Newznab Documentation

Release 0.2.3-dev

Newznab

Jun 19, 2017

Contents

Below you'll find a collection of useful bits of information about setting up and running your own Newznab install. Please be sure to read through the overview, install and FAQs very carefully.

CHAPTER 1

Overview

About

Newznab+ is a PHP/Smarty application, which supports the indexing of usenet headers into a MySQL database and provides a simple web based search interface onto the data.

It includes simple CMS facilities, SEO friendly URLs and is designed with the intention of allowing users to create a community around their index.

For information on how to install, please refer to INSTALL.txt

To discuss visit us on IRC at `irc.synirc.net #newznab`, or use the web client.

Newznab+ is licensed under a non redistributable license. For details, please refer to *LICENCE.txt*. An open source (GPLv3) edition of Newznab is freely available for those wishing to create derivative works.

How it Works

Usenet groups are specified, message headers (binaries and parts) are downloaded for the groups which match regex, releases are created from completed sets of binaries by applying *regex* to the message subject. Releases are categorised by *regex*ing the message subject. Metadata from *TvRage*, *TMDb*, *Rotten Tomatoes*, *IMDb* and *Amazon* are applied to each created release.

Choosing Newsgroups

Groups can be manually entered if you know the name. Groups can also be bulk added when specified as a regular expression. For example if you want to index the groups `alt.bin.blah.*` and `alt.bin.other` use the value `alt.bin.blah.*|alt.bin.other`.

Updating Index (populating binaries + parts)

The recommended way to schedule updates is via the dos and unix start scripts in `/path/to/newznab/misc/update_scripts/`. Make sure you set the paths correctly for your installation. If you are running on unix environment, there is an experimental multithreaded update binaries script named `update_binaries_threaded.php` and `backfill_threaded.php`. You can alter the number of threads used by editing the `maxChildren` setting. For more information on updating and backfilling, see the docs for *update scripts*.

Categorization

Most categorization of releases is done at the time of applying the *regex*. However if no category is supplied for a *regex* then `\www\lib\category.php` contains the logic which attempts to map a release to a site category. Site categories are used to make browsing NZBs easier. Add new categories by updating the category table, and adding a new `Category` constant, then map it in the function `determineCategory`.

Missing Parts

When headers are requested from the usenet provider, they are asked for in number ranges e.g. 1-1000, 1001-2000 etc. For various reasons sometimes the provider does not return a header, this is not always because the header does not exist, there may be some synchronization going on at the providers end. If a header is requested but not returned, we store a record of this in the table `partrepair`. Each time *update_binaries* is run an attempt is made to go back and get the missing parts. If after five attempts the parts can still not be obtained, Newznab gives up. When *update_releases* runs, if a release is seen to have missing parts it will not be released until four hours after it was uploaded to usenet. this is so a chance has been made to repair all missing parts. After four hours a release will be created anyway and its down to the quality of the *PAR* files to determine whether a release can be correctly unpacked.

Backfilling Groups

Since most usenet providers have 800+ days of retention indexing all that information in one shot is not practical. Newznab provides a backfill feature that allow you to index past articles once your initial index has been built. To use the feature first set the back fill days setting in the group(s) to be backfilled to the number of day you wish to go back, making sure to set it higher than the number of days listed in the first post column. Once set run the `backfill.php` script in `misc/update_scripts`. Groups can be backfilled to a particular date using the script `misc/update_scripts/backfill_date.php` using the syntax:

```
php backfill_date.php 2011-05-15 alt.binaries.groupname.here
```

You can use the `_threaded` version of this script if on linux.

For more information on backfilling, see *Update Scripts*.

Regex Matching

Releases are created by applying *regex*s to binary message subjects. Different *regex*s are applied to binaries from different newsgroups. Catchall *regex*s are applied to any binaries left unmatched after the group specific matching. A category can be associated with a *regex*, which will allow the processing of groups like `a.b.inner-sanctum` which contain a combination of different binary types.

Regex Details

Regexes are used to parse the subject header to create definable release names. There are two named capturing groups used for this, 'name' and 'parts'.

The (?P<name>) capturing group is used to define the final release name as well as the text that the binaries are grouped on. It is required to use this named capturing group in the *regex*.

