0 is "**Append FIELD TYPE (numeric code) after each field name**" which places after each field name an integer value that represents the data type of the field. You also have the ability to specify via a pull-down list how the data type integer is displayed, either plain (no enclosing characters) of with a range of different enclosing characters (the default being enclosing parentheses), thus:



"Before Values" and "After Values" of Fields

Let's examine in more detail the various settings at work. The next illustration shows the effect of default field name suffix being used, that is, the chevron string " >> " (a space followed by two Greater Than symbols followed by another space), and with the field type option is switched off:



To see the field changes you click on the **FIELD CONTENTS - AUDIT TRAIL** twisty, and then on the twisties that appear underneath it. Two buttons are provided as a convenience for quickly expanding or collapsing all of these twisties.

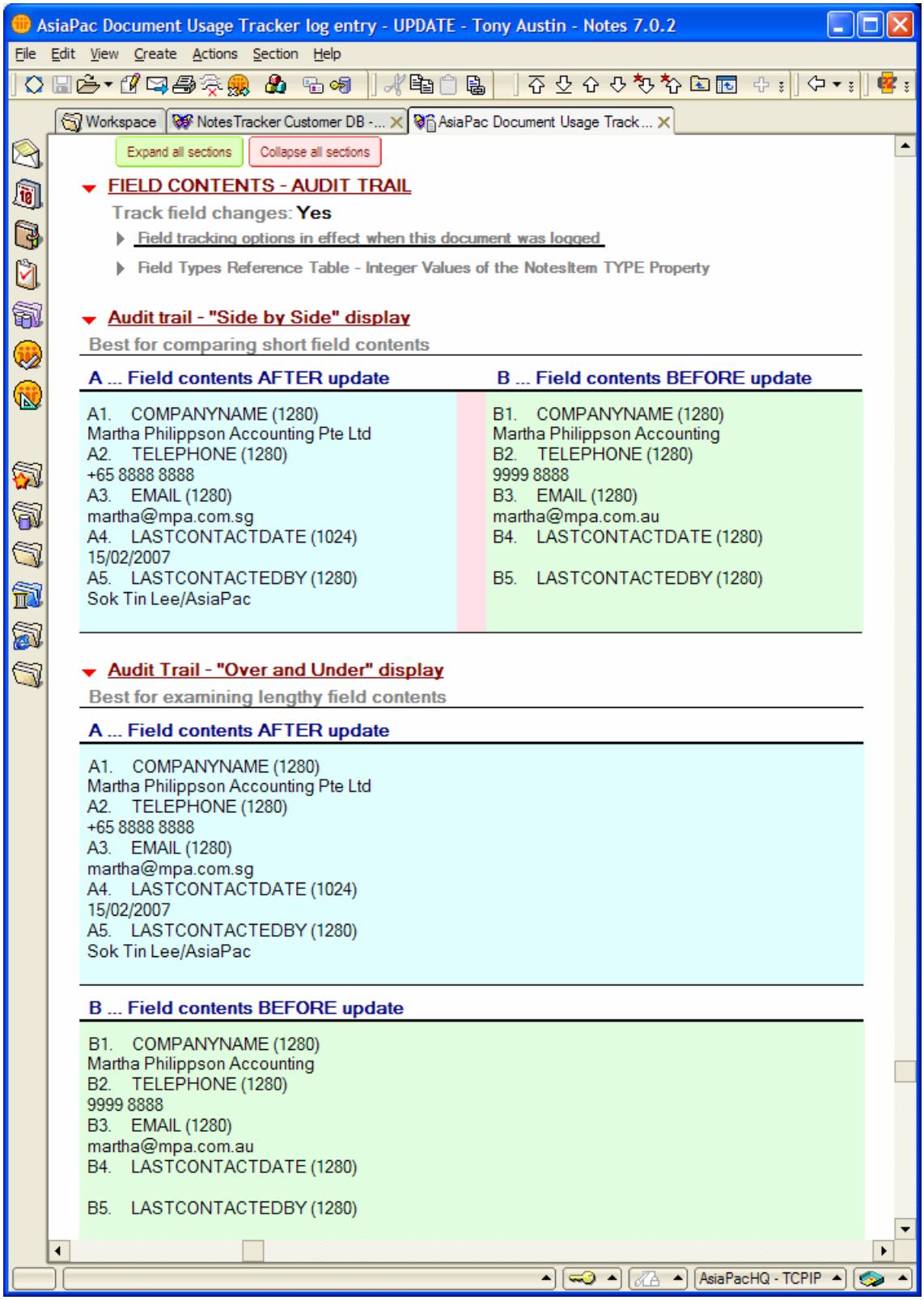
The first of the four twisties (grey color) lists the field tracking options that were in effect in the database at the instant that the document was opened. (Remember that NotesTracker reads the configuration document for the database for each and every logged event). The second twisty (also grey color) lists the various integer values of field types (rich text, number, date, etc), further described a little later In this guide.

This example shows that the TELEPHONE and PRODUCTCATEGORIES fields existed prior to the update and that their contents were modified, whereas the LASTCONTACTDATE field was null (didn't exist prior to the update event).

The "Side by Side" and "Over and Under" twisties provide alternatives for display the field contents. Select the one that is best for displaying the particular contents of a given document.

Note: for your convenience in analyzing the usage log entries, the field names are prefixed with a sequentially incremented field identifier (for "Before Images" B1, B2, B3, … and for "After images" A1, A2, A3, … where of course "B" stands for "Before" and "A" for "After"). You should understand that due to the way that the field's data is gathered by NotesTracker, by looping through all of the field items in a document, there isn't a guaranteed one-to-one relationship between these sequence numbers. Thus the field labeled A1 is not necessarily the same as the one labeled B1, and so on. Nevertheless, the sequence numbers often do finish up being identical, or at worst fairly close together.

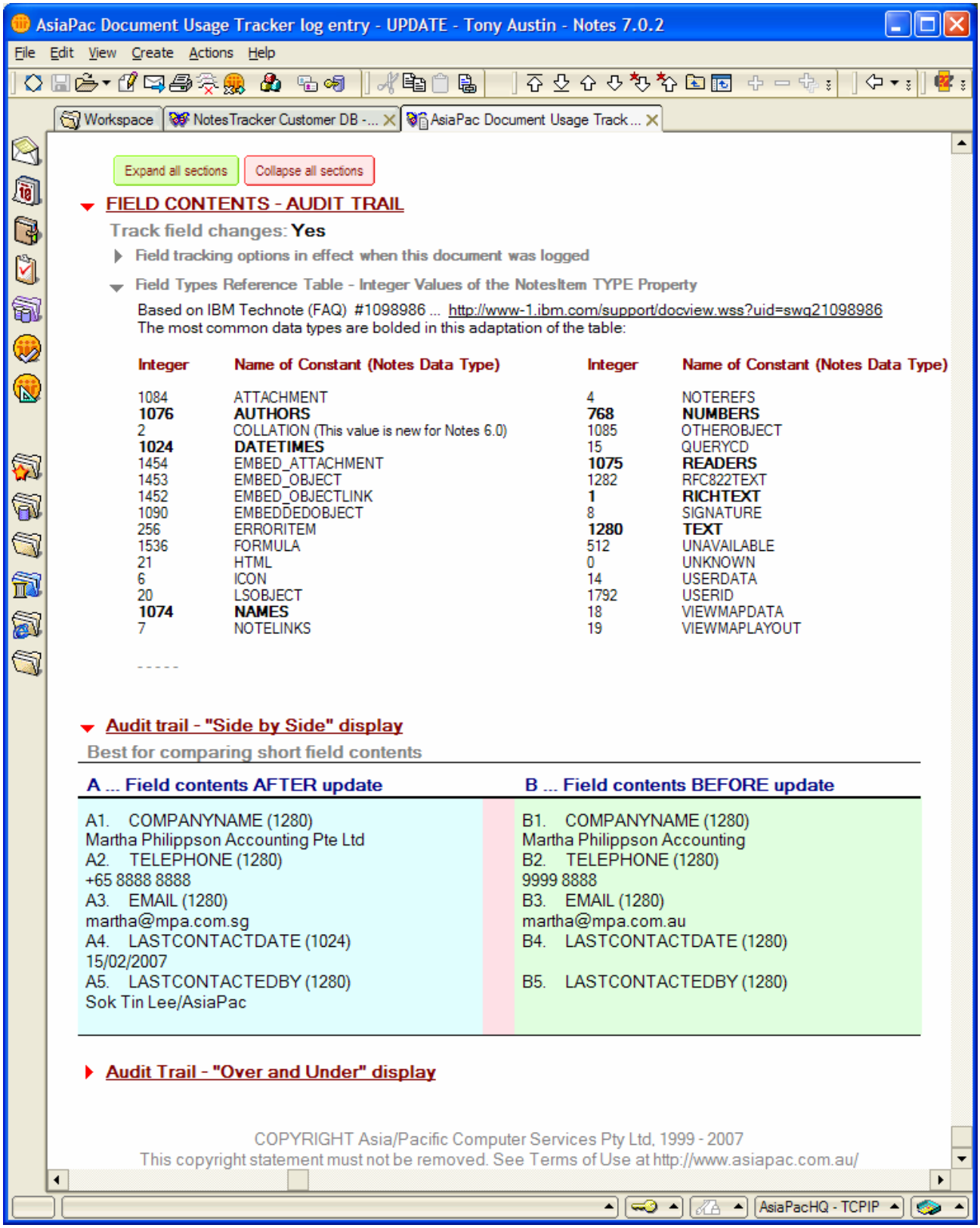
If you specify the special value of ***NL** (an asterisk followed by an uppercase N and an uppercase L), then a New Line character is inserted between the field name and the field contents. Using this value in conjunction with "Force all field names to UPPER CASE" can lead to better legibility in the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



The LastContactDate and LastContactBy fields were null or blank before the document was updated, and clearly show as blank lines in the “before images”.

Field Types

In this example (and new in NotesTracker Version 5.0), also in effect was the option "**Append FIELD TYPE (numeric code) after each field name**" using the default value of "**(enclosed in parentheses)**". When you click on the "Field Types Reference Table" twisty you can easily interpret the field type of each field in the before and after image lists, as in the following figure. The data types (name of constants) are arranged in ascending name order (not numerical value):



Note: it was decided to use the numeric value for field types (rather than name of the constant) to conserve space.

Knowing the field type for each field can be very useful (even necessary) in some situations, and could help with data fidelity management or application debugging. The above illustration indicates, for example, that the telephone "number" field is actually of type text (1280) rather than numeric (768).

Considerations for Tracking Database Accesses via Web Browsers

A major feature incorporated into NotesTracker (starting at Version 4.0) is the option to track “front end” accesses against your Notes databases carried out via web browsers.

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 cannot in and of themselves distinguish between the **creation** and the **update** of a Notes document.

This is in distinct contrast with the Notes Client environment where document creates and update events can be detected separately logged as such. 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. A fundamental design decision for NotesTracker was to record only a single event when a new document is created or an existing document is updated, no matter how many times the document is saved during the time that the document is open.

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 order to determine what fields are changed, the before images are obtained by retrieving their values from the so-called “back end” Notes document (that is, by again retrieving the saved document from the database), only then having a set of before images to compare with the after images stored in the current web page.

It is important to note that the “back end” Notes document so retrieved just might gave been updated in the intervening period -- seconds, minutes or even hours – since the web page was originally posted to the user’s browser window, (In the Notes Client environment the before images are recorded as soon as the document is opened, and 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”.)

That all being said, testing shows that in many if not most cases NotesTracker’s web browser update Usage Log entry seems to give an accurate representation of the actual field changes.

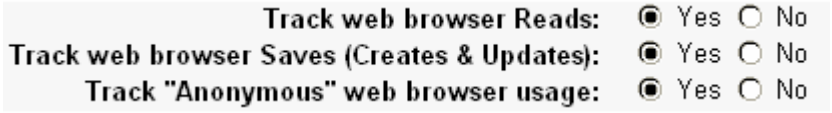
Web Browser Tracking – Performance Considerations

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 might 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 “the general public” -- non-authenticated users, to whom Domino gives the user name **"Anonymous"**.

You may be able to significantly reduce usage tracking overheads (CPU load, memory and disk space on your Domino servers) by specifying that you do not want anonymous user accesses to be tracked> Naturally enough, if you really do need to know about anonymous access, this will reduce the subsequent analytical value of the usage logs. In some cases (such as where you only allow authenticated browser access against specific Notes databases by your staff, your customers, your suppliers or your business partners), an idea may be to retain the logging of anonymous accesses but only for Web Page 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 Field Contents

The contents of Rich Text fields (such as the commonly-occurring "Body" field in mail memos) are not fully logged by NotesTracker. Only a simplified **plain-text rendition** of a Rich Text field's contents is written to the Usage Log document. Attachments and embedded objects are not captured by NotesTracker, nor are text styles (such as bold, underlined and italic) retained.

This was a deliberate NotesTracker design decision. It was felt to be unnecessary and beyond the purposes and scope of NotesTracker to log changes to the complex structure of Rich Text fields. Firstly, it was regarded as undesirable to make log copies of attachments (and to a somewhat lesser extent embedded objects), since it was likely that as this would dramatically increase the disk space requirements for the NotesTracker Repository database. Secondly, it was thought that in most cases it will be sufficient for many purposes to determine from just the simple text rendition how the contents of a Rich Text field have changed. (Or, putting it another way, the assumption is that many users of NotesTracker will be satisfied to understand just the textual changes that have been made to Rich Text fields.)

If this is a problem for you, please contact Asia/Pacific Computer Services to discuss possible alternatives.

Nominating Fields to be Suppressed from the Logging of Changes

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

Reasons for wanting to suppress the logging of some fields are discussed a page or two on.

The bottom half of the **Field Tracking Options** tab has several options that enable you to remove in various ways those fields in a database that are of no interest to you as regards their contents. Such fields would include special fields used internally by Notes (such as the so-called $ fields), and fields that are only used for content display (rather than permanent storage in the database) or as hidden work fields. NotesTracker will faithfully record them all unless you specify otherwise.

In the next illustration, the section circled in red surrounds the controls used to specify unwanted fields. The default values are shown.

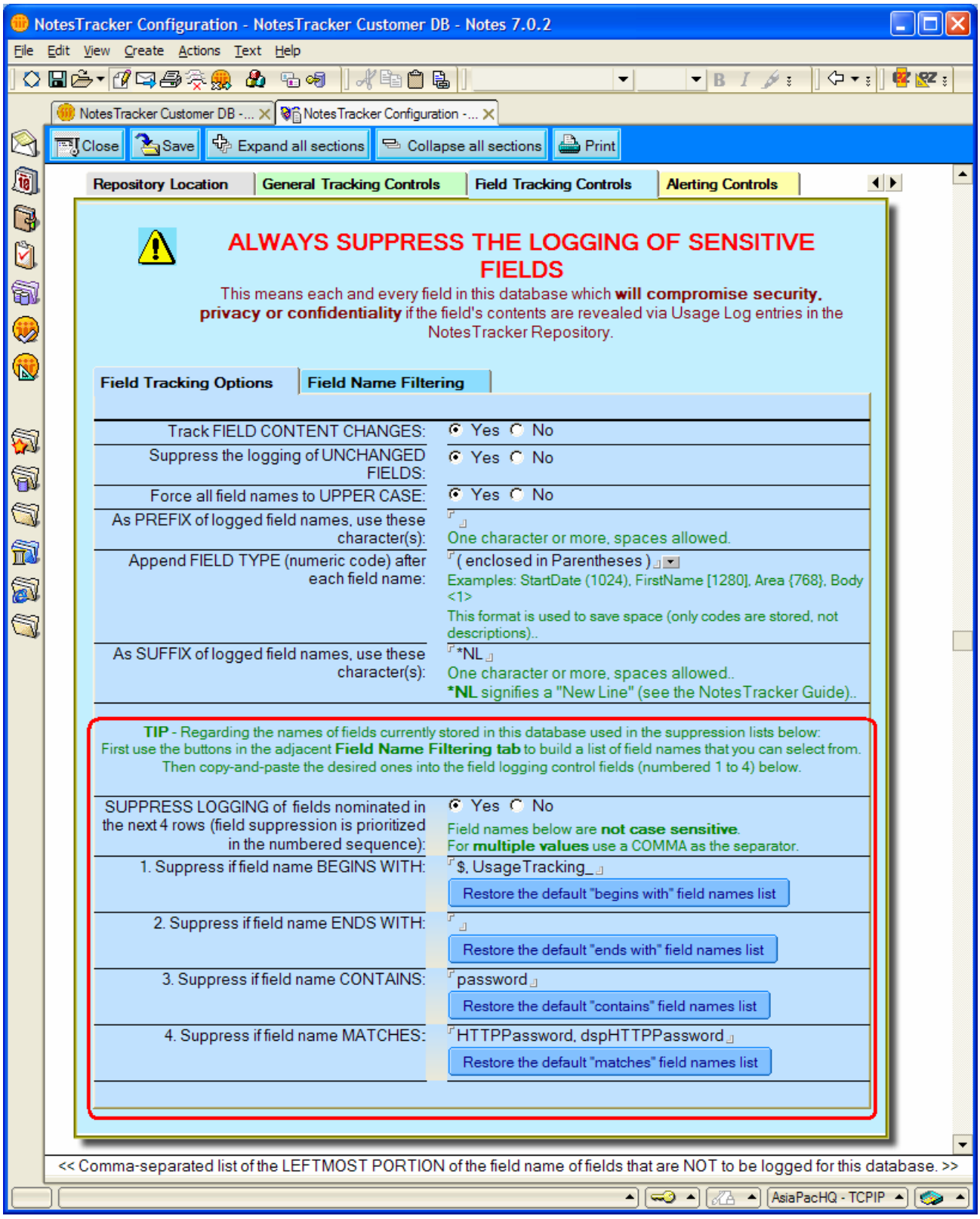
Wild-Card Filtering of Fields

As you can see, the default is that fields are suppressed if their field names **begin with** "$" and "UsageTrack" (the reason for these two is explained a few paragraphs later). You can also suppress fields having name that **end with** or **contain** or **match** specified strings. These are explained more a few paragraphs further on in this guide. These four "filter" fields provide a lot of versatility in the nomination of fields to be suppressed.

When you save the NotesTracker configuration document, these filter fields are saved too. Therefore if you modify or extend these filters, they are recalled the next time that you open the configuration document. There are some filter strings that apply in a generic fashion to Notes (and NotesTracker), but there will be other filter strings that are unique to each database, so saving them is a handy feature. If you want to go back to scratch, each field has a button associated with it to restore the NotesTracker defaults for that filter type.

As the green Help text indicates, you enter multiple values using a comma as the separator character.

The four long buttons provide a convenient way to restore NotesTracker's default values to the filter fields. (It's a simple matter for your Notes developer to tailor the names affected by these buttons to better meet the requirements of a given database or of 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.)



There are **at least two good reasons** why you would not want to log the before/after images of certain of the fields that were changed in a document edit:

- 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 HTTP Password field in a Person document in the Domino Directory database), personal information (salary, medical, HR, etc), plus such things as board decisions, product strategies, sales results, and mail memo contents. What would be the point of placing such information in highly-restricted databases if logging reveals it?

- 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.

Since there may be dozens or even hundreds of fields in a form, so using the three types of **wild-card specifiers** can greatly reduce the burden for field name elimination. Otherwise, you must list the field names explicitly.

Let's go into a little more detail about the filter strings. Firstly, you specify the field names via simple strings that are (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: **$**, **UsageTrack** (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. **Exact match** (do this when the above filters aren't suitable) – such as: **form**, **author**, **httppassword** or **HTTPPassword** (uppercase or lowercase or mixed case makes no difference, except perhaps to legibility).

Since there often are very many fields in a database, NotesTracker provides a means of easily discovering their names so that you can add the names to one or other of the four filter fields just discussed.

Going one step further, how do you determine what are the names of the fields in the current database? The solution is to switch to the adjacent "**Field Name Filtering**" tab. This was designed to generate a list of the names of all fields in the ***forms*** in the database.

Clicking buttons in the top section of this tab generate lists of field names, and another button at the bottom of the tab can be used to remove "noise" or "clutter" in the field name lists so that you can hone in on the field names that are of value for usage tracking purposes.

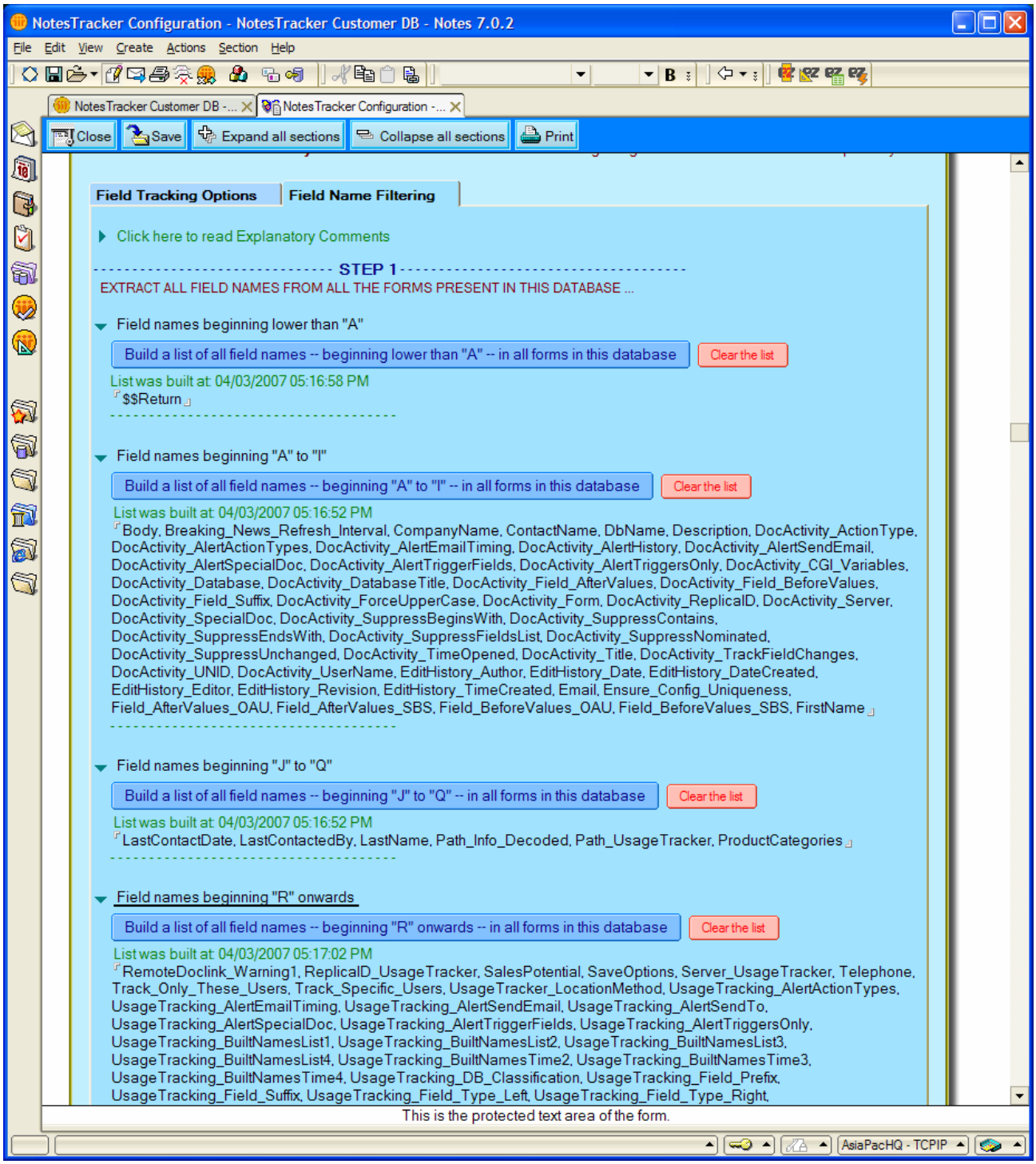
Note: the buttons do not gather the names of fields that are stored in the ***documents*** that are currently stored in the database. A database may contain documents that were not created (or updated) via forms presented to the "front end" (the user interface, either Notes Client or Web browser). Such things as agents and other "back end" processes can store data fields in one or more of the database's documents, and some or all of this data may be allotted field names that do not exist in the database's forms. The buttons have been designed to gather field names from all the forms (a rapid process) rather than all of the documents, since this could be a very slow process in a database holding a large number of documents. If there are any such fields that you want to track, it's a simple matter to type them in manually.

In NotesTracker Version 4.3 a single button was added to generate the list of 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. (This had to be done because it was discovered that the single button ran into memory issues. There is a Notes limitation that crops up when the database has many hundreds of fields, and splitting the function into separate buttons was the best way to overcome the limitation. It makes little if any difference to the effectiveness of NotesTracker.)

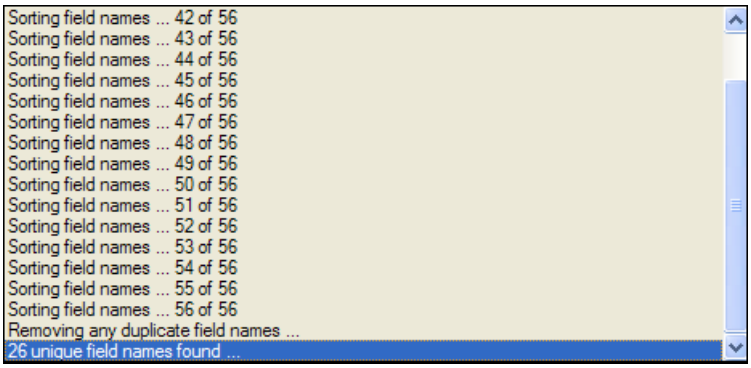
The steps involved are easier to carry out than to describe. They should only take seconds or at most a minute or two to perform. All the same, a brief description follows.

Listing the Names of Fields in the Database, and Field Name Filtering

Go to the **Field Name Filtering** tab and open the twisties in the top section labeled **STEP 1**. Then click on the four long buttons (dark blue) labeled "**Build a list of Field Names …**" (for lower than "A", beginning "A" to "I", beginning "J" to "Q", and beginning "R" onwards). The resultant field names lists will of course vary from database to database, but you should get a result like this:

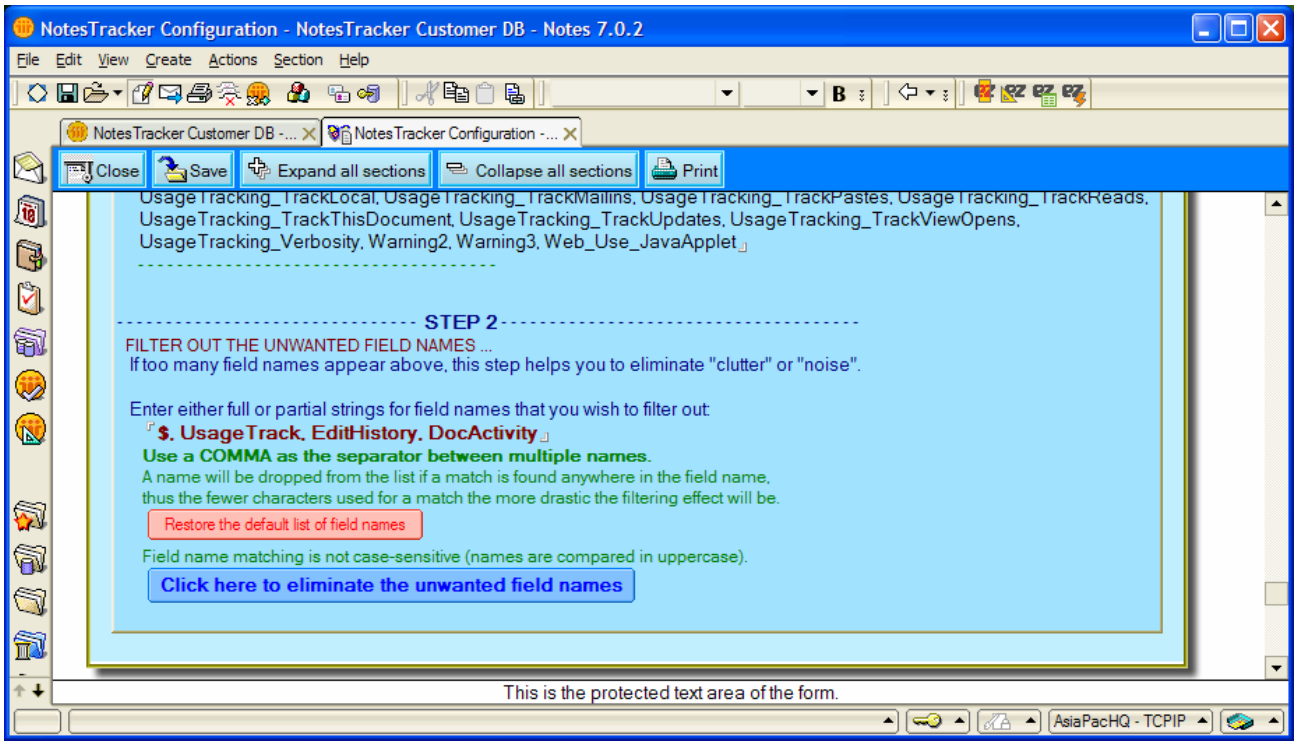


The generation of the field name lists should be very fast, usually a few seconds, but will take a little longer if there are lots of fields in the database. You can watch progress via the Notes status line, thus:



If you make a mess with any of the lists, just click on the associated "**Clear this list**" button (pink color) and click the button again to regenerate that list of field names.

Next, scroll down to **STEP 2** at the bottom of the Field Names Filtering tab:



The idea of the Field Tracking Controls tab is to nominate fields that you wish to track. But even a moderately complicated database can contain hundreds of field names, making it rather difficult to "see the wood for the trees" just due to the sheer size of the field name lists.

Coming to your rescue is a simple means of quickly filtering out unwanted field names, via an operation that you can carry out very rapidly either a single time or repetitively until the lists are reduced to manageable proportions.

There is a default string of characters ("$, UsageTrack, EditHistory, DocActivity") to act as a filter. Use the default list as provided or modify it to suit your filtering needs.

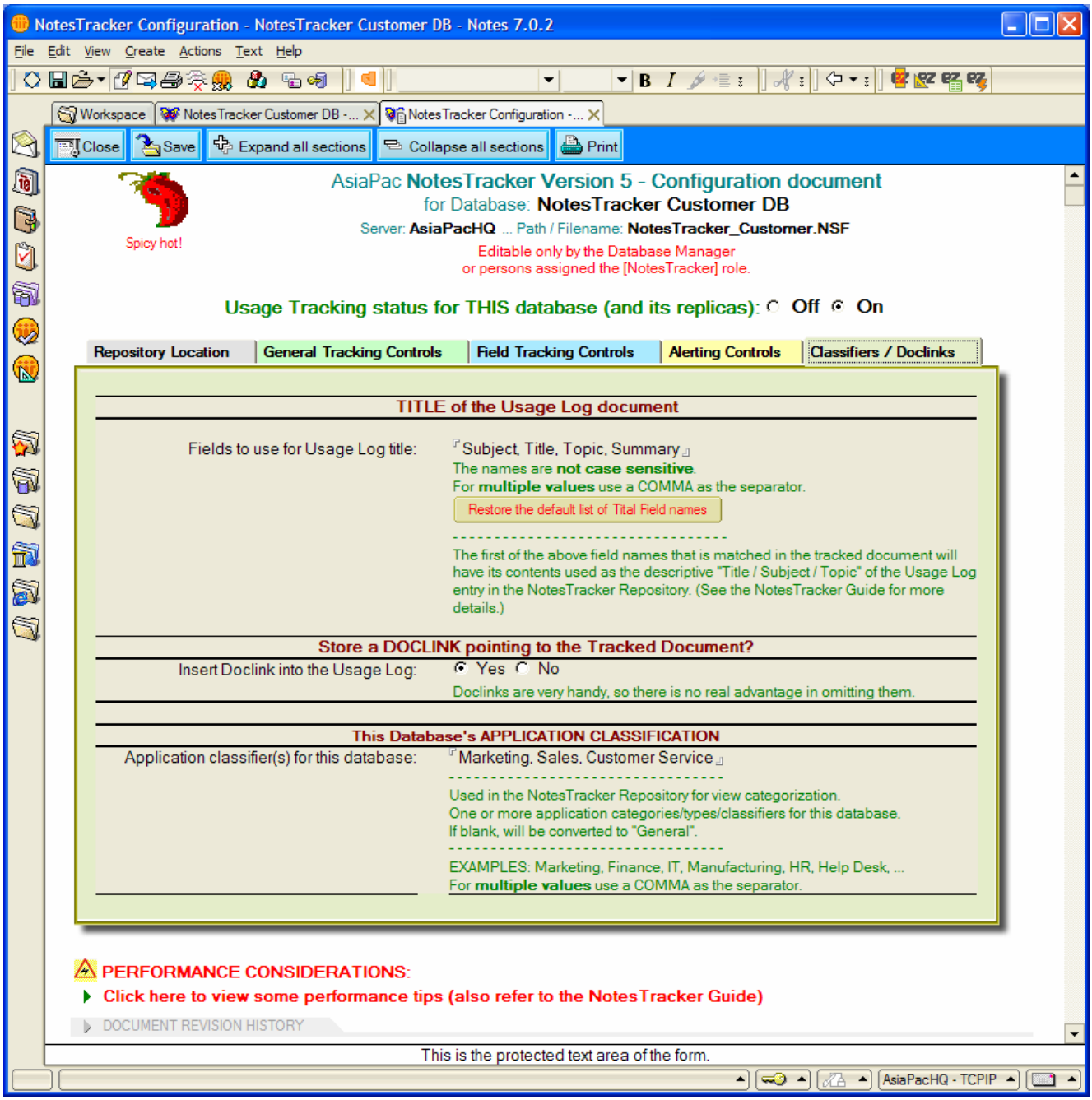
Now click on the long button at bottom (dark blue color) labeled "**Click here to eliminate the unwanted field names**" and in a flash the four lists generated in STEP 1 will become smaller.

