Version 2.0.2

User's Guide and Technical Overview

Revised July 18, 2017

Table of Contents

Evernote and NixNote Overview

Evernote is company which provides note synchronization tools across multiple platforms. Evernote provides clients for Windows, macOS, iOS, Chrome OS, Android, Windows Phone, BlackBerry, and web based clients. Their goal is to be a “remember everything” type of service which allows you to search text, handwritten, images, and attachments regardless of where you are or what platform you are using. More information can be found at http://www.evernote.com.

Absent from their support is Linux. When asked about Linux their usual response is that the market is too small to warrant a client and Linux users can use the web interface to access their notes.

NixNote is designed to use Evernote's public API to provide a native full synchronization client for Linux users.

NixNote's feature set includes:

- Full synchronization of all notes and attachments. This includes linked and public notebooks as well as private notebooks.
- The ability to create, edit, and delete notes, tags, notebooks, and saved searches.
- The ability to search notes and the ability to index some attachments.
- The ability to use the image text recognition features that Evernote provides.
- Support for multiple Evernote accounts.

NixNote History

This project began several years ago as a side project written in Java. The initial goal was not to provide a full client, but to allow me to learn more about Evernote and to attempt to fulfill a few personal needs. Eventually it grew into a full client and was released under the name “NeverNote” in the hopes that others found it useful.

NeverNote was eventually renamed to “NixNote” at Evernote's request because they felt the name was too similar to Evernote and didn't want to confuse their users.

NixNote went through several releases until it reached 1.4. At that time, I decided to begin a rewrite to bypass some of the issues I had encountered with the Java and my . The overall goal of the new NixNote 2 was to provide a better user experience while still providing as many of the same features as NixNote 1.x. There will most likely not be any new releases of the Java based NixNote 1.x beyond version 1.6.

Differences between Evernote and NixNote

I've tried to keep NixNote with the same feature set as Evernote. Despite this, there are some differences. Some of these I hope to correct as time allows, some I'll never be able to correct due to limitations, and some I may never correct unless someone asks since I don't use them..

Features in Evernote that NixNote doesn't have or does differently:

- The search syntax is slightly different. I think all the same search features work, but NixNote allows any term to be negated, where Evernote does not. For example, searching for the term “notebook:inbox” will find any note in the inbox notebook, but searching “-notebook:inbox” will search for any note containing the word inbox and doesn't contain the word “notebook”. NixNote interprets this differently. It will search for any note that is not in the notebook inbox.

  NixNote uses SQLite's FTS search engine. It searches only words that begin a match. For example, if you have a document with the text “Mississipi”, you can search for “Mississippi”, “Miss” or “Miss*” and find the note. You cannot, however, search for “issipi” and find the note because the word “Mississippi” doesn't start with “issippi”. To get around this limitation, you can search for “*issippi” and find Mississippi. This is because the search word starts with a “*”. This tells NixNote to handle the search a bit differently. The downside of this is that the search will be slower.

- Twitter, Facebook, and LinkedIn sharing. I don't use them so I don't plan on implementing sharing to those services. If you need sharing to them let me know and I'll consider it, but it isn't high on the list of priorities.

- Audio notes are not directly supported. Evernote allows you to directly record a message, but NixNote does not. The best workaround is to create the audio file in another application and attach the file to a note.

- Ink notes are not possible. Evernote doesn't provide an API for ink notes and reverse engineering the file format is something I can't do. You can view ink notes as an image, but that is it. Many of Evernotes non-Windows clients have this same restriction.

There are a few small features that NixNote has that Evernote doesn't. None of them are earth shatteringly cool features, but I like them.

- You can close or open notebooks to make them disappear. This is similar to the selective sync feature, but it is different. It allows you to do full synchronization of a notebook, but you can hide notebooks you don't frequently use. I use this feature for some older reference material that I

seldom need, but I don't want to forget. This just hides them in the interface so when I search I don't need to look through all those old notes. The important thing to remember is that this does NOT impact any synchronization of a note.

- Totals for tags and notes are dynamically updated filtering notes. For example, if you select the notebook “inbox” the counts beside the tags change to only be the count of notes matching your selection criteria.
- Editing the source of a note is possible, so if you REALLY want to mess with the formatting or just want to see what it looks like you can do an Edit/View Source. I don't recommend it for everyone since you can really mess things up, but it can be useful at times.
- Notes can be “pinned”. A pinned note will always appear in the note list regardless of any other selection criteria. If a notebook is closed, however, a pinned note will not be visible. If you have the tray icon enabled, pinned notes are also quickly available via a menu option.
- You can “quick link” from a note to another note by highlighting text and selecting “Quick Link” from the context menu. It will then search for any notes with titles matching the selected text.
- LaTeX formulas can be inserted as PNG images and edited later when needed. You must have the program “mimetex” installed for this to work.
- HTML entities can be entered directly via a dialog box.

Known Limitations

There are a few known limitations with NixNote that users should be aware of when using the program.

- The largest problem with NixNote deals with pasting of notes from a web browser or any other HTML formatted application (like LibreOffice or MS Office). To provide a constant experience across devices, Evernote is very strict on the structure of a note. Notes that do not adhere to their formatting rules cannot be uploaded to Evernote. HTML web pages & other HTML formatted applications are typically not well formatted. NixNote makes an attempt to clean them up so they can be synchronized, but this is a “best effort” attempt. If you paste from a web browser you run the risk of getting a formatting error an “invalid note structure” error. You can edit the HTML of a note directly to try and correct the problem by going into “View/Show Source”. I typically recommend either doing a “Paste as Unformatted” text or using Evernote’s web clipper to get around this limitation.
- Ink notes cannot be edited. As stated above, Evernote doesn’t provide an API for ink notes so you can view them on a Linux client, but not edit them. The Windows client doesn’t support viewing ink notes at this time.
- Not all window managers on Linux support all of the possible features. There are a lot of different window managers with a lot of different options. It is impossible to support all of the options and there is no easy way to get around some of the limitations. Because of this, some features may not work on some window managers.
- NixNote has the ability to minimize and/or close to the system tray on some window managers. This can cause data loss on some systems if a note is modified and a user does a shutdown before a note is saved. There are a few ways to mitigate this problem.
    - On Linux you can reduce the probability by going under Edit/Preferences and enabling the “Intercept SIGHUP” in the Advanced tab. This option is explained more fully below.
    - On Windows, you can reduce the probability by going under the Edit/Preferences tab and enabling the “Use libtidy directly” option. On Windows there is no way to intercept the SIGHUP signal. When the “Use libtidy directly” option is not selected, NixNote will try to start a separate process to save a note when it shuts down. If this is when the system is shutting down or a user is logging off it is possible that Windows won’t

allow the process to start. Enabling this option keeps the tidy operation in the same process, so Windows allows it to save properly.

- On both Windows & Linux, you can reduce the auto-save interval in Edit/Preferences under the Advanced tab to a lower value.

Installation

NixNote can be installed in various ways depending upon the distribution you are using.

Debian based distributions (including Ubuntu) can use the deb packages. Once downloaded use the command “sudo dpkg -i <filename>” and the deb will be installed.

RedHat based systems cad download the RPM. Once downloaded use the command “sudo rpm -uvh <filename>” and the RPM will be installed.

For both RPM and DEB packages, you will most likely need to download and install any dependencies your system is missing.

If you are using a distribution other than a RedHat or Debian based system you can download the tar file. Once downloaded and unpackaged you can install it by running the “sudo ./install.sh” script from the directory. There is no automatic checking of dependencies with a tar file, so you will need to validate the dependencies manually.

As a final resort, you can download the source and compile it yourself. The build instructions are in the development section later in this document.

Once installed, NixNote should appear under the “Networking” section of your menus. The first time you run NixNote, it will create a database in ~/.nixnote/ directory.

Plugins

There are two plugins currently available for NixNote. These are separated into separate plugins for a couple of reasons.

The first reason is that it reduces the libraries needed to run NixNote. The plugins require specific versions of various libraries and not all distributions provide them. By separating them into plugins, the base NixNote program can continue to be used without requiring people to do a lot of effort to find & manipulate the required libraries.

Another reason is that not everyone wants the features provided by the plugins, so there are fewer dependencies needed and reduces the overall size of NixNote.

Finally, the features provided by the plugins are not supported on Windows because they require specific Linux libraries and I haven't had the time to look for Windows alternatives. Having them as plugins makes it easier to port NixNote to Windows.

Here are the two plugins available:

Spell Check:

The Hunspell plugin (libhunspellplugin.so) allows for a dialog box to spell check a note. This uses the Hunspell library to check the spelling of a note's text against the system default language. At this time the system only checks for your default language. Multiple languages are not supported.

Webcam:

The webcam plugin (libwebcamplugin.so) will allow you to use your webcam to take a picture and produce a static image as a note. It really just creates a png file of whatever the camera Is looking at, but it can be useful in some instances. After the note is created, text can be entered into the note.

Interacting with Evernote

Initial Synchronization

NixNote is designed to interact with Evernote and synchronize any data you have. The first time you synchronize it will download all of your data. The first sync can take a very long time based upon the size of your database.

If it encounters problems and crashes on the first sync, you should be able to restart the program and continue the synchronization. It will take up the synchronization where it stopped.

Storing your userid and password

NixNote does not have access to your userid and password. Authorization is granted through Evernote directly when you first synchronize. This gives NixNote an authorization token that is good for one year. When the year expires you will be prompted for again for your userid and password and another token will be given to NixNote. If you desire, you can revoke NixNote's access at any point via your account on http://www.evernote.com.

Synchronization intervals

NixNote will synchronize every 15 minutes by default. You can customize this under the Edit/Preferences dialog in the “Sync” section.