The (?P<parts>) capturing group defines the total number of parts needed in order to make a release. Most posters include the total number of binaries in the subject header however some do not. When the (?P<parts>) capturing group is omitted from the *regex*, newznab will wait 4 hours after the postdate of the last binary before making them into a release to ensure the final release is complete. This capturing group is optional.

Regex Updating

*Regex*s in the system with IDs in the range 0-10000 are system defined and are updated centrally and are retrieved from Newznab's server. Every time `processReleases` is run, a check will be performed to see if you have the latest *regex*s. If you do not want this check to be made then set `site.latestregexurl` to `null`.

NZB File Storage

*NZB*s are saved to disk, in a compressed gzip format, at the location specified by `site.nzbpath` in subdirectories based on the first character of the release guid; this just makes the directories a bit easier to manage when you have thousands of `nzb.gz` files. The default path is `/website/../nzbfiles`.

Spotnab

Spotnab allows systems to share information between each other based on discovery and approval of found sources. The implementation is based loosely on how spotweb works. With Spotnab you can fetch comments (and potentially other information) from other newznab sources and apply them to your own comment section. Fetched content is scanned and decrypted using a password that the newznab server who posted it chooses to share with you. Since all newznab servers choose to share or not, Spotnab will only populate your database with comments from the sources who choose to share they're comments with you.

Additionally you can post comments; send all the comments made by your local users of your site to usenet encrypted by your own secure private key. Only those you share your public key will be allowed to decrypt it for their own server.

SSL Usenet Connection

Install the OpenSSL extension, set in `config.php`:

```
define ('NNTP_SSLENABLED', true);
```

Importing & Exporting NZBs

NZB files can be imported from the admin interface (or *cli*). Importing is a convenient way to fill the index without trawling a large backdated number of usenet messages. After running an import the `processReleases` function must be run to post process releases. *NZB*s can also be exported based on system categories.

Import script lives in /misc/update_scripts/import.php Usage: php import.php [path(string)] [usefilename(true/false)] [dupecheck(true/false)] [movefiles(true/false)] [overridecategory(number)]

Google Ads/Analytics

To integrate Google Analytics and AdSense, provide the AdSense ad module IDs into the site table for the `searchbox` (bottom of menu). Providing an Analytics ID will include the Analytics javascript in the footer.

Admin

Admin functions all live under the URL `/admin/` which is only accessible by users with admin role. Set `users.role` to be 2 on the users you wish to be admins.

TvRage/TVDB

After `processReleases` is called, an attempt is made to determine the *TvRage* IDs for every release which looks like its TV. This also works out the series/episode columns. The data in the *TvRage* table will become populated from best guesses from the *TvRage* search API. If some of these guesses are wrong, you can manually edit the *TvRage* data in the admin interface and use the remove link to wipe any releases which have that `rageid` and then manually call 'process tv' which will attempt to relink rage data. When a new release is created it goes in with `release.rageid = -1` when TV is processed, the `rageid` either goes to the best guess, or to -2, which indicates that either no match could be made or the release isn't perceived to be TV.

NFO

*NFO*s are attempted to be retrieved using a queuing method. There will be a number of attempts to get an *NFO* before giving up.

Caching

Caching of queries results to aid performance and is supported by using memcache or file. In the `config.php` file edit the `CACHEOPT_METHOD` constant to either `memcache` or `file`. You can additionally configure the memcache server/port address. There is a default caching TTL of 15 minutes, which when enabled, is applied to queries in the main browsing lists.

IMDb, TMDb and Rotten Tomatoes

If enabled, and if an *IMDb* ID is found in the *NFO*, the application will attempt to use that *IMDb* ID to get general data about the movie (title, year, genre, covers, etc.) from themoviedb.org. If no entry is available from *TMDb* then an attempt to gather the info from `imdb.com` is made. Any results are stored in the `moveinfo` table, with covers/backdrops being saved to the `images/covers/` directory.

3rd Party API Keys

In order to do lookups to *TMDb*, *Rotten Tomatoes* and *Amazon*, API keys are used. Newznab ships with some default keys, but due to the restrictions on use of APIs, it is strongly suggested you go and get your own API keys for each service and save them using the site edit page.