If the lists are still to long, just modify the filter string using additional starting characters (separated by commas) for unwanted fields that are common in this particular database, the click the bottom button again.

Within minutes, if not seconds, seconds you will have short lists of field names that you can now copy to the clipboard and paste into the various tracking suppression lists under the Field Tracking Options tab.

Specifying a Database's Title, Doclink and Application Classification

The rightmost tab in the configuration document (labeled "**Classifiers / Doclinks**") allows you to specify several general-purpose settings:



These settings apply to every type of action performed against the database, with the just one exception: Doclinks are not relevant for document deletions.

Setting up a Meaningful “Title / Subject / Topic” Field for each Usage Log Entry

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

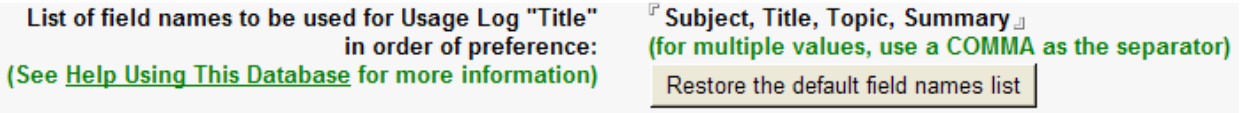
NotesTracker produces “usage log” documents, or entries -- one for each operation (Create, Read, Update, Delete, or view switch) in every one of your databases being tracked. These entries are your means of “usage metric analysis”, and the entries are presented as various categorized views in the NotesTracker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

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

The “UsageTracking_Title” Special Tracking 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” or “Subject” or “Topic” or “Summary” for that document – taking precedence over the contents of any of the fields in the field names list.

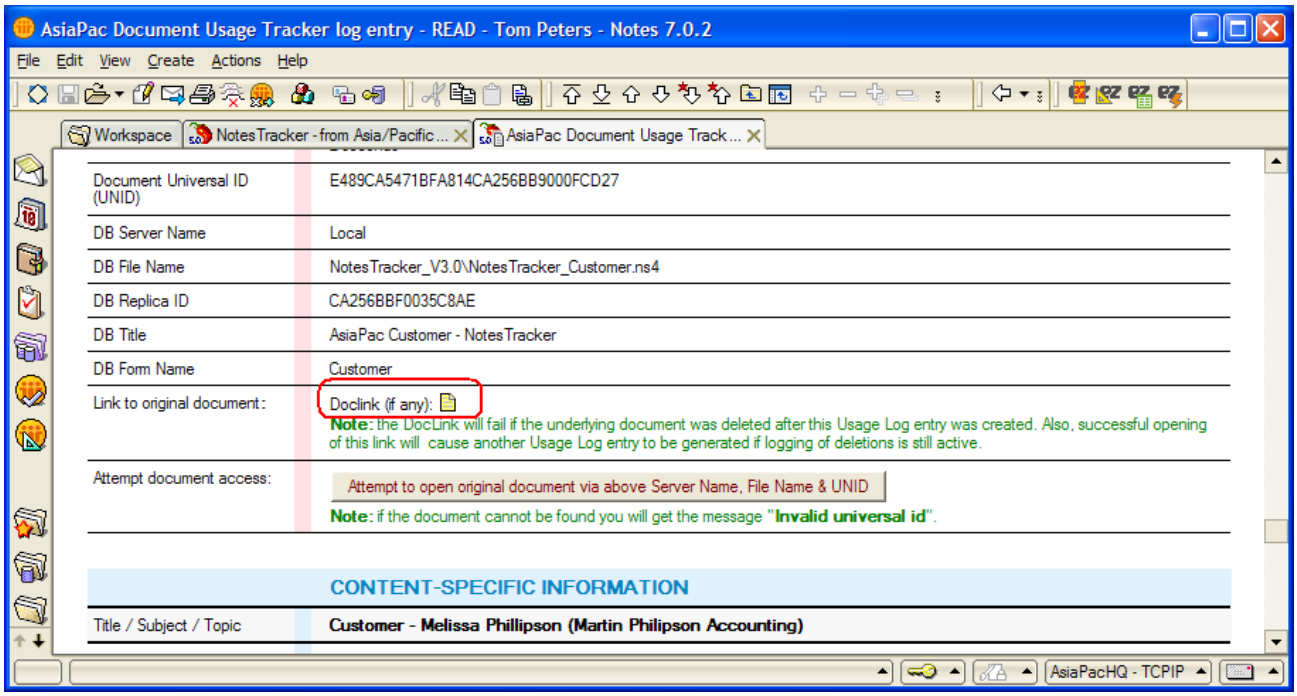
There is a detailed discussion of this field later on, in the Developer Topics section, since it is an aspect of usage logging that your developer must set up.

Suppressing the Logging of Document Links

NotesTracker can optionally store in the Usage Log document a **Document Link** -- often abbreviated to "**DocLink**" or "Doclink" -- that points to current Notes document for each Read, Create or Update.

Note: storing of a DocLink is never done for a document deletion event. Obviously, unless the deletion request fails, no document will be left in the database for such a link to be valid. Also, DocLinks are not applicable to View Open events, but they do apply for document pastes and mail-ins..

The DocLink makes it easy for you to open the original document when you are examining a particular Usage Log entry, simply by clicking on the DocLink icon.



In earlier versions of NotesTracker, the DocLink was *always* stored in each Usage Log entry, but starting with NotesTracker Version 4.0 the following setting in the NotesTracker configuration document gave the option not to store any DocLinks for the current database, by changing the "**Yes**" default to "**No**":

Selecting "No" perhaps will save a tiny bit of system resources -- processor time, disk space -- but probably not much.

Generally it's probably wise not to deselect this option unless you're quite sure that you don't mind losing the convenience of having these links in the Usage Log documents. Furthermore, Doclinks can play a part in the dynamic opening of documents for portal pages, discussed elsewhere in this guide, and this functionality will be lost of Doclinks are suppressed.

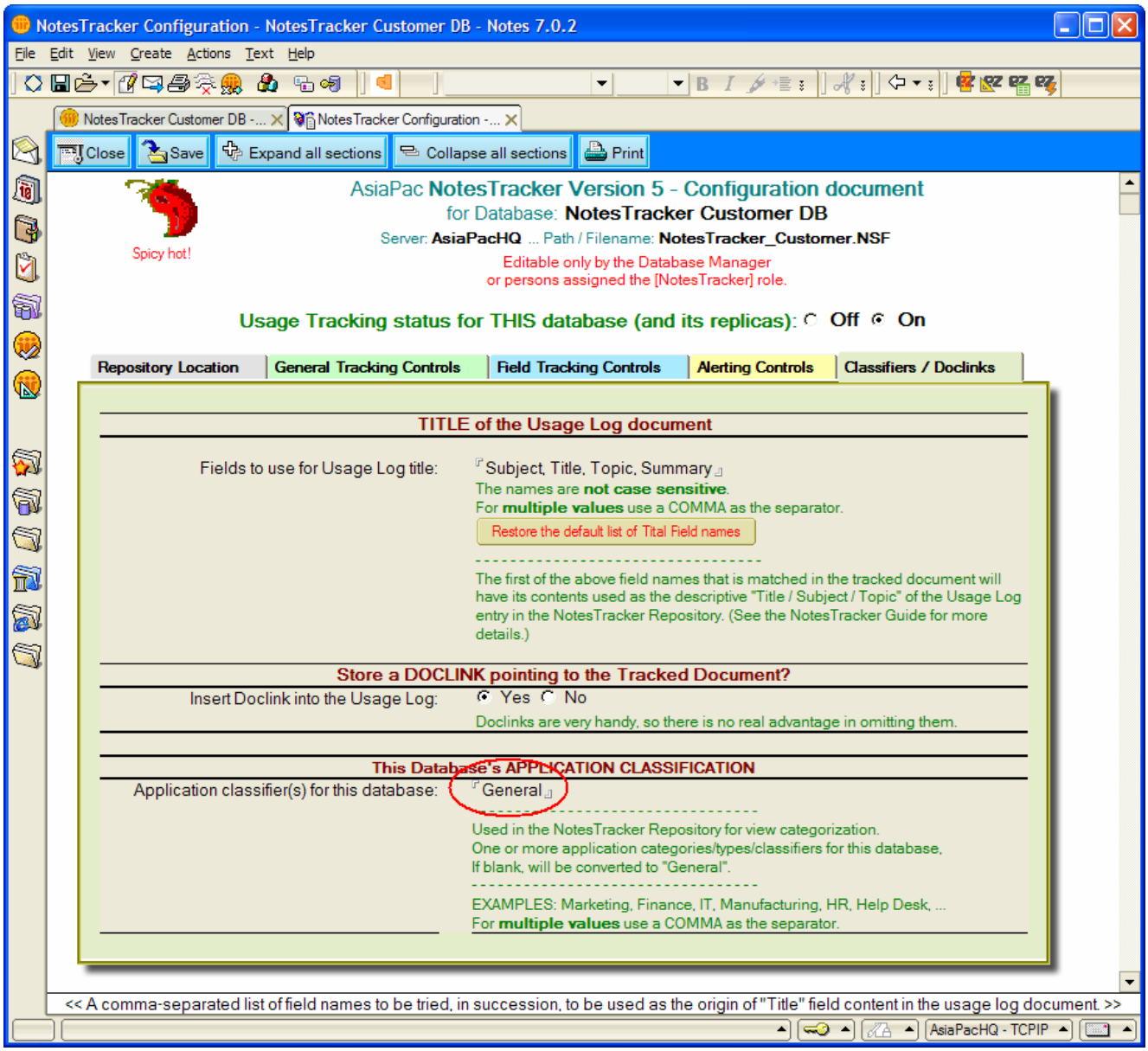
In particular, the existence of a DocLink in each Usage Log document is essential for the nifty "Breaking News" technique described later (in the Developer Topics section) to operate properly. For this technique to succeed, the DocLink must not be suppressed.. Otherwise, when the Usage Log document is opened via the Breaking News view, there is no DocLink present that will allow automatic displaying of the underlying Notes document. The end user will not want to see the Usage Log document (the technical content of which is of absolutely no interest to him/her). For this "Breaking News" approach, you only ever want the user to see *the original Notes document* (the one that was originally tracked and which holds the interesting content).

Note: you might be interested in the article **Managing Lotus Notes DocLinks with LotusScript** at http://searchdomino.techtarget.com/tip/0,289483,sid4_gci1236108,00.html?track=NL-348&ad=580227&asrc=EM_NLT_1019499&uid=5774849 which shows code for two agents, the first of which that builds a list of all of the DocLinks in a database. The second agent is then run to determine whether or not the target document behind each DocLink still exists.

Specify a Database's Application Classification

A mind but handy option (new in NotesTracker Version 5.0) is the ability to specify, for analysis and reporting purposes, the "classification" or "category" of the current database.

This value defaults to "**General**" (and the view category used for Usage Log entries created by NotesTracker releases prior to Version 5.0 will also be set to "General"):



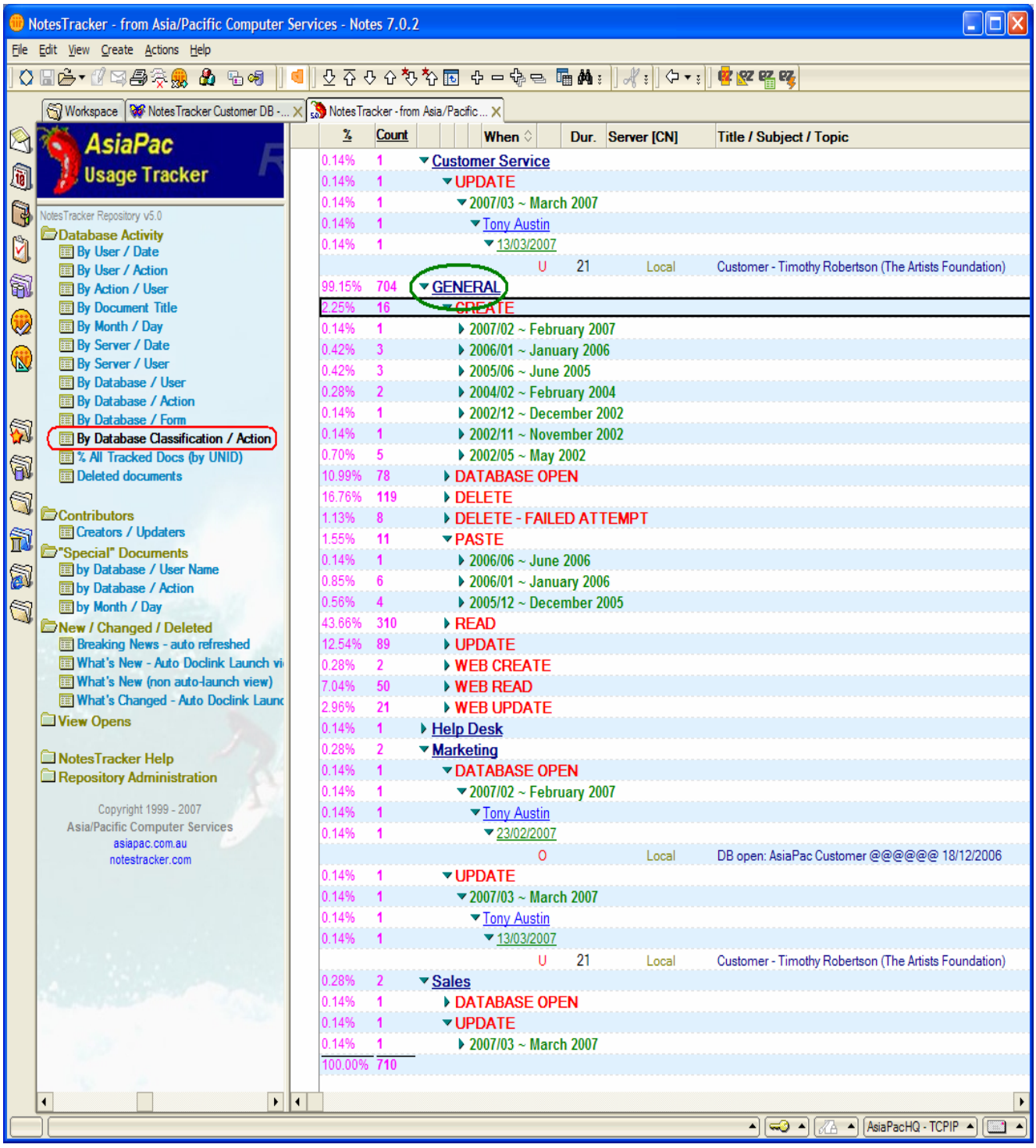
Put one or more entries in the classification field to indicate the database's nature. Enter the classifiers as a string of characters, using commas as separators between the classifier entries. There is no need to enter spaces between entries. Leading and trailing spaces in each entry will be trimmed out.

These entries will be featured as category heading in Usage Log views, so it pays to check their spelling

You should discuss this option with the person(s) who "own" or "sponsor" the current database application and/or who will be spending most time analyzing the Usage Log.

Choose whatever is most relevant as classifiers for the current database. Thus, for a CRM database you might put:
Marketing, Sales, Customers, Project Tracking, Appointments

If you leave this field blank -- and for compatibility with Usage Log entries created in a NotesTracker version prior to Version 5.0 (where this Classifier field was not generated) -- the special value of “GENERAL” appears in the “**By Database Classification / Action**” view, as shown in the following illustration:



This feature was added to NotesTracker Version 5.0 so that you will have a new perspective from which to analyze database usage patterns and trends as you track your databases over the months and years.

Proactivity – e-Mail Alerting, and "Special Documents"

Following a NotesTracker user's suggestion, a new tracking feature was added in Version 5.0 that will be of great interest and value to existing users and potential NotesTracker users.

The user wanted to be able to nominate what from now on we will call "**special documents**" -- specific documents in a database that being of extra interest or in some way more important than other documents in the database.

In the NotesTracker Repository, the special documents are given **their own separate Usage Log views**, so that they stand out as soon as they appear in the Repository.

And even better if, instead of just passively keeping a watch on the repository, you can specify that you want to be **proactively alerted** via e-mail whenever one of these special documents is "touched" in some way (created, updated, deleted, etc). Going one step further, you could even try using a third-party Domino add-on that detects the arrival of each such e-mail message and immediately triggers an SMS or pager alert.

Two major new features in NotesTracker Version 5.0 can handle these requirements, with minimal developer involvement:

- **SPECIAL DOCUMENTS:**
  The ability to easily nominate any document in a database as a "special document"

- **E-MAIL ALERTING:**
  The ability to nominate some or all of the action types to automatically send an e-mail message to delegated recipient(s).

  The document actions that you can select (via checkbox) are any combination of "the usual suspects" … Create, Read, Update, Delete, Paste and Mail-in.

  For **special documents**, you can specify the e-mail alert to be sent **for any action** or just for **the selected actions**.

  For **updates**, you can go one step further and list which **specific fields** must be updated in order to trigger an e-mail alert.

Let's consider two examples – and there would be many others -- of how e-mail alerting for "special documents" would come in handy.

Firstly, in any moderate-to-large sized organization, Domino administrators are likely to be more focused on certain strategic Domino servers than on the others: They are in one way or another critical to high operational availability and service levels: typically hub servers involved with replication, mail routing, Web serving, etc (plus, almost certainly, any server dedicated to the top executives in the organization -- after all, "You've got to keep the boss happy!").

The other Domino servers would be regarded as being, relatively speaking, of somewhat lesser importance. It follows that the administrators would be more interested in changes happening to or deletion of certain Server documents than others. So (starting in NotesTracker Version 5.0) they would designate the Server documents for these strategic servers as "special documents" (probably deselecting document Read operations as being of little interest). Perhaps they might even nominate certain Server Document fields to act as triggers: security-related fields (such as Full Access administrators), programmability restrictions (Run unrestricted methods, etc), performance-related fields (the likes of Maximum number of threads), and so on.

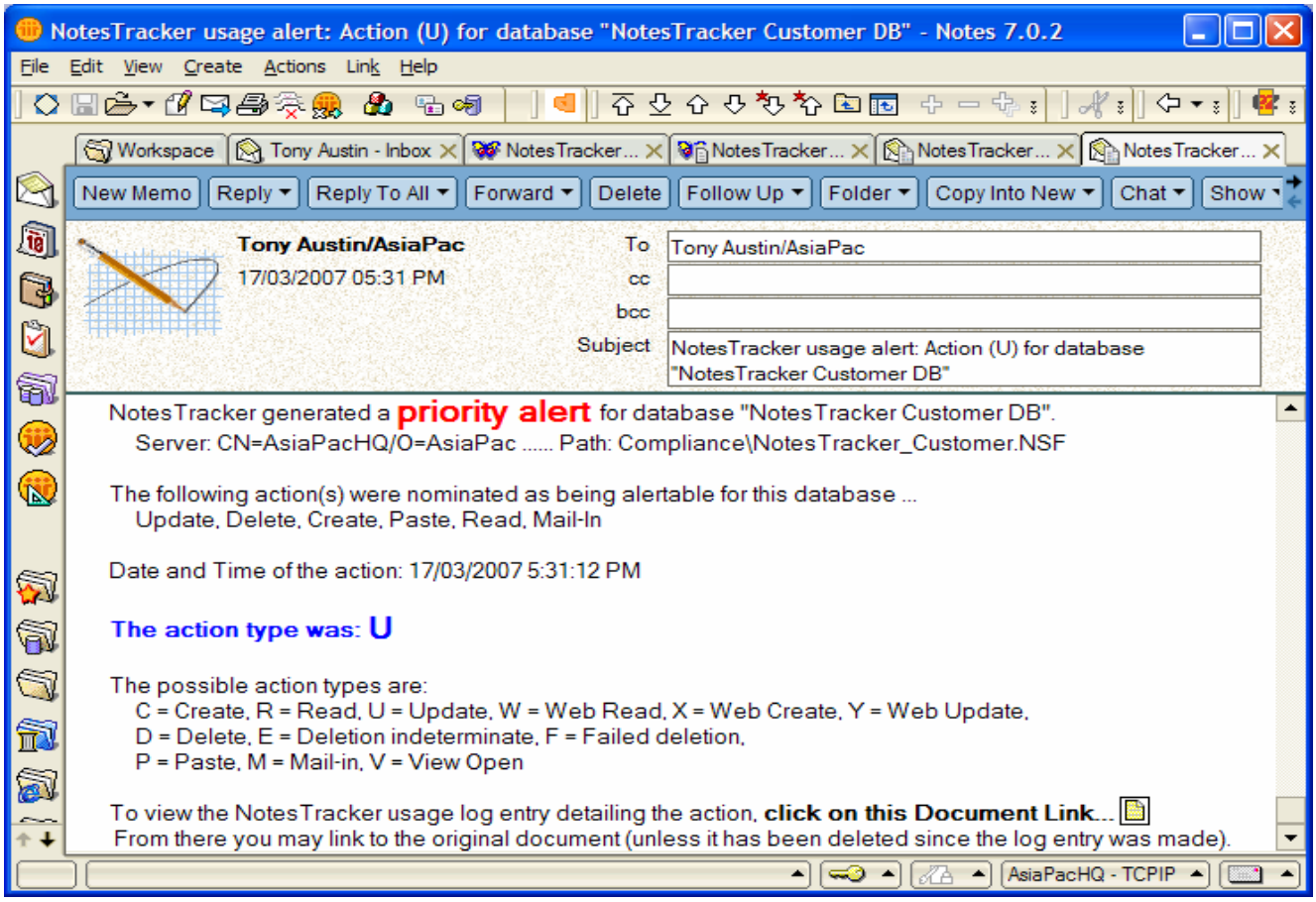
A second example is a Customer Relationship Management (CRM) application where sales people would benefit from being proactively alerted via e-mail as soon as there is activity against specific customers or prospects: for example, a Sales Call Report document being created or a customer's address and telephone number being changed.

For CRM applications like this, usually there will be certain customers (or prospects, or business partners) who are of considerable importance to the organization's current sales and marketing efforts. Therefore it would be extremely beneficial to put such customers onto a watch list, and with NotesTracker V5 you would do this by nominating as "special documents" some specific documents in the CRM database relating to these customers. After that, not only would a passive watch on these customers (via the continually-updated NotesTracker usage repository) but also you could go to the next step and have e-mail alerts generated as soon as the special documents are touched. Then later on, when the marketing or sales focus moves to other customers, you could easily and quickly change which of the documents in the database are nominated as being "special" and thereby easily adapt to a rapidly changing marketing and sales environment.

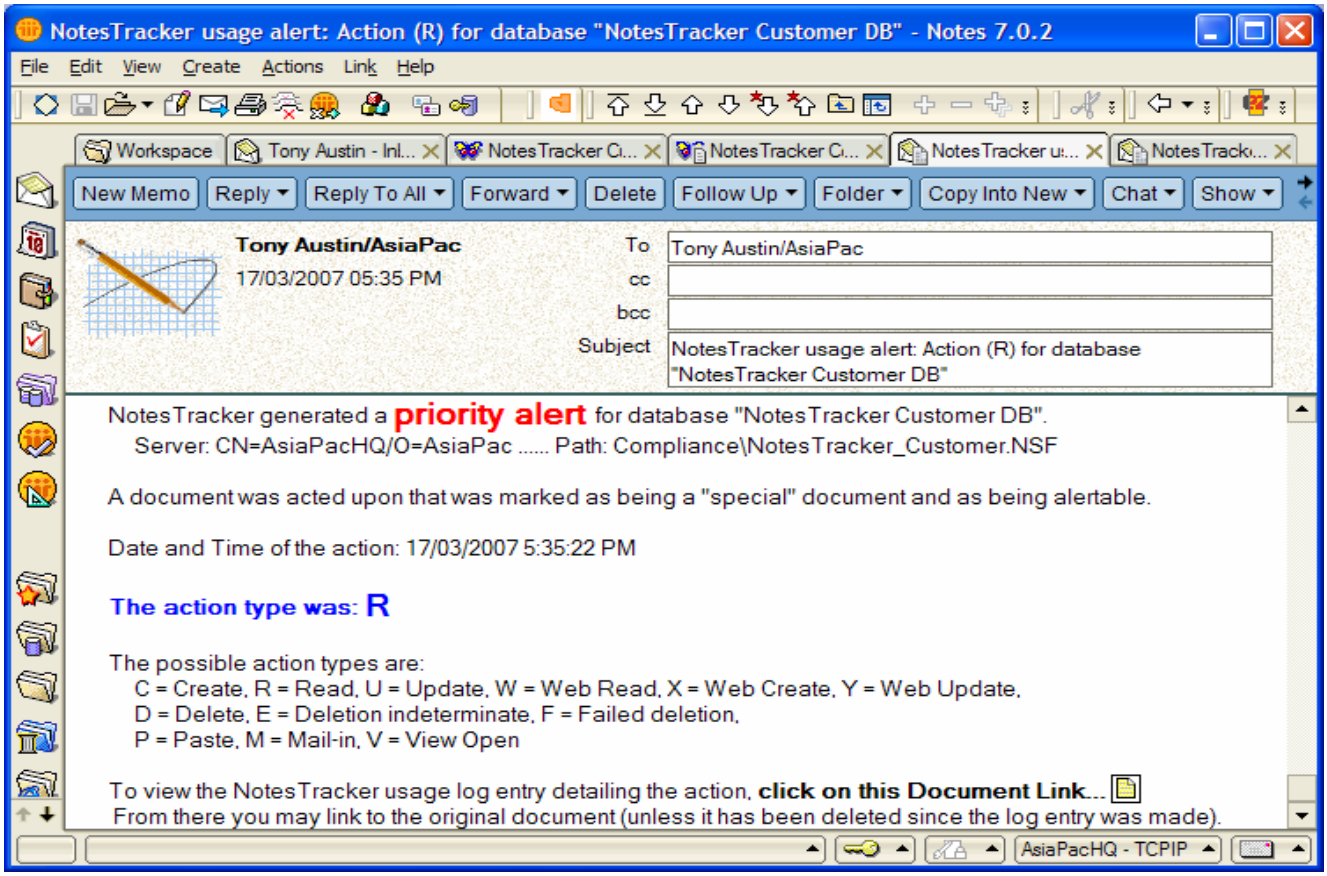
Example of NotesTracker e-Mail Alert Messages

Each addressee who is specified to be sent e-mail alerts for a given database will receive an incoming mail message in one of two layouts.

For specified event(s) happening to general documents (those not marked as “special documents”) …



For specified action types (or all action types, if so selected), when the document is a “special document” …

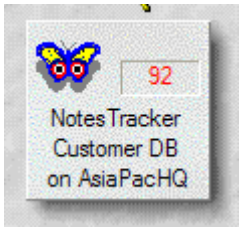


The “UsageTracking_SpecialDoc” Special Tracking Field

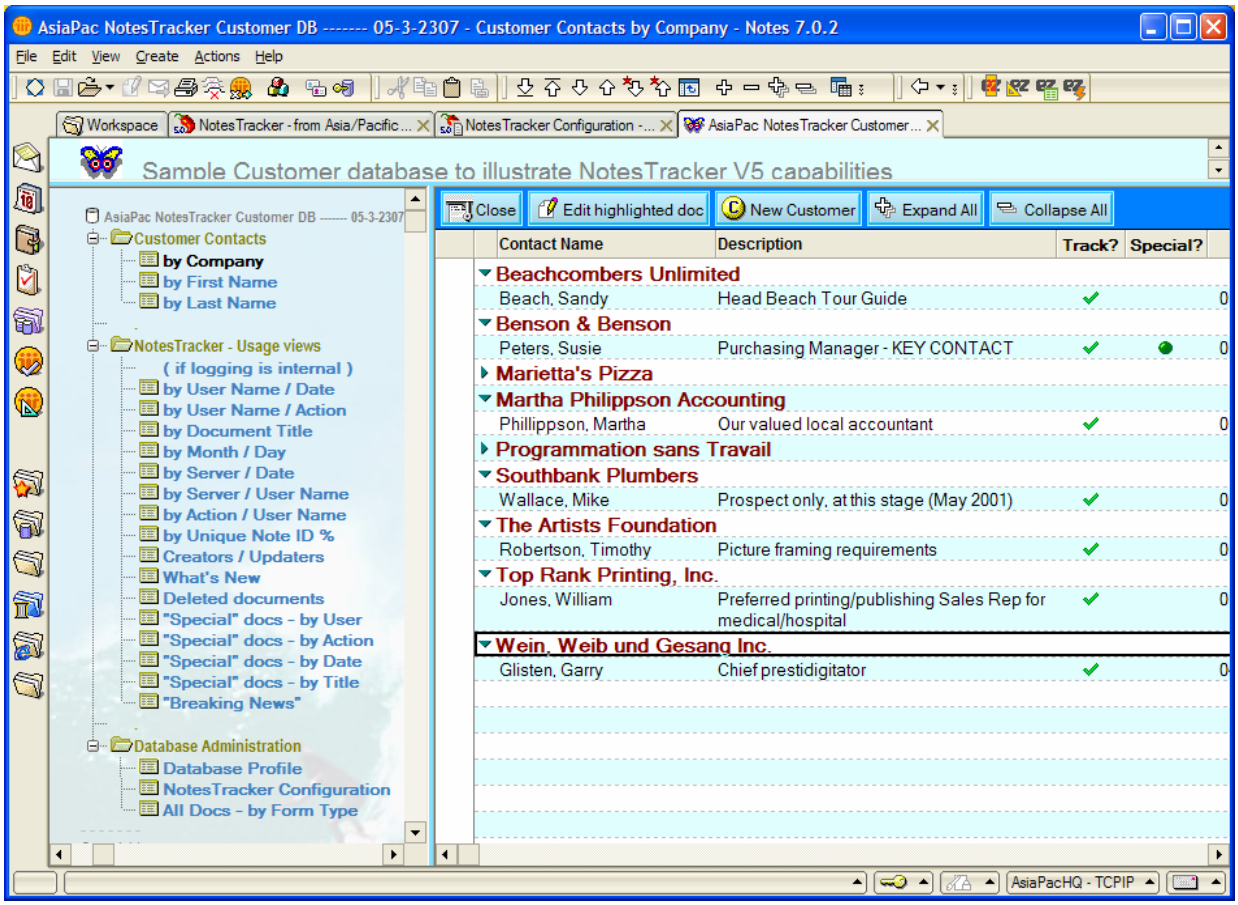
In the Developer Topics section, it is explained how, with some quick and simple database design changes, your Notes developer can:

- Set up a special NotesTracker plain text field, called “**UsageTracking_SpecialDoc**”
- Arrange for that field to be populated (either user edited, or under program control)
- Ensure that the field can be used to generate e-mail alerts
- Adjust how the field affects the Usage Log views in the NotesTracker Repository database for maximum impact

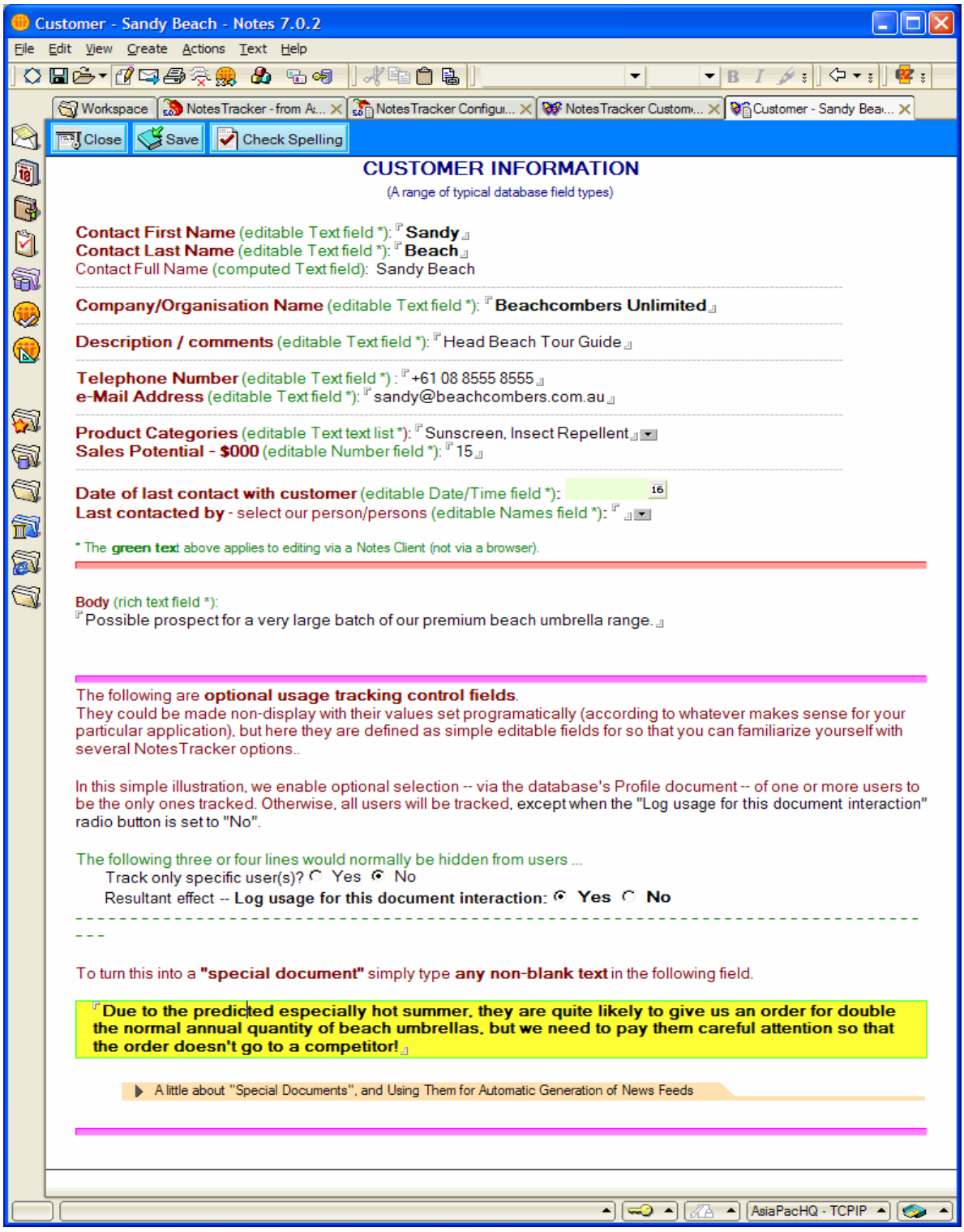
But let’s assume that you have a database with a form already containing the UsageTracking_SpecialDoc text field. A good example of this is the example **NotesTracker Customer Database** that is part of the NotesTracker distribution package:



This database has a design with the only intent being to clearly illustrate a range of NotesTracker capabilities, so its design is very straightforward: a single Customer form a few views (Customers by Company Name, by contact First Name, by contact Last Name) as well as a set of NotesTracker usage log views and a few administration/control views.