There are a few other options you can customize if you wish.

- “Enable Sync Notifications” – This provides a visual message from the system tray when a synchronization is complete. You must have the tray icon enabled (under the Appearances) tab for this to be enabled.
- “Show successful syncs” - When enabled, both successful and unsuccessful syncs will display via a popup message from the tray icon. When not selected only errors will show a message.
- “Popup message on sync errors” - When enabled, a popup dialog will appear when a sync error occurs.
- “Sync on Startup” – This option will automatically log you into Evernote and start a sync when you start the program. If you haven't granted NixNote account access via the login page, this option won't work.
- “Sync On Shutdown” – By choosing this option NixNote will do a final synchronization when it shuts down. This option will only work if you have granted NixNote access via the login page at least once. Depending upon

the amount of data, this could cause NixNote to run for a while in the background.

- “Sync Automatically” - When set, this determines how often NixNote will attempt to sync with Evernote.
- “Restart sync on API limit” - This is an experimental feature that will automatically restart a sync when NixNote encounters an API rate limit exceeded error with Evernote. The wait amount will vary depending upon the time to the top of the hour. See below for more details on the API rate limit set by Evernote.
- “Enable Proxy” - Enable proxy configuration when connecting with Evernote.
- “Enable Socks5” - Enable Socks5 proxy support.
- “Proxy Hostname” - The hostname of the proxy to use when communicating with Evernote.
- “Proxy Port” - Port to use on the Proxy server.
- “Proxy Username” - Proxy username to use when connecting to the proxy server.
- “Proxy Password” - Password to use when connecting to the proxy server.

The initial sync with Evernote can take a long time, but it should be quicker after it completes the first sync.

Evernote also limits the number of API rquests that can be done in an hour. This is to prevent a rogue application from flooding their servers. If you encounter this, you’ll see an error “API Rate Limit Exceeded” and you will need to wait for up to one hour to continue to synchronize. When you grant NixNote access to your account, this limit is much higher than it will be a few days later. Evernote does this to allow for the initial sync with the understanding that later, incremental syncs, will need to perform less API requests.

Emptying the trash

When you delete a note it is moved to the trash. The next time you synchronize that note should then be moved to the trash in your Evernote account. This works the same as any other Evernote client.

There is one difference in behavior between NixNote and the way Evernote’s official clients handle the trash. When you empty the trash on an Evernote client it is

permanently removed from your local database and when you synchronize it is permanently removed from your online account.

NixNote doesn't work this way. In NixNote, when you empty the trash you can no longer restore it and the note vanishes from your trash, but when you synchronize it will remain in your trash on in your online account and is not permanently deleted. If you empty the trash on another Evernote client, the note is then permanently removed from NixNote. If you restore the note from Evernote's trash it will reappear in NixNote.

This is due to an Evernote security restriction. In the past, they had third party API developers delete the wrong notes and caused a lot of work for their support team to restore the notes. To prevent this they do not allow other clients to permanently delete notes.

Using multiple Evernote accounts

NixNote will allow you to access multiple Evernote accounts at one time, but these accounts are stored in different databases. To do this, look under the "File" menu option and click "Add Another User". This will give you a dialog where you can specify the name of this account and (if desired) allow you to change to a different Evernote service. After creating a new account, you can switch to that account and do a sync. Authorization is not shared between accounts, so you can have multiple active accounts in NixNote at one time.

Using NixNote

Notes

Ink notes

NixNote allows you to create a text note and synchronize it with your Evernote account. You can't create or edit ink notes, however viewing ink notes is possible. When you synchronize it should pull an image down and show it to you. The note is read-only and you can't change the contents.

Evernote doesn't provide an API or any documentation on the file formats for ink notes, which is not helpful for determining how to create and edit them. The Evernote non-Windows clients cannot edit ink notes, so I'm not expecting them to provide this ability to other third party clients any time soon.

Merging notes

You can merge the contents of multiple notes together. This is done by selecting multiple notes in the note list and choosing "Merge Notes". The order of the merged notes is the same order in which you selected them.

Note history

If you are a premium user, you can restore older versions of notes from Evernote's servers. This is under the "View" menu option "Note History".

Duplicating a note

NixNote allows you to duplicate an existing note. Any data or file attachments are treated as separate notes. This can be useful if you wish to use one note as a template for other notes.

Background color

You can change the background color of any note by right clicking in the note and choosing the "Background Color" option. This is a non-standard feature and it may not work on all Evernote platforms. Depending upon your web browser it may appear properly in the web interface, but editing it in the web interface will remove the color.

Viewing note lists

You can customize the list of notes and you can customize where it appears. By default it is on the right hand side of the screen just above the note editor. By selecting "View/Narrow List View" or "View/Wide List View" you can move it from appearing above the note to appearing between the note and the left hand panels.

If you don't wish to change which columns are visible in the note list, you can customize them by right clicking the title bar and choosing the columns to view.

The “wide view” and “narrow view” are separate so customizing one does not update the other. This is to allow for things like a thumbnail in the narrow view where you have more vertical space, but hiding it in the wide view where you want to be able to see a larger number of notes.

Printing

Printing under Linux requires that CUPS be installed. If any text is selected, then only that portion of the note is printed, otherwise the entire note is printed.

Emailing

Notes and their attachments can be sent via email, but the server settings must be configured. See the “Preferences” section below for more details.

Encrypting Text

Plain text can be encrypted or decrypted, but attachments or images cannot. Evernote's client has this same restriction. To encrypt or decrypt text you must have Java installed. Once Java is installed, you can select the desired text and right-click to bring up the context menu. From there you can choose the Encrypt Text option.

Note HTML Source

You can view the HTML source of a note under the View/Show Source menu option. Doing this allows you to change the markup of a note more directly and overcome some of the editor's limitations, but it also means you can really mess things up. NixNote won't prevent you from producing really badly formatted HTML but it will still try to clean up anything you create so it can synchronize with Evernote properly.

Note Links

Note links can be created in NixNote by right clicking on a note in the note list and selecting “Copy as URL”. You can then paste the URL into another note. When you click on the link, the note will open in a new window.

Notebooks

Creating, editing, and deleting notebooks

Notebooks can be created or renamed the same as any Evernote client. When you create a notebook you have the option of creating it as a local or synchronized notebook. Local notebooks are saved only on your hard drive and never go through Evernote's servers. If you use local notebooks I highly recommend doing frequent backups to preserve any data in the event of a crash.

You can move notes from synchronized notebooks to local notebooks. If you move a note to a synchronized notebook it will be created on Evernote the next time you synchronize. If you move from a synchronized notebook to a local notebook that note is moved to the trash on your Evernote account the next time you synchronize.

Renaming a notebook works the same as any other Evernote client. The next time you synchronize it will update the name on Evernote. NixNote will not permit two notebooks to have the same name and it is case insensitive, so you can't have a notebook called "My Data" and another called "my data". NixNote considers them to be the same. Evernote's clients work this way too.

Deleting a notebook is possible only if all notes in that notebook have been deleted or moved to other notebooks. It doesn't allow you to delete a notebook and all notes in one operation.

Stacks

NixNote supports stacks the same as other Evernote clients. To set a stack right click on the notebook and select "Set Stack" or choose the option "File/Notebook/Set Stack". There, you can set a stack name or choose an existing stack for that notebook. To remove a notebook from a stack simply blank out the stack name and click "OK". It will then be removed from that stack.

Opening and closing notebooks

NixNote has a non-standard feature to permit you to show or hide specific notebooks. This allows you to, for instance, hide all your personal data on your work PC or hide notebooks you seldom use.

To open or close a notebook, choose the option "File" menu under "Open/Close Notebooks". A dialog box will appear where you can choose the notebooks you wish to view.

Please note, this does NOT change any synchronization behavior. Any changed notes are synchronized regardless if the notebook is opened or closed and all data remains on your hard disk. If you don't wish to have this feature, then you need to do a selective synchronize.

Tags

Tags in NixNote work in much the same way as other Evernote clients. You can't have duplicate tag names and selecting a parent tag does not show notes containing the children of those tags. This is to mimic Evernote's interface.

When you create a tag it is placed in the top level of the tree. If you want it somewhere else you need to drag it to that tag and it will be moved to be a child of that parent tag.

Merging Tags

You can merge multiple tags into one tag. To do this, select the tags you want merged (including the one you want them merged with). Right-click in the tag menu and select the "Merge" menu option. This will bring up a dialog box of the selected tags. Choose the tag you want them merged into and click OK. Any notes with any of the selected tags will be changed to the one tag. Please note that the old tags are not deleted.

Saved Searches

Saved searches work the same as in the Evernote client. I believe the full syntax is supported, but as Evernote announces new search keywords NixNote may be a little behind implementing them.

The current items supported are:

- Words in the title of a note (intitle:)
- Tags (tag:)
- Notebook (notebook:)
- Author (author:)
- Source (source:)
- Source Application (sourceapplication:)
- Created (created:)
- Updated (updated:)
- Subject Date (subjectdate:)
- Todo (todo:)
- Recognition Type (recotype:)
- Stack (stack:)
- Longitude (longitude:)
- Latitude (latitude:)
- Altitude (altitude:)

In addition to the search terms, you can prefix with a term with a negative (i.e. -tag:) and it will negate the statement. Using this, you could search for all notes that don't match the criteria (for example, all notes not tagged with a value).

Favorites

Favorites in NixNote works much the same way as Evernote's windows client. Dragging a note, notebook, tag, or saved search to the favorites menu allows for quicker access to those notes.

Please note that favorites are not synchronized across clients. Evernote has not yet provided an API interface to permit this.