Content/CMS

Pages can be added to the site with SEO friendly URLs via the `/admin/`.

Skinning & Themes

Avoid custom edits to code and stylesheets to make updating painless.

Override any styles by creating a folder `\www\templates\<yourtheme>\`. Stick any custom images, views, scripts or styles in `\www\templates\<yourtheme>\images\`. Then pick the theme in the `admin/site-edit` page. Your styles and pages will override the existing default pages.

Web API

www.sitename.com/api? provides API access to query and retrieve NZBs. Call www.sitename.com/apihelp to see help doc with all available options. Users either have to be logged in or provide their `rsstoken`. Users can use their `rsstoken` to access both rss + api. Full details of the API and how to implement it are provided in the *Web API* docs.

Debugging

Switch `php.ini` `error_reporting` to `E_ALL` and ensure logging to browser is enabled.

Development

Here is a brief overview of the location of various Newznab components. For more detailed information, see the appropriate sections in the docs.

`\db\schema.sql` The latest database schema. You should be able to rerun in and create new blank schema.

`\db\patch\` Database upgrade patch files. If you update from svn you will need to apply all patches since last update.

\db\cache If file based caching is enabled the cache objects are stored here.

\misc Used for general docs and useful info, nothing in here is referenced by the application.

\misc\update_scripts Shell, batch scripts and php files to call the updating of index from cli

\nzbfiles Default folder for all gzipped NZBs to be stored.

\www\install Installer files.

\www\lib\framework A few general classes for db/http code.

\www\lib\smarty Copy of a fairly recent Smarty lib, used for template rendering.

\www\lib All classes used in the app, typically named same as its database entity

\www\covers All covers downloaded for releases.

\www\pages Controllers for every frontend page in the system.

\www\admin All php pages used by the admin.

\www\templates\default\views\admin All templates used by the admin pages.

\www\templates\default\views\frontend All templates used by the user pages.

\www\templates\<yourtheme> Blank area for implementation specific UI customizations.

\www\templates\default\scripts Javscript dumping ground.

\www\templates\default\styles Default theme css (don't edit, extend with your own theme).

Hall of Fame

(just some of the) people who've helped along the way:

iota@cyberarmy	regexs, sessions
enstyne@cyberarmy	regexs
fatfecker@newznab	mediainfo, ffmpeg, tv
gizmore@wechall	password, hash
lhbandit@nzbsorg	yenc, nntp, bokko, dev
dryes@nzbsorg	anidb
pleo@newznab	sphinx, mobile, docs
lordgnu@newznab	powerspawn, threading
bb@newznab	dev
keyvan@newznab	backfill
troph@bhw	performance
kevin123@newznab	compression,theming
wafflehouse@newznab	compression,db
jayhawk@nzbsu	testing, icons
midgetspy@sickbeard	rage, api
dogzipp@dognzb	dev
andrew@newznab	inno
zdefect@newznab	anidb
ueland@newznab	installer
ensi@ensisoft	api
hecks@tvnzb	rar api
michael@newznab	dev

Continued on next page

Table 1.1 – continued from previous page

l2g@newznab	nfo,spotnab
danza@newznab	dev
sakarias@newznab	testing
pairdime@sabnzbd	jquery, css
pmow@sabnzbd	headers, backfill
poutine@newznab	recaptcha
robv@newznab	init.d
bigdave@newznab	testing
duz@sabnzbd	yenc
inpheaux@sabnzbd	design, nzb
spooge@newznab	testing
sy@newznab	testing, regexs, amazon
magegminds@newznab	lighttpd rewrite rules
trizz@newznab	lighttpd rewrite rules
emanon@newznab	testing
fubaarr@newznab	testing
mobiKalw@newznab	testing
crudehung@newznab	nginx rewrite rules
f0rmed@newznab	testing
frikish@github	theme
www.famfamfam.com	icons
wally73@newznab	dev