Here is an example of the simple Customer form, with the **UsageTracking_SpecialDoc** special field being the one with bright yellow background near the bottom. Of course, users of the database have no inkling of this field's significance.. To them it's just another ordinary text field on the form:



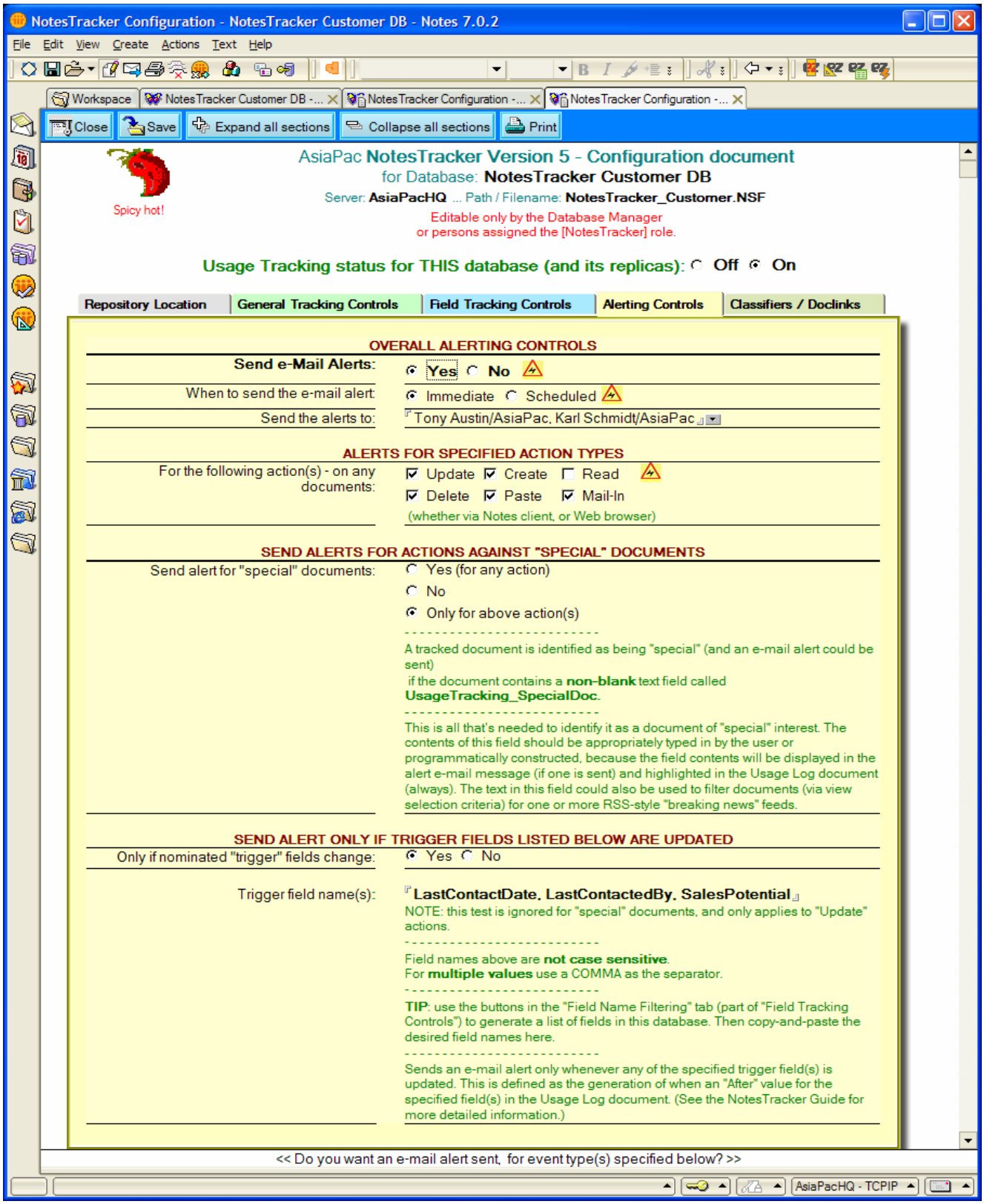
As soon as a user enters some text – and it can be *any* text – into this field, then as far as NotesTracker is concerned this is a "special document" and that's all there is to it! (it is also possible for your Notes developer to make this a non-editable field, and even a hidden field, the content of which depends on either some other field(s) that the user has populated, on some algorithm relationship between other fields in the form or read in from elsewhere, or a combination of the two. This is discussed in the Developer Topics section.)

As far at the database administrator is concerned, it is not the field itself that matters, but how it affects the NotesTracker Configuration document, so let's now switch across to this.

Alerting Controls

It is via the Alerting Controls tab that you control whether, to whom and under what circumstances e-mail alerts should be sent out by during usage tracking.

Some options are hidden at certain times. For example, the selection of “trigger fields” only appears for field updates; that is, if you select the Update action type via the check box in the central section of this tab.



Alerts for Specified Action Types

Generally you will want to keep alert e-mail traffic within reasonable bounds, both for technical reasons (network traffic, mail server overheads) and so that you don't get a flood of alerts that lead to alerting to lose its effectiveness. Therefore you can specify which of action type(s) can generate alerts for the current database.

Selecting a Subset of Action Types

Usually you can switch of alerting for document Reads (which, as stated elsewhere, typically make up 80 to 90 percent of database actions). There may be certain situations where alerting for Reads has value, however you should weigh up the benefits against the disadvantages. Alerts for document Pastes and Mail-ins might be useful, but often won't be. It is most likely that alerts will useful for document Creates, Deletes and Updates.

For Updates, you can go one step further by nominating a subset of the database's fields that can generate alerts if modified. If you are tracking the Domino Directory, say, then you might be especially interested in changes to fields in the Server Document like as "Full Access administrators" (field name "FullAdmin") and :Maximum number of threads" (field name "AdminPMaxThreads"). Or in a Travel Request application, the travel booking department might want to be alerted if an employee makes changes to fields such as "DepartureDate" and "ArrivalDate" (and probably one or two others).

Trigger Fields

Such fields are termed "trigger fields" and you specify "**Yes**" for the "**Only if nominated "trigger" fields change**" option. Only then does the "**Trigger field name(s)**" field become visible, and it is here that you list the names of the desired trigger fields.

TIP: to get spelling for the trigger field names, use the "**Field Name Filtering**" tab (under the "Field Tracking Controls**"** tab), described elsewhere in this guide. Copy-and-paste the field names from there to avoid transcription errors.

Alerts for Actions against "Special" Documents

If you do not want e-mail alerts to be sent out for "special documents", you would select the "**No**" radio button in the "**Send alerts for "special" documents**" area of this tab.

If you want e-mail alerts always to be sent out for special documents (regardless of the action type), you would select the "**Yes (for any action)**" radio button.

Otherwise you would select the "**Only for the above actions(s)**" radio button (and in the above example, e-mail alerts would only be sent out for document creates, updates and deletes).

Specifying When e-Mail Alerts are to be Sent

The option "**When to send the e-mail alert**" allows you to specify whether:

- You want the alerts to be sent "immediately" (that is, synchronously). The alert mail is sent when the document is closed. This is a dependable event in the Notes Client environment, perhaps less so in the Web browser environment (if a user simply closes the browser, say, rather than clicking on the Submit button to update the document).

- You want the e-mails to be sent later (asynchronously), via a scheduled agent in the NotesTracker Repository database. This will send out the e-mail alerts in batches, which may be a better performance option but you should carry out tests to determine whether or not this is so for your Domino environment. The schedule period specified for this agent determines how soon after the tracked event the alerts are sent out.)

Immediate versus Scheduled Sending of e-Mail Alerts

The sending of e-mails places a load on your Domino servers and communications network.

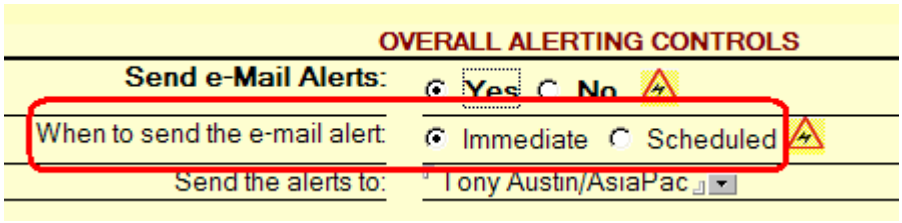
Therefore **you should be cautious about how many NotesTracker e-mail alerts get generated.**

Alerts can be generated for "special documents" and/or for regular document events (Creates, Reads, Updates, Deletes, Pastes).

Performance would be affected especially for Reads, which occur very frequently and typically make up some 80 to 90 percent of all database activity.

By all means make use of the alerting capabilities of NotesTracker, that's why they were added to NotesTracker. Just be sure to use alerting judiciously, striking a balance between their business value and the additional load they place on system resources.

If you do not need alerts to be delivered "synchronously" (at the same time as Usage Log entries being generated), you can change the e-mail delivery option from "**Immediate**" to "**Scheduled**" in the following part of the configuration document:



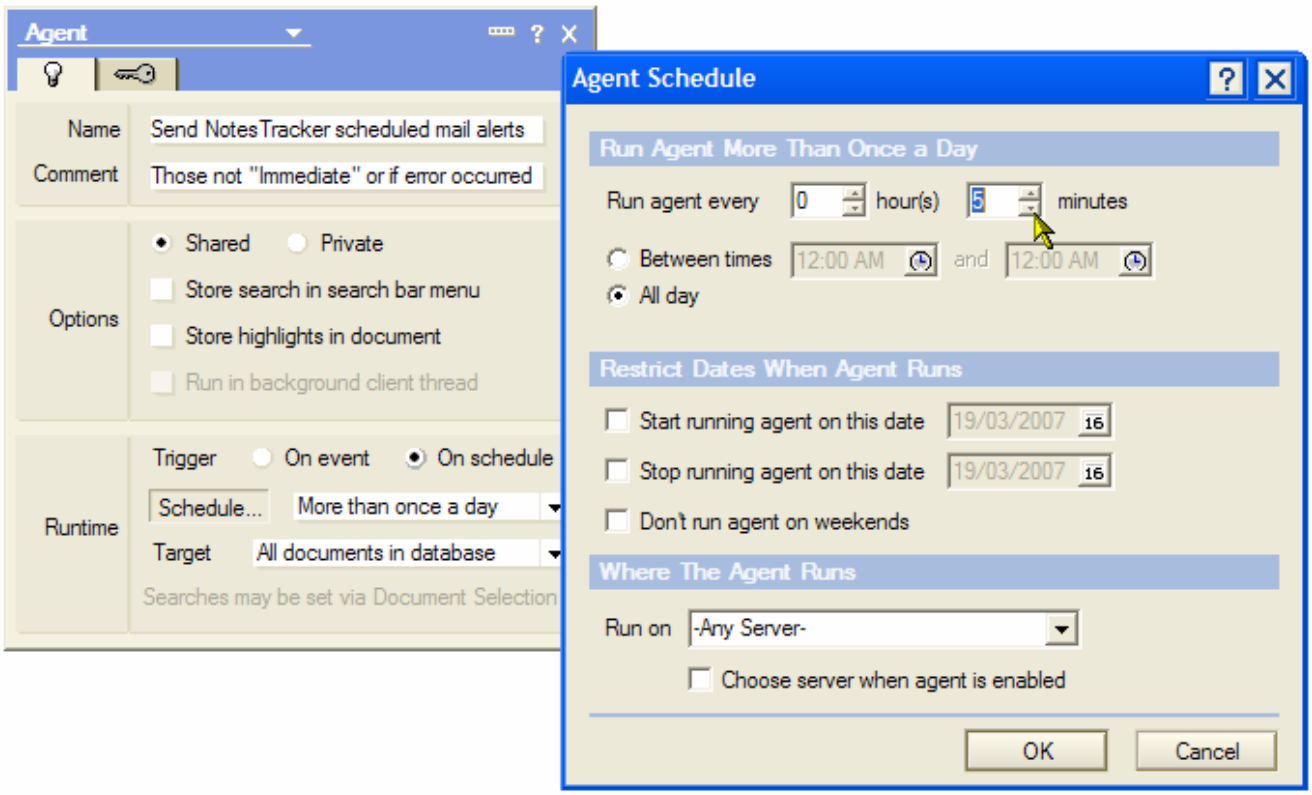
The value "Scheduled" will be stored in the UsageLog document, and e-mail alerts will not be sent synchronously for the current database. An agent in the NotesTracker Repository (or one added to the current database, if usage logging is "internal") would then send the e-mail alerts at a scheduled frequency. This frequency would be adjusted by you to best balance your Domino server workload against the desired promptness of alert message delivery.

Whether it is better to send alerts synchronously or via agents (asynchronously) will be dependent on your Domino infrastructure and on the nature plus usage patterns of each of your application databases. Only general guidance can be given here. Programming considerations for the immediate versus scheduled sending of NotesTracker e-mail alerts will be found later on, in the Developer Topics section of this guide.

Let's look into these matters a little more.

Considerations for Scheduled Alerting

The mail is sent out by a scheduled agent named “Send NotesTracker scheduled mail alerts” and, as always, it is incumbent upon the Domino Administrator to set the appropriate schedule options for the agent.



Experiment with the scheduling frequency (the “Run agent every xx hour(s) xx minutes” fields) so as to strike the optimum balance between alerting effectives and Domino server workload. For testing purposes, the minimum value of 5 minutes gives good turnaround, but this value is almost certainly far too low for your production environment.

Alerting Effectiveness, and Getting the Balance Right

Think about it. If you select “Immediate” alerting, the Notes Client workstation’s resources perform most of the work, and the alert message is sent within a second or two. (If you are using a Web browser, the work is performed on the Domino server, but the message still goes out in a second or two.) Will the recipient(s) notice the incoming alert message immediately? They might, but in many cases probably not until some minutes or even hours later.

Maybe, then, setting the agent’s schedule frequency in hours rather than minutes is adequate.

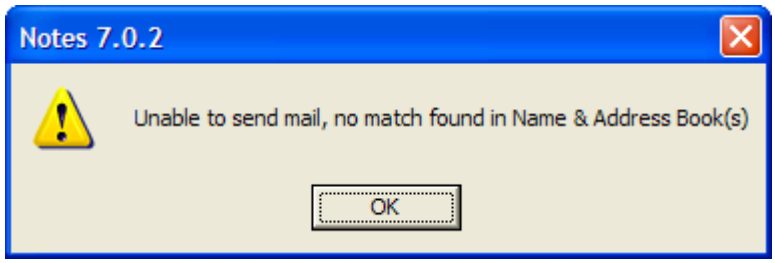
Alert Mail Recipients, and Testing of Alert Scheduling

If you indicate that e-mail alerts are to be sent out, then you must select (via the "**Send the alerts to**" address dialog) the valid Notes Mail address of at least one mail recipient.

It is essential that you carefully test to ensure that -- at the time that the alert is going to be sent -- the recipient(s) have valid entries in the Domino Directory or other address book(s).

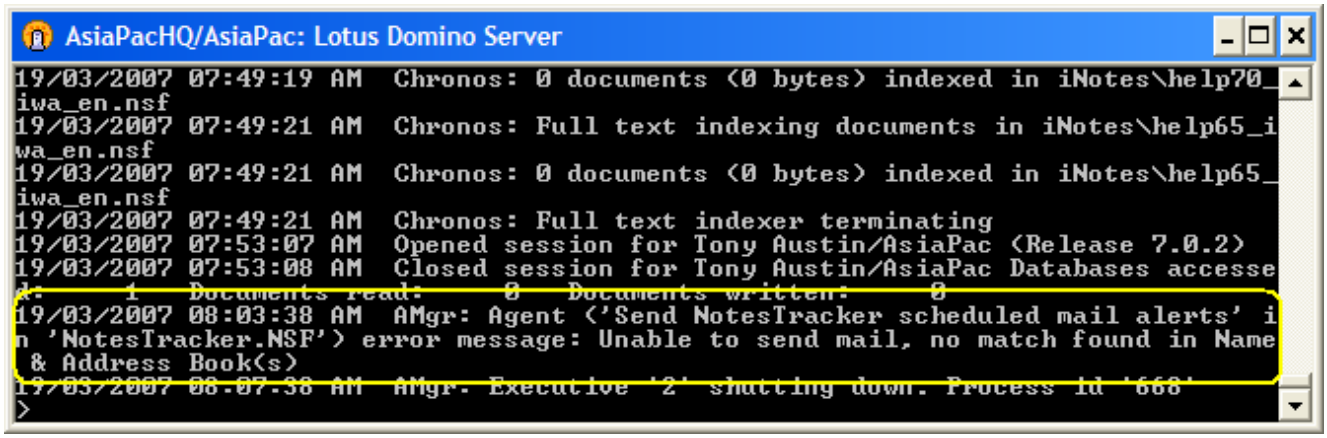
TIP: First carry out your tests using the "Immediate" option. If you intend to use the "Scheduled" option, then carry out the same tests and carefully examine the Domino console log for mail addressing errors.

It is a very good idea to use deliberately an invalid e-mail address (for example, "**No Body/There**") to force an error in both the Immediate and Scheduled alerting situations. In the former, as soon as you go to close the current document you get a mail addressing "**no match found**" error dialog box, like this:



In this case, it is quite easy to work back and determine any recipient address that is incorrectly specified. (For example, correct the "No Body/There" error and the dialog box should not appear.)

However, as with most server-based agents, the invalid e-mail address presents a less direct error message on the Domino console (repeated, of course, at every schedule period):



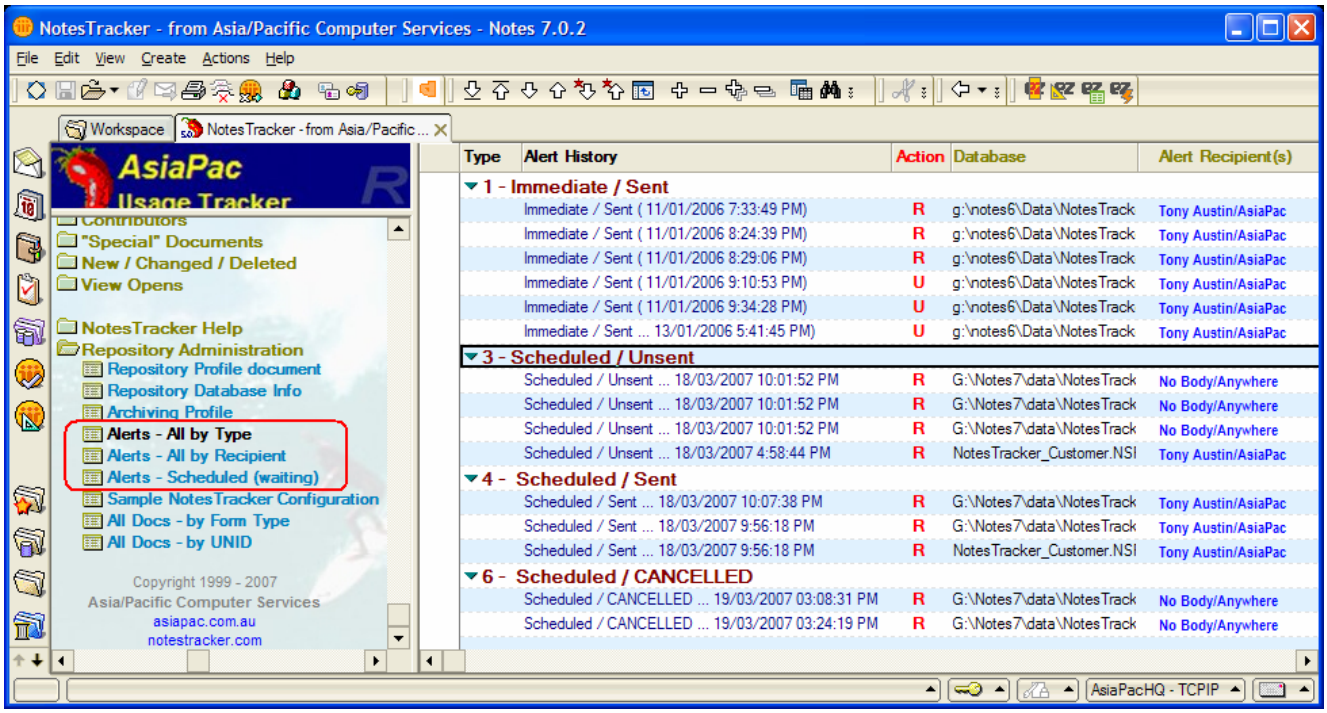
As usual, the error message will get buried in amongst all the other console messages, but a t least you have the agent name and database name as a starting point for your sleuthing work.

Managing and Cancelling Scheduled Alerts

Proactive e-mail alerting is a major new feature added in NotesTracker Version 5.0.

Alerting Views

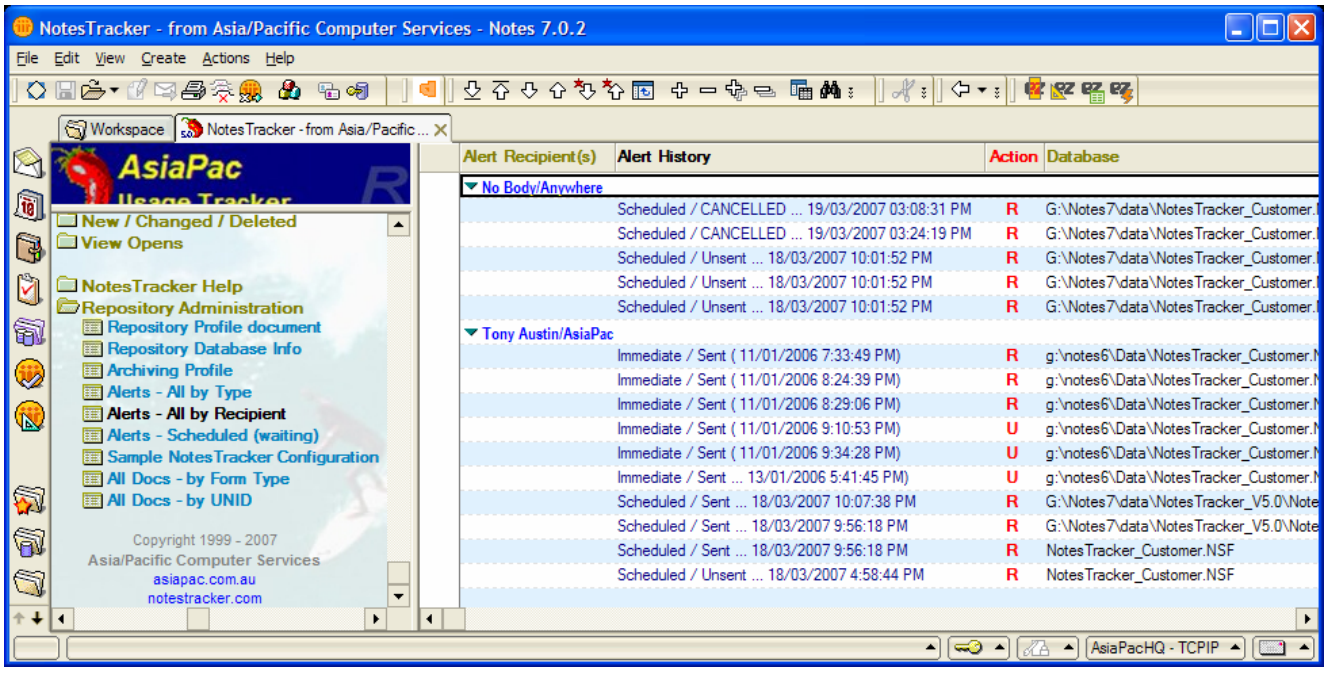
In the Repository Administration section of the NotesTracker Repository database, there are some views for viewing and managing e-mail alerting, for all of the databases that are being logged to this repository. These views are circled in red in the following illustration:



It is important to note that the rows in these views represent normal Usage Log documents, not some sort of special alerting documents. Do not make the mistake of deleting these documents, thinking that you are just deleting information about alerts. If you delete any of them, you are losing valuable usage tracking history.

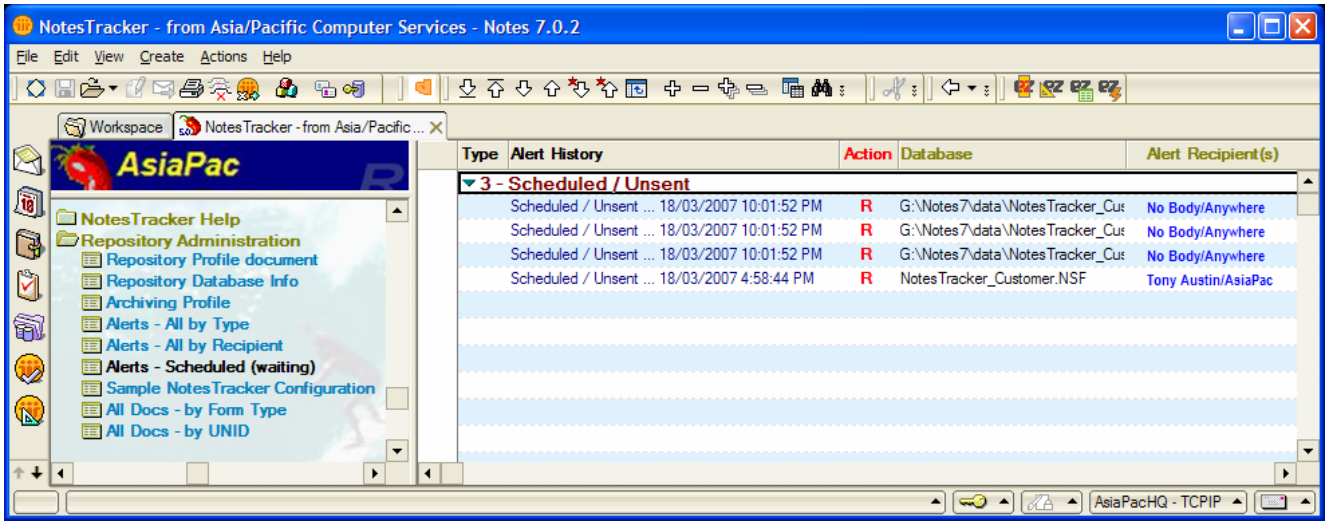
The view shown above ("**Alerts – All by Type**") gives you an overall picture of the alerting progress, categorized by alert status: whether "immediate" or "Scheduled" alerts, and their fate (successfully sent, yet unsent, in error, or cancelled).

The second view ("**Alerts – All by Recipient**") also displays every alert, but categorized by recipient:



We will see in a moment how this view can help with the management of scheduled alerts.

The third view ("**Alerts – Scheduled (waiting)**") displays the Usage Log documents for only those actions which have been scheduled to send alert e-mails and which have not yet been sent. That ism they are awaiting the next scheduled running of the agent (the name of which is "**Send NotesTracker scheduled mail alerts**"). In other words, it gives you a snapshot of the alerts that should be sent out next. It will look something like this:



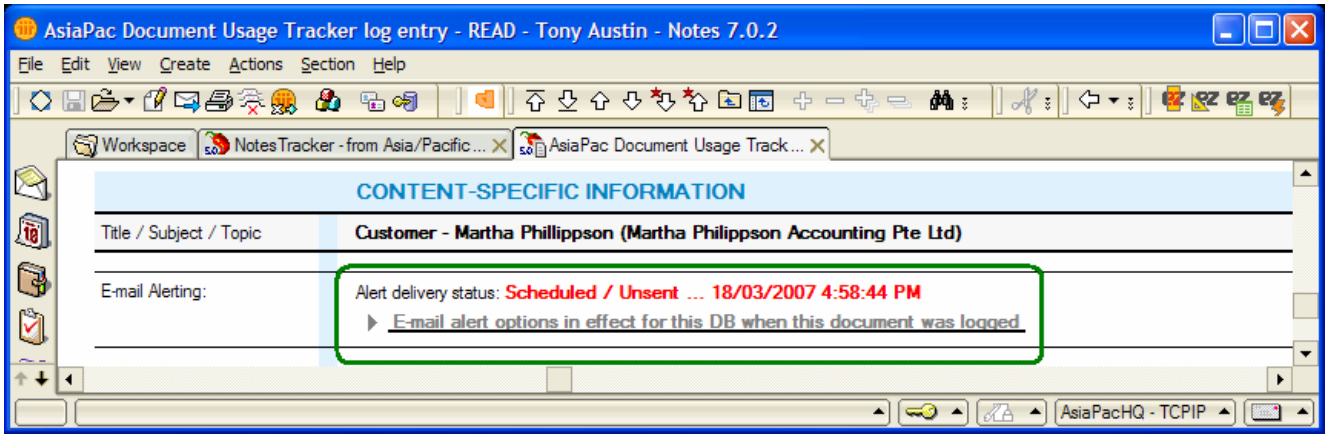
If all goes well, all of these entries should disappear -- from the view, not from the Repository database -- at the next scheduling agent run.

Managing Failures in NotesTracker e-Mail Alerting

What can go wrong with e-mail alerting, and how can problems be managed?

In the case of **"Immediate" alerting**, as described earlier there will be (in most cases) the "no match found" dialog box presented to the user, caused by incorrect recipient addresses being entered into the tracked database's NotesTracker Configuration document. This should not happen (if the administrator does adequate testing to ensure that valid mail addresses are entered). Even if this does occur, the error is obvious as an "in your face" dialog box that should lead to an early amendment of the Configuration Document.

With **"Scheduled" alerting**, the resolution is not quite so obvious. Firstly, you will start getting agent error messages on the Domino console similar to the one shown a page or two earlier. You should be able to correlate that with one of the rows in the third view, let's say the last one (for 18/03/2007 4:58:44 PM), and open the Usage Log document to see information in the "E-mail Alerting" section of the table, circled in green in the following illustration:

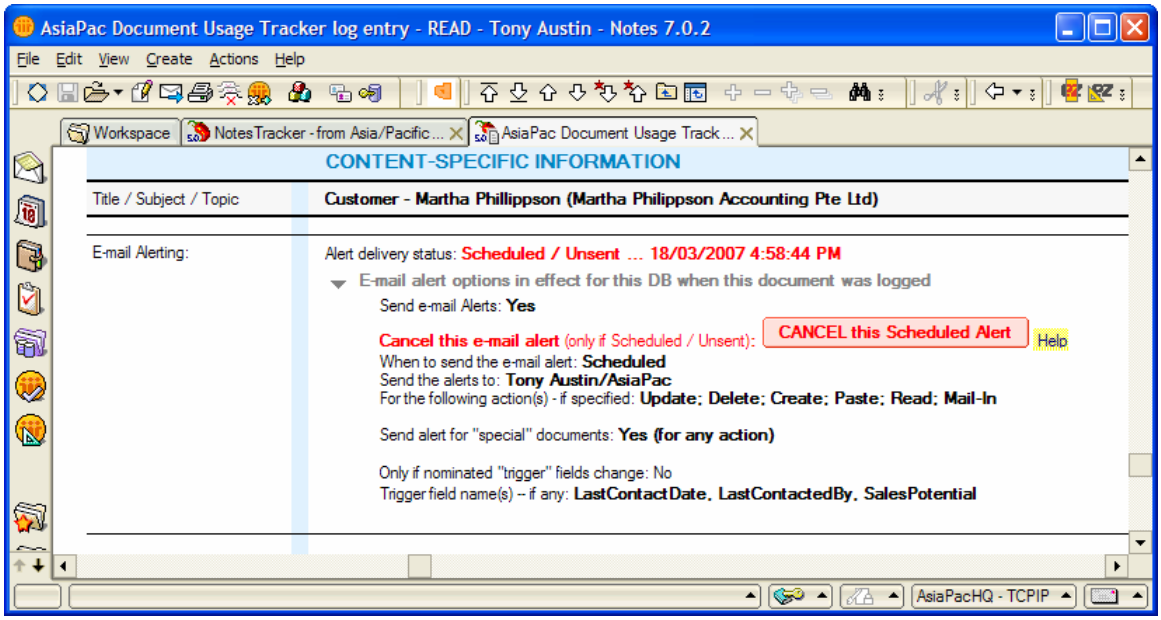


Let's suppose that there is a problem in sending this particular alert (the Send command in the agent fails, probably due to an incorrect recipient address). This effect of this is that this row remains in the view, longer that a single agent schedule interval and potentially indefinitely. This will result in a console error message being generated it once at each agent run interval.