Deleting Notes

Deleting a note moves it to the trash. It can be restored at any time until the trash is emptied. When the trash is emptied, the note is not permanently deleted from NixNote until you empty the trash on an Evernote client. The section “Emptying the trash” above has more details.

Screen Clipper

NixNote does have a screen clipper, but it may not work on all systems. To use it either click on the “Screen Capture” note button in the toolbar (click on the down arrow in the New Note or New Webcam button). Another way to get it is to go to Edit/Preferences/“Appearance” tab and check “Show tray icon”. Once you see the tray icon, you can right click to bring up a menu and choose “Screen Capture”.

If the screen clipper doesn't work for you or you have a favorite screen clipper, you can mimic this behavior by setting up an import folder (look under Tools/Import Folders) and save your clipped images to that directory.

Colored Notes

NixNote allows you to change the background color of a note. This isn't officially supported by Evernote, but it will display properly on some of their clients. You can set the background color by right clicking in the note and selecting “Background Color”. Be aware that editing the note on an Evernote client or in the web version of Evernote may remove this color. It depends upon the client.

Keyboard Shortcuts

Most of the menu items in NixNote can be customized. To do this, you must create a shortcut.txt file in your ~/.nixnote directory. For more information on customizing shortcuts, look in /usr/share/nixnote2/shortcuts_howto.txt. After making any changes, you must restart NixNote for the changes to take effect.

Themes

NixNote does have the ability to change most of the icons and some of the colors within the application. To do this, you should use /usr/share/nixnote2/theme.ini as a template. There can be multiple themes in a file and you can choose to override some icons or all of them. You can edit /usr/share/nixnote2/theme.ini and the customizations will be available to all users, or you can create ~/.nixnote/theme.ini in your home directory and those changes are only available to you. Entries in

~/.nixnote/theme.ini override the settings in /usr/share/nixnote2/theme.ini. The ones in /usr/share/nixnote2/theme.ini override the default application icons. If an icon is not specified in a theme.ini or if the file it references do not exist, the default applicationt icon is used. To choose a desired theme, select them via the menu option under Edit/Icon Theme. Most changes happen automatically, but some desktop environments may require a restart of NixNote or the user to logout and back in for all icons to be refreshed.

Preferences

Under Edit/Preferences there are various options that can change the appearance and behavior of NixNote.

The first tab you will see is the Appearance tag.

- “Show tray icon” - If you window manager supports it, this will show an icon in the tray of your status bar.
- “Show splash screen on startup” - When starting NixNote, display a splash screen until the startup is complete.
- “Display PDFs Inline” controls how NixNote displays notes with PDF attachments. If selected, it will try to create a PNG image of the first page of the PDF. The user can then use the arrow keys above the image to browse through the PDF without leaving NixNote. If this value is unchecked, the PDF will be displayed as a small icon which, when clicked, will launch the default PDF reader. Please note: There are currently issues with Gnome using this feature. It is recommended that if you run Gnome, you do not enable this feature.
- “Show missed reminders on startup” - If NixNote wasn't running when a reminder was due, this option will show it when NixNote starts the next time. If unchecked, you will not receive any notifications of missed reminders.
- “Show notebook and tag totals” - If enabled, this will dynamically update the number of notebooks and tags. The total is in two parts. The first number is the number that matches the current filter criteria and the second is the total number of notes in that notebook/tag. If only one number is shown, it means none of those notes are filtered.
- “Always start minimized” - Always start NixNote as a minimized application. This is particularly useful if you have it set to autostart and minimize to the tray.
- “Start automatically at login” - Starts NixNote whenever you login.
- “Disable note editing on startup” - If this selection is checked, you will not be able to edit any notes when starting NixNote.
- “Focus on Note Title on New Note” - If selected, the cursor will be placed on the note title, rather than the note body when creating a new note.
- “Confirm Deletes” - Prompt for verification prior to deleting an item.

- “Limit Editor to Web Fonts” - Only show fonts that are directly available in the Evernote Web application. Other fonts are not shown in the editor's font list.
- “Show note grid list” - Show grid lines in the note list. If not selected, no lines are shown.
- “Alternate note list colors” - Display alternating row colors on the note list.
- “Set author on new notes” - Attempt to automatically populate the author on any new notes.
- “Preview fonts in editor dialog” - When enabled, the font list in the editor dialog will show a preview of each font available. This helps you see visually how each font will appear, but it can also slow down the initial selection of this dialog.
- “Startup Behavior” - This controls what notes you see when NixNote starts.
    - “Restore Selection Criteria” - Whatever search and selection criteria you had when NixNote shut down the last time will be restored.
    - “Select Default Notebook” - Always start NixNote and just view any notes in the default notebook.
    - “View All Notebooks” – View everything when NixNote starts
- “Tray Icon Click Action” - If you have the tray icon enabled, this option controls what happens when you click on the icon.
    - “Show/Hide NixNote” - Show or Hide NixNote.
    - “New Text Note” - Show NixNote and create a new text note.
    - “New Quick Note” - Show NixNote and create a new note in a separate window.
    - “New Screen Capture” - Crete a screen clip note.
- “Tray Icon Double Click Action” - This is the same as the “Tray Icon Click Action”, except that it controls what happens when you double click the icon. This is known to have issues on some window managers.
- “Tray Icon Middle Click Action” - This is the same as the “Tray Icon Click Action”, except it controls what happens when the middle mouse button is clicked on a the tray icon.
- “Default GUI Font & Size” - These allow you to specify the default font and font size used within the application. Please note that this does not change

the size and font that the editor uses. You need to restart NixNote for this change to take effect.

- Default Editor Font & Size” - These allow you to specify the default font and font size used when viewing notes. Please note that this does not change the size and font used in other areas of the application. You need to restart NixNote for this change to take effect.

The next tab is the “Locale” tab. It is used to customize how dates and times are displayed.

The “Sync” tab discussed in the "Synchronization” section above.

Next is the “Search” tab. This tab controls how NixNote resolves searches and vairous behaviors when selecting notebooks or searching for text within a note.

- “Clear Notebook Selection on Search Text Changes” - If this option is checked, NixNote will unselect any notebooks in the left hand panel whenever the search string is changed. If this option is not checked, no reset will be performed. This option is useful if you usually search within a specific notebook. The default is unchecked.

- “Clear Tag Selection on Search Text Changes” - If this option is checked, NixNote will unselect any tags in the left hand panel whenever the search string is changed. The default is unchecked.

- “Clear Search Text on Notebook Changes” - If this option is checked the search text will be cleared whenever a new notebook is selected. The default is unchecked.

- “Show Any Matching Tags When Selecting Multiple Tags” - This option controls how notes are filtered when multiple tags are selected. If checked, any note that matches ANY selected tag is displayed. When unchecked, any note that matches ALL selected tags is displayed. The default is unchecked.

- “Minimum Recognition Weight” - When Evernote does text recognition of images, it returns a list of possible words in the image along with a weight. The higher the weight, the more certain Evernote is of a match. This value is used to determine the minimum recognition certainty for NixNote to consider it a match when searching. Changing this value does not require you to reindex your database.

- “Index PDFs Locally” - This tells NixNote to index any PDFs it receives locally using Poppler. If unchecked, PDFs are not indexed. Disabling this feature can improve performance and reduce your overall database size, but it comes at

the cost of being able to search PDFs. Any PDF information received from Evernote (including any image recognition) is still stored and searchable. Changing this does also not re-index any previously saved PDFs, but only impacts behavior on new or modified PDFs.

The “Email” setting controls various debugging settings. If confiured properly, NixNote will permit you to send any notes & their attachments via an email message.

- “SMTP Server” - The email server to use when sending emails.
- “Server Port” - The port to use on the SMTP server.
- “Connection Type” - The type of connection to use when sending email.
- “Userid” - The userid used to login to the SMTP server.
- “Password” - The password to use when logging into the SMTP server.
- “Sender Email” - The email ID to use as a reply-to address.
- “Sender Display Name” - The name to display on outgoing email messages.

The “Thumbnail” settings controls how thumbnail images are generated. Thumbnails are generated periodically as notes are added or updated. These settings can have negative impact on performance if set too aggressive, or cause thumbails to be generated very slowly if set to low. In addition, thumbnail generation can be eliminated entirely if you do not plan on using them. There are four main settings.

- “Disable thumbnail generation” - Do not generate any thumbnails. If checked, this reduces some of the work that NixNote needs to do and it will save some disk space. It also means that you cannot view any new or altered note thumbails in the note list. If unchecked, thumbnail generation will continue and any notes needing a thumbnail should have one generated.
- “Images to generate per interval” - This controls how many thumbnails to generate before going to sleep. Setting this too low will cause NixNote to sleep frequently between thumbnails. Setting it too high will cause it to go to sleep very infrequently. Valid values are 1-9999. The default is 1.
- “Minimum scan interval (in seconds)” - This controls the shortest period of time NixNote will sleep when generating thumbnails. It uses this interval if it did not generate all of the needed thumbnails, but the maximum number of images per interval has been reached. The default is 5 seconds.
- “Maximum scan interval (in seconds)” - This controls the longest period of time NixNote will sleep when generating thumbnails. When NixNote has

completed all of the necessary thumbnail work, it will go to sleep for this number of seconds before looking for new work.

The “Exits” tab is discussed in the “Exits” section below.

The “Advanced” setting controls various debugging & andvanced settings.