CHAPTER 2

Install

Prerequisites

- **PHP**
  - php version 5.4 or higher
  - sessions enabled
  - memory limit at 1GB or more
  - minimum execution time of 60+ seconds
  - make sure you update the `php.ini` for both web and cli
  - php `register_globals` off
  - php5-curl library installed
  - php_openssl library installed (if connecting to SSL usenet provider or using spotnab)
- GD Imaging Library w/PHP integration
- PEAR
- **MySQL**
  - `max_allowed_packet = 12582912`
  - `group_concat_max_len = 8192`
  - timezone set to php's
- **Apache**
  - script timeout of at least 60 seconds
  - mod_rewrite enabled
  - AllowOverride All set in conf to allow `.htaccess` to specify rewrite rules
- **3rd Party API Keys (recommended to get your own api keys)**

    - TMDb (signup @ http://api.themoviedb.org/2.1/)
    - Amazon (signup @ https://affiliate-program.amazon.com/gp/advertising/api/detail/main.html)
- **Deep RAR Password Inspection**
    - unrar version 3.9 or higher
- **Thumbnailing and Media Info (if deep rar inspection is enabled)**
    - ffmpeg
    - mediainfo

Installation

1. Unzip/svn checkout the code into a folder in your web directory. do NOT move directories around.
2. Create your website/vhost in apache and point to the /www/ directory.
3. Browse to http://yournewznabserver/install.
4. Once installed activate only one or two groups to test with first, this will save you time if it is not working correctly.
5. Run the update_binaries.php and update_releases.php scripts in miscupdate_scriptsvia command-line.
6. If updating was successful then you can continue to setup your site and configure the update scripts for auto-updating. windows users use the script `/misc/update_scripts/win_scripts/runme.bat` linux users use the `/misc/update_scripts/nix_scripts/newznab_screen.sh`

Updating

1. Run `svn update` or in windows (using tortoisesvn) right mouse button the folder and click `SVN Update`
2. Run any new patches since your last update by executing the script `/misc/update_scripts/update_database_version.php`
3. Delete contents of cached smarty files in `/www/lib/smarty/templates_c/*`

Running Outside of Web Root

Set /.htaccess RewriteBase to your virtual directory

Sample Apache VirtualHost File

Add this to your existing VHOST file modifying your values for ServerName, Server Alias, and DocumentRoot. You should find this under /etc/apache2/sites-enabled/ (000-default):

```
<VirtualHost *:80>
    <Directory /var/www/newznab/htdocs/www/>
        Options FollowSymLinks
        AllowOverride All
        Order allow,deny
```

```
        allow from all
    </Directory>

    ServerAdmin admin@example.com
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot /var/www/newznab/htdocs/www
    LogLevel warn
    ServerSignature Off
</VirtualHost>
```

CHAPTER 3

Guides

Memcached with Newznab Server

This guide will help you install memcached on your new server (assuming you've just followed the Ubuntu 14.04.2 guide).

Note: This assumes you know linux in general, and know how to use CLI.

Guide written by Stifler. Find me on synIRC #newznab

Version 0.1 - Initial Release

Version 0.2 - Added memcache install and confiure;

Step 1: Install Memcached

Memcached is in the Ubuntu repository and is up to date.:

```
sudo apt-get install php5-memcached memcached php5-dev
```

Easy!!! You now have memcached. Now it should have automagically started and also restarted your Apache2, but in case:

```
sudo service memcached start
```

Step 2: Lets get some monitoring

INSTALL

Lets install `phpMemcacheAdmin` so we can monitor memcached. It can be found here : http://blog.elijaa.org/

At the time of writing version 1.2.2 was the latest, so in terminal:

```
cd /tmp
wget http://phpmemcacheadmin.googlecode.com/files/phpMemcachedAdmin-1.2.2-r262.tar.gz
cd /var/www
sudo mkdir phpmemcacheadmin
cd phpmemcacheadmin
sudo tar -zxvf /tmp/phpMemcachedAdmin-1.2.2-r262.tar.gz
```

Note: replace `phpMemcachedAdmin-1.2.2-r262.tar.gz` with whatever the current version is.

Serving the page

Time to get apache2 to serve the new webpage.:

```
sudo nano /etc/apache2/sites-available/phpmemcacheadmin.conf
```

Now paste this into the file:

```
<VirtualHost *:12323>
                ServerName   phpmemcacheadmin
                UseCanonicalName Off
                ServerAdmin  "webmaster@example.com"
                DocumentRoot "/var/www/phpmemcacheadmin"
                CustomLog  /var/log/phpmemcacheadmin-access_log common
                ErrorLog   /var/log/phpmemcacheadmin-error_log
</VirtualHost>
```

Save the file.