Assuming that sooner or later you realize that this row will never generate an alert e-mail, you will want to have the row removed from the view. But you must not simply delete the row (that is, delete the Usage Log document) as explained previously!

The solution is to follow the following simple procedure:

1. Double-click on the view row to open the underlying Usage Log document, and click on the twisty in the “E-mail Alerting” section of the table, when you will see the following:



   This shows that the alert status of this Usage Log entry is “**Scheduled / Unsent**”

2. If you have the [Admin] role in the NotesTracker Repository’s ACL, you will be able to see the line containing a pink-colored button labeled “**CANCEL this Scheduled Alert**”
3. Click on this button
4. The alert delivery status should change to “**Scheduled / CANCELLED** … “ appended with the date and time that you clicked the button.
5. The row with the pink button will not be visible next time the Usage Log document is opened. This prevents repeated cancellations of any given alert.
6. It would be nice -- if you can work out who is/are the proper alert recipient(s) – to contact the alert recipient(s) and explain that you have had to cancel the alert.
7. It is suggested that, as a courtesy and if you are able to establish the correct e-mail addresses, you send an e-mail message containing a Doclink to the Usage Log document so that the recipient(s) can examine the nature of the alert.
8. See to it that the relevant NotesTracker Configuration document is amended so that the error does not continue.

Using this rather painless approach, maybe with modifications to cater for variations in the error cause, you should be able to resolve alerting errors fairly easily.

NotesTracker Database Size, View Indexes, and Performance Considerations

The Build-up of Usage Log documents, and View Index 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.

Of course, it's not only document contents that take up space in a database. Keep in mind that view indexes 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.

As a guide, one user of NotesTracker found that some 60,000 Usage Log entries used close to 1 GB of disk space, and you may get similar results.

Disk Space Management – Archiving Agent

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. The archive agent is discussed a little further on.

Index Sizes of "Breaking News" Views

In the Developer Topics section near the end of this guide, you will find a discussion of NotesTracker's capability to easily build newsfeeds (RSS-style views).

A proliferation of new views in the NotesTracker Repository ordinarily might be a worry for database administrators. However it is likely that every such view will have a view selection formula that keeps the number of selected documents quite small, and hence the view's index size should not turn out to be of much concern.

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 cognizant 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.

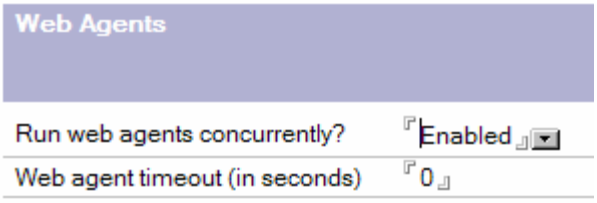
Some other related performance considerations are discussed several pages above, in the section on Web Browser usage tracking. For some general background on Domino web server performance, refer to the Lotus product documentation and to IBM developerWorks articles such as:

- The Architecture of the Domino Web Server – Part 1
  at http://www-10.lotus.com/ldd/today.nsf/9148b29c86ffdcd385256658007aaa0f/89fb100bf3d93ec68525645200615a95
- The Architecture of the Domino Web Server – Part 2
  at http://www-10.lotus.com/ldd/today.nsf/b1d67fedee86c741852563cc005019c5/79abe7c0a0137699852564650057702a
- Introduction to Domino performance tuning
  at http://www-128.ibm.com/developerworks/lotus/library/ls-perf_intro/
- Troubleshooting application performance: Part 1: Troubleshooting techniques and code tips
  at http://www-128.ibm.com/developerworks/lotus/library/app-troubleshooting1/
- Troubleshooting application performance: Part 2: New tools in Lotus Notes/Domino 7
  at http://www-128.ibm.com/developerworks/lotus/library/app-troubleshooting2/?track=FG_DDM
- Coding faster lookups in IBM Lotus Notes and Domino
  at http://www-128.ibm.com/developerworks/lotus/library/notes-lookups/index.html

Tip – Allowing More 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

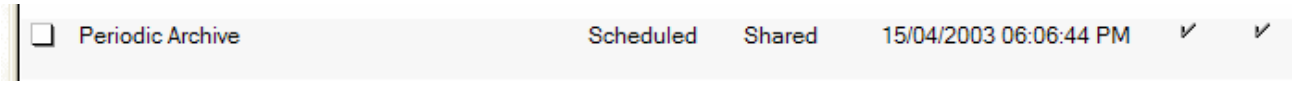
You can experiment with this by editing the **Internet Protocols** section of the Server Document:



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

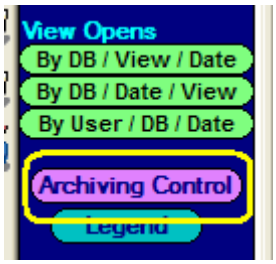
An important feature of NotesTracker (added in Version 4.2) is an archiving agent.



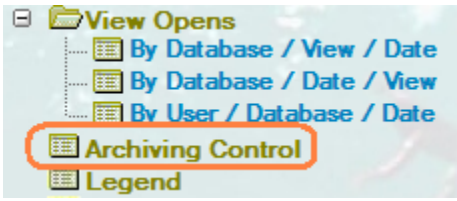
TIP: this agent is largely based on the archive agent in the Lotus R4 mail template, and you may find additional information about using it in the Lotus documentation.

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).



or



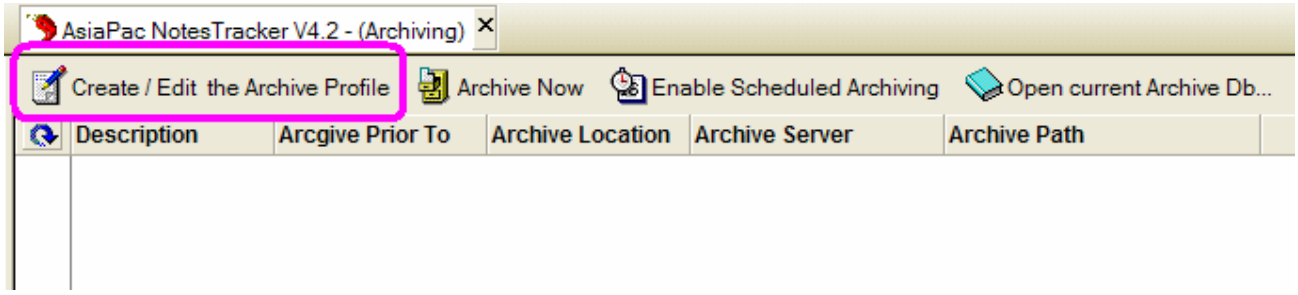
The agent is designed to work similarly to the archiving agent in your Notes Mail database.

Setting Up NotesTracker Archiving

The archiving view is designed to hold a single NotesTracker Archive Profile document. You must create and configure this profile document before the agent will run.

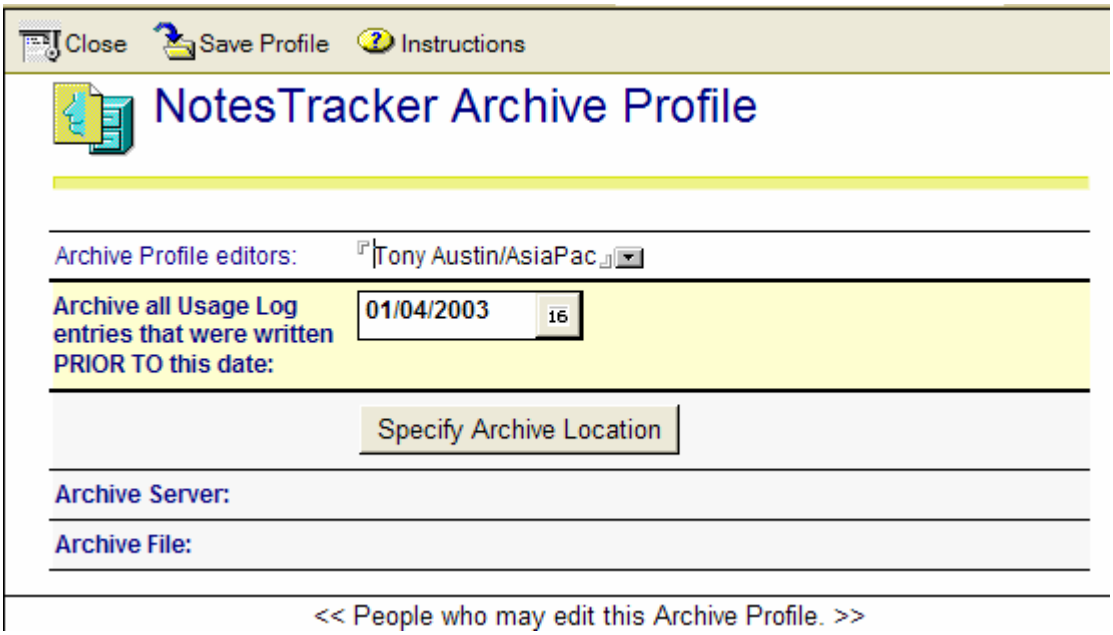
Note: you must have **Manager** or **Designer** access level rights to be able to view the action buttons described below for setting up and running the archiving agent.

The very first time that you click on the Archiving Control 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.

While it is being edited, the **NotesTracker Archive Profile document** will look like this:



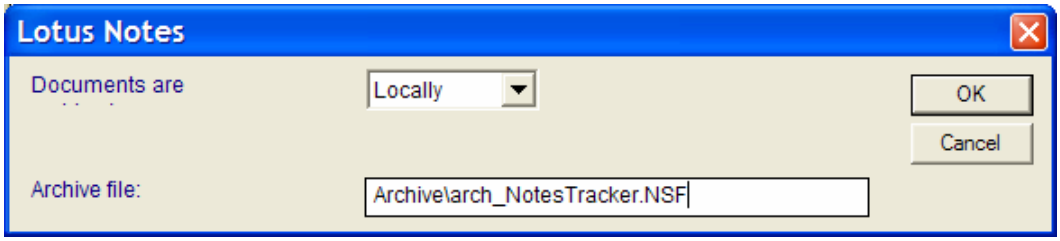
You need to specify three things:

1. The person(s) who are allowed to edit the NotesTracker Archive Profile document.

2. 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 (and is largely dependent on the rate at which Usage Log documents are generated, plus how long you wish to keep them online for the viewing of historical usage patterns and trends).

3. 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.

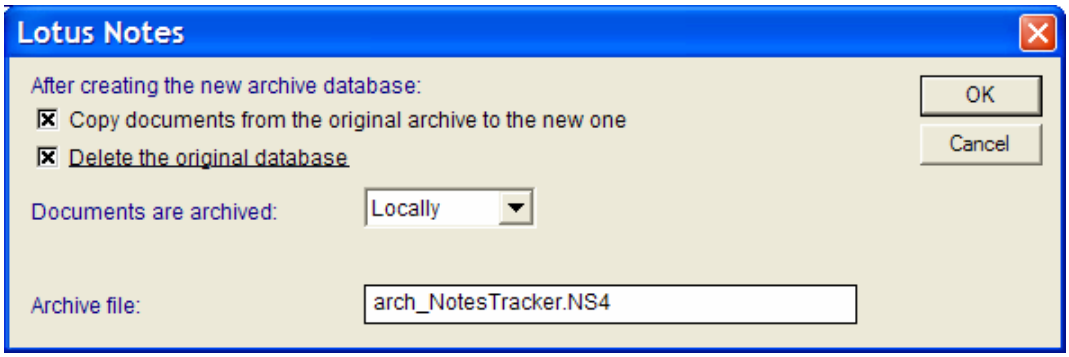
   The archive file name is the path to the Notes database which will receive the archived Usage Log documents. The file name can be preceded, in the normal fashion, by a directory path (in the proper fashion for the operating system, and relative to the Domino data directory).

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

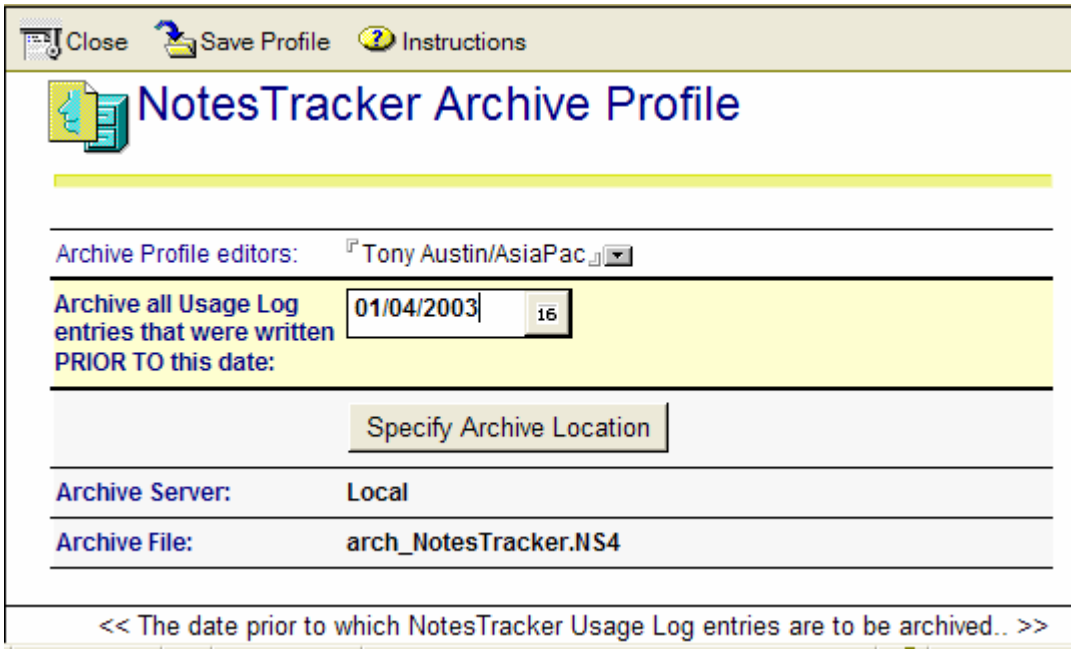
   Note: the archive database can be any database to which you have rights to add documents. However, it will make life easier for you if the archive database is a copy of the NotesTracker Database (which is the default), since this will contain all the regular NotesTracker views. This enables you to view 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.)

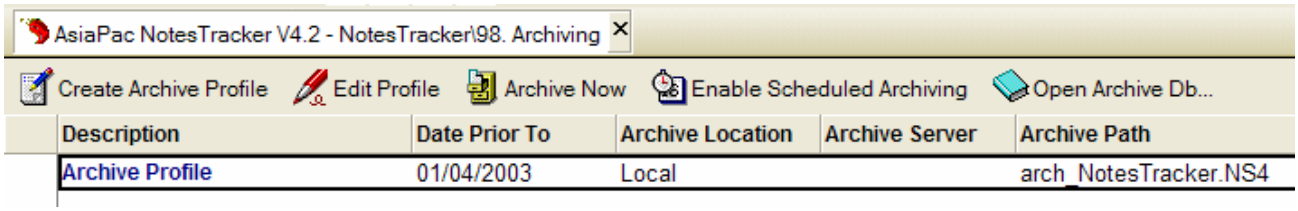
   If you click the button after the archive database has been created, you will see a different form of the dialog box:



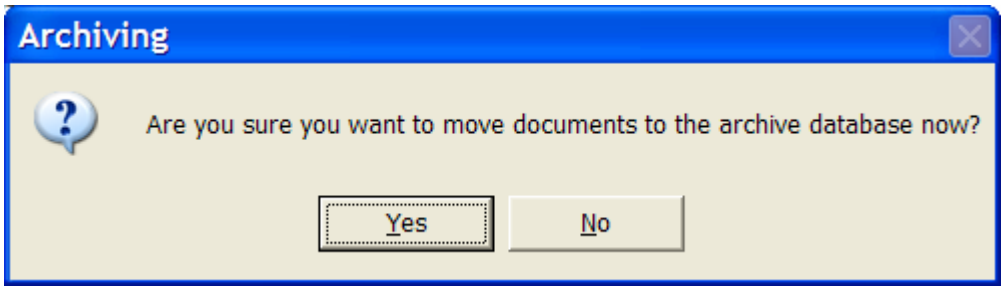
A completed Archive Profile looks something like this:



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

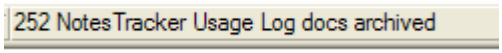


At this stage, you can click the **"Archive Now"** action bar button to cause the archiving agent to run immediately. You are asked for confirmation:

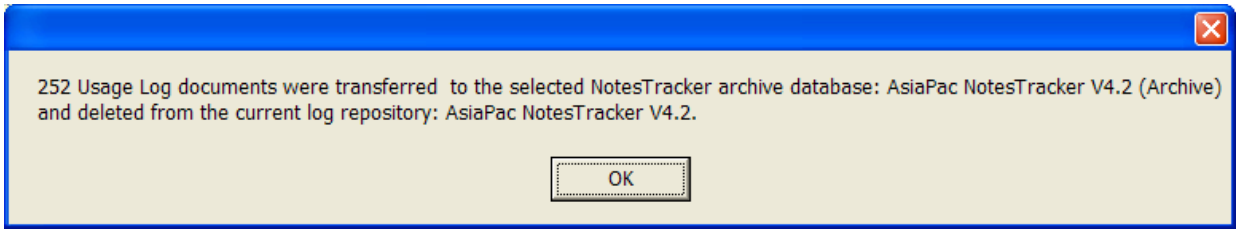


If you confirm by clicking the "Yes" button, then the agent runs on your Notes Client workstation.

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:



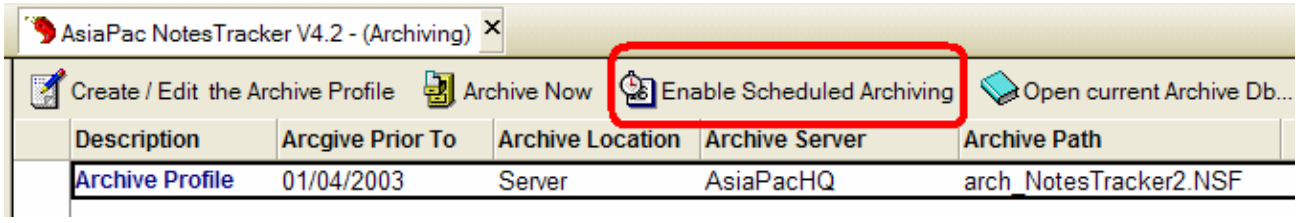
Upon successful completion you will see a dialog box like the following:



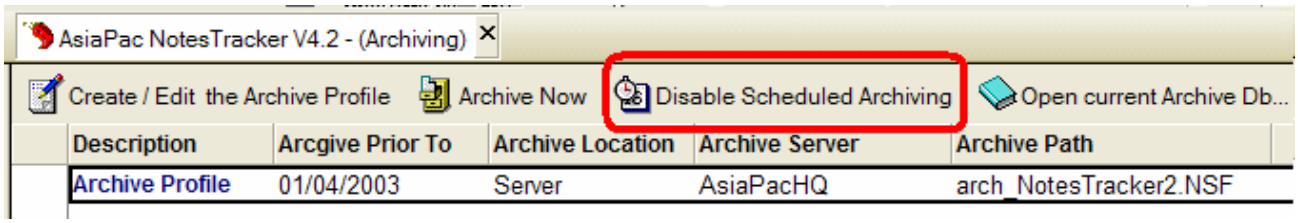
Alternatively, you can **schedule the Archiving Agent to run later**. You do so using normal techniques for setting up the agent's schedule (daily, weekly, monthly, etc) and for activating the agent... It is beyond the scope of this NotesTracker Guide to explain this any further.

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 in order to analyze 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 organization to organization. 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.

As mentioned earlier, on NotesTracker user reported that some 60,000 Usage Log documents took up 1 GB of disk space.

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.

Click on the “Open current Archive Db” button to open the database 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 minimize 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 organization 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.)

Adapting a Database's Design for Usage Tracking – Overview

For each database that is to be tracked, you only need to follow a few simple steps.

With just a little practice, it should take you only minutes or seconds per design element (form, view, etc) to implement usage tracking in a database!

The Planning Stage

At the outset, it is very important that you meet with the executive sponsor, knowledge manager or database owner for each database and draw up a **list of actions that are to be tracked for each database** and other reporting options, such as what sort of "document title" is to be stored for each tracked document (as explained elsewhere). You might choose to track all forms 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. You may want to track just Notes Client actions, or just Web browser actions, but perhaps both. You might or might not decide to track document deletions. You generally won't want to track view usage, and so on.

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. Begin by just tracking document Creates, Reads and Updates, before attempting to track other actions, and then add other actions when you get start getting the hang of things. Only then start implementing NotesTracker in other databases that have more complex designs: those with many forms and views, and with complex code structures (the likes of the Notes Mail template or similar). In the simple databases, experiment with the various NotesTracker Configuration Document settings, checking that usage logging occurs properly under all circumstances in both the Notes Client and the Web browser environments.

Design Modification - The Golden Rule

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

How often have you seen software "enhancements" or "fixes" that produce noticeably different behavior 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).

Naturally, **all design changes always must be followed by adequate** testing, to ensure that the application works exactly the same as before (regression testing) and also that the added NotesTracker functions work as anticipated.

Regression testing ensures that the application performs without functional or usability impairment – sometimes referred to as the **"*Do No Harm*"** design principle -- and that the application's performance and reliability remain acceptable. Verification that all desired usage tracking options work satisfactorily covers such matters as (a) all expected types of usage log entries being generated in the NotesTracker Repository database; (b) e-mail alerts messages being sent; and (c) absence of NotesTracker error messages at Lotus Notes client workstations or Web browsers.

Design Steps – Summary Table

These are explained in full detail on subsequent pages. Not all steps need necessarily be performed against each database, although several steps must be. Due to interdependencies, some but not all steps must be performed in the order shown.

	DESIGN MODIFICATION MADE TO THE TARGET DATABASE	WHEN THE STEP MUST BE CARRIED OUT	UPGRADING FROM EARLIER VERSION	ASSISTED BY FAST PROPAGATE
STEP 1 Page 95	**1A, 1B**. Copy in the Usage Logging, CGI Variables and Simple Edit Tracking Footer subforms	If document Creates, Reads, Updates are to be tracked.	Enhanced (replace the subforms)	✔ (button 3A or 3B)
	1C. Insert the Usage Logging subform in each form being tracked..	Once per form being tracked.	No action needed.	(Not applicable)
STEP 2 Page 96	Copy in the AsiaPac_UsageTracker script library.	Always.	Enhanced (replace the library)	✔ (button 3A or 3B)
STEP 3 Page 97	Copy in the Database Script subroutines.	Always.	Enhanced (replace the subroutines)	✘
STEP 4 Page 99	**4A**. Set up the "UsageTracking_Title" special field	For each form where a meaningful "title" must be set for a Usage Log entry (if and where needed).	Unchanged	✘
	4B. Set up the "UsageTracking_SpecialDoc" special field.	For each form involved with identifying "special" documents field (if and where needed).	New feature	✘
STEP 5 Page 106	**5A**. Add the NotesTracker Configuration view.	Always.	Enhanced (replace)	✔ (button 3A or 3B)
	5B. Add the NotesTracker Configuration form.	Always.	Enhanced (replace)	✔ (button 3A or 3B)
	5C. Create and edit the NotesTracker Configuration document.	Always.	Edit the new V5 options.	✔
STEP 6 Page 107	Optionally, add Usage Log views and a Breaking News view. Check that all existing views work properly.	Only for "internal" usage logging (rather than using an external NotesTracker Repository).	Enhanced (replace)	✔ (button 3B)
	Add Usage Log archiving capability.			
STEP 7 Page 109	Set up or suppress usage tracking at the individual document level (via the "UsageTracking_TrackThisDocument" special field).	Only if this is a tracking requirement. (Probably rarely used.)	Unchanged.	✘
STEP 8 Page 112	Enable Web browser usage tracking (various sub-steps).	Only if Web browser tracking is required.	Enhanced (replace)	✔ (button 3A or 3B)
STEP 9 Page 114	Check that the database still has the original default view and default form. (Set them if missing.)	These settings will be overridden by Fast Design Propagate (or can be disrupted by other causes).	Unchanged	✘
STEP 10 Page 115	Implement the tracking of View Opens (various sub-steps).	Notes Client only, and only if desired (rarely used).	Enhanced (replace)	✘

LEGEND: ✔ Assisted by the "Fast Design Propagation" feature; may require manual work too. ✘ Always requires manual work.

Note: Some considerations for upgrading to NotesTracker V5 are given on page 121.

Step-by-Step – the Design Changes in Detail

STEP 1 – Add the NotesTracker Subforms

STEP 1A – Add the CGI Variables Subform

(Once per database)

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

Name/Comment	Alias
CGI Variables 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 – Add the Document Usage Tracking subform

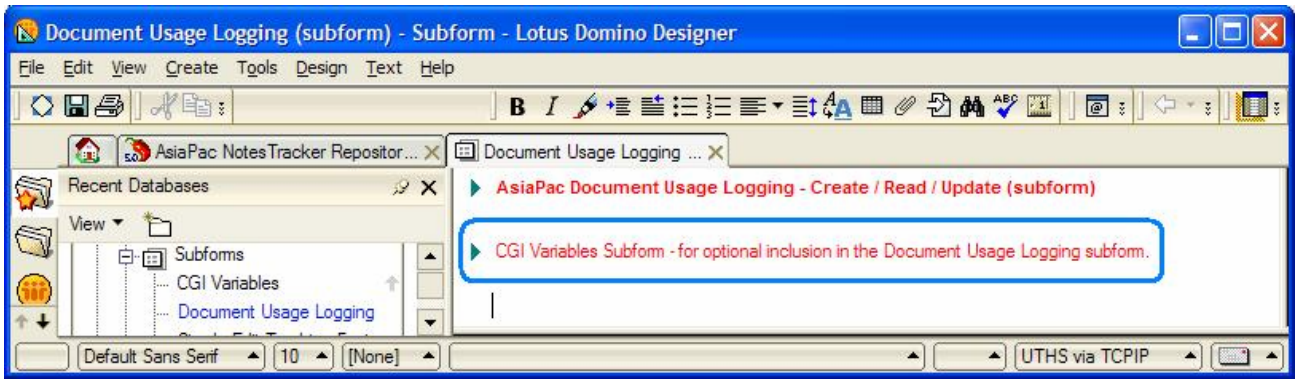
(Once per database)

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

Name/Comment	Alias
Document Usage Logging (subform) 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 fields defined in the CGI Variables subform are only needed if you implement Web usage tracking and then also select that CGI variables are to get logged. However it was decided to make it *a routine step* to include this subform in the design (always carrying out STEP 1A before STEP 1B), since STEP 1A is a trivial operation and feedback showed that doing this causes less trouble overall.

STEP 1C – Insert the Document Usage Tracking Subform into Each Form

(Once per form that is used for documents being tracked)

Insert the subform named "**Document Usage Logging (subform)**" into each form that will be used to create, read or edit any document that you want to track. It is probably best to insert the subform at or near the top of the form.

Note: inserting this subform into a form will also insert the "**CGI Variables**" subform nested within it. (This is why the CGI Variables subform must already be present in the target database.)

STEP 2 – Insert the NotesTracker Script Library

(Once per database)

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

STEP 3 – Modify the Database Script

((Once per database)

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.**

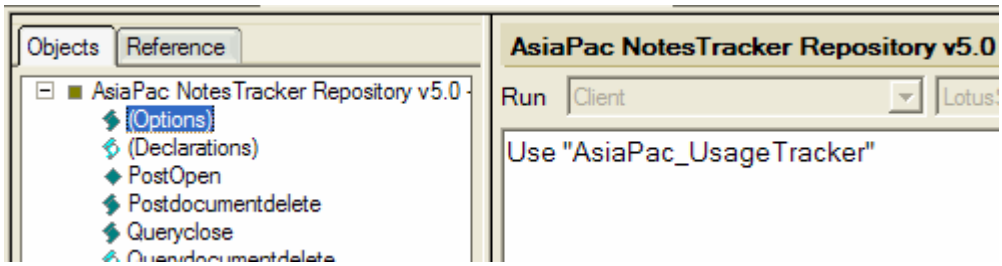
Because there may already be code in some of the routines in the database script, the NotesTracker code cannot just be copied in (either manually or using the "Fast Design Propagation" feature) and must always be modified by hand.

Note: If you already do have code in any of the affected subroutines, you must determine how to merge the NotesTracker code into the existing code so that current database operations are not affected.

STEP 3A – Setting the "Use" Statement for the Database Script

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

Use "AsiaPac_UsageTracker"



TIP: There is a trap that you can avoid: Do not use a copy-and-paste operation *from this guide* to transfer the above statement (**Use "AsiaPac_UsageTracker")** into the Database Script. Experience shows that when you do this and try to save the Database Script, you will get an unexpected error message when you try to save the Database Script:

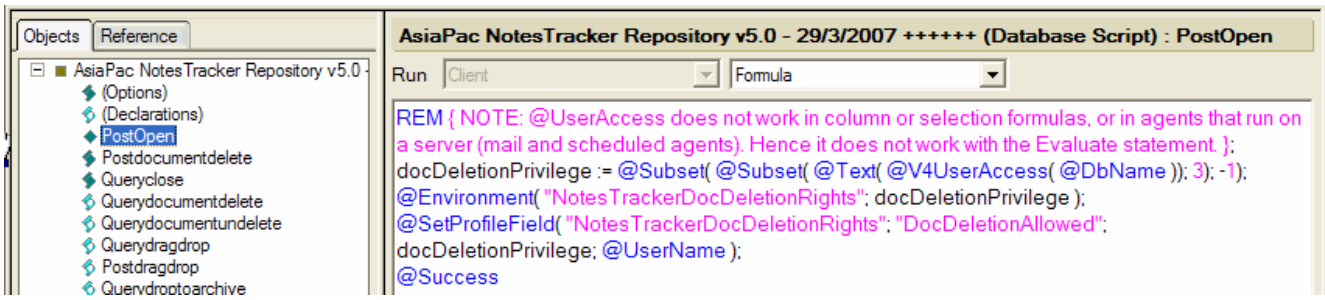
Not a constant: "ASIAPAC_USAGETRACKER"

This happens because the copy-and=paste usually transfers invisible character(s) that are regarded invalid by the LotusScript compiler, and so get rejected by it. The statement *appears* to be valid, but since it is invisible character(s) that prevent the script from compiling you will be quite puzzled by the script not being able to be saved!

Therefore, it is best to **type this statement** into the Database Script (all 26 characters of it) rather than copying it.

STEP 3B – Setting the Postopen Subroutine for the Database Script

Go to the Database Script of the NotesTracker Repository, open the Postopen subroutine, go to the target Database Script's Postopen event and change the language from LotusScript to Formula, then copy-and-paste its entire clipboard contents into its **Postopen** subroutine:



STEP 3C – Setting the Postdocumentdelete Subroutine for the Database Script

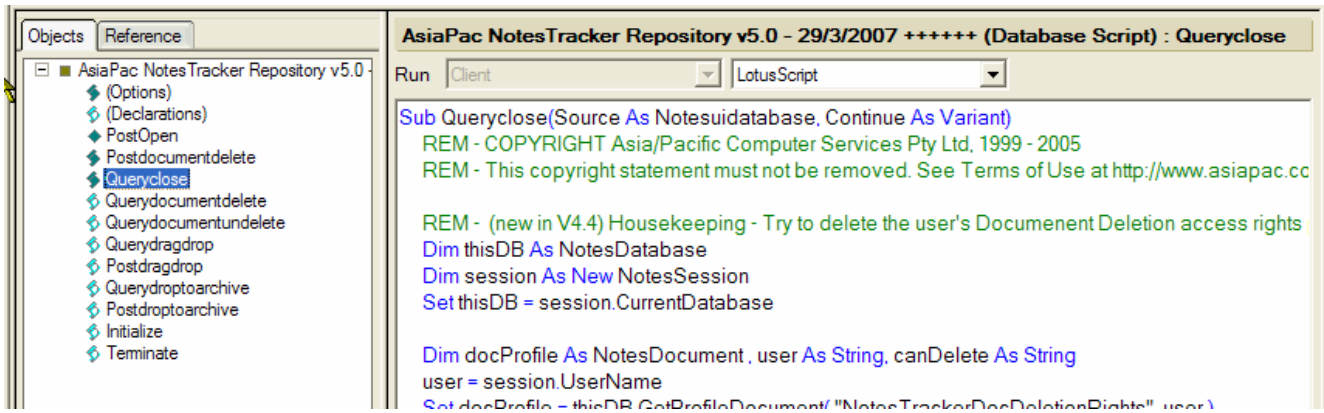
Go to the Database Script of the NotesTracker Repository, open the Postdocumentdelete subroutine, then copy-and-paste its entire clipboard contents into the target database's **Postdocumentdelete** subroutine:

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)

STEP 3C – Setting the Queryclose Subroutine for the Database Script

Go to the Database Script of the NotesTracker Repository, open the Queryclose subroutine, then copy-and-paste the entire clipboard contents into the target's **Queryclose** subroutine:



STEP 4 – Set Up any Usage Log Entry "Title" or "Special" Fields

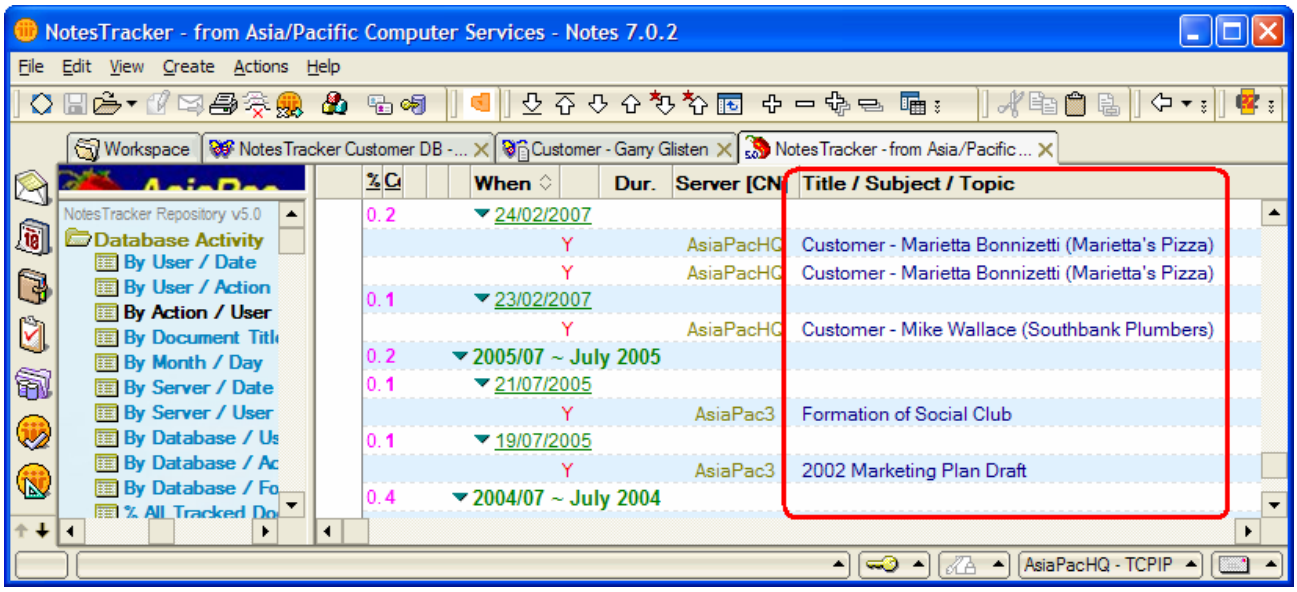
STEP 4A – Setting Up Usage Log Entry "Title" Fields in the NotesTracker Configuration Document

The aim with this step is to ensure that a meaningful "title" or "topic" or "subject" or "summary" descriptor appears in each Usage Log document in the Repository database. This is crucial for making sense of the Usage Log views.