- “Disable uploads to server” - When checked, notes will only be downloaded and no changes done locally will be sent to Evernote. This is useful if you wish to test Evernote without running the risk of corrupting your notes.
- “Show LID column” - This adds an additional column to the note list that shows the LID (local ID) for each note. This is primarily used for debugging and most people really don't care about it.
- “Disable Tag Sorting” - Some people have reported a bug with some non ASCII characters. Checking this will disable the automatic sorting of tag names within the display. Some people have reported that this keeps NixNote from having lockup issues.
- “Disable image search highlighting” - When enabled, image text will not have matches highlighting when performing a search.
- “Strict note checking” - When selected, NixNote will be more strict in trying to validate that a note's structure matches Evernote requirements. Anything not matching Evernote standards is removed.
- “Force UTF8 Encoding” - Evernote doesn’t always seem to send the <xml> header with notes created or edited in the web interface. This can cause an issue with some special characters. As a workaround, this will add the <xml> tags to notes that are missing it. This is experimental and disabled by defalut, but if you are having issues where special characters are being displayed improperly this may correct the issue.
- “Intercept SIGHUP” - This is a Linux-only option.When enabled, it will prevent a logoff or shuhtdown until NixNote does a final save of any modified notes. Without this, there is a risk that NixNote will not have time to save any modified notes before Linux kills the process. With this enabled, NixNote will have the time to save any changes, but if an error is encountered it could potentially cause the system to lock up while logging out.
- “Use multiple threads to save note contents” - This is an experimental feature that will multi-thread the saving of a note’s contents. This can improve performance, but could also cause issues on some systems. If you do not encounter performance problems when saving or editing notes, it is recommend that this not be enabled. This value requires a restart to take effect when changed.

- “Use libtidy directly” - This is an experimental feature that requires a bit of explanation. In the past, NixNote starts a separate process to call the program “tidy”. This helps clean up some of the HTML in a note so it can synchronize with Evernote. When enabled, this feature calls libtidy directly rather than starting a separate process. This should slightly improve performance. It also gives NixNote the opportunity to save a note on Windows when a user logs off or does a shutdown. This is experimental because it could alter the behavior of NixNote and I wanted to give users the ability to use the older, better tested save method.

- “Auto save interval” - This is the time (in seconds) since the last change that NiixNote will wait before automatically saving a note. This can be useful if you want to be sure a note is changed in the event of a NixNote or system crash. The lower the interval, the more quickly it will save, but it also adds to overhead if set too low. The default is 300 seconds (5 minutes).

- “Message Level” - Specifies the amount and type of messages that are written to the log file. Normally, this should be set to “Info”, but changing it can be helpful when problems occur.

    - “Trace” - Produces a lot of messages about the various routines that NixNote is calling internally. It is useful in identifying potential bottlenecks. This message level shows all other messages as well as trace information.

    - “Debug” - Show all debugging messages. Debugging also shows informational, warning, and error messages.

    - “Warning” - Show only warnings and critical errors.

    - “Error” - Only ever show critical error messages.

    - “Fatal” - Show only the most critical messages that cause NixNote to fail.

    - “Info” - Show any informational or error messages. This is the normal message level.

Indexing

Interval and synchronization

Under normal operations, NixNote will index a note when it is saved. If needed, an additional background process can be enabled by passing the command line parameter “--enableIndexing”. This allows yout to optionally reindex the entire database but it can also cause some performance issues.

Reindexing specific notes or the entire database

Indexing your entire database takes time, but if you want to do it again you can tell NixNote to do this by choosing the “Tools/Reindex Database” option. Please note that this is only possible by passing the command line parameter “--enableIndexing”.

When the entire database re-indexed the search words for that note are not immediately deleted, but are deleted just prior to indexing that note.

You can determine how many notes need to be indexed by looking at the “Tools/Database Status” menu option.

OCR Data

When Evernote gets a note, it will OCR any images in a note and (if you are a premium member) it will OCR any images in PDFs. Evernote stores the OCR text and assigns a value to it. The higher the value the more certain it is that it found a word. The next time NixNote syncs, Evernote will send that OCR data and NixNote can then add it to its search database.

Data which has not passed through Evernote’s servers (local notebooks and unsynchronized notes) does not contain any OCR data.

Attachments

NixNote has the ability to index PDF, Microsoft Office (Excel, Outlook messages, Word, and PowerPoint), ODF, PDF, and text documents. This isn’t perfect but in my limited testing it seems to work reasonably well. This does, however, increase the database size immensely if you have a lot of attachments. You can enable or disable this under “Edit/Preferences” in the “Indexing” section. If you enable this option it will NOT reindex any existing attachments, so you may need to reindex your entire database to get all of your attachments indexed.

Editing Attachments

By clicking on a file attachment the default program for that file type is launched. If that program allows editing any changes made will be updated in NixNote. For

example, if you open an image in a note using an external editor, any changes to that image will be reflected in the note. This works the same as Evernote's clients.

Tools

Import folders

Import folders are used by NixNote to automatically import data when it appears in a directory. There are two types of import folders.

The first type is “Import and keep”. This type of directory is monitored while NixNote is running. If a file is moved into that folder or an existing file is modified it will create a new note in NixNote with the file name as the title. If NixNote is not running then any new or changed files in this directory will not be imported.

The second type is “Import and delete”. This is the same as “Import and keep” except for two things. The first difference is that the file is deleted when the note is created. The second difference is that if a file exists in that directory when NixNote starts it will import that file and create a note, so that even if NixNote is down a note will be created when it starts.

Both types of imports allow for recursive scanning of subdirectories, but please BE CAREFUL. If you tell it to import/delete it will IMPORT AND DELETE. Sometimes this can be disastrous if you setup a really bad location (like root).

If NixNote is running, it will notice the new (or changed) file as soon as it is written. It might not appear in your note list, however, until the search is refreshed.

Import/export of data

Under the File menu there is an option to export and import notes. Exporting notes creates a NixNote export file with the extension of .nnex and contains information about any selected notes. This includes notebooks, tags, and resources.

Restoring a note from a NixNote export does not replace any existing notes and it does not create any notebooks or tags. An imported note is treated as a new note.

Backup/restore of data

The backup and restore process is similar to the import/export process with a few exceptions.

The first difference is that everything is backed up (not just the selected notes). This means that it will take longer to backup a database than to export a few notes.

Restoring from a backup is also different. I only recommend using this when you have an empty database. Restoring will not create any new notes. Any restored notes should have the same GUID, tags, and be in the same notebook as it was when it was backed up. It also will restore notebook, tag, and saved searches.

Linked Notebooks

With a few minor exceptions, NixNote handles shared notebooks in a manor similar to Evernote. You cannot share individual notes or share notes via Twitter, LinkedIn or Facebook. To share a notebook, you must setup the share using an Evernote client or Evernote's web interface.

LaTeX Formulas

NixNote has the ability to add LaTeX formula images. This will only work if you have the “mimetex” program installed. Unlike NixNote 1.x versions, Internet access is not required.

The first way to insert a formula is to type the formula in a note, highlight it, and select “Insert LaTeX Formula” via the context menu.

The second way is to select “Insert LaTeX Formula” via the context menu with no text selected. This will cause a dialog box to appear and you can enter the formula.

Any LaTeX formula is treated by NixNote as a bitmap image. As a result, the image is sent to Evernote where the normal OCR processing is done. Once this is done, you can search for text in the image the same as any other Evernote client. Since other Evernote clients also see this image, you can search on those clients too.

NixNote does allow you to edit a LaTeX formula by clicking on a LaTeX image.

Command Line Options

help or --help or ? or -? or --?

Show command line options and exit.

start <option>

This is the default option when none others are specified. This simply starts NixNote normally.

Start options:

--accountId=<id> : NixNote configuration to use. This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

--dontStartMinimized : This overrides the setting in the Preferences dialog box and will force NixNote to start in a non-minimized state.

--disableEditing : This overrides the setting in the Preferences dialog box and will not allow editing of notes when starting NixnNote.

--enableIndexing : This will enable the background indexing thread to do work. This normally is not needed unless you want to do a full database reindexing. Enabling this option can cause performance problems until the database is reindexed.

--openNote=<id> : Open a specific note on startup. This parameter is optional. If not specified the last viewed note will be opened.

--forceSystemTrayAvailable : NixNote attempts to determine if the window manager supports tray icons. If it cannot verify that tray icons are supported it will not show a tray icon. This option forces NixNote to permit a tray icon regardless if NixNote determines a tray icon is supported.

--startMinimized : Override the Preferences dialog option and start NixNote in a minimized state.

--syncAndExit : Start NixNote, but don't display the GUI. After starting, synchronize with Evernote and exit. This is the same as the “sync” option below, except that the “sync” option below will force NixNote to do a sync if the GUI is available. If another NixNote is running, this option will not do a sync.

sync

Synchronize with Evernote. If NixNote is already running, the GUI instance will start a sync. If NixNote is not already running it will start NixNote without the GUI, do a sync with Evernote, then exit.

--accountId=<id> : NixNote configuration to use. This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

shutdown

If another instance of NixNote is running, this option will request it to shut itself down.

--accountId=<id> : NixNote configuration to use. This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

show_window

If another instance of NixNote is running, request it to show itself. If minimized, it will be un-minimized.

--accountId=<id> : NixNote configuration to use. This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

query <options>

Perform a command line query. The results are displayed back to the terminal. The same syntax that the GUI uses is supported via the command line.

Query options:

--accountId=<id> : NixNote configuration to use for the query. This defaults to "1". If you do not have multiple NixNote configurations you can ignore this option. This parameter is optional.

--search="<search_string>" : Query to use when searching the NixNote database. This parameter is optional, but if you don't specify it all notes will be returned.

--delimiter="<delimiter>" : When displaying results, the results are separated by this string of characters. It defaults to "|" and this parameter is optional.

--noHeaders : Don't display column headers when showin the search results.

--display="<output_format>" : Output fields to show and the format to use in displaying them. The output syntax is in the form of %<field><length><:>. This parameter is optional. The padding & truncation statements are also optional for each output field.