Now we need to listen on the port 12323:

```
sudo nano /etc/apache2/ports.conf
```

Find the line `Listen 80` and add `Listen 12323` under it. Then save the file.

Now we have to enable the site:

```
sudo a2ensite phpmemcacheadmin
```

Finally, we need to enable the phpmemcacheadmin some write permissions:

```
cd /var/www/phpmemcacheadmin/
sudo chmod 777 Config
```

Finally, restart apache so the page is served:

```
sudo service apache2 restart
```

Now you should be able to browse to `http://yourserver:12323` and see the monitoring page.

Step 3: Memcache extension for PHP

Now we need to install and setup the memcache extension. First we have to install it:

```
sudo pecl install memcache
```

That will build the extension and install it in `/var/lib/php5`

Next we need to enable the extension in php.ini.

Note: There are two php.ini files, edit both of them and make the same changes to both files.

```
sudo nano /etc/php5/cli/php.ini
```

and:

```
sudo nano /etc/php5/apache2/php.ini
```

Find these are labelled **Dynamic Extensions** and add the following line:

```
extension=memcache.so
```

Save the file. Once you have changed both files, you can now restart apache2.

Note: Stop your screen script(s) first.

```
sudo service apache2 restart
```

Now you can restart your screen scripts.

Step 4: Enable Memcached and configuration

Firstly we need to enable newznab to use memcache:

```
sudo nano /var/www/newznab/www/config.php
```

Find `define('CACHEOPT_METHOD', 'none');` and change it to:

```
define('CACHEOPT_METHOD', 'memcache');
```

Save the file and restart apache2:

```
sudo service apache2 restart
```

If you need to tweak the settings for memcached, they are located in this file `/etc/memcached.conf`

Done!!

Newznab on Ubuntu 11.10

This guide covers installing Newznab+ on a fresh copy of Ubuntu 11.10 64-bit server and assumes a basic knowledge of using Linux via the command line.

TODO: Install and configure Memcache?

Install Ubuntu

Grab a copy of Ubuntu from their website and install it. Once the install is complete, log in, fire up a terminal window, do a full upgrade and reboot (entering your password when prompted):

```
sudo apt-get update
sudo apt-get -y upgrade
sudo apt-get -y dist-upgrade
sudo reboot
```

You'll probably want to install SSH if you don't have it already:

```
sudo apt-get install -y ssh
```

Additionally, we're going to be compiling some code from source, so we'll need the tools to do so:

```
sudo apt-get install build-essential checkinstall
```

Install Newznab

Once Ubuntu is installed and updated, we can begin installing Newznab. Before we get started though, we need to decide where to install Newznab. Typically, web-related stuff goes in `/var/www/`, so lets put Newznab in `/var/www/newznab`. Let's create the directory and set it writeable by our user for now:

```
sudo mkdir -p /var/www/newznab
sudo chmod 777 /var/www/newznab
```

Now we can begin installing the prerequisites.

Prerequisites

Newznab as a few dependencies, so let's start with installing the required software. According to the install docs, Newznab needs the following software:

1. PHP 5.2+
2. GD Imaging Library w/PHP integration
3. PEAR
4. Database: MySQL (or Percona)
5. Web Server: Apache (or Nginx)

Its worth mentioning a few things at this point. The first is that the default version of MySQL that comes with Ubuntu is kind of old (version 5.1). Therefore, we'll also cover installing a higher-performance version (Percona version 5.5) instead. Additionally, Apache is listed as a requirement, however it is possible to use a different web server instead. Therefore, we'll go over how to use the blazing-fast Nginx as well for those who don't want to use Apache.

The following software is optional, but we'll also cover installing it and setting it up:

1. Unrar
2. FFmpeg
3. Lame
4. MediaInfo

5. Sphinx

By the end of this guide we'll have a fully working Newznab+ install with all optional components working as well.