For each form, determine if there is already a field named "**Title**" that contains a suitable text string which adequately identifies each document. If there is, then the contents of this field should be suitable to be used for the Usage Log title too,

CONTENT-SPECIFIC INFORMATION	
Title / Subject / Topic	**Customer - Anatole Piquet (Programmation sans Travail)**

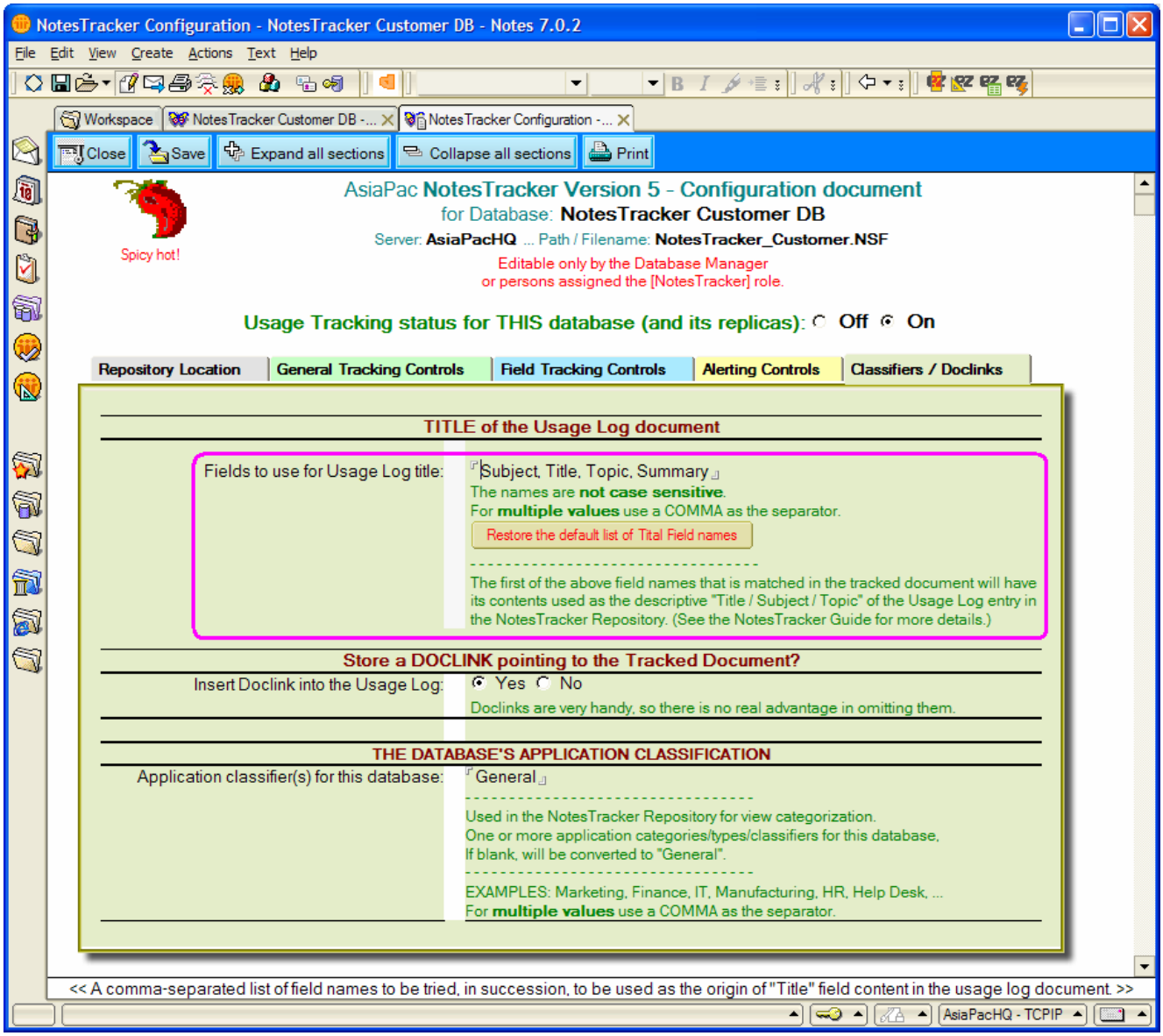
There may be no such string. Another common situation is that a field named "Title" does exist in the document, but it is only holds a personal salutation (such as "Mr.", "Mrs.", "Ms", "Dr", "Prof"), and it would be ridiculous to have these (Mr., Mrs., Miss, etc) in the Title / Subject / Topic column that appears in all of the Repository views:



But what if there is no field named **Title** in the current document?

In the NotesTracker Configuration document, under the **Classifiers / Doclinks** tab, there is the ability to specify a list of field names that NotesTracker should use to search for successively in the document being logged. NotesTracker will use the contents of these nominated fields for the "title" for the Usage Log document.

The list of default field names is shown in the following illustration, and you can edit this list to suit the current database:



The field names are scanned by NotesTracker in the order that they are listed.

However, in some circumstances such a simple list of alternative field names may not enable NotesTracker to generate a meaningful or appropriate title for the Usage Log entry. What then?

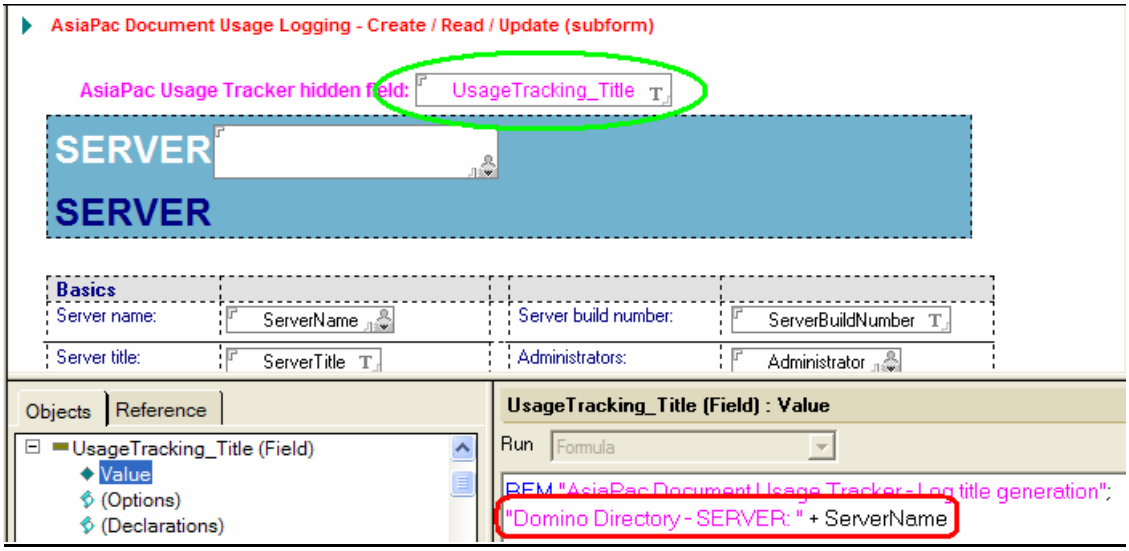
STEP 4B – How to Set Up a "UsageTracking_Title" Special NotesTracker Field

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

NotesTracker gets over this predicament by the simple method of enabling you to set up special text field named "**UsageTracking_Title**" which normally (but not necessarily) would be **computed-for-display** and **hidden** and which you should programmatically populate with a realistic title / subject / topic / summary Usage Log descriptor.

What you would put in this field is hard to state specifically. And depends rather a lot on the nature of the current database plus what sort of data is held in the current document. In practice, with a little bit of experimentation you should easily be able to come up with a suitable algorithm for populating this field. You might, say, experiment with concatenating the "Category" field or even the "Form" field -- or whatever other useful fields are available in the document -- with the current system date and time (always storing the concatenated result as a text string).

To get some more ides, examine the forms in the Domino Directory R4.6 database that is included in the NotesTracker distribution package. Each form has a formula that relates to the nature of the form. Here's the Server Document, for instance:



This is all quite easy to do in practice. Indeed, it's another case of the description of the process being harder than carrying it out.

Note to users of NotesTracker Version 1 and Version 2: Prior to Version 3, this special field was named "UsageTracker_Title" and not as it is (for field naming consistency reasons, from Version 3.0 onwards) "UsageTracking_Title". Therefore when upgrading to NotesTracker Version 3 or later from earlier versions you would have to make the necessary adjustments to your forms designs accommodate this change of field name.

STEP 4C – Specifying and Handling "Special" Documents

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

As mentioned earlier in this guide (see **Proactivity – e-Mail Alerting, and "Special Documents"**), a major new feature introduced with NotesTracker Version 5.0 is the ability to nominate certain documents in a database as being "special" and thereby give them some sort of extra focus and treatment -- say, highlighting the documents in the Usage Log Repository so that they can more readily be examined, or sending out e-mail alerts about them as soon as they are updated or deleted.

The question is, how do you nominate a particular document in a database as being special? And the NotesTracker solution is a simple one: just by adding another "special field" to that document and putting some sort of meaningful text into it.

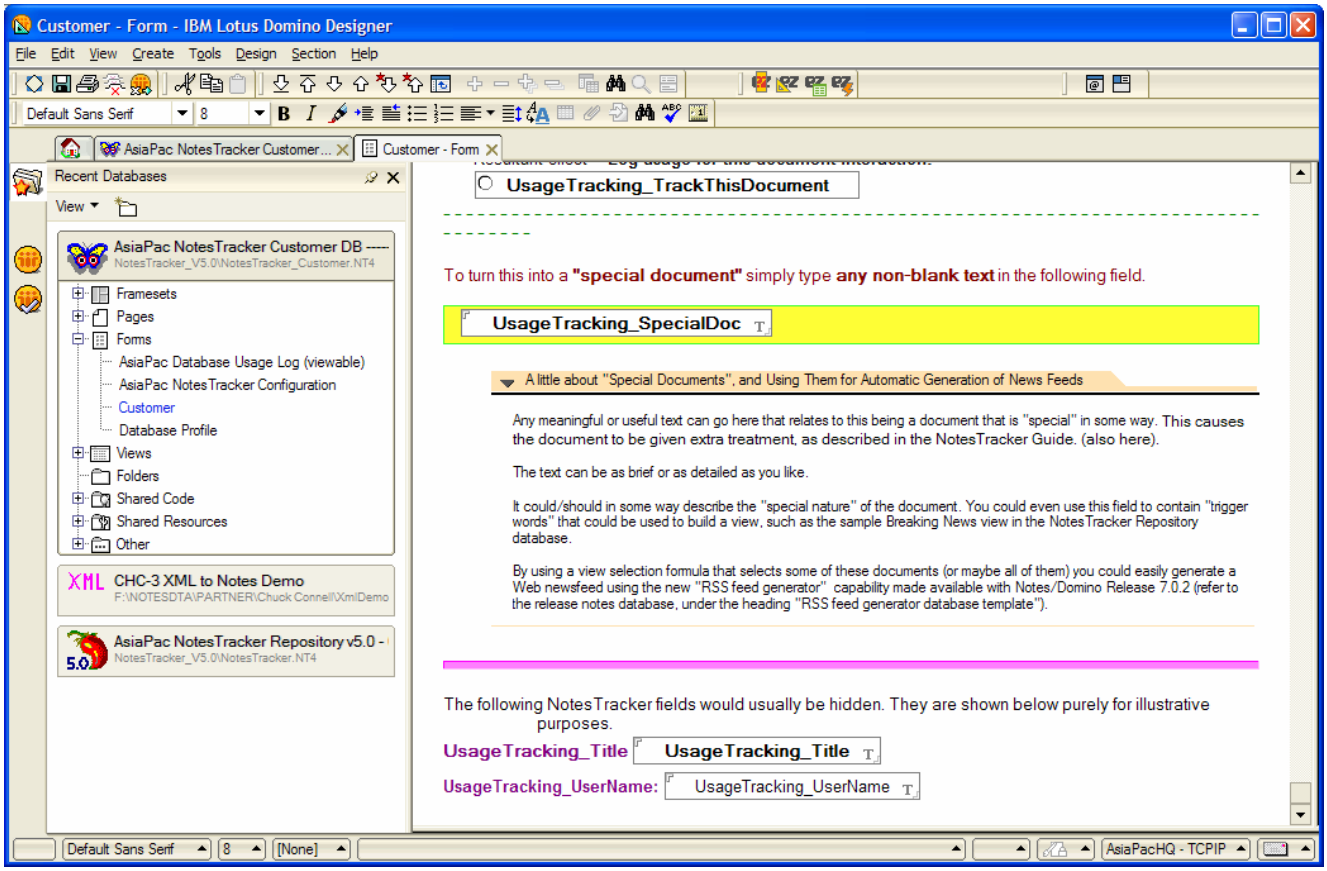
STEP 4D – How to Set Up a "UsageTracking_SpecialDoc" Special NotesTracker Field

You (as Notes developer) just arrange for a document to have a field that is named "**UsageTracking_SpecialDoc**" (of type "Text"), and then either:

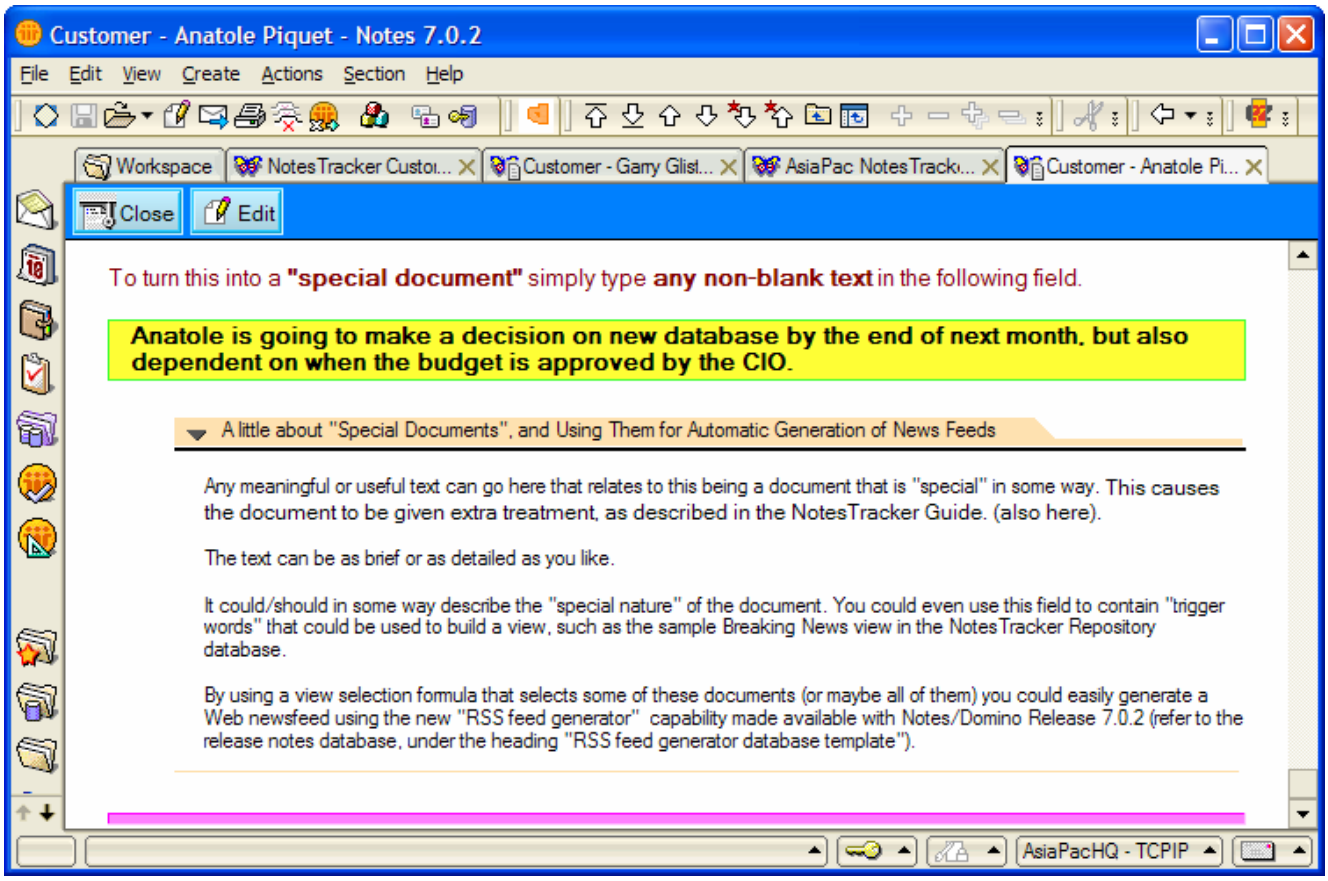
- Have the **user** enter any non-blank character(s) into the field;
  For example, in a travel request application, the user could type "I require vegetarian meals on board" or "Please put me in a seat near one of the emergency exits that provides extra legroom" or similar.
  or
- **Programmatically** set the field so that is non-blank.
  For example, assemble the field's contents from data in other fields in the document (or from other documents in the same or a different database, or from sources external to Notes/Domino) – it's a case of "anything goes."

That's all there is to it! The user isn't aware of it, but as soon as NotesTracker detects that this field is non-blank it treats the document as being "special" and from then on does a few extra things when handling it. Although it's extremely simple to implement, there are many ways you can use this to your advantage. Indeed, the possibilities are virtually limitless.

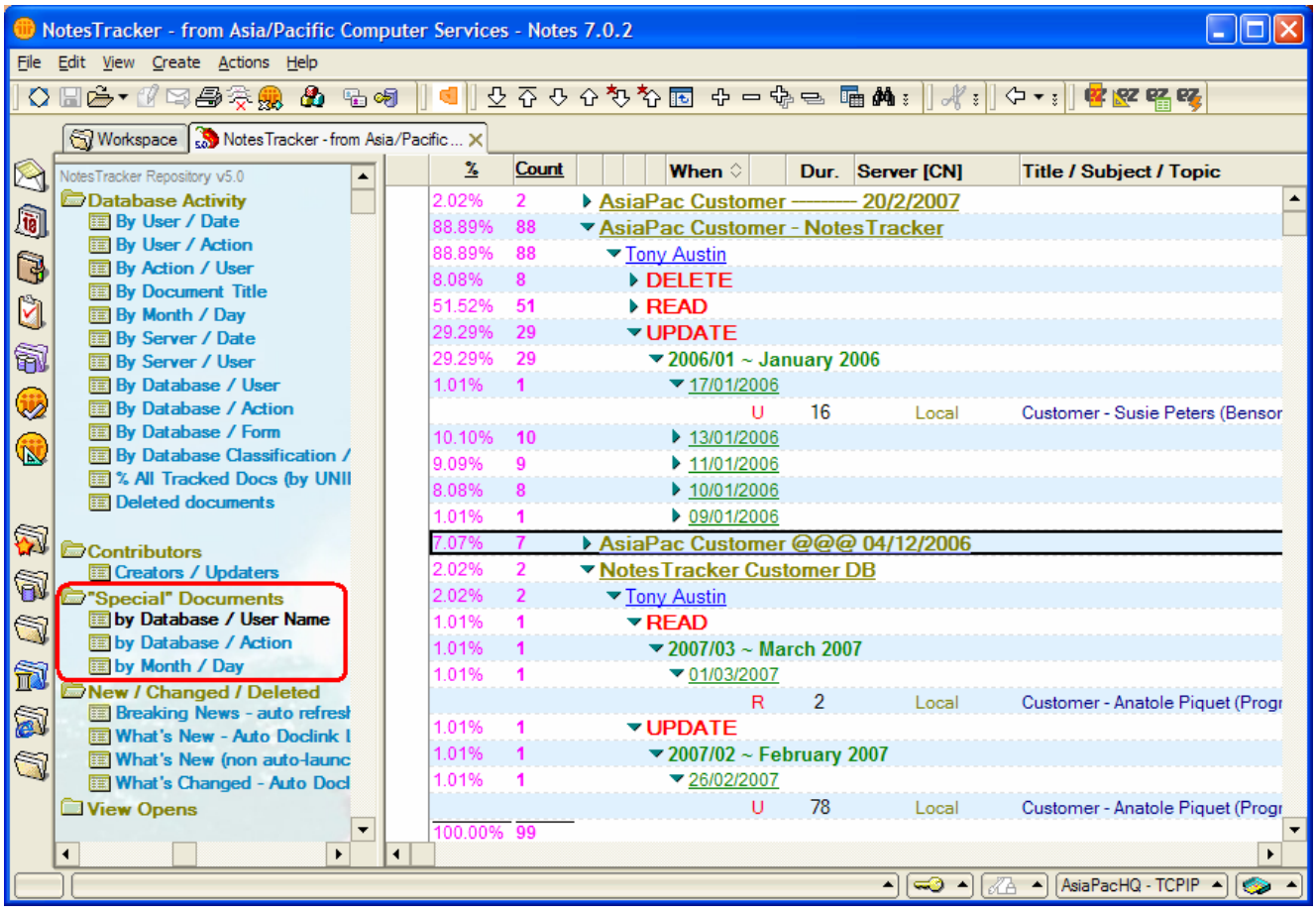
It's easiest to give an example. In the **NotesTracker Customer DB** provided as part of the NotesTracker distribution package, there is just a single "Customer" form, and in that form (inside a table with a bright yellow background) you will find the requisite "**UsageTracking_SpecialDoc**" plain text field, as shown below:



In this example, the user merely types some relevant text into this field (anything meaningful, as long as it's made non-blank) and that's all there is to it. Here's an example:

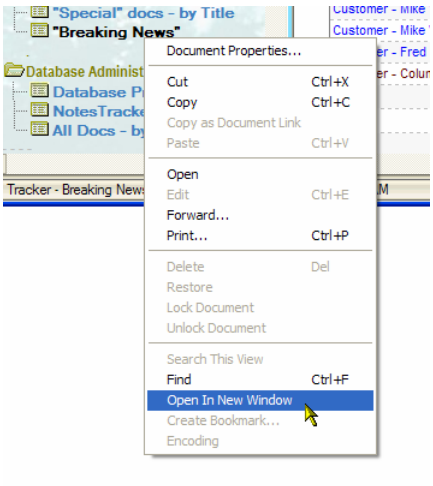


Documents like this can be viewed in isolation (from documents that are not so designated) and so highlighted and watched for in the NotesTracker Repository, via one of several new views. For example:



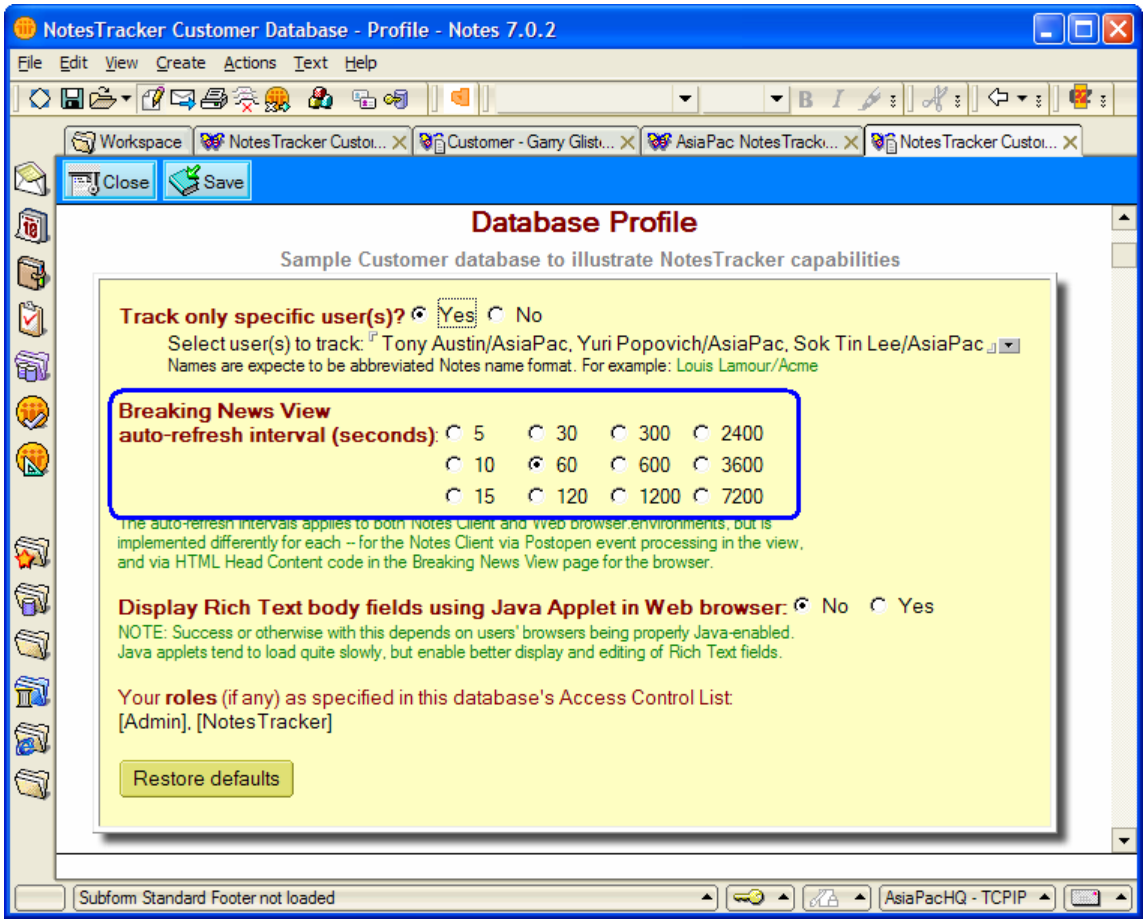
Instead of just visiting the Repository database every now and then to see what special documents have arrived, you might choose to be more proactive by making such a view into a “Breaking News” or RSS-style of style of view. Then by keeping the view open in a completely separate Notes window, you would see new “special” documents arriving in the view at every view refresh cycle. How to do this is described further on in this guide, see: Breaking News View for Inclusion in a Portal Page or RSS Feed

There are several variations of such a “**Breaking News**” view in the **NotesTracker Customer DB** database, under the “New / Changed / Deleted” navigator. You can open it in the main Notes window as shown, or right-click on the view heading and select “**Open In New Window**” so that you can continue to keep an eye on what’s changing (as the view gets periodically refreshed) and keep a better watch on things:



Note: unfortunately, the “Open In New Window” capability seems to be unavailable (is grayed out) in the Lotus Notes 8 Standard Client. (This is so for the Beta 2 release, March 2007. Let us hope that it will be reinstated for the final release.)

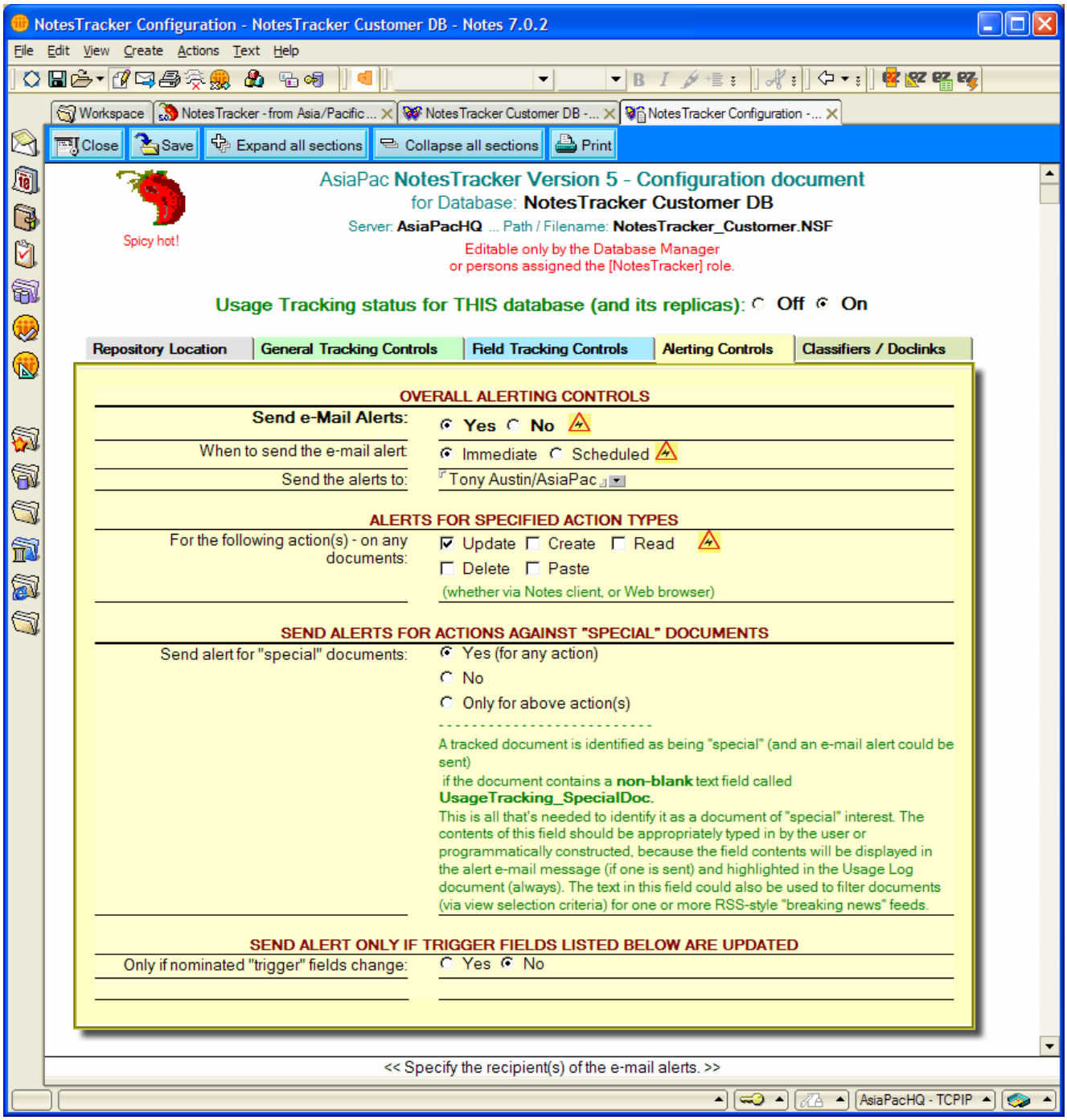
You can try changing the view’s refresh interval by editing the Database Profile document:



Note: the view refresh interval also applies to watching this Breaking News view via a Web browser.

Another way to be proactive with “special” documents is to switch on one of the options (in the database’s NotesTracker Configuration document, as discussed earlier, see **Alerting Controls**) that cause the triggering e-mail alerts.