Display output fields:

%i : Display the note Id.

%t : Display the note title.

%n : Display the notebook containing the note.

%g : Display the tags assigned to the note.

%c : Display the creation date of the note.

%u : Display the last date the note was updated.

%e : Show if the note is synchronized with Evernote.

%s : Show the source URL of the note.

%a : Show the author of the note.

%x : Show if the note has a todo item.

%r : Show if the note reminder time.

%v : Show the time the reminder was completed.

Display output padding format:

<length> - any numeric number. If the resulting field is shorter than this length the result is padded to <length>. If it is longer, the result field is not truncated. Specifying this is optional.

<length>: - If the length ends with ":", the resulting field is padded to the length specified, but if the resulting field is longer it is truncated to that length. Specifying this is optional, and if not specified no truncation is done.

Example #1: Search for any note with the text "Famous" and "Author". The results are displayed with " * " separating the fields. The output format will show the note Id, the note title (padded to 10 characters and truncated if longer), and the notebook of the note.

nixnote2 query --search="Famous Authors" --delimiter=" * " --display="%i %t10:%n"

Example #2

Seacrh for any note with the text "Fred". The default field separation character (|) is used. The fields returned are the note Id, the notebook (padded to a minimum of 5 characters but not truncated), and the author of the note (padded to 7 characters and truncated if longer).

nixnote2 query –search="Fred" --display="%i%n5,%a7:"

readNote

Display the text of a specific note. This will only show any plain-text contents. Attachments and images are not shown.

readNote options:

--accountId=<id> : NixNote configuration This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

--id=<id> : The id of the note. This parameter is required.

appendNote

Append text to an existing note.

addNote options:

--accountId=<id> : NixNote configuration This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

--id=<id> : The ID of the note to append. This parameter is required.

--notebook=”notebook_name” : Notebook for the new note.

–tag=”tag_name” : Name of the tag to assign to the new note. For multiple tags, specify this option multiple times.

--attachment=”file_path” : File to attach to the note. For multiple attachments, specify this option multiple times.

--delimiter=”delimiter_character” : Delimiter for attachments within a note. When specified, any attacments will be added to the note where an attachment delimiter is specified. The default character is %%.

--noteText=”note_text” : The actual note body. To add attachments add the delimiter character in the note along with any –attachment parameters. If this parameter is not specified, then data will be read from stdin.

addNote

Add a new note to the database. Note: new notes are not seen immediately if another instance is running.

addNote options:

--accountId=<id> : NixNote configuration This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

--attachment=”<file_path>” : File to attach to the note. For multiple attachments, specify this option multiple times.

--delimiter="<delimiter_character>" : Delimiter for attachments within a note. When specified, any attacments will be added to the note where an attachment delimiter is specified. The default character is %%.

--created="<date_created>" : The date & time the note was created. This defaults to today's date & time. The date & time format is in yyyy-MM-ddTHH:mm:ss.zzzZ. For example, --created="1999-01-03T20:30:00.000Z" to specify January 3, 1999 at 20:30.

--updated="<date_updated>" : The date the note was last updated. This defaults to today's date & time. The date & time format is in yyyy-MM-ddTHH:mm:ss.zzzZ. For example, --updated="1999-01-03T20:30:00.000Z" to specify January 3, 1999 at 20:30.

--reminder="<datetime>" : Set a date & time for a reminder. This is an optional field that must be set for a future date & time. The date & time format is in yyyy-MM-ddTHH:mm:ss.zzzZ. For example, --reminder="2021-01-03T01:30:00.000Z" to specify January 3, 2021 at 01:30.

--noteText="note_text" : The actual note body. To add attachments add the delimiter character in the note along with any –attachment parameters.

alterNote

Modify notebook, add tags, or remove tags for selected notes.

alterNote options:

--accountId=<id> : NixNote configuration This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

--search="<search string>" : Search for notes matching this criteria. All notes matching are modified.

–-id="<id list>" : A list of note IDs to alter.

--notebook="<notebook name>" : Move the notes to this notebook.

--addTag="<tag name>" : Add this tag to the matching notes. If you want to add multiple tags you need to specify this parameter multiple times.

–-delTag="<tag name>" : Remove this tag from matching notes. If you want to remove multiple tags you need to specify this parameter multiple times.

--reminder="<datetime>" : Set a date & time for a reminder. This is an optional field that must be set for a future date & time. The date & time format is in

yyyy-MM-ddTHH:mm:ss.zzzZ. For example, --reminder="2021-01-03T01:30:00.000Z" to specify January 3, 2021 at 01:30.

--reminderComplete" : Set a date & time a reminder was marked as complete.

--reminderClear : Clear the reminder from a note.

deleteNote

Delete a specific note.

deleteNote options:

--accountId=<id> : NixNote configuration. This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

--noVerify : Don't prompt for verification prior to deleting the note.

--id=<id> : The id of the note. This parameter is required.

emailNote

Email a specific note. You must have setup the Email settings for this to work properly.

emailNote options:

--accountId=<id> : NixNote configuration to use for the query. This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

--id=<id> : The id of the note to email. This parameter is mandatory.

--subject="<subject>" : Subject line for the email. This statement is optional. If not specified the note title will be used.

--to="<email_address>" : Recipient address. Multiple --to=<email_address> will send the note to multiple recipients. This parameter is optional, but at at least one --to or one --cc or one --bcc must be specified.

--cc="<email_address>": Carbon copy recipient of the note. To specify multiple recipients, use multiple --cc="<email_address>" statements. This parameter is optional, but at at least one --to or one --cc or one --bcc must be specified.

--bcc=<email_address> : Blind carbon copy recipient of the note. To specify multiple recipients, use multiple --bcc=<email_address> statemnts. This parameter is optional, but at at least one --to or one --cc or one --bcc must be specified.

--note:=”<text>” : Additional notes to prepend to the note text. This parameter is optional.

export

Extract notes from the NixNote database. Notes are stored as NixNote extract files (nnex) and can optionally be deleted after being extracted. NixNote may not be active when this is run.

export options:

--accountId=<id> : NixNote configuration to use for the query. This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

--id=”<note_id_1, note_id_2 ...>” : A list of comma or space separated note IDs to extract.

--search=”search string” : Extrcat all notes that match the search string.

--output=<filename> : Export file name. This is required.

--deleteAfterExtract : If specified, notes will be moved to the trash after the extract is complete.

--noVerifyDelete : If “--deleteAfterExtract” is specified, this option removes the console verification prior to deleting the notes.

import

Extract notes from the NixNote database. Notes are stored as NixNote extract files (nnex) and can optionally be deleted after being extracted. NixNote may not be active when this is run.

import options:

--accountId=<id> : NixNote configuration to use for the query. This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

--input=<filename> : Export file name. This is required.

backup

Perform a NixNote backup. NixNote may not be active at the time this is run.

backup options:

--accountId=<id> : NixNote configuration to use for the query. This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

--output=<filename> : The output file for the backup. This field is mandatory.

openNotebook

Open a notebook that had been previously closed.

openNotebook options:

--notebook="<notebook>" : The name of the notebook to open. You can specify multiple notebooks by using multiple –notebook= statements.

--accountId=<id> : NixNote configuration to use for the query. This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

closeNotebook

Close a notebook.

closeNotebook options:

--notebook="<notebook>" : The name of the notebook to close. You can specify multiple notebooks by using multiple –notebook= statements.

--accountId=<id> : NixNote configuration to use for the query. This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

signalGui

Send a command to a running instance of NixNote from the command line.

signalGui options:

--show : Show NixNote if it is hidden or minimized.

--synchronize : Start synchronizing with Evernote.

--shutdown : Stop NixNote.

--screenshot : Take a screenshot and create a new note.

--openNote : Open a note in NixNote. The –id=<note_id> tag must also be specified.

--openNoteNewTab : Open a note in a new tab. The –id=<note_id> tag must also be specified.

--openExternalNote : Open a note in an external window. The –id=<note_id> tag must also be specified.

--newNote : Create a new note.

--newExternalNote : Create a new note in an external window.

–id=<note_id> : The ID of the note to open. (used with the openNote, openNoteNewTab, & openExternalNote options).

--accountId=<id> : NixNote configuration to use for the query. This defaults to the last used account. If you do not have multiple NixNote configurations you can ignore this option.

Exits

Exits are user written scripts that gain control at various points within the execution of NixNote and provide the opportunity to customize NixNote. For example, you could automatically assign tags to a note based upon the text contents within a note or check that a note is assigned to the proper notebook.

There is a sample exit in the “sample_exits” directory of the NixNote installation path. This example will check the start of a note for a specific date format. If it is found it will assign the creation date to match and will change the note’s notebook to “Journal”.

Exits are written in either JavaScript (for Qt5 builds) or QtScript (for Qt4 builds) and are configured through the Preferences dialog under the “Exits” tab. There are currently two exits supported. They are the “Note Load” exit and “Note Save” exit.

Exits, while providing a lot of functionality, can also cause some strangeness or data errors if your exit is coded incorrectly. Please be careful when coding exits.

Below are the exits currently supported:

Note Load Exit:

This exit is called when a note is first loaded into the NixNote editor. When a new note is selected, it is called again and so forth. If a note is refreshed (for example if a note’s source has changed through the editor) it is *not* called again.

The script you write will have access to a variable called “note”. It has the following methods available:

- log(*string*) → write a string to the NixNote log. This is useful for debugging.
- setTitle(*string*) → Change the note’s title.
- getTitle() → Returns a string of the current note’s title.
- isTitleModified() → Returns true if a note’s title has been modified by the exit.
- setContents(*string)* → Change a note’s body to a specific string. PLEASE BE CAREFUL. Evernote is very picky about their note format. Also be careful when editing a note that has an image or attachment. There are very specific things that are needed in a note to keep everything together.
- getContents() → Returns a string of the XHTML body of a note.