Try it yourself in the example **NotesTracker Customer DB** database. If you wanted Lee Sok Tin and Yuri Popovich to receive e-mail alerts for any action type that occurs against any special document in the database, then you would set the Configuration Document like this:



STEP 5 – NotesTracker Configuration

STEP 5A – Add the NotesTracker configuration view

(Once per database)

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

Name/Comment	Alias
NotesTracker\NotesTracker Configuration view	vwNotesTrackerConfig

STEP 5B – Add the NotesTracker Configuration form

(Once per database)

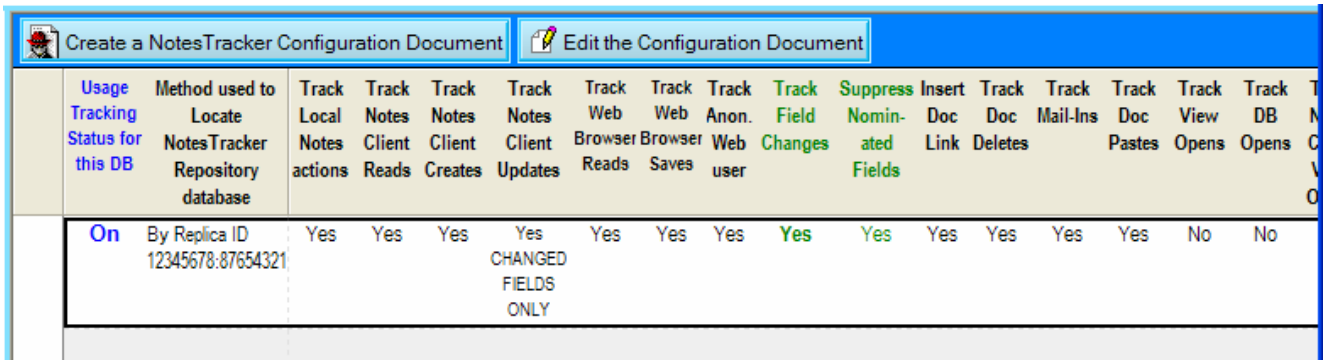
Add the **NotesTracker Configuration form** to each database being tracked.

Name/Comment	Alias
AsiaPac NotesTracker Configuration	UsageTrackerConfig

STEP 5C – Create the NotesTracker Configuration Document

(Once per database)

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



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 – Implement “Internal” Usage Tracking

(Only if desired)

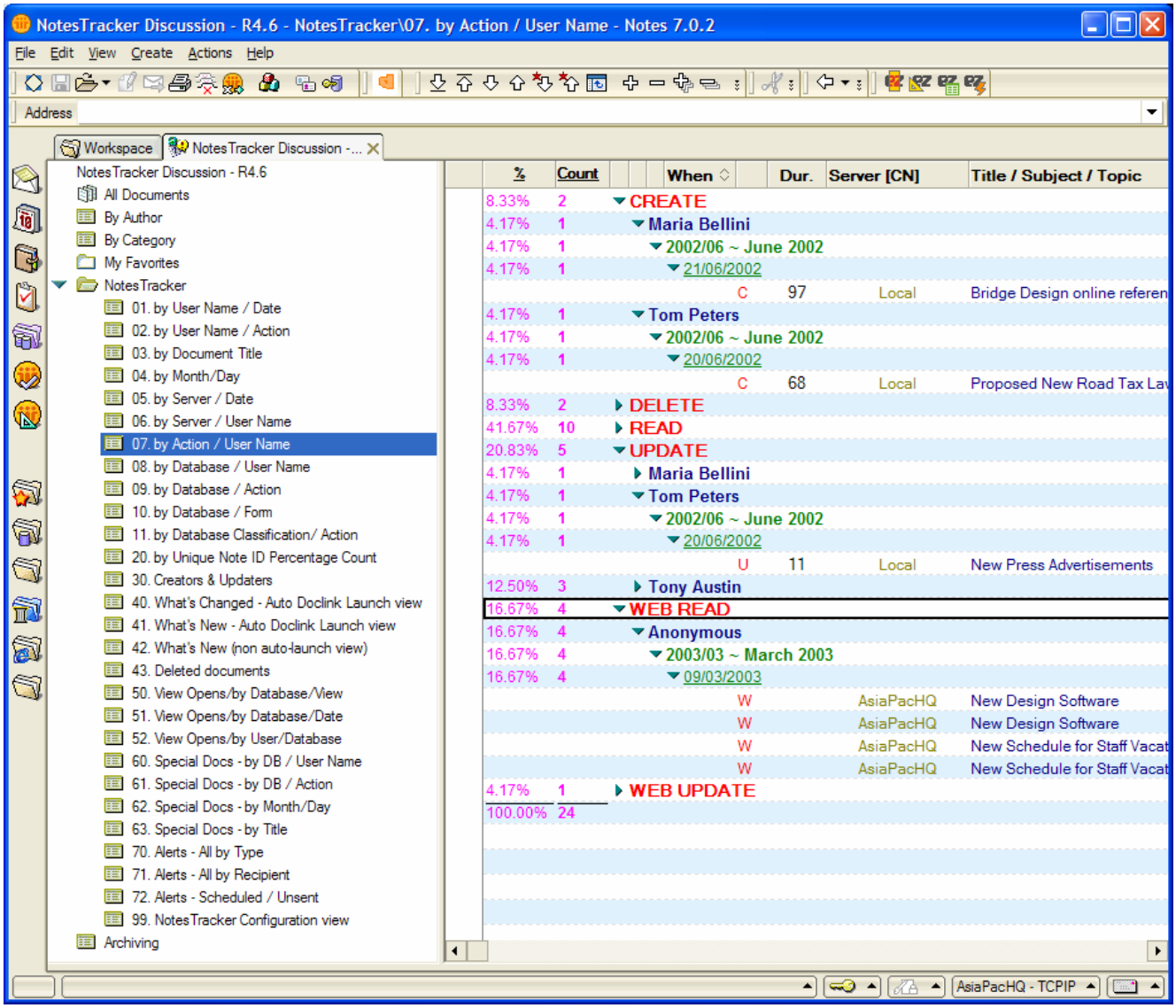
This step is only required if you decide to implement “internal” or "self-contained" or “internal” usage tracking, introduced earlier in the Overview section and the Administration Topics section.

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:

STEP 6A – Adding the NotesTracker Usage Log Views

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 analyze document usage. The Fast Design Propagation design tool (discussed later in this guide) can do this for you I a matter of seconds.)

In a “raw” form these views should look something like this:



STEP 6B – Check that All Existing Views Still Operate Correctly

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.

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 behavior and "user experience" of the database.

STEP 6C – Set Up the Archiving of Usage Log documents

The internally-stored Usage Log documents will build up over time. The NotesTracker Database contains a **Usage Log Archiving** capability that you can incorporate in your own databases.

There are some simple operational considerations for Usage Log archiving explained earlier on, in the Administrator Topics section of this guide, and you should understand these before you make design changes.

You should copy-and-paste into your database (from the NotesTracker Repository):

- The agent named "**Archive Selected Documents**"
- The"**NotesTracker Archive Profile**" form
- The "**Archive Log**" form
- The hidden **Archiving** view

Note: if you find yourself doing this repetitively and would like shave off some time, add the names of these four design elements to the "Names of Design Elements" list in the Fast Design Propagation tool.

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.

STEP 7 – How to Suppress Usage Logging for Individual Documents

(Only if desired)

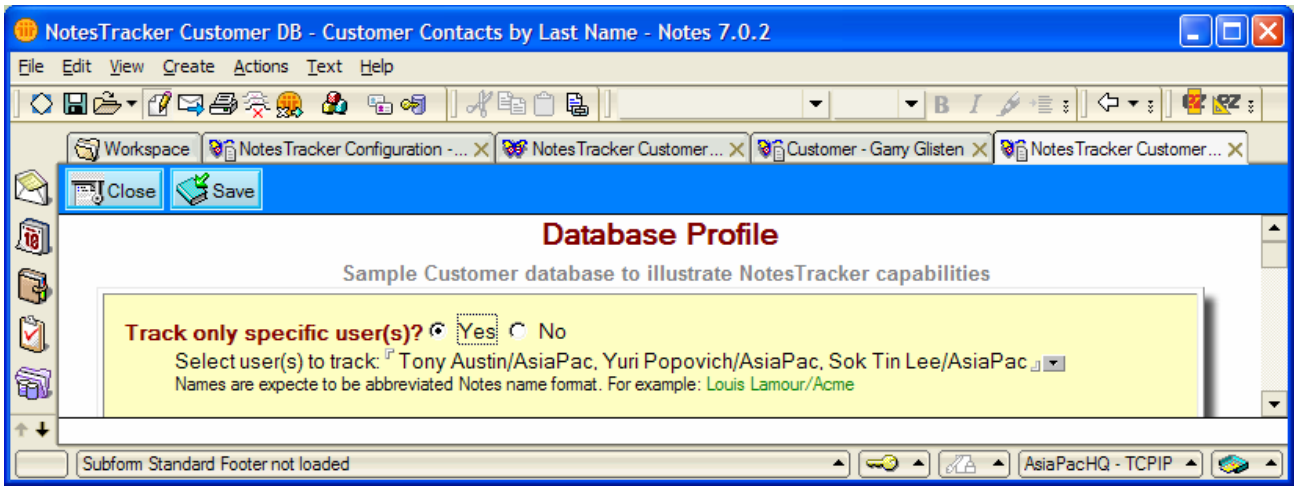
What if you want to ignore or suppress the tracking of certain documents? These might be documents that are of no interest or value for tracking purposes, that would just ad ”noise” to your usage metrics. Or they might be documents that contain sensitive information, such as personnel data, salary data,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

To do this, you merely add a text field called "**UsageTracking_TrackThisDocument**" to the pertinent form or forms in the database. Then populate the field with the string value “**No**” (or “NO”) to stop the document from being tracked. The code in the NotesTracker Queryclose event checks for the existence of this field in the document. If it finds that the field exists and contains the value "No" (or "NO"), it bypasses usage tracking for that particular document.

The example customer database (named "AsiaPac NotesTracker Customer DB") has this capability implemented to give you a feel for how it might be used in practice.

Instead of simply setting the "UsageTracking_TrackThisDocument" to “Yes” or “No”, it takes a slightly more interesting tack by using the capability to enable you to specify that usage tracking is to be carried out only for a selected list of users. (This would be a very common and useful variation on the usage tracking. Just think laterally about situations where only activities of certain users should be tracked: for training purposes, for application testing purposes, for security and compliance audits, and lots more.)

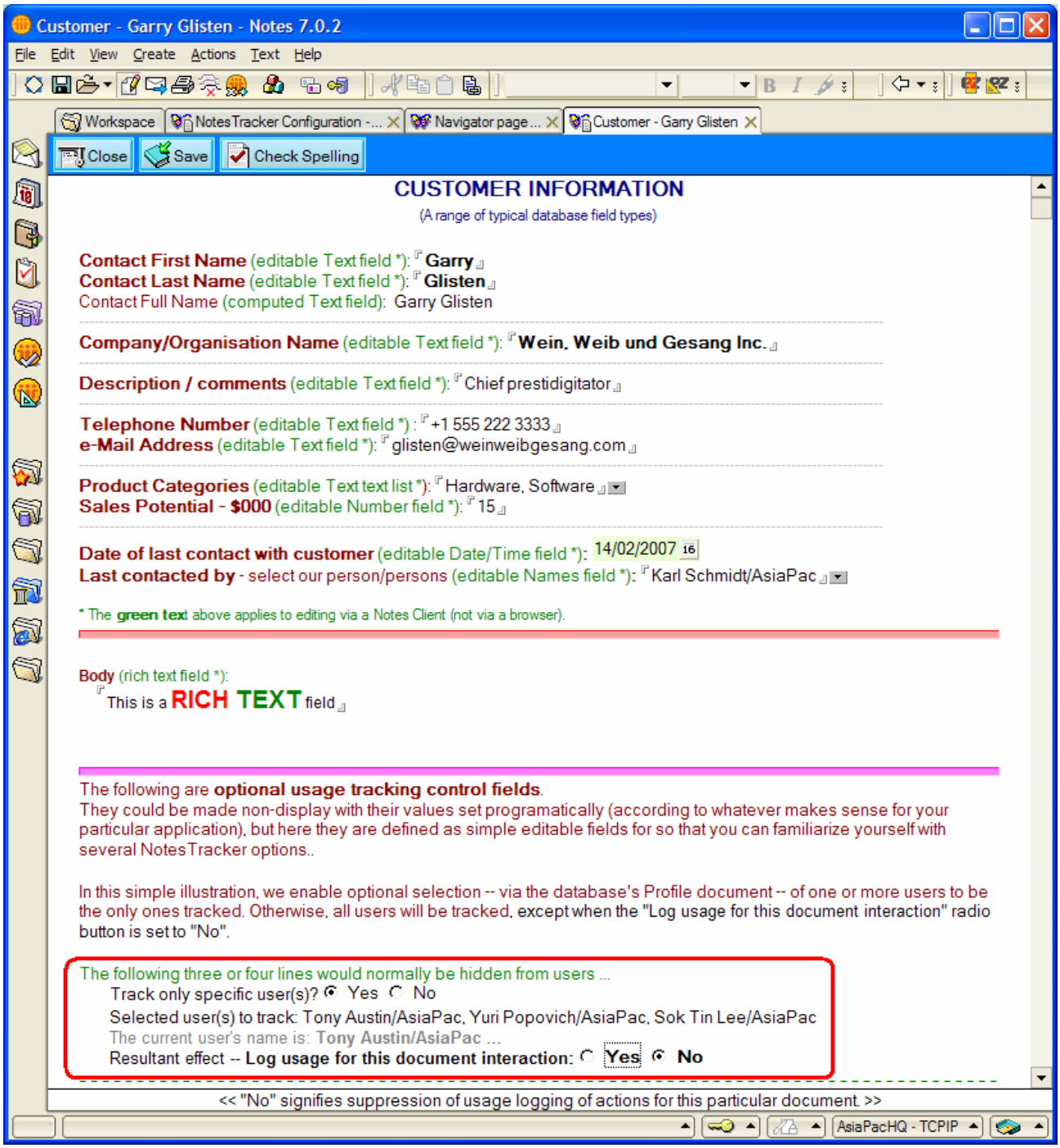
Firstly, in the Customer database’s profile there is a field (at the very top of the form) which enables you to specify whether or not you want only to track actions performed by specific users:



If you specify “**Yes**” in the radio button field, a field appears on the next line using which -- as the administrator of this application – you specify the Notes names of only those users you that you want to track. (For the purposes of this simple illustrative database, an address dialog is used for name selection, but it could be designed to be some other type of selection list.)

If you select “Yes” for this button in the Database Profile, then in the Customer form (discussed next) the specified users’ names are “locked in” and cannot be changed. So this is one method that you could use to enhance an existing database to enforce per-user usage tracking. In such a case, you would need to ensure that any user cannot edit the Database Profile and so remove his or her user name from this list.

Having specified the user names, we can turn to the Customer form, which looks like this (focus on the section circled in red):



Here we see that the names of the users specified to be tracked have been brought in from the Database Profile, and it is only these users that it is desired for NotesTracker will track for the database.

And lo and behold, the current user is indeed one of those users nominated to be tracked.

Normally the user's actions would be tracked, unless some other NotesTracker control option turns tracking off. To illustrate this, the form allows you to choose via a simple radio button field whether or not usage tracking should occur. (You almost certainly would not do it this way in a proper application, but here we allow it just to make a point.)

If the user selects "**Yes**" causes the radio button field on the next line "**Log usage for this document interaction**" -- and since this is the special "UsageTracking_TrackThisDocument" it will cause NotesTracker to log this document interaction.

However this user has behaved in a rather shady manner! He has decided that he wants to avoid having his activities tracked, therefore he has clicked on the "**No**" radio button and this stops usage tracking of this document interaction by the user.
The above is a contrived example, admittedly, but it does illustrate **one method for prevent a specific document from being tracked** (regardless of other NotesTracker settings).

It also demonstrates clearly **a simple way to designate that only certain users of the database are to be tracked**.

You will find that setting up this sort of thing is very easy to accomplish.

As long as the value of the "UsageTracking_TrackThisDocument" field contains, at Queryclose time, the value "No" or "NO" then the NotesTracker immediately exits the Queryclose event and no Usage Log document is created. This check is carried out at the very beginning of the Queryclose event, in line with other NotesTracker functions that follow the “exit as early as possible” golden rule in order to the overheads of usage tracking to the bare minimum.

Note: if the "**UsageTracking_TrackThisDocument**" field has any value other than “No” (or “NO”), or if the field is blank or null, or if the field doesn’t exist in the document, then usage tracking proceeds normally.

In practice, the "UsageTracking_TrackThisDocument" 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 algorithm.

STEP 8 – Enable Web Browser Usage Tracking

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

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

Adding the NotesTracker WebQueryOpen and WebQuerySave Agents

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 …

Switch to the Webqueryopen event, which defaults to:
@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:

Switch to the WebQuerySave event, which defaults to:
@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.)

Note: Some database forms will already contain WebQueryOpen and/or WebQuerySave agent(s). It is fairly likely that you can add the NotesTracker agents without affecting the behavior of your existing web agents. For example:

```
@Command( [ToolsRunMacro]; "(your_existing_agent)" );
@Command( [ToolsRunMacro]; "(NotesTrackerWebQueryOpen)" )
```

However, due to the inherent complexities of the way that such web agents might interact, you might have to do some extra work to ensure that the addition of the NotesTracker agents does not affect the outcome of the other agents.

Programming and Testing the NotesTracker Web Agents

Generally, NotesTracker is designed to trap any errors arising from omission, misapplication or execution-time problems related to its design elements. The design philosophy is to quietly exit at the point of error without performing usage logging and without disrupting the host application (so that users should not notice anything untoward). Therefore if usage logging does not take place as anticipated, you should review your code and carefully check one step at a time for the presence and correct installation of all of NotesTracker's 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.

TIP: Web agents can be rather tricky to get working, and debugging them can be a painful exercise, so you may get some value from the tip "**An Efficient Way to Debug Domino Web Agents**" a little further on in this guide.

What is being Measured by the NotesTracker Web Agents

The two Domino form events activate the agents (via the ToolsRunMacro command in the event) as follows:

- WebQueryOpen occurs before Lotus Domino converts the document being opened to HTML and sends it to the browser. You can change initial field values and do other pre-processing. NotesTracker records each such interaction as a "Web Read" action (Action Type of "W").

  Note: you may be a little surprised by finding that multiple Web Reads get recorded by NotesTracker for what (at first sight) seems to be only a single interaction with a form. This is of no significance whatsoever to the end user of the application, but NotesTracker is recording things more precisely.

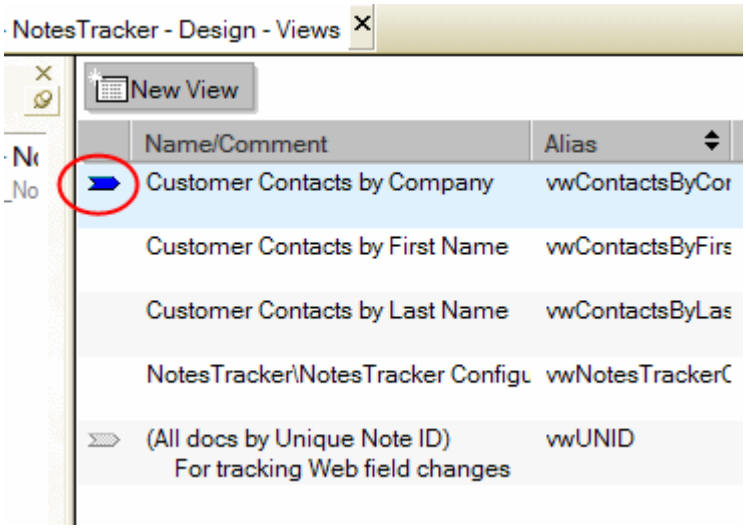
  When a form opens a Web Read is always recorded. If you have a Close button in the form this will be recorded as another Web Open event. But if you move away from the form by pressing the browser's Back button (or close the entire browser window, say, by using the "X" at the top right corner of the window) then NotesTracker will not record this as another Web Open interaction.

- WebQuerySave occurs before a Web document is saved. NotesTracker records this either as a Web Create event (Action Type "X") if it's a new document or as a Web Update event (Action Type "Y") if it's not a new document. When you click the "Submit" button -- or whatever button, such as "Save", that you have built into the form to override the default "Submit" button – there will be two events logged: a Web Read followed immediately by a Web Save. Such is the nature of transactions performed by Web servers.

STEP 9 – Ensure the Database has a Default View and Default Form

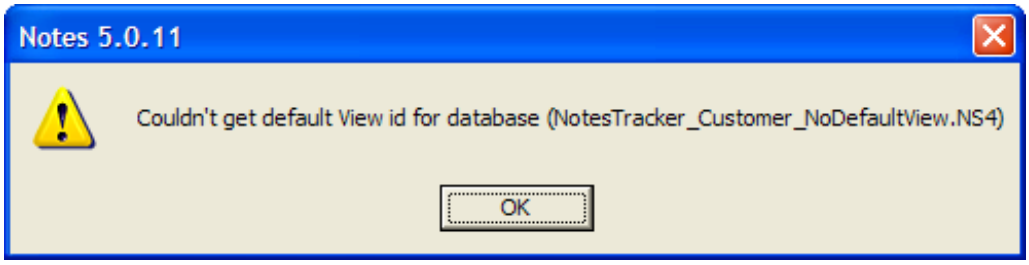
(Once per database)

Every Notes database should have a default view and a default form. A default view 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:

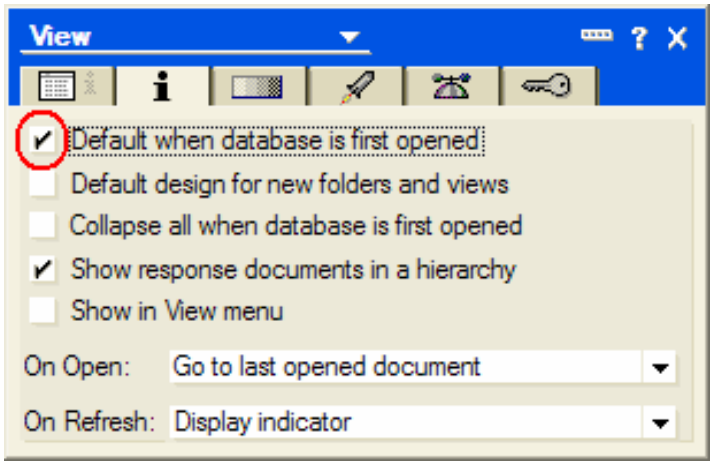


Note: the blue arrow default view indicator (circled in red above) was changed to a yellow star in Notes 7: 

It is possible for the view itself, or just its designation as being the default view, to get deleted (although thankfully this tends not to happen much). The absence of a default view may not have much impact on normal use of the database, but then NotesTracker throws a "**Couldn't get default View for database**" error dialog like the following at the time that a trackable document is being closed:



To eliminate this error, you must specify the database's default view, done simply by opening the appropriate view's design and setting the "**Default when database is first opened**" property, as follows:



Note: You should do the same sort of thing to ensure that the database's **default form** is correct (unchanged from its original form name)

STEP 10 – Set Up the Tracking of View Opens (Notes Client only)

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

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, the Usage Log document will have, instead of the view name being logged as "(View name not obtainable in Release 4)" you will see a meaningful view name: 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.

Fast Design Propagation – a New Designer Productivity Tool in NotesTracker V5.0

NotesTracker Version 5.0 introduced a significant feature that should be of special interest to Notes developers, architects, IT managers, and anybody else involved in NotesTracker planning and deployment.

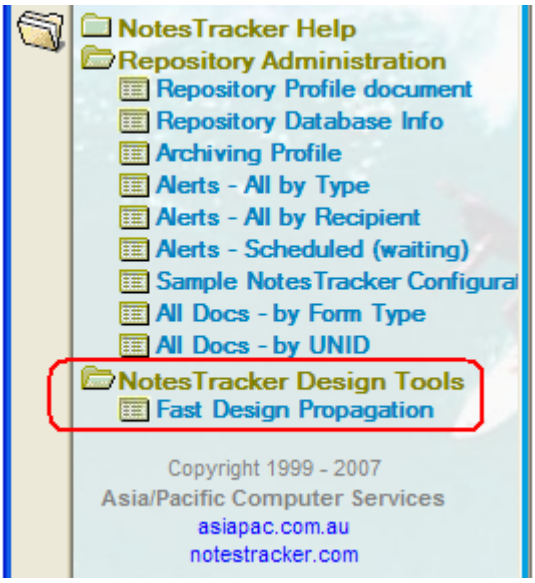
It is called "**Fast Design Propagation**" and is the copying of selected NotesTracker design elements to the target database in a matter of seconds (compared with minutes, or even tens of minutes, prior to Version 5).

It can perform most, but not all, of the repetitive or "rote" operations described in the ten steps just above. It does *not* have the capability of performing other NotesTracker programming procedures (coding, forms design, etc) such as setting up "Usage Log title" or "special document" fields described further on in the guide.

All the same, many of your Notes databases will require very little more than what fast design propagation provides. By affording a significant reduction in developer time **it now should be quite economical and feasible for you to build NotesTracker into quite a few more of your Notes/Domino applications**

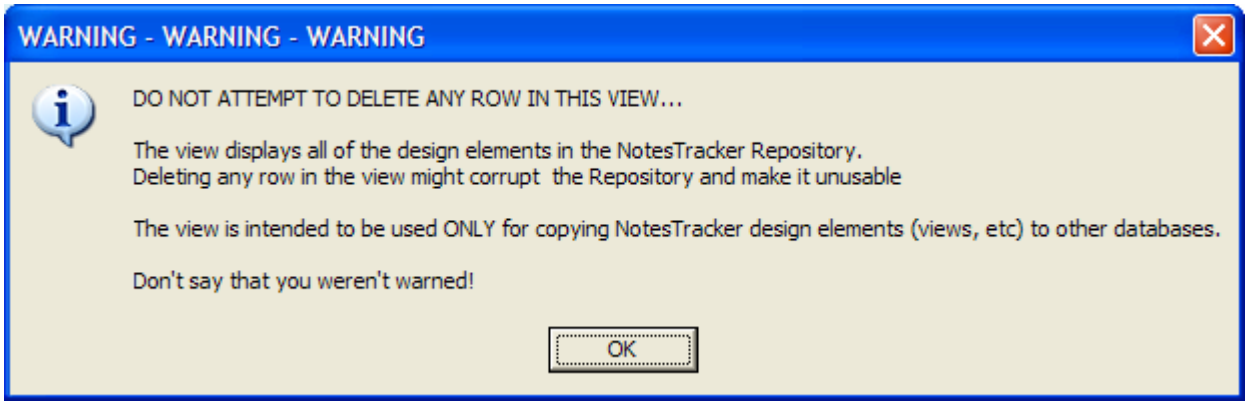
Fast Design Propagation – Steps

A new **NotesTracker Design Tools** section appears at the bottom of the navigator column, and when it is expanded you will see the new "Fast Design Propagation" menu item, like this:

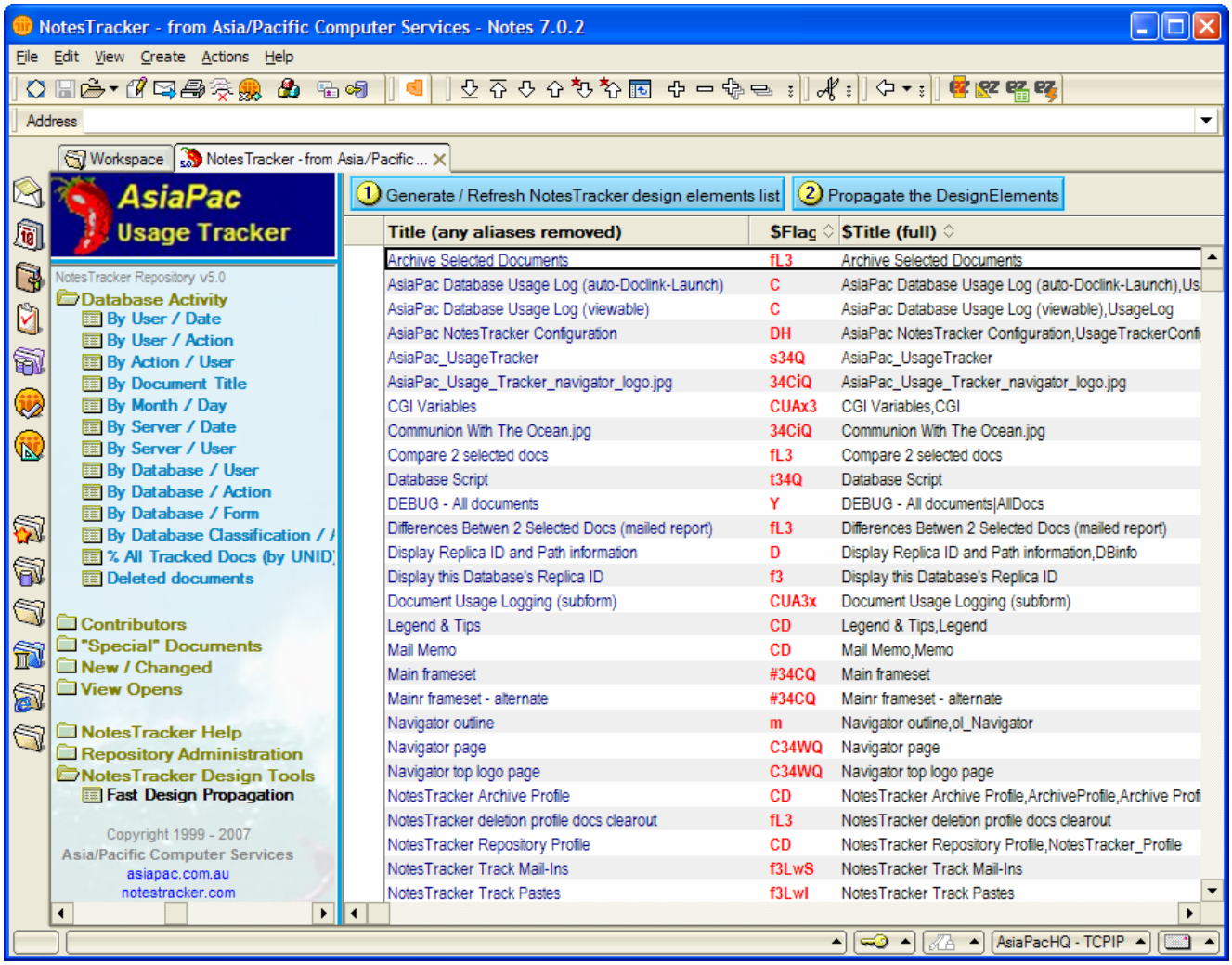


You should ensure that it is visible only to those having the new **[Designer] role** since it could be considered "dangerous" in the hands of those who do not understand Notes development in general and what it adds to target databases in particular.

When (as a database Designer) you click on the "Fast Design Propagation" menu entry, you are presented with a warning dialog:



This warning applies to the view of all design elements in the NotesTracker Repository, which gets displayed next:



The view (which will be empty the very first time) is populated by clicking on the "**Generate / Refresh NotesTracker design elements list**" button:

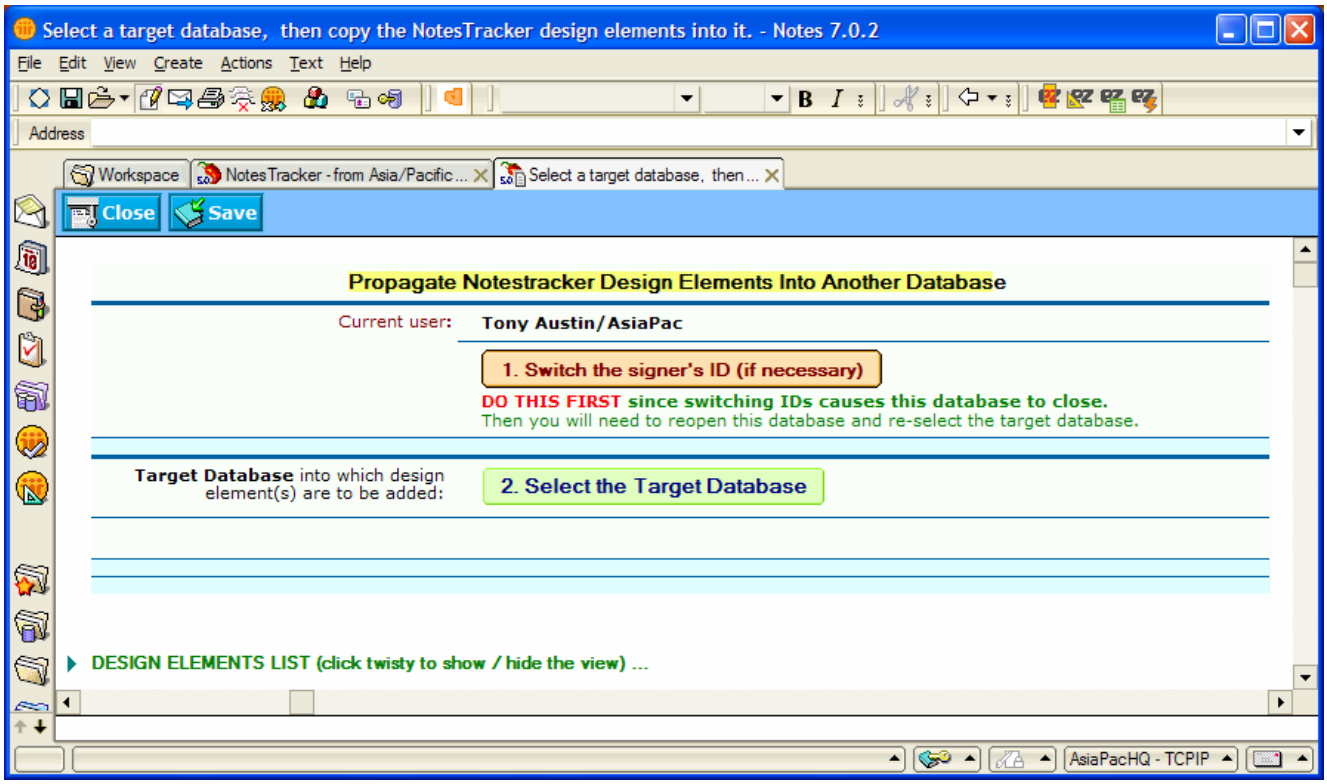
You certainly do not have to understand any of the details shown in this view, such as what the **$Flags** value signifies for each design element. (You might work out what some of the flags mean, but will probably be stumped by others!)

Only one thing is important for NotesTracker's purposes: the exact spelling of the names of the assorted design elements that it uses. (These are the design elements discussed in the numbered steps of the previous section.)

All of the Repository database's design elements are listed in **the first column of the view** with any alias names stripped off. This is the required format for the "Fast Design Propagation" feature. The third column is included in the view only as a matter of interest, and it includes the alias name(s) for each design element.

Once the view is generated (or refreshed), you click on the "**Propagate the Design Elements**" button:

The first time that you do this for, you will see the following:



The form has been designed to be self-explanatory and as easy to use as possible.

If you do not have Designer rights to the target database, you use the top button ("1. **Switch the signer's ID**") to switch to the appropriate designer's user ID. (Lotus Notes will close the database, and you'll need to make your way back to the "Propagate the Design Elements" button again.)

Next you click the button labeled "**2. Select the Target Database**" which presents you with the conventional Lotus Notes "Choose Database" dialog for selecting the database into which you want to copy the NotesTracker design elements.

Once you have selected the target database, the form is automatically refreshed and displays the next step in the procedure, where you specify the design elements that are to be propagated:

- Whether or not any design elements are to be copied: "**Copy**" or "**Do Not Copy**"
  and, if so, you can edit a list of the individual design elements' names.

- Whether or not the NotesTracker Configuration Document is to be copied: "**Copy**" or "**Do Not Copy**"

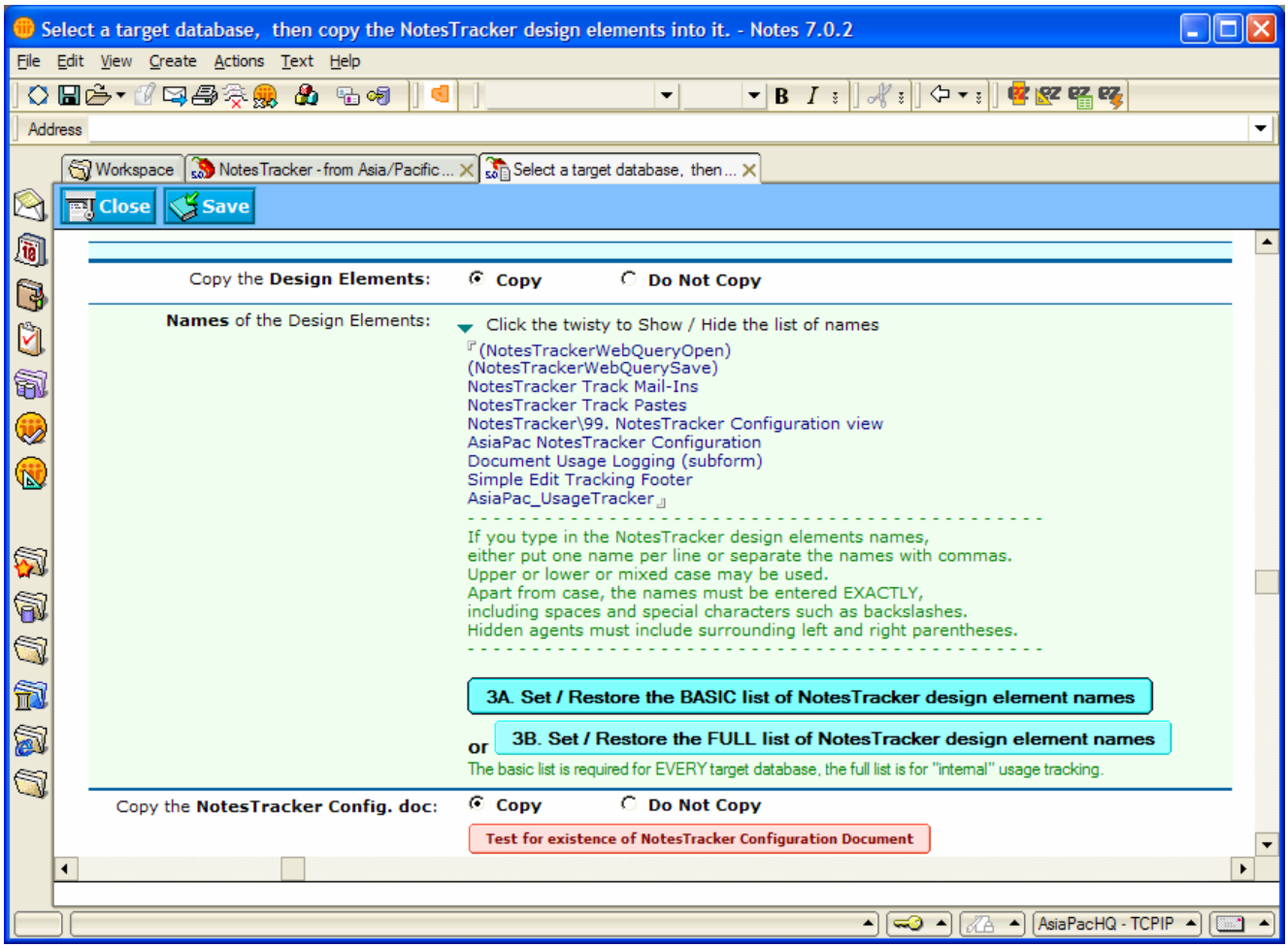
  If you select "Copy" then the NotesTracker configuration document stored in the current NotesTracker Repository database will be copied to the target database.

  It is not necessary to make this copy of the configuration document already exists in the target database (for example, during a NotesTracker version upgrade). In fact, this is why the "Do Not Copy" option exists.

  Note: unexpected behavior might ensue when the target database is next opened -- depending on the value of some of the options in the Configuration Document (such as the tracking of database opens or view opens) – until you have the opportunity to edit the document and change whichever of the options are causing that behavior.

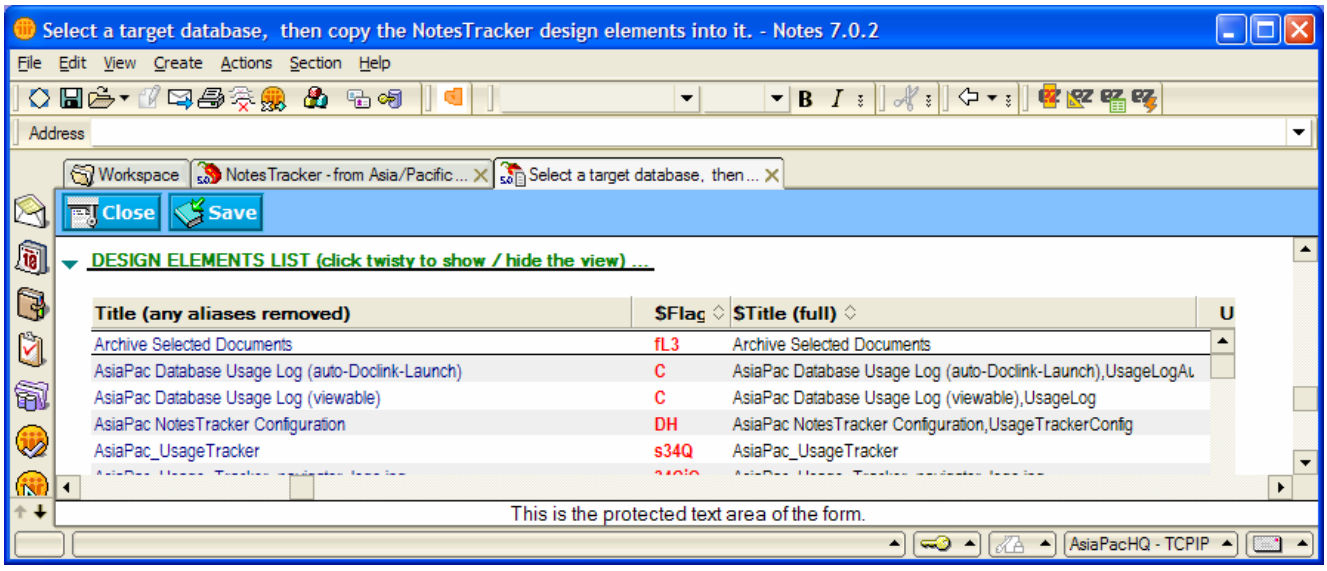
  TIP: to avoid any such unexpected behavior, it is probably wise to ensure that usage tracking is turned off in the Repository's configuration document *before* you propagate it to other databases.

Button 3A generates a list of the basic NotesTracker design elements, those that you would probably put into every target database, as follows:

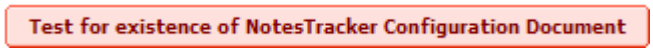


Button 3B generates a list that also includes the views (and other design elements) that typically would be required for "internal" usage tracking in the target database. You might decide reduce the number of additional views if you think that they're not likely to be required.

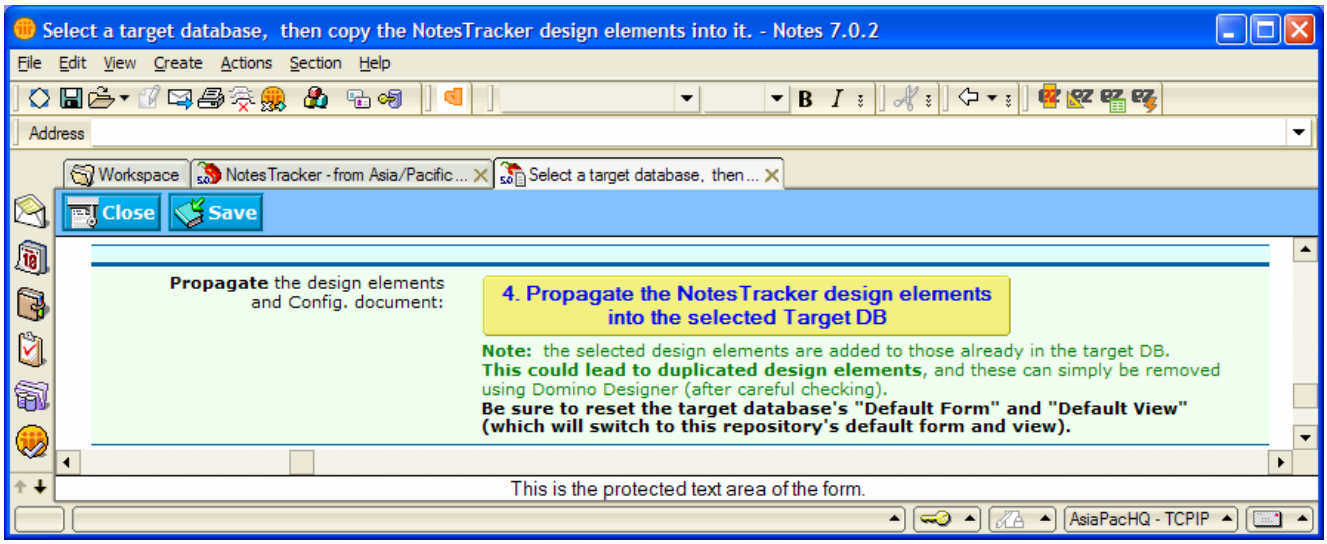
At the bottom of the form is a collapsible section that contains the design elements list view, embedded here purely for your convenience in checking the exact spelling of design element names:



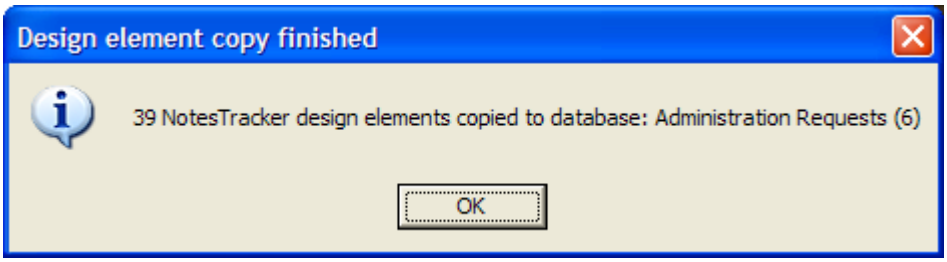
If you decide to propagate the NotesTracker Configuration document, you should use the button labeled "**Test for existence of NotesTracker Configuration Document**" to verify that the current Repository database does contain this document.



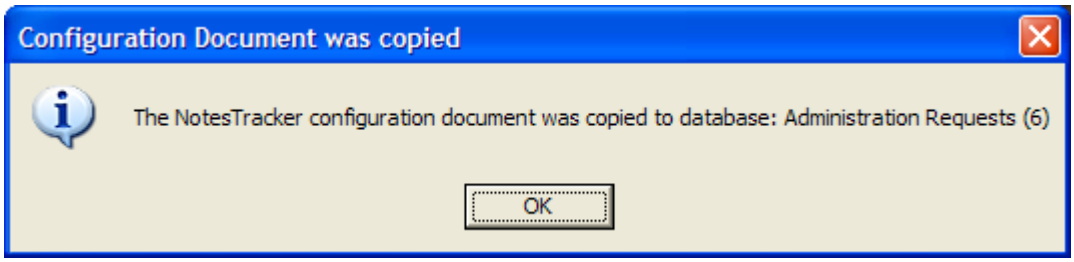
Once you have done all of the above, click the large yellow button at the bottom labeled "**Propagate the NotesTracker design elements into the selected Target DB**":



Assuming that you have used Button 3B earlier, typically you should see the following dialog box:

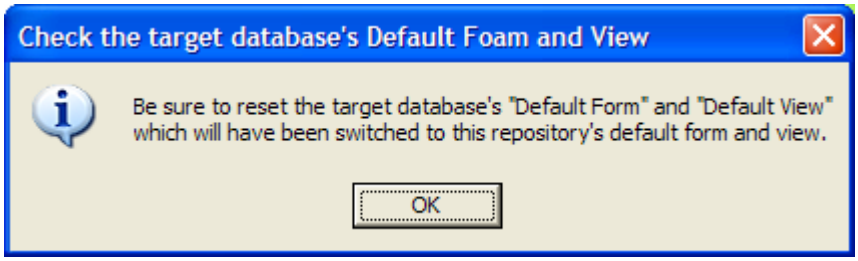


Followed immediately by:



Fast Design Propagation – Follow-up steps

Finally there is the following **important reminder** (necessary, because the Copy process copies the "default view" and "default form" flags into the target database, thus overwriting these two settings in the target)



Note: the design elements and NotesTracker Configuration document will be copied to (appended to) the target database each time that you click the yellow button. Be sure to check for duplicates – NotesTracker forms, subforms, views, agents, script libraries and configuration documents -- and delete any that are superfluous. (There may be unintended side effects if duplicates are present in the target database.)

It is good practice to **make a backup copy of the target database's design template beforehand**, so that you can revert to it if desired (for example, if you decide this to be easier than removing the duplicate design elements).

NotesTracker Version Upgrade Considerations

If you already have a database that incorporates NotesTracker at a previous level (say, Version 3 or Version 4), what has to be changed when upgrading the database's design to incorporate NotesTracker Version 5.0?

Assuming that you have not tailored any of the NotesTracker design elements, the design steps summary table on page 93 gives a hint, and this section goes a little deeper into this matter in the same step-by-step sequence.

1. The Usage Logging subform has been enhanced significantly for V5, and you must replace the old subform. Of course, you do not need to re-insert the subform into any forms (since it is automatically inserted by Notes). The other two subforms mentioned have not changed significantly.

2. The AsiaPac_UsageTracker script library has changed substantially, and you must replace it.

3. The subroutines in the database script must be replaced.

4. The reasons for using a "UsageTracking_Title" special field have not changed, so you do not need to worry about this. The "UsageTracking_SpecialDoc" special field is new for V5, so there are no upgrades considerations for this.

5. The NotesTracker Configuration form and view have been significantly enhanced, so you must replace them.

   After this, you should go into edit mode and then save the NotesTracker Configuration document in order for all the new V5 option fields to be allocated their default values. Better though will be to work through the new options and features and change some of them to take advantages of the new V5 features (e-mail alerting, database classification, etc).

6. The NotesTracker usage log views have been enhanced, so you should replace them. (This can be done by deleting all of the views and using Fast Design Propagation to quickly insert the new view designs.)

7. Suppressing usage tracking at the individual document level has not changed in V5.

8. The NotesTrackerWebQueryOpen and NotesTrackerWebQuerySave agents have been enhanced and must be replaced.

9. It is still important to check that the database's default form and default view have not been changed, particularly if you use the Fast Design Propagation tool (which tends to change the defaults).

10. If you are tracking view opens – a specialist use of NotesTracker – then you need to carry out this step since the view open tracking design has been enhanced in V5.

Our testing has shown that upgrading NotesTracker to V5 level is not an onerous job. It just requires the usual degree of care and regression testing.

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. It is important always to "cut" so that there's only ever a single instance of the debug statement in your code, which makes it far easier to ensure your code is free of debug statements when you finish.

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 better still just: **z% = 1 / (z% - z%)**

We generally use the following form:

z% = 1/(z% - z%) ' ############# DEBUG – 23 MARCH 2007 ##########################

The string of hash symbols (#) makes the statement stand out. This makes it much easier to locate, just by scanning (or performing a search for) a string of hashes. It is less likely that you will overlook it and leave it in your agent when you have finished your debugging session. It is highly recommended that you only have a single copy of this debug statement in your code at all times. Otherwise, you may not be sure which of several statements is causing the zero-divide error. It quickly becomes a habit to use a single debug statement, and to cut-and-paste it elsewhere in the code stream as your detective work proceeds.

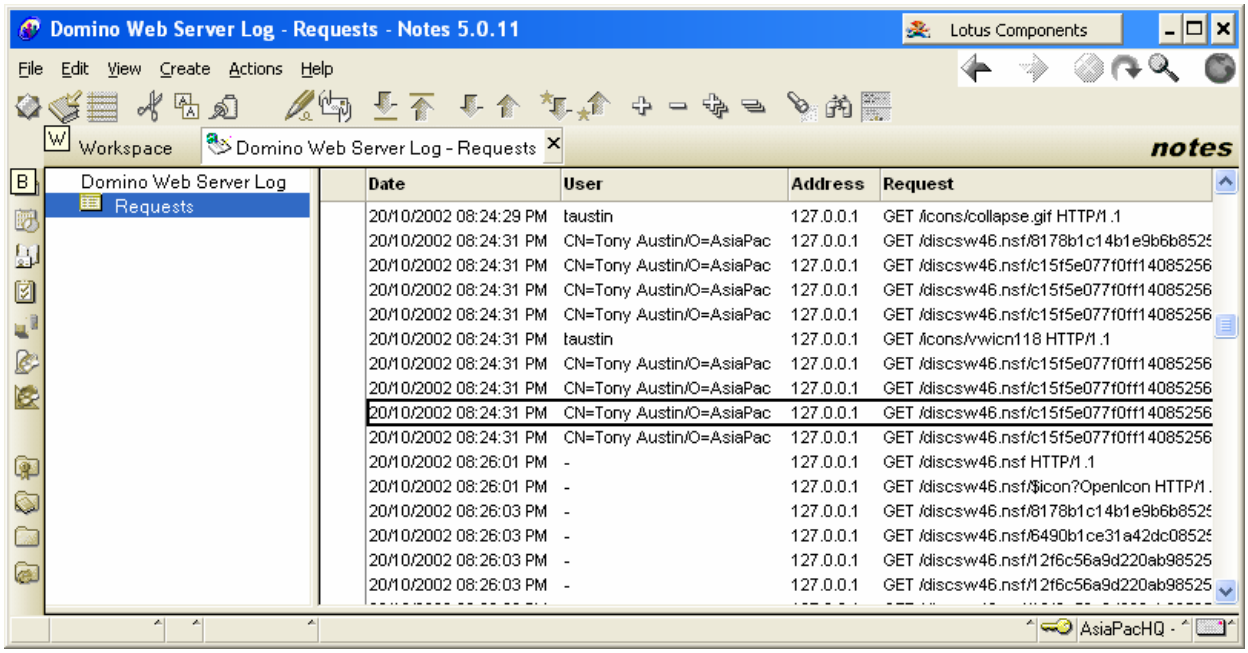
For maximum efficiency and convenience, we run a Domino test server on the same workstation that we are using for the Domino Designer. Then we watch for the Zerodivide error messages to appear on the Domino Console as they occur, side-by-side with the browser window. This gives feedback an instant after we trigger the error-causing action in the web browser window. (A remote Domino Console on your workstation is the next best thing, or a Domino server system close by.) An

arrangement like this can substantially improve your debugging turnaround time, compared with trying to debug events occurring on a distant Domino server.

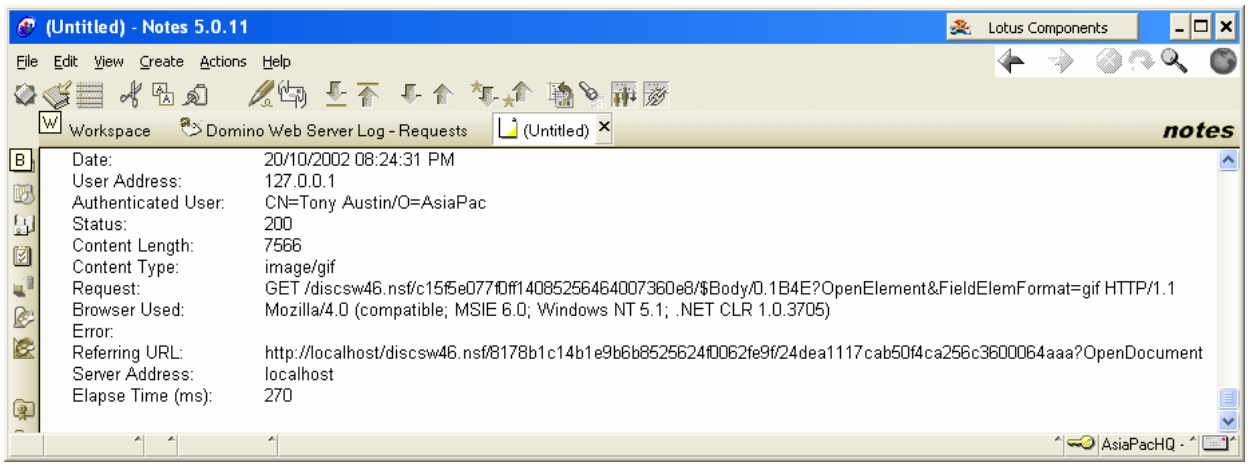
Why Web Browser Tracking was added to NotesTracker

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 analyzers).

Commencing with Version 4, NotesTracker in contrast:

- 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 analyze 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 & Coding Conventions Used by NotesTracker

NotesTracker code plus other design elements (such as the main NotesTracker subform and script library) were designed not only to be as easy as possible to incorporate into your database applications but also to cause minimal, if any, disruption to or interference with the existing code in these applications.

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

For instance, as you're modifying or debugging your code you will come across:

- 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 "**UsageTracke**r" or "**Usage Tracking**" in these names should make it easy for you to discriminate between NotesTracker code and all the other code in your databases. As well as this, the convention is followed that:

- Database fields (permanently stored fields) have names that start with an uppercase letter.

- Work fields (temporary variables that are never stored) 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).

NotesTracker uses design element names in general -- forms, views, agents, etc, and not just fields --were chosen that were felt to be as meaningful as possible rather than being cryptic. Line spacing and comments were used wherever it was thought they would help you to understand and navigate the code more easily. Coding "trickery" was avoided in the interests of clarity and ease of maintenance.

You will find that the color purple has been used throughout for such things like hidden fields and developer comments. This color was chosen (rather than red, say) because it is rarely used for anything else and it stands out quite well. And because it is not seen by end users it will not cause any negative "poor color choice" comments.

Tracking of Document Deletions (and Deletion Attempts)

The Importance of Deletion Tracking

In some of your database applications, it is vital to know as much as you can about document deletions (when, by whom, etc). On the other hand, in other databases the deletion tracking will be of little or no interest.

Deletion of documents usually occurs during normal database operations. It might be a travel request database where you want to know the culprit who deleted your airline booking request, or a CRM application in which all the documents for your key client or prospect suddenly disappear. It might even be your Notes Mail database, when a person who is delegated to use it inadvertently or otherwise happens to delete an important mail memo.

Then there are cases where document deletions occur for less innocent reasons, even criminal activities. It can be important to find out that **deletion attempts** are being carried out, even if the attempts fail because the perpetrator lacks deletions rights in the database concerned. In applications like these, you definitely need the forensic capability that NotesTracker's tracking of deletions offers!

NotesTracker Version 5.0 introduced the ability to specify that a document deletion will trigger the sending of an e-mail alert, so that somebody is quickly made aware of the deletion and if necessary can react to it promptly.

Design Issues with the Postdocumentdelete Event

Going by its name, the **PostDocumentDelete** subroutine (part of the Database Script library) should allow you to track the actual deletion of documents.

However, experience gained during the development and testing of NotesTracker's deletion tracking code has shown that this is not always what it does. Actually, we have found that what this event enables you to track should be termed more accurately the **attempted deletion** of documents.

The Domino Designer Help for PostDocumentDelete states, somewhat obscurely, that the event "occurs just after a document is deleted (cleared or cut)." Then a little later it 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."

So, if the user is not authorized to delete documents the deletion request will fail. In effect, this means that the triggering of the PostDocumentDelete event belies its name and is not reliably ***post*** (after) the actual removal of the document from the database.

Therefore you should regard a triggering of this event as measuring a ***request*** for document deletion – so perhaps it should really be called the "**PostDocumentDeleteRequest**" subroutine!

To our knowledge there is no capability to properly handle this indeterminate (you might even say misleading) situation, at least at the LotusScript level of coding (and at the time of writing, up to and including Release 7.0.2).

How Deletion Tracking was improved in NotesTracker Version 4.4

In a "best efforts" attempt to provide more accurate/meaningful results, for NotesTracker Version 4.4 we modified the design of the earlier deletion tracking code. We did so by adding a database Queryopen routine that determines whether or not the user is authorized to delete documents in the database. It writes this authorization information into a Profile Document specific to the current user, so that deletion rights of all users get separately recorded.

Then in the PostDocumentDelete subroutine – even though, 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"). However, we found that 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" (and the letter E was chosen because it lies in the alphabet between D and F).

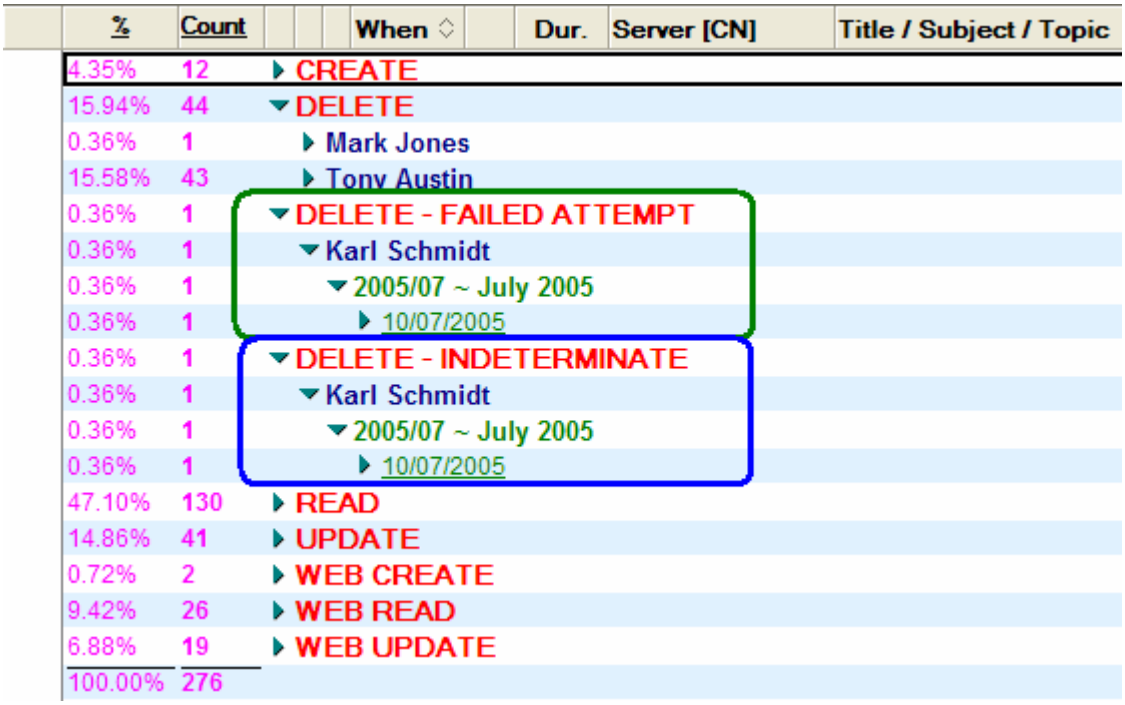
Finally, in the database's Queryclose event we do some housekeeping, trying to delete the no-longer-needed Profile Document. If a user's authorization changes, then next time a new Profile Document (or the existing one, if the attempt to remove it failed) will be created with the updated authorization information.

To examine the code for the above, open the "AsiaPac_UsageTracker" script library, and examine the following two subroutines:

1. UsageTracker_SaveDocDeletionRights
2. UsageTracker_PostDocDelete

These are called (in that sequence) from the Postdocumentdelete event in the database script.

From NotesTracker Version 4.4 onwards, this results in a different appearance in the Usage Log, with the new "F" and "E" entries that look like this:



%	Count	When	Dur.	Server [CN]	Title / Subject / Topic
4.35%	12	CREATE			
15.94%	44	DELETE			
0.36%	1	Mark Jones			
15.58%	43	Tony Austin			
0.36%	1	DELETE - FAILED ATTEMPT			
0.36%	1	Karl Schmidt			
0.36%	1	2005/07 ~ July 2005			
0.36%	1	10/07/2005			
0.36%	1	DELETE - INDETERMINATE			
0.36%	1	Karl Schmidt			
0.36%	1	2005/07 ~ July 2005			
0.36%	1	10/07/2005			
47.10%	130	READ			
14.86%	41	UPDATE			
0.72%	2	WEB CREATE			
9.42%	26	WEB READ			
6.88%	19	WEB UPDATE			
100.00%	276				

Action type "**F**" is translated as "**DELETE – FAILED ATTEMPT**" (shown in the green rectangle in the illustration).

Action type "**E**" is translated as "**DELETE – INDETERMINATE**" (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).

How to Set a Usage Tracking Title for Document Deletions

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.

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

When a user has a view open and deletes a document (or multiple documents at a time), naturally enough there is no a "front end" document open while each document deletion operation is being carried out. This in turn means that there is no way for NotesTracker's Querydocumentdelete subroutine to set a meaningful "title" for each document via the technique of having a **Computed for Display definition for the UsageTracking_Title field** as is done for other types of actions (Read, Update, etc). Rather than recording nothing (a null value) for the title, NotesTracker uses the value stored in the document's **Form** field.

Even when the Form field is used, it can be nigh impossible later to understand what was the nature and content held by any given deleted document. As mentioned elsewhere, there can be no Doclink for a deleted document. You just don't know what the document was from the content perspective (only who deleted it, the database name and Replica ID, the document's UNID value, the date and time of deletion, etc).

However, in many cases it will be important to you that a meaningful title gets recorded for each deleted document. If so, you must store the "UsageTracking_Title" field as **a permanent field in the document** -- in advance, obviously (before the document is deleted). It is only such pre-existing permanent fields that the Querydocumentdelete subroutine can retrieve for storage in the title field of Usage Log documents.

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).

"Breakings News" – NotesTracker's Generic Way to Populate Newsfeed Views for Intranet and Web

We think that NotesTracker offers a **unique design capability**, enabling you in an extremely painless, quick-and-easy fashion to build and populate "What's New" or "Breaking News" views. These views can then be used in all sorts of interesting ways.

Three example 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 example 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.

Contact Asia/Pacific Computer Services 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

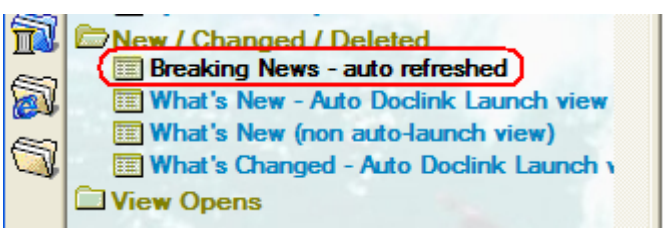
Sample views 40 and 41 are "auto Doclink launch" views. They were designed with additional code that is tailored for placement in an embedded view in a frame on your portal page. When the user double clicks to open a document within the "What's New" or "What's Changed" frame, instead of opening the NotesTracker 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)

Tip: If there is not enough information in a given Usage Log document in a view such as the above, with a little programming you can use the Replica ID and document UNID to open the original document to retrieve its base data (rather than just the data in the Usage Log document). Starting with NotesTracker Version 5.0 another approach would be to set up the "**UsageTracking_SpecialDoc**" field (described earlier in this guide) with a text string that can be parsed (say, using the @Explode command) into separate subfields that can as column values for the Breaking News view.