- getContentsPlainText() → Returns a string of a note's content in plain text. All of the XHTML is removed.
- isContentsModifie() → Returns true if the note's contents have already been modified by the script, otherwise it returns false.
- isContentsDirty() → Returns true if a note needs to be synchronized with Evernote, otherwise it returns false.
- setContentsDirty(*bool*) → If set to true, a note will be synchronized with Evernote the next time a sync is performed. If set to false, a note will not be synchronized. Note: Changes later to this note may reset this vaue. For example, setting this to false in your script, but later changing a note's title will result in this being set as needing synchronization. Also note that setting this to false may result in a note on your local machine being different than a note on Evernote's servers.
- setNotebook(*string*) → Set a notebook to the string passed. Note: A notebook by that name must already exist. If an invalid or unknown notebook is passed the notebook will not be changed.
- getNotebook() → Returns a string of the current notebooks' name.
- isNotebookModified() → Returns true if the script has already modified the note's notebook, otherwise it returns false.
- insertTag(*string*) → Add a tag to a note. The tag must already exist. If it does not exist, the tag will not be assigned.
- removeTag(*string*) → Remove the tag identified by the string from a note.
- getTagAt(*int*) → Return a specific tag from a note. This is used to access individual tag members from the list of tags assigned to a note. The tags do not appear in any specific alphabetic order.
- getTagSize() → Return the number of tags assigned to a note.
- getCreationDate(*string*) → Return the note's creation date in the format requested by the parameter.
- getUpdatedDate(*string*) → Return the date a note was last updated in the format requested by the parameter.
- getSubjectDate(*string*) → Return the note's subject date in the format requested by the parameter.
- setCreationDate(int) → Set the note's creation date. The value passed is the number of miliseconds since January 1, 1970.

- setUpdatedDate(int) → Set the note’s last updated date. The value passed is the number of miliseconds since January 1, 1970.
- setSubjectDate(int) → Set the note’s subject date. The value passed is the number of miliseconds since January 1, 1970.

Note Save Exit:

The note save exit is called every time a note’s *contents* are changed. It is not called when any other attributes of a note are changed. it uses the same properties as the Note Load Exit. A note’s contents are saved when you leave a note, when you do a Ctrl+S (the default save shortcut) or at a user-defined interval. If a note’s contents have not changed, this exit is not called.

One thing to consider with this note is that by modifying a note’s contents the note is reloaded. This causes the note to be repositioned from the top.

Ubuntu Scope

For people using the Unity desktop, there is a scope that can be installed to allow for searching of notes via a scope. The current scope is a python script and can be slow. Since Ubuntu has discontinued Unity, this will program has been discontinued.

Getting Help

I try to provide some level of support for NixNote but it obviously isn't my full time job and this software is provided as-is, so please be patient.

If you encounter a problem running NixNote please use the steps below in providing as much information as possible.

1.) Read this document. Maybe you'll find the answer has already been documented.

2.) Check that you are using the most current version. Sometimes bugs are fixed in a later release and people just need to upgrade.

3.) The preferred contact method to get help is to fill out a ticket on GitHub at the address https://github.com/baumgarr/nixnote2/issues. You can also fill out an issue at https://sourceforge.net/p/nevernote/_list/tickets.

4.) Set the message level to “Debug”. Recreate the problem and send the messages.log in your ~/.nixnote/logs-x directory. I may need a different log setting later, but “Debug” is a good place to start.

5.) Include as much information as possible regarding the problem. The more information you can provide the easier it is to diagnose the problem. Information that may help diagnosing the problem includes:

- If you are using Linux, what distribution are you using?
- What is the release number of the Linux distribution or what version of Windows are you using?
- What window manager are you using (Gnome, KDE, Unity, etc).
- Did you install this via rpm, deb, tar file, or some other method.
- Did this work in the past and just broke, or is it something that never worked? If it worked in the past, in what release did it work?
- Is the problem with a shared/linked notebook that is owned by another user? If you own it, is it shared with other users?
- Is this a problem with one note or is it with multiple notes?
- If this is caused by a specific note (or a specific type of note), do the notes contain a complex mixture of images and/or attachments?

- Try running nixnote2 from the command line to see if there are any odd stack dump messages that happen around the time of the error.

Known Issues

PDF Preview causes NixNote to crash.

On Gnome, for some reason, enabling the PDF Preview feature will cause NixNote to crash. There is no workaround for this issue other than to disable it in the Preferences dialog.

Some shortcuts don't work.

There are typically three reasons shortcuts don't work.

The first reason is that the shortcut isn't setup properly. Look under the Help menu for the "Shortcuts" option. This will display what shortcuts are currently defined. Please make sure the shortcut you want is actually defined correctly.

The second reason is that the window manager is intercepting them. The window manager will see any key combination first and could be stopping it before it ever gets to NixNote. There isn't anything NixNote can do about that and the only option is to change the shortcut in NixNote or the window manager.

Finally, there is a problem with Unity and NixNote when it is complied against Qt5. As far as I know there isn't anything NixNote can do about it. The only available options are to recompile it against Qt4 or to switch window managers.

NixNote has issues when opening the dialog box to enter my Evernote username & password.

On some window managers NixNote will not display the window to enter your Evernote userid & password. The way around it is to set the environment variable "QT_STYLE_OVERRIDE=gtk2" (without the quotes). This will change the appearance of NixNote, but it should allow you to enter your Evernote userid & password. After entering this information and getting a token from Evernote, you do not need to set this again.

The tray icon doesn't appear or NixNote vanishes when minimize to tray or close to tray options ore specified.

There could be a couple of issues when this happens, but basically not all window managers handle the tray icon in a way that Qt can handle. Sometimes building it from the source on your system will correct the problem and, if you are using Qt5, using Qt4 will correct the issue. Unfortunately there is very little that NixNote can do about the problem.

Frequently Asked Questions

What Evernote features does NixNote support?

NixNote supports most of the same features that Evernote's Windows and Macintosh clients support. The major exceptions are the lack of ability to edit ink notes, only a subset of the search syntax is supported, and it doesn't have the ability to post to Facebook, LinkedIn, or Twitter.

I start NixNote, but nothing happens. What is wrong with this program? Is it broken?

Most likely you are missing some type of dependency. Open a terminal window and issue the command “nixnote2”. Look for any console messages about missing or incompatible libraries.

Can I do a client-to-client synchronization or synchronize without going through Evernote?

No. NixNote was designed to be a client to Evernote. It currently doesn't have the ability to synchronize directly with other clients. You can, however, do a backup of your database and restore it to another client. It isn't a true synchronization but it can reduce the amount of network activity needed when you start NixNote on a machine for the first time.

Is Evernote giving anything?

Evernote has been extremely supportive in NixNote's development. Any question I've asked has been answered and they've been very helpful with things like encryption and ink notes.

Why don't you support creating ink notes?

Evernote doesn't provide an API for ink notes. In order to support creating ink notes I'd need to reverse engineer their existing file format. That simply isn't something I have the time or ability to do at this point. The only reason read-only ink notes are supported at all is because it pulls the image from the web site. Even Evernote has limited ink note support. Only their Windows clients support creating ink notes. Other clients are read-only just like NixNote.

What about NixNote 1.x?

NixNote 1.x had some limitations that were too difficult to overcome without doing a complete rewrite of the program. Initially I resisted this because starting over was a massive task, however as time wore on it became more obvious that something needed to be done. Some of the features of NixNote 1.x required older SSL libraries that were difficult to find, the libraries were becoming increasingly outdated and causing issues with newer versions of Java, and the simple fact that it was never initially designed to be a full client. Eventually the only choice available was to rewrite it.

The old NixNote 1.x ran on macOS and Windows. Will there be a Windows/Mac version of NixNote 2?

There may some day be a NixNote 2 for Windows if I ever get the time to port it, but there will probably never be a NixNote 2 for macOS because I don't own a Mac.

Can you implement feature ___?

Maybe. The best thing to do is to open a request at GitHub https://github.com/baumgarr/nixnote2/issues. I don't promise anything but many of the current features are the results of people asking. Things like auto-import folders, LaTeX formulas, and viewing ink notes are all things that were added by request.

Can it run on a thumb drive?

Technically there isn't anything preventing it from running on a thumb drive. You'd need to setup your .nixnote directory to point to the thumb drive via a symbolic link and it should work. I would think it would be difficult, however, to make sure the dependencies are installed on any computer you happen to use.

Is there a way to set global hot key so I can press Ctrl-F from anywhere and bring up NixNote?

Yes, kindof. There isn’t a way to for NixNote to set a global hot key that works on all (or even most) window managers. There is, however, a crude workaround. If your window manager allows user-defined global keys you can set that key sequence to issue a command to NixNote via a command line option. Look at the “signalGui” command line option to specify what command to issue. Please note that NixNote must be running for the signalGUI to work.

How can I help?

There are several ways to help. If you enjoy programming and like looking through awful, poorly documented, difficult to understand code then you can always help improve it by helping to develop it. If coding isn't something you enjoy (or you looked at how it is written and were too frightened to continue) you can contribute by identifying bugs or offering suggestions for improvement, or helping translate it to other languages.

Can I contribute money?

If you like NixNote enough to pay for it, please take whatever amount you feel it is worth and donate it to your favorite charity. If you think about it, you can send me an email and let me know.

Development

This section is for anyone who wants to know how NixNote works at a programming level. The goal of this section is not to document every class and method, but to give a general overview of how NixNote works.