Breaking News View for Inclusion in a Portal Page or RSS Feed

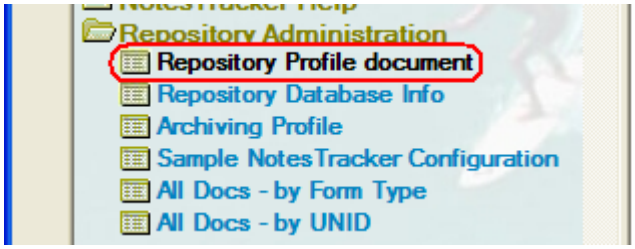
Starting with NotesTracker Version 5.0, there is a navigator entry labeled "**Breaking News – auto refresh**":

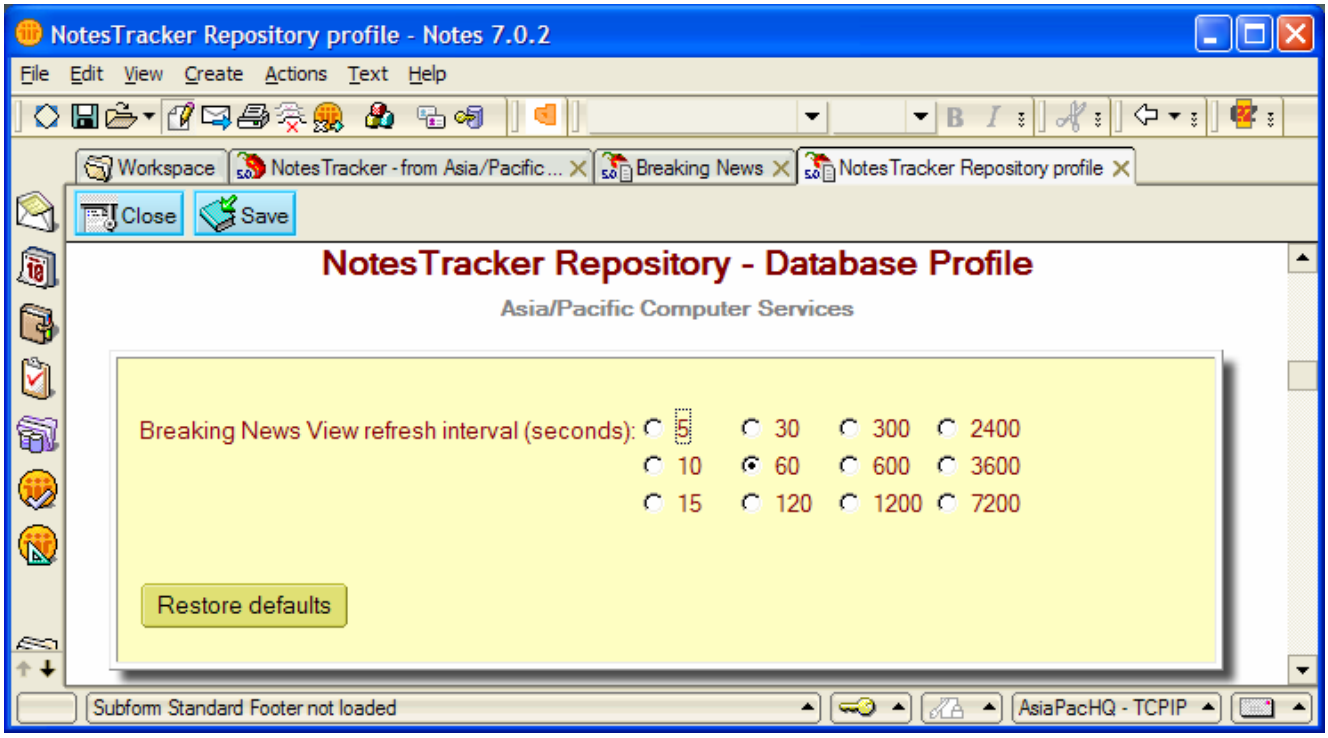


As implemented (and remember this is only an example), when this item is clicked it causes a document to be opened which holds an embedded "Breaking News" view, named "**(Breaking News Items)**".

Note: for the purposes of this example, the document is placed in Create mode and with a 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, and so is never meant to be saved).

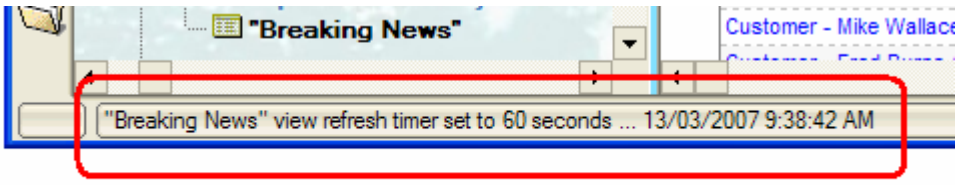
When the document is opened, a value is retrieved from the NotesTracker Repository's database profile for the refresh interval (in seconds):



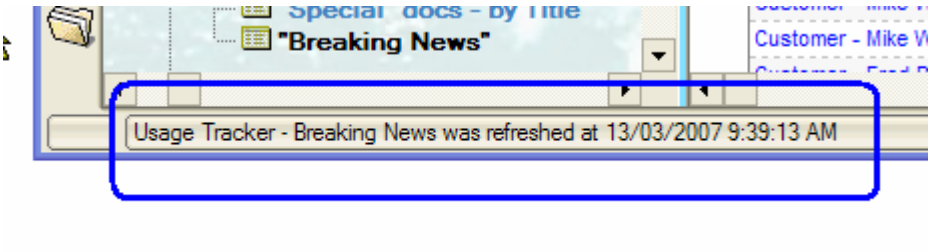


You can experiment with changing the refresh interval with values ranging anything from 5 seconds to 2 hours. (The selected value is displayed at the bottom of the view, as shown in the next illustration.)

The retrieved value of the refresh interval is displayed in the Notes status line when the Breaking News view is opened:



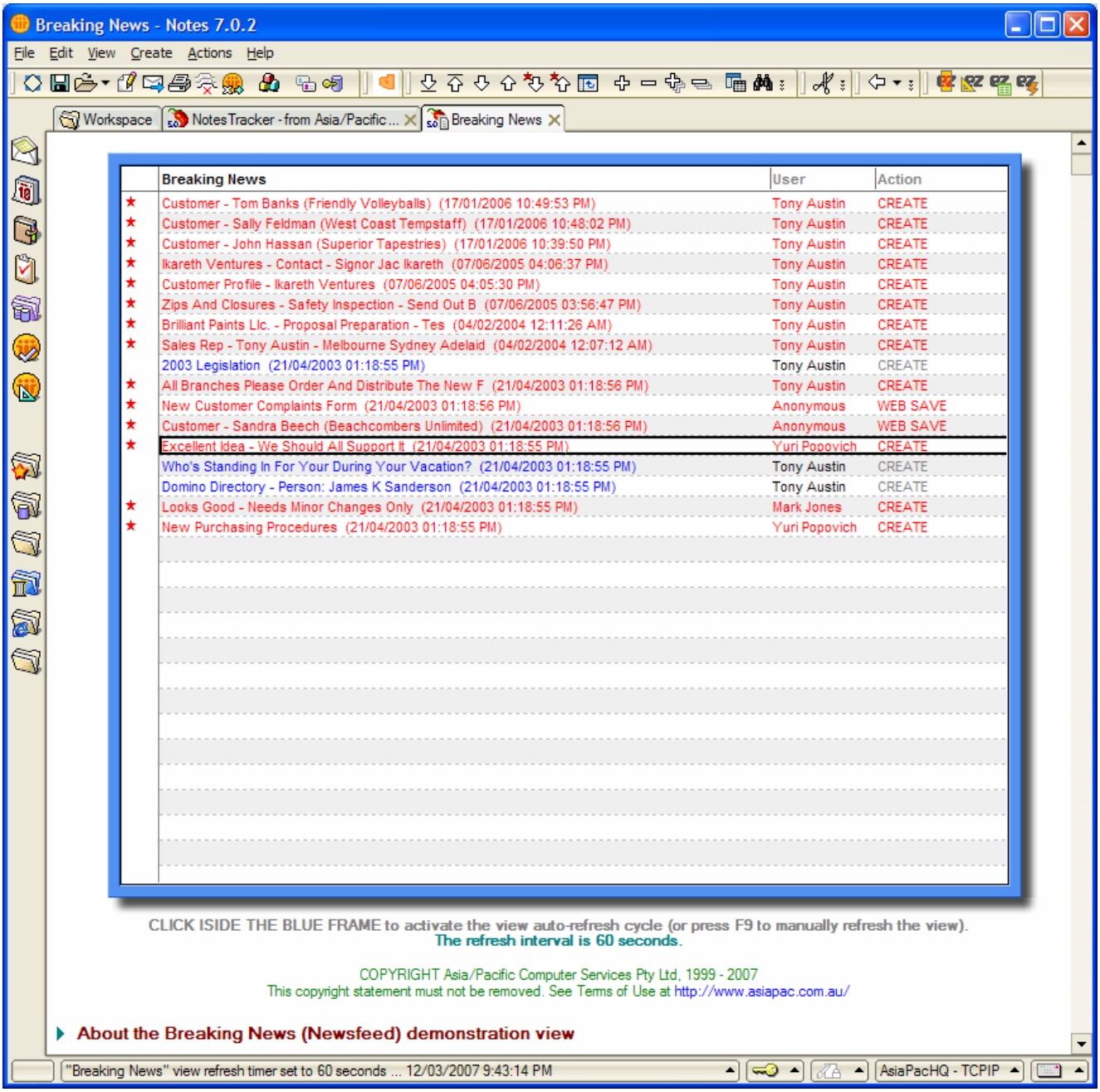
Similarly, every view refresh operation is displayed in the status line:



Breaking News example view in the NotesTracker Repository database

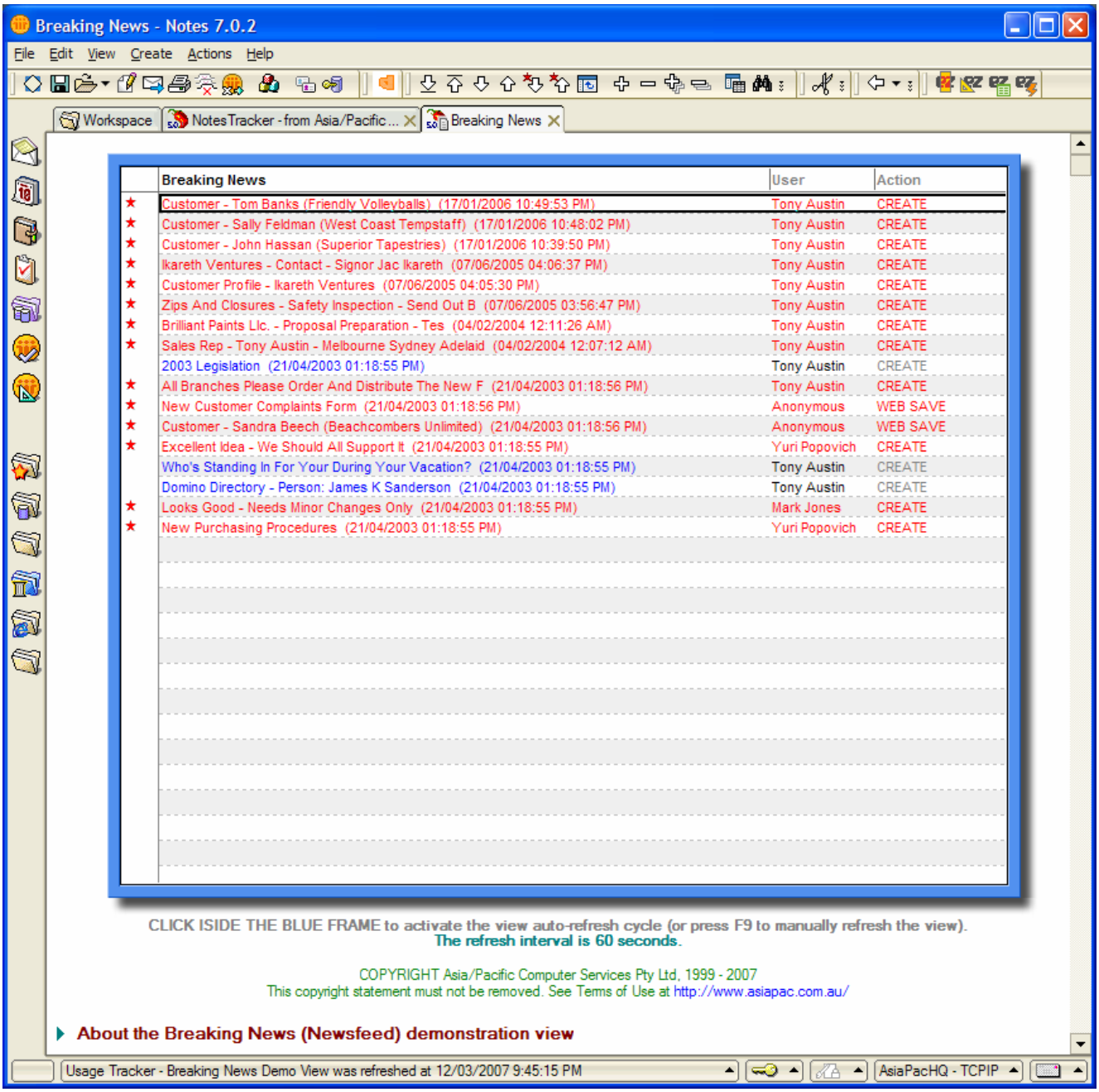
This is a regular Notes view, but with two “twists” to the way that things happen. Firstly, it gets automatically refreshed; and secondly, rather than opening the Usage Log documents that are selected to appear in the view, it is not these that you open by double-clicking on a row in the view.

For example, to see the latest news on "Excellent Idea – We Should All Support It", you move the view selection pointer (the dark rectangle) to that particular row in the view:



Then you simply double-click on it to open the underlying database document, that is, the original document that was being tracked.

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 color indicating that this document has not yet been opened in the Breaking News view):



This demonstration embedded Breaking News view has only two columns:

- A hidden column, SORTED in descending date/time sequence (the date/time that each document Create was logged)
- A text column containing the "Title" field from the Usage Log document, and with the column heading "Breaking News".

For your own purposes, you can easily set up all sorts of views to show whatever is relevant in a given situation. Contact Asia/Pacific Computer Services if you want some more ideas about this.

This is very much a native Lotus Notes Client implementation of what in the Internet is commonly referred to as a "newsfeed" (or "news feed") -- of which there are two types, RSS feeds and Atom feeds. However, it's worth noting that this was first implemented in the precursor to NotesTracker in the late 1990s, before such newsfeeds were as well known and popular as they are now in the mid-2000s.

There are two aspects of this view's design that are uncommon. You should examine the view's Postopen event to see how the timed refreshing of the view operates. Then refer to the view's Form Formula to see that the view causes use of a special form against the underlying documents in the view. The form's alias name is "**UsageLogAutoLaunch**" and it has an aspect that is a little uncommon, namely, the Launch Property "**Auto Launch: First Document Link**". And what is the document link in the underlying Usage Log document? What else but a link to the original document that was being tracked, at some time, in some database, on some Domino server (or "perhaps "local"). If a given document is still in existence and reachable from your local workstation, then if you double-click on a row in the embedded view this should open the original

document via the DocLink in the Usage Log entry. (If the document is not on the local workstation or a nearby Domino server, it might take the usual short while to locate and retrieve it.)

Finally, how does the view refresh timer work? 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
```

This moves the "selected document" focus back to the top of the view, which is where NotesTracker will keep placing documents that have just been logged.

This is quite analogous to the optional HTML tag in a web page's HEAD section, like this:

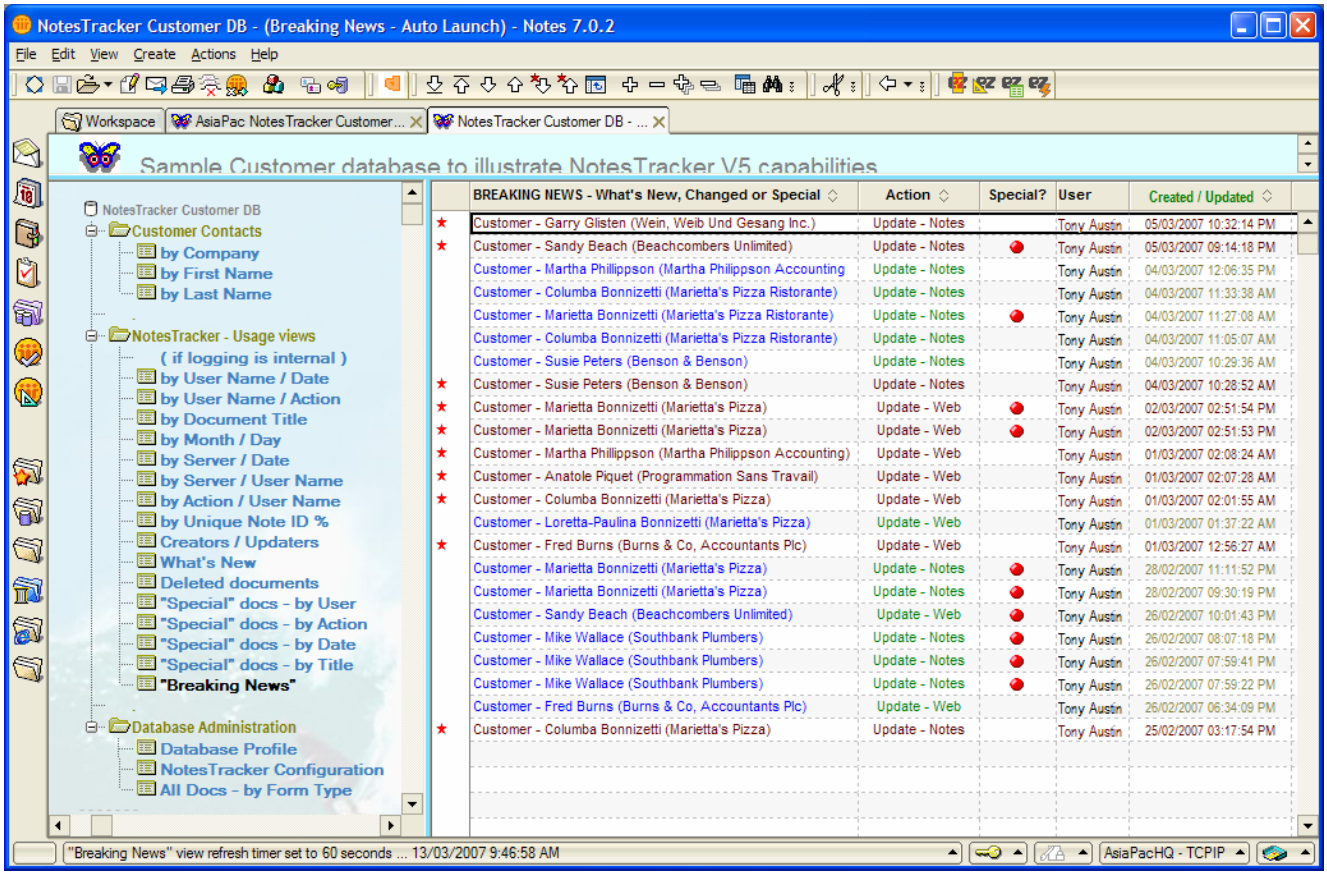
```
<meta http-equiv="refresh" content="900">
```

This, of course, causes the web browser to reload the web page every 15 minutes (900 seconds).

You are able to test this by accessing (through a browser) the example database "**NotesTracker Customer DB**" that is part of the NotesTracker Version 5.0 distribution package. If you are a licensed user of NotesTracker (rather than an evaluator), you should examine the design of the page named "**Breaking News View page.**" You will see that the HTML Head Content retrieves the very same refresh interval value as for the Lotus Notes client! This value, as described earlier, is retrieved from the database's profile document, with the same default refresh interval of 60 seconds (and settable to anything between five seconds and two hours via a radio button in the profile document).

Breaking News example view in the NotesTracker Customer DB

There is a slightly different example view in the **NotesTracker Customer DB** which looks like this:

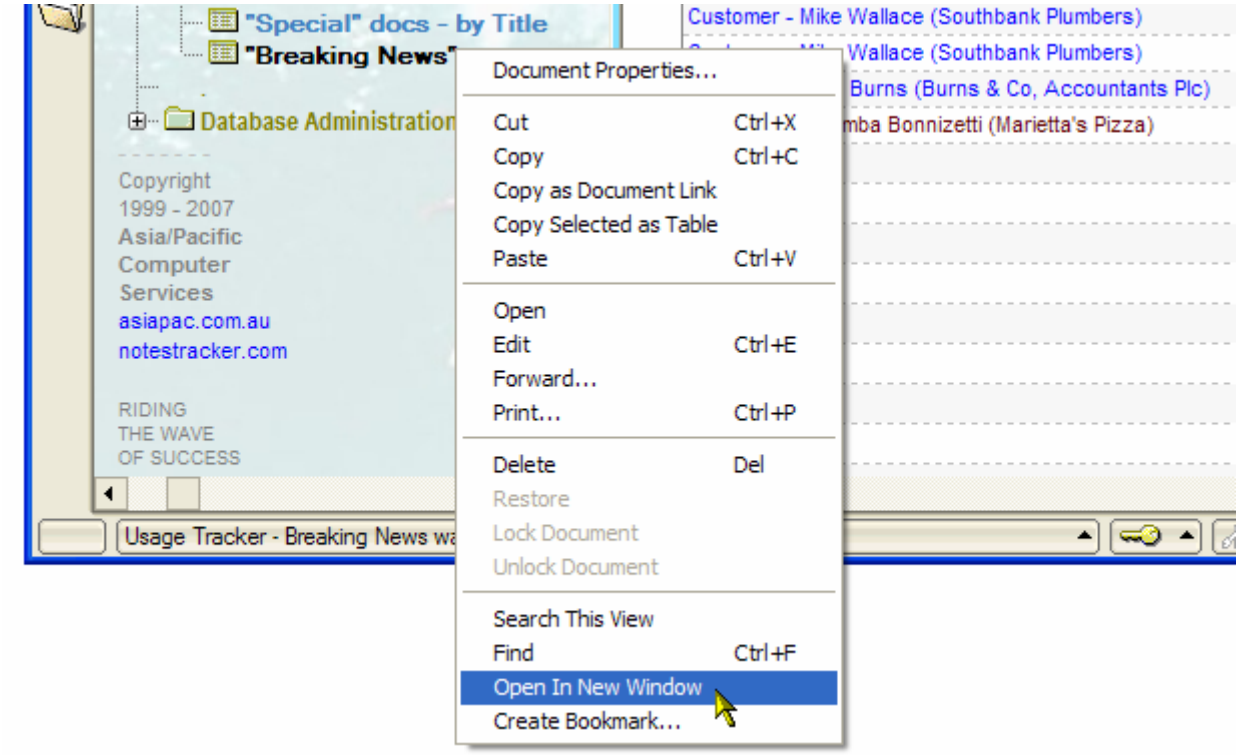


Apart from containing what is perhaps a more realistic set of columns than does the previous example, this view is designed the same way. The refresh interval is retrieved from the Database Profile document, and everything operates as explained just above about the Breaking News example view in the NotesTracker Repository database.

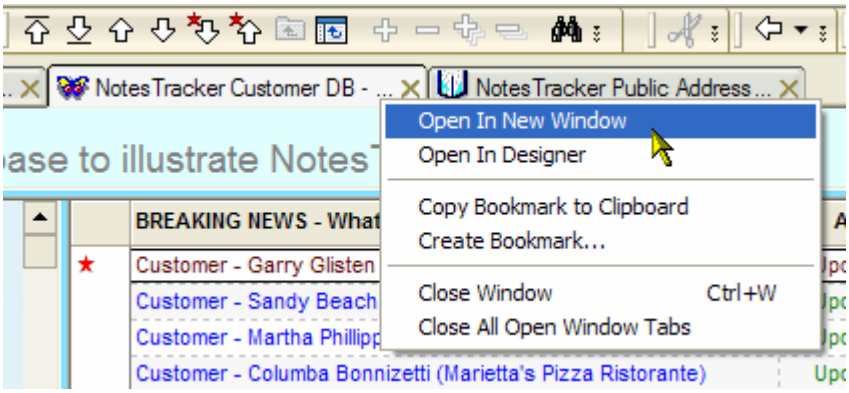
The problem with using the Breaking News view in this manner is that as soon as you tab to another window in the Notes Client you lose sight of the view, and so the automatic refreshing of the view does very little for you.

There's a way out of this conundrum, and not only is it very easy to do but it's an effective way to keep watch on the Breaking News view.

TIP: simply **launch the view in a new window** by right-clicking on the navigator entry, like this:



Alternatively, right-click on the tab for the Breaking News view, like this:



With the auto-refresh interval set to something realistic -- let's say between five and fifteen minutes -- all you need do is glance at the window to see the latest items appearing (at the top of the view) soon after the Usage Log entries arrive!.

Effective, and Easy to Implement

This exemplifies what NotesTracker offers you: an almost perfect RSS/Atom type of news feed, triggered by events happening in just about *any* of you Notes database applications.

And it's extremely easy to implement, using the simple design techniques outlined in this guide! This includes setting the refresh interval as an easily-edited field in a database's profile document, which will apply for both Notes and Web browser instances of a view.

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

Essentially, as with the **NotesTracker Customer DB** example database, whatever may be built into a meaningful Notes view may appear in a portal page and viewed using either a Notes Client or a Web browser.

They need not be just document Creates of course, but could be document Updates (instead of or as well as Creates), or have any sort of document Selection formula plus column sorting and categorization arrangement that generates a meaningful view. By their very nature, you would such Breaking News views to be based on a small subset of the Usage Log documents in the NotesTracker Repository, which in turn means that their view indexes are not likely to be of concern to your Notes administrators.

Therefore you should not worry too much about a proliferation of Breaking News views. Use them freely and to your organization's advantage. You have great opportunities -- based upon NotesTracker's ability to log database usage from all sorts of databases deployed all around your Domino network, and using the simple techniques described earlier in this guide – to generate with ease a range of newsfeed-style Notes views that can be used in your applications. And they can be viewed equally well in both your Notes Client and your Web browser portal pages.

What's more, the opportunities have been boosted in NotesTracker Version 5.0, with new features such as being able to nominate "special documents" which by their nature somebody in your organization wants to keep an eye on (and so would be prime candidates for inclusion in newsfeeds).

The newsfeeds that you build can be populated *in parallel with* e-mail alerts (another new feature in NotesTracker Version 5.0) to keep your database users "in the know" at all times!

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 efficient way to keep up with news, you really should do so!

We have some *general information about RSS feeds* on our web site. Refer to either of the following:

- http://asiapac.com.au/Links/KM.htm#Newsfeeds_Webfeeds_RSS (web server in Australia)
- http://notestracker.com/Links/KM.htm#Newsfeeds_Webfeeds_RSS (web server in USA).

There are also some relevant articles in various forums and weblogs that might give you some ideas about RSS and Domino, such as the excellent **OpenNTF** site at http://openntf.org/

There is also the following article at IBM developerWorks, which has some good background information:

- Building RSS feeds for Lotus Domino Document Manager
  at http://www-128.ibm.com/developerworks/lotus/library/ddm-rss-feeds/

RSS Feeds – Supported Directly in IBM Lotus Notes Domino Release 7.0.2

Even better things have happened with the release 7.0.2 of Notes/Domino. There is now an IBM-supported RSS feed generator database template that contains a collection of agents and script libraries designed to produce RSS feeds for views in Domino databases, including E-mail, Calendar, and Contact entries from a user's database, Corporate contacts, and Discussions.

The IBM Lotus Notes/Domino 7.0.2 Release Notes for information about using this RSS capability. Refer to "**RSS feed generator database template**" in the "New in this release" section.

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 unauthorized 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.

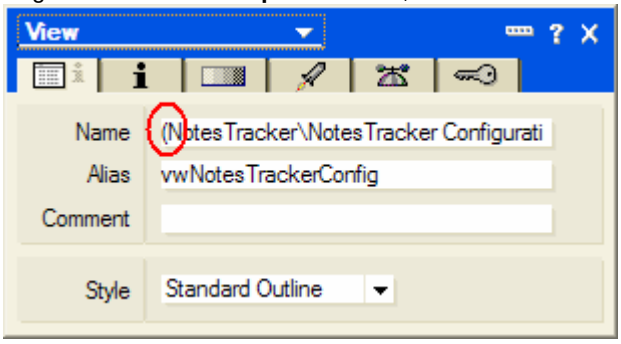
While this "edit history" tells you who made changes to the configuration document and when the changes were made, it does not control who can make changes.

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:



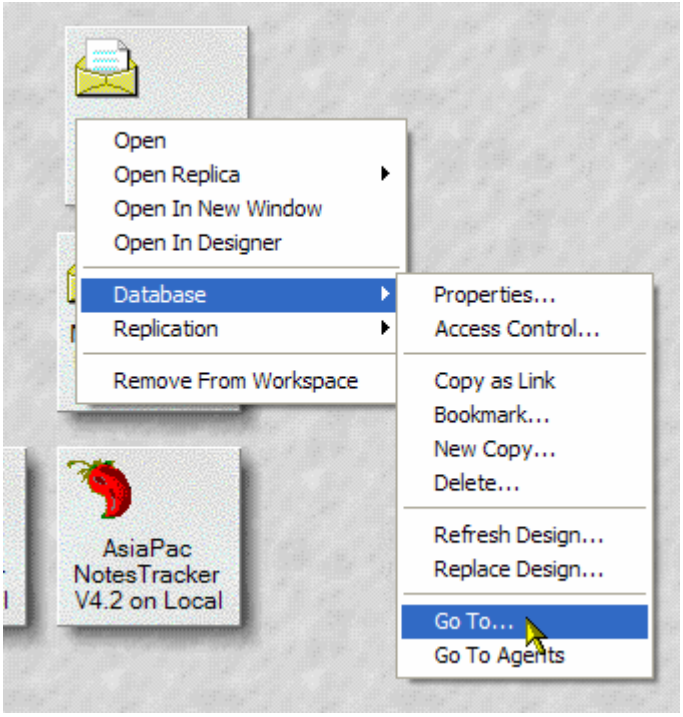
Note: be particularly sure not to alter (or remove) the View Alias name “vwNotesTrackerConfig” since this exact alias name is critical to the successful running of the NotesTracker code.

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

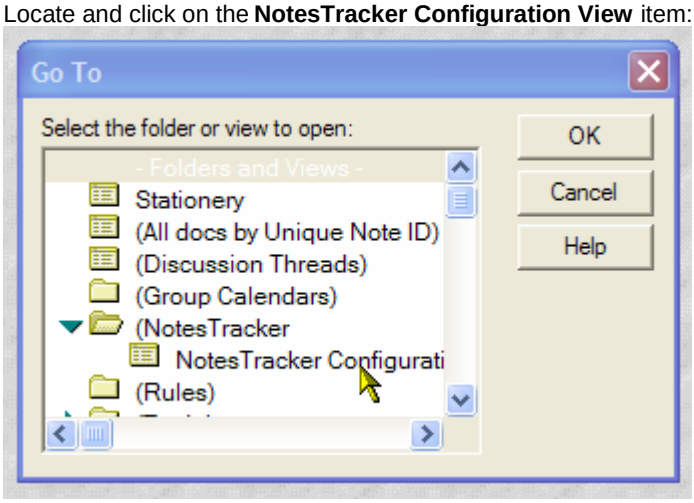
The answer is to use a “trick” that is not known by many Notes users (and for our purposes it is best that you don’t publicize this method).

The trick method is:

1. Go to the Notes workspace
2. Hold down simultaneously the **Ctrl** 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:



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.

Better to use a Profile Document for NotesTracker Configuration?

Although for historical reasons NotesTracker was not designed and developed this way, it would be possible to store the configuration data as a Profile Document rather than as a regular document accessed via a configuration view.

This would have some advantages, such as:

- It would be easy to hide the configuration document from those who should not ever see it
- There would not be the requirement to copy in the configuration view and then create the NotesTracker configuration document in each tracked database. This would mean a small reduction in Notes developer effort, and less to go wrong operationally (such as the view getting removed, and the configuration document getting duplicated or deleted).
- Probably slightly better performance.

You could make the necessary changes yourself, if you really like this approach. Otherwise, if there is enough feedback from NotesTracker users requesting a changeover to the Profile Document approach, then it might be implemented as a standard feature in a future release of NotesTracker.

Send any feedback on this to NotesTrackerSupport@asiapac.com.au

Extended Usage Analysis and Reporting

Tailoring of Views, and Interfacing with External Analysis Tools

In the NotesTracker Repository database, a set of views have been put together in the expectation that they will meet at least your initial requirements for usage analysis (by user, by date, by action type, by server, by database, special documents, contributors, breaking news, and so on).

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-categorized, 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 categorizes 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 nature and extent of analysis that can be performed using Notes views alone. 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

Another approach is put forward by Chuck Connell, of CHC-3 Consulting (Woburn, Massachusetts, USA). This is described in an article of 8th March 2007 that Chuck authored for SearchDomino.com: "A user-friendly and flexible data export agent for Lotus Notes" at http://searchdomino.techtarget.com/tip/0,289483,sid4_gci1246642,00.html There is an example database available on CHC-3's free downloads page at http://www.chc-3.com/downloads.htm. At the time of writing this guide, the example is described as follows:

> **Flexible Data Export from Notes** (download link http://www.chc-3.com/downloads/export2.zip) – "An improved version of my popular mail-merge example. This sample allows you to define any number of data export configurations within a Notes application. The data can be used by Microsoft Word or Excel, or just about any other program. Users can create their own data export setups, without programmer help! (The zip file contains both ND6 and ND5 versions.)"