I use Qt Creator and the project files are all on GitHub. To download everything, do a “git clone https://github.com/baumgarr/nixnote2 nixnote” and a directory called “nixnote” will be created (of course, you'll need to have git installed before you can clone the project). Simply open up Qt Creator and open the .pro file. After the .pro files is opened and the dependencies are installed, you should be able to build the project. You'll need to execute qmake to prepare the project for building.

Other distributions might have different package names than the ones listed below, but these lists should give you a good idea of what is needed to build NixNote on other Linux distributions.

This is elementary, but be aware that an X window system will need to be present for NixNote to run properly.

Setting up a programming environment in Debian/Ubuntu

These are the packages needed for building NixNote using any Debian/Ubuntu based distribution. Use the “sudo apt-get install <package_name>” command to manually install them. You do not need both Qt4 & Qt5 libraries. You just need to install for the version of Qt you are building against.

- **git-core** → Repository tool to manage changes.
- **qtcreator** → IDE environment.
- **qt4-dev-tools** → Qt4 development tools.
- **qt-sdk** → QT development kit (for Qt4 builds).
- **libqt5webkit-dev** → Webkit development files (for Qt5 builds).
- **libpoppler-qt4-dev** → Poppler tools for PDF support (for Qt4 builds).
- **libpoppler-qt5-dev** → Poppler tools for PDF support (for Qt5 builds).
- **libopencv-dev** → OpenCV libraries used for webcam support.
- **libhunspell-dev** → Hunspell libraries used for spell checking.
- **libboost-dev** → Boost C++ libraries.
- **libboost-test-dev** → Boost C++ libraries.
- **libboost-program-options-dev** → Boost C++ libraries.
- **libevent-dev** → A library to execute a function when a specific event occurs on a file descriptor.
- **automake** → Tool used for makefiles.
- **libtool** → A shared library tool for developers.
- **flex** → The Fast Lexical Analyzer.
- **bison** → A general-purpose (yacc-compatible) parser generator.
- **pkg-config** → A package configuration system that manages compile/link flags.
- **g++** → C++ compiler.
- **libssl-dev** → SSL libraries for secure communications.
- **tidy** → Utility used to convert HTML to XML.

- **libtidy-dev** → Tidy development libraries
- **libcurl4-openssl-dev** → Text based web browser used to download ink note images.
- **cups** → (not required to build NixNote, but required for note printing support)

Setting up a programming environment in Gentoo

These are the packages needed for building NixNote for any Gentoo-based distribution. Use the “emerge -uDNav <package_atom>” to manually install each development dependency:

- **dev-vcs/git** → Flag to enable or disable options for prebuilt (GRP) packages (eg. due to licensing issues). Used for NixNote's version control.
- **dev-qt/qt-creator** → Flag to enable or disable options for prebuilt (GRP) packages (eg. due to licensing issues) used for NixNote development.
- **app-text/poppler** → A PDF rendering library. Needed for exporting to PDF.
- **media-libs/opencv** → A collection of algorithms and sample code for various computer vision problems.
- **app-text/hunspell** → A spell checker.
- **dev-libs/boost** → Boost libraries for C++.
- **dev-libs/libevent** → A library to execute a function when a specific event occurs on a file descriptor.
- **sys-devel/automake** → Used to generate Makefile.in from Makefile.am
- **sys-devel/libtool** → A shared library tool for developers.
- **sys-devel/flex** → The Fast Lexical Analyzer.
- **sys-devel/bison** → A general-purpose (yacc-compatible) parser generator.
- **dev-util/pkgconfig** → A package configuration system that manages compile/link flags.
- **sys-devel/gcc** → The GNU Compiler Collection.
- **dev-libs/openssl** → A full-strength general purpose cryptography library (including SSL and TLS).

- **app-text/htmltidy** → Tidy the layout and correct errors in HTML and XML documents.
- **net-print/cups** → Not required to build NixNote, but required for note printing support.

Setting up a programming environment in Fedora

These settings were tried under Fedora 22. Other versions of Fedora may need different packages or different settings. To install the packages, use “dnf install <packagename>” as root to manually install each package.

- **qt5-base-devel** → base Qt5 libraries (for Qt5 builds).
- **qt-devel** → Qt development libraries (for Qt4 builds).
- **qt5-devel** → Qt development libraries (for Qt5 builds).
- **qtwebkit-devel** → Qt Webkit development libraries (for Qt4 builds).
- **qt5-qtwebkit-devel** → Qt Webkit development libraries (for Qt5 builds).
- **poppler-qt-devel** → Poppler libraries for PDF support (for :Qt4 builds).
- **poppler-qt5-devel** → Poppler libraries for PDF support (for Qt5 builds).
- **hunspell-devel** → hunspell development libraries for spell checking.
- **boost-devel** → Development libraries for Boost C++ libraries.
- **opencv-devel** → Development libraries used for webcam support.
- **libtool** → Developer shared library tools.
- **flex** → The Fast Lexical Analyzer.
- **libcurl-devel** → curl developmentlibraries used for downloading ink note images.
- **bison** → A general-purpose (yacc-compatible) parser generator.
- **tidy** → Utility used to convert HTML to XML.
- **tidyp** → Tidy libraries
- **tidy-devel** → Tidy development libraries
- **automake** → Utility to build NixNote from source.
- **gcc-c++** → C++ compiler.

- **cups** → CUPS printing support. This is only needed if you wish to print.
- **rpm-build** → Build RPM packages. If you don't plan on building an RPM you can ignore this.

If you wish to have printing support, CUPS will need to be installed.

General Notes on Building NixNote

On some systems (like Fedora) I had to setup a symbolc link for libhunspell-1.3.so to libhunspell.so for the library to be found. You could also modify the build to include this specific version.

One other thing to be aware of is that on systems that have Qt5 installed, the “qmake” command is named “qmake-qt4” to avoid conflicts with Qt5.

After doing a “qmake” or “qmake-qt4” you need to run a “make” to compile the actual program. This can take several minutes depending upon the speed of your machine, so please be patient. When the complie is done you should find an executable “nixnote2” in your git directory and you can run it right there.

Please note that there is no “make install”. If you wish to have it installed and setup for all users, the easiest thing to do is to go into the package_scripts directory. From there, you can do a “./build.sh” and the script will walk you through creating package file. Once the package file is created, you can move it to another location and install it normally.

If you are using the tar.gz file, there is an “install.sh” included that does the installation. It is expecting you to run as root and it does **NOT** do any checking for other packages.

Programming Language

NixNote is written in C++ using the Qt libraries. There is a Java module that is used for encryption & decryption of notes, but this is not built as part of the normal qmake build. NixNote will work without this module, but you cannot encrypt or decrypt notes without it.x

Database

NixNote uses a sqlite database for everything except attachments and thumbnails. The database is found in the ~/.nixnote/db-<account> directory. For most people, you will only have one account so it will be ~/.nixnote/db-1. Attachments are stored in ~/.nixnote/db-<account>/dba directory. Thumbnails are stored in the ~/.nixnote/db-<account>/tdba directory. Automatic notes imported via the command line are stored in the ~/.nixnote/db-<account>/dbi directory. The format of these notes should be in valid XML format and are deleted when they are imported into the normal database.

If you have multiple accounts, they will each have their own ~/.nixnote/db-<account> directory. When an account is deleted the database and all attachments are removed.

Evernote data

Each note, resource, saved search, notebook and tag have a global unique identifier (GUID) assigned to them and they have a sequence count. The GUID is unique among all Evernote accounts, so there should not be any duplicates. This GUID is used to identify these resources across different Evernote clients.

Each NixNote entry has a LID number. This is a simple sequential number that is assigned when the item is created. The number is NOT unique among different accounts or databases and only exists within a single client. Using it removes the need to update a large number of GUIDs when a note is synchronized.

The synchronization number indicates the current version of the note. When a note is updated on Evernote's servers it gets a higher sequence number. This sequence number is assigned by Evernote (not NixNote). A sequence number is unique within a users account, so no two GUIDs will have the same sequence number. A note which has a sequence number of 1, when synchronized, may next have a sequence number of 243.

This sequence number is used to indicate the position at which you last synchronized. When starting a sync, you pass the highest number you've ever received and it will give you data starting from that point.

Threads

NixNote has several threads of its own that it uses for processing.

The first thread to start is the main (or NixNote) thread. This is the master thread that controls most other threads. There is a restriction that no Widgets (pretty much anything GUI) can exist outside the main thread. This means that any fields that are presented to the user and any updates the user make must pass through this main thread. Doing something foolish in this thread can really hurt performance.

This main thread spawns several other threads:

- SyncRunner thread
- IndexRunner thread
- CounterRunner threads

Communication between the threads is done via the Signal API in Qt. There are a few boolean primitives that are shared, but these are used mostly to interrupt the

child threads (typically done during a shutdown). All threads should remain active for the life of the program.

SyncRunner thread and synchronization processing

The SyncRunner thread handles most of the interaction with Evernote. It is the process that deals with synchronizing any notes and resolving any conflicts. Most of the time this thread is asleep and will wake up when the synchronization timer has expired, or when the user clicks the synchronization button to do a manual sync. If the SyncRunner encounters a problem it will post an error message back to the main thread and disconnect from Evernote.

When the SyncRunner thread is running, a global flag should be set to indicate a sync is in progress. This will prevent notes from being indexed or thumbnails from being created. The logic is that they shouldn't waste CPU doing work for a note that may change in a few moments.

The SyncRunner thread goes through some “interesting” logic to handle everything. This is a rough overview of the flow.

The first thing it does is it tries to validate your authentication token with Evernote. Assuming that is successful, it continues. If it isn't successful it is done and the thread waits for the next request. If you don't have an authentication token it will open a connection to Evernote prompting you for permission to access your account.

The next thing it does is to get the User's account information. This is used to determine if a user is a premium member, their Evernote email address, and their quota usage.

The next information it retrieves is from Evernote is the sync status of your account. This will give you the highest sequence number on your account that Evernote is aware of. If this count is higher than yours then you need to get data from Evernote. If this number is the same as yours then you don't need to download any data.

The sync status also contains a date. This date is a “drop dead” date. If the last time you synchronized is before this date, then Evernote no longer has all the individual changes you need to synchronize. If you are in this state, you will need to do a full synchronization.

If you need to get information from Evernote, the first thing you tell it is the highest sequence number you received in the past. This tells Evernote where to begin sending data. Evernote sends data in “chunks”.

A chunk of data contains any items (notes, resources, tags, notebooks, and saved searches, etc...) which have changed. It contains records indicating deleted records a well as new and changed records. A chunk also contains the highest sequence

number in that chunk. This sequence number is saved so we know where to begin the next time we get data.

Notebooks, saved searches, and tags are pretty straight forward. Any change from Evernote will be added or updated. If there is a conflict, Evernote wins. For example, if you renamed a notebook on the web interface and you edited it on NixNote, the Evernote name will win and you'll lose the NixNote change.

Note changes are a bit different, since we need to deal with potential conflicts.

If a note hasn't been changed locally, but it was changed remotely then we just update the note in the database, flag it to be reindexed, and move to the next note.

If the same note has been changed locally and on Evernote, then we have a conflict. In this case, the note that was changed in NixNote is given a new GUID by NixNote. It is then moved to a local notebook called “Conflicts”. If a “Conflicts” notebook doesn't exist, it creates one and moves the note into it. It is then the users responsibility to decide what to do with the conflicts. If the user simply moves the note from the conflicts notebook back into a synchronized notebook, it is treated as a new note and will get a new GUID.

The next part of a sync is to look at what has changed locally. When a note, notebook, saved search, or tag is changed NixNote flags that note as “dirty”. Dirty items are items that need to be synchronized.

NixNote goes through its database looking for any records which are flagged as “dirty”. Those records are then sent to Evernote. When the note is sent, Evernote will send back a new sequence number for that record. That sequence number is saved back with the note. If it is a new record, the GUID will change as well, so any references to that GUID need to be updated.

The tricky part about sending changes to Evernote are tags. Before a new child tag can be created, its parent tag needs to exist at Evernote.

For instance, if I were to create TAG1 and TAG2 and set TAG2 as a child to TAG1, neither tag has a valid Evernote GUID so, when I synchronize, I need to be sure to synchronize TAG1 first, get the new GUID, update the parent GUID for TAG2, then finally synchronize TAG2. If there is an error with TAG1 and it can't be synchronized, then TAG2 will not be synchronized either. Most of this is accomplished with lists to keep track of what tags are ready to be synchronized.

Finally, the last thing the synchronization process does is to synchronize linked notebooks. This is essentially the same as synchronizing your own account as described above, except you go against the notebook owner's shard. The other major difference is that changes to tags and notebooks in linked notebooks are not synchronized (you can't update another person's notebooks or tags). This is an Evernote restriction. NixNote will let you change them, but that change is only on

that system and it isn't synchronized. Tags and notebooks are stored a little differently in the database to denote that they are linked instead of tags you created.

Each linked notebook is a different sync. If you have 10 linked notebooks, the sync process is run against each notebook individually. The initial sync of a linked notebook seems to be very slow. It looks like it sends the data in chunks (the same as synchronizing your account the first time), but it will send empty chunks if you are not authorized for any data in that particular chunk. For example, if I can receive a chunk that contains nothing.

IndexRunner Thread

Please note, that by default this thread is no longer used. It can be enabled via the command line option “--enableIndexing”. The primary reason it is disabled is due to the fact it was causing excessive amounts of CPU and (for some reason) causing occasional sync errors. Enable it at your own risk.

When enabled, NixNote maintains a separate database for searching. Adding words to this database is handled by the IndexRunner thread. It also permits the entire database to be re-indexed in the background.

The first thing the IndexRunner will do is to determine if any notes need to be indexed and it will parse them and add individual words to the database. Note resources are not done at the same time as the contents of a note.

Once all notes are done it will index note resources (attachments and OCR data). The OCR data is received from Evernote and is in XML format. Once the OCR data is added it will then handle indexing their attachments.

CounterRunner Threads

The notebook, tag, and trash lists in the GUI maintain a count of notes. The actual counting, however, is done in a separate thread to improve user performance.

Once a count is complete, the total is passed back to the main thread target via signals. If a thread is counting and another count request comes in, the existing count is canceled and the counting is restarted. For example, if it is counting and you switch notebooks a new count is ordered and the existing count is aborted. This is done to prevent meaningless counts and improve user responses.

Program Directory Hierarchy

The NixNote GIT repository contains the following hierarchy:

certs → Digital certificates used to authenticate Evernote.

cmdtools → Classes used to support command line options.

communication → Classes used for communication with Evernote. Typically (but not always) called from the syncrunner thread.

dialog → popup dialog boxes.

qevercloud → Clasess used for Evernote Thrift communication.

filters → Classes used to filter database records from GUI classes.

gui → Most of the classes that provide visual elements to the users.

html → Classes used to format notes from HTML to Evernote's ENML markup data and vice versa.

images → Icons, PNGs, and other picture elements.

java → Java classes used to encrypt and decrypt text (RC2 only).

logger → Handles messages to the application log file and the user's log file.

man → Unix man files.

models → Classes used for Qt's Model/View structure.

oauth → Classes used to popup the initial Evernote window granting an authorization token (Obsolete).

package_scripts → Scripts used to create the rpm, deb, and tar files.

qss → Style sheets for Qt's theming support.

reminders → Classes used to support reminders within notes.

settings → Classes used to maintain program and user settings. IT also helps maintain where the different directories used by Evernote are.

spell → Contains a user's custom dictionary for hunspell.

sql → Most of the classes that are used to maintain database records. There are a few other locations that access the database directly, but most of the DB I/O happens here.

threads → Subthreads syncrunner, counterrunner, and indexrunner.

translations → Qt Linguist language translations.

utilities → miscellaneous classes used for various utilities.

watcher → Classes that deal with auto-import of notes.

webcam → Classes that deal with webcam notes and interfaces.

xml → Classes used for nnex and enex imports and exports.

Command Line Options (for developers)

NixNote allows for various command line options. These options are handled primarily by classes in the cmdtools directory.

Because I cannot guarantee that the database NixNote uses won't be corrupted if multiple programs access it simultaneously, NixNote does provides command line support in two different ways. Which method NixNote uses is dependent upon if another Nixnote is running. It uses the shared memory segment to determine if another copy is running.

Command line logic if no other NixNote is running

This is the most straightforward method. If there is no other NixNote running we don't need to worry about corrupting the database. As stated above, the shared memory segment is used to determine if another NixNote is running. If no other Nixnote is running NixNote simply does what the user requested and exits.

Command line logic if another NixNote is running

This situation is more complex, because the GUI NixNote is using the database and I don't want to risk corruption by accessing it via another program.

To get around this situation, NixNote uses a few different methods to communicate with the other instance of NixNote. For clarification purposes, we'll refer to the other NixNote as the GUI NixNote and the command line NixNote as the non-GUI NixNote.

When the non-GUI NixNote is started and finds a GUI NixNote running, it will typically pass t he request to the GUI NixNote via the shared memory segment and (depending upon the request) wait for a response.

For these types of requests, this is the sequence of events:

1. The non-GUI NixNote determines that another GUI NixNote is running.
2. The non-GUI NixNote builds a request. If a response is requested, then it creates a second shared memory segment for the response. The key for the new shared memory segment is included in the request to the GUI NixNote so it can reply with the return code.
3. After building the request, the non-GUI NixNote will write the request to the shared memory segment.
4. The GUI NixNote will constantly check the shared memory segment for a request. If one is found, it reads it and clears the shared memory segment (so it doesn't do the same action twice).

5. If a reply is requested, the GUI-NixNote will attach to the shared memory segment and write the reply. After writing the reply it detaches from the non-GUI's shared memory segment.

6. The non-GUI segment will wait for a few seconds for a reply. If none is received it reports the error and exists. If a response is received it displays the response to the user.

Doing this type of back-and-forth communication works well for small requests, but there are some requests which may contain larger amounts of data, so alternate methods are used.

For adding a new note, the process works in much the same manor as above but there are a few changes.

1. The non-GUI NixNote does not use the shared memory segment to initiate the note's creation by the GUI NixNote. It uses the database dbi directory instead. The file written in this directory is an XML file detailing the note to be created. This file also includes a shared memory key to use when responding to the note creation request.

2. The GUI NixNote notices the file was created and it reads the XML file. The XML file is used to create the note. After creating the note, the GUI-NixNote will write the newly created note's ID to the respons shared memory segment.

3. The non-GUI NixNote will listen on the response shared memory segment for a few seconds. If a response is received it display's the newly created note's ID to the user. If a response is not received it displays an error message.

For doing a query, the process is similar except in reverse.

1. The non-GUI Nixnote writes the query in the shared memory segment the same way as other requests are done.

2. The GUI NixNote reads the request from the shared memory segment.

3. The GUI NixNote then does the query, but it writes the respons to a file in the user's tmp directory. It then uses the response shared memory segment to notify the not-GUI segment that the query is complete.

4. The non-GUI NixNote waits for the response from the GUI-NixNote. If none is received it displays an error. If One is received it uses the response shared memory segment to find the file containing the response and displays it to the user.