PHP

Let's start by installing PHP and required PHP extensions:

```
sudo apt-get install -y php5 php5-dev php-pear php5-gd php5-mysql php5-curl
```

The install docs also say that PHP needs a few tweaks. Using your favorite text-editor (I'm using `nano`) open up `php.ini`:

```
sudo nano /etc/php5/cli/php.ini
```

Scroll down and look for `register_globals`, make sure it is set to:

```
register_globals = Off
```

Next, look for `max_execution_time`, and set it to (2 minutes):

```
max_execution_time = 120
```

Next, look for `memory_limit`, and either set it to `1024M`, or `-1` for no limit:

```
memory_limit = 1024M
```

While you're here, you should also set PHP's `date.timezone` (a list of available timezone settings can be found here):

```
date.timezone = Europe/London
```

That should do it for PHP for the time being (we'll come back to it when we configure the web server).

Database

You have a few options here, we'll cover two:

1. Use MySQL 5.1
2. Use Percona 5.5

I'd recommend that you use Percona, but the choice is yours. Pick one and follow *one* of the sections below (**do not** install MySQL and Percona).

MySQL 5.1

If you decided to use MySQL, simply install it via `aptitude`:

```
sudo apt-get install mysql-server-5.1 mysql-client-5.1 libmysqlclient-dev
```

Percona 5.5

Installing Percona requires a little extra work upfront, but in the long run it is generally worth it as Percona is a high-tuned fork of MySQL. The first step is to add Percona into aptitude:

```
gpg --keyserver  hkp://keys.gnupg.net --recv-keys 1C4CBDCDCD2EFD2A
gpg -a --export CD2EFD2A | sudo apt-key add -
```

That retrieves and installs the GPG keys needed to add Percona's aptitude repository. Now we need to tell aptitude about the repositories:

```
sudo sh -c "echo  \"\n#Percona\" >> /etc/apt/sources.list"
sudo sh -c "echo  \"deb http://repo.percona.com/apt lenny main\" >> /etc/apt/sources.
↪list"
sudo sh -c "echo  \"deb-src http://repo.percona.com/apt lenny main\" >> /etc/apt/
↪sources.list"
```

Now update and install it:

```
sudo apt-get update
sudo apt-get install -y percona-server-client-5.5 \
                        percona-server-server-5.5 \
                        libmysqlclient-dev
```

It'll ask for a password to use for the MySQL root user, so pick one and remember it.

Web Server

OK, now for the web server. Once again, you have multiple choices and we'll cover two:

1. Apache
2. Nginx

Apache is "easier" and generally more flexible, but it also tends to be a system hog. So, if you can't make up your mind it is probably best to take the easy route and install Apache. If you feel like squeezing the most out of your machine and you are a little more skilled with configuring software, Nginx might be a good choice.

Again, pick one and follow the steps below (don't install both).

Apache

First install it:

```
sudo apt-get install -y apache2
```

You'll also need to configure the `php.ini` for Apache, so follow the section above about configuring PHP (from above), but this time, edit the file `/etc/php5/apache2/php.ini`:

```
sudo nano /etc/php5/apache2/php.ini
```

Let's create the site configuration file for Apache. Open up a new file:

```
sudo nano /etc/apache2/sites-available/newznab
```

Here is a template you can use for this file. You should change settings where appropriate for your setup:

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    ServerName localhost    # You might want to change this

    # These paths should be fine
    DocumentRoot /var/www/newznab/www
```

```
    ErrorLog /var/log/apache2/error.log
    LogLevel warn
</VirtualHost>
```

Now we need to disable the default Apache settings, enable the one we just created for Newznab and enable `modrewrite`:

```
sudo a2dissite default
sudo a2ensite newznab
sudo a2enmod rewrite
sudo service apache2 restart
```

Nginx

First install it:

```
sudo apt-get install -y nginx
```

Now, for Nginx there are multiple ways to serve PHP files. Probably the best is `php-fpm`, which is basically a daemon that runs and serves PHP to Nginx. So, let's go ahead and install that now:

```
sudo apt-get install -y php5-fpm
```

You'll also need to configure the `php.ini` for FPM, so follow the section above about configuring PHP (from above), but this time, edit the file `/etc/php5/fpm/php.ini`:

```
sudo nano /etc/php5/fpm/php.ini
```

Just for good measure, restart the daemon:

```
sudo /etc/init.d/php5-fpm restart
```

Let's create the configuration file for Nginx. Open a new file:

```
sudo nano /etc/nginx/sites-available/newznab
```

Here is a template you can use for this file. You should change settings where appropriate for your setup:

```
server {
    # Change these settings to match your machine
    listen 80 default_server;
    server_name localhost;

    # Everything below here doesn't need to be changed
    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    root /var/www/newznab/www/;
    index index.html index.htm index.php;

    location ~* \.(?:ico|css|js|gif|inc|txt|gz|xml|png|jpe?g) {
            expires max;
            add_header Pragma public;
            add_header Cache-Control "public, must-revalidate, proxy-revalidate";
    }

    location / { try_files $uri $uri/ @rewrites; }
```

```
    location @rewrites {
            rewrite ^/([^/\.]+)/([^/]+)/([^/]+)/? /index.php?page=$1&id=$2&subpage=$3␣
↪last;
            rewrite ^/([^/\.]+)/([^/]+)/?$ /index.php?page=$1&id=$2 last;
            rewrite ^/([^/\.]+)/?$ /index.php?page=$1 last;
    }

    location /admin { }
    location /install { }

    location ~ \.php$ {
            include /etc/nginx/fastcgi_params;
            fastcgi_pass 127.0.0.1:9000;

            # The next two lines should go in your fastcgi_params
            fastcgi_index index.php;
            fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    }
}
```

Let's make sure that the Nginx logs directory exists and is writeable:

```
sudo mkdir -p /var/log/nginx
sudo chmod 755 /var/log/nginx
```

Now, let's disable the `default` Nginx site handler and enable our `newznab` configuration:

```
sudo unlink /etc/nginx/sites-enabled/default
sudo ln -s /etc/nginx/sites-available/newznab /etc/nginx/sites-enabled/newznab
```

Finally, let's restart Nginx:

```
sudo service nginx restart
```

Extras

unrar

Unrar is used to extract files from releases. Installing unrar is trivial:

```
sudo apt-get install -y unrar
```

FFmpeg

Once again, the version of FFmpeg in Ubuntu's repository is old and crusty, so lets build a newer one and also add x264 support. Start by grabbing the source code we need:

```
wget http://ffmpeg.org/releases/ffmpeg-0.8.7.tar.gz
wget ftp://ftp.videolan.org/pub/videolan/x264/snapshots/last_stable_x264.tar.bz2
sudo apt-get install -y libfaac-dev libjack-jackd2-dev \
                        libmp3lame-dev libopencore-amrnb-dev \
                        libopencore-amrwb-dev libsdl1.2-dev libtheora-dev \
                        libva-dev libvdpau-dev libvorbis-dev libx11-dev \
```

```
                        libxfixes-dev texi2html yasm zlib1g-dev \
                        libdirac-dev libxvidcore-dev
```

Let's build `x264` first:

```
mkdir x264
tar --strip-components=1 -jxf last_stable_x264.tar.bz2 -C x264
cd x264
./configure --enable-static
make
sudo checkinstall --pkgname=x264 --pkgversion="3:$(./version.sh | \
                  awk -F'[" ]' '/POINT/{print $4$5}')" \
                  --backup=no --deldoc=yes --fstrans=no --default
```

Now, onto `FFmpeg`:

```
cd ../
tar xvfz ffmpeg-0.8.7.tar.gz
cd ffmpeg-0.8.7/
./configure --enable-gpl --enable-libfaac --enable-libmp3lame \
            --enable-libopencore-amrnb --enable-libopencore-amrwb \
            --enable-libtheora --enable-libvorbis --enable-libx264 \
            --enable-nonfree --enable-postproc --enable-version3 \
            --enable-x11grab --enable-libdirac --enable-libxvid
make
sudo checkinstall --pkgname=ffmpeg --pkgversion="5:0.8.7" --backup=no \
                  --deldoc=yes --fstrans=no --default
hash x264 ffmpeg ffplay ffprobe
```

Lame

Lame is used for processing audio samples:

```
sudo apt-get install -y lame
```

MediaInfo

MediaInfo is used to gain information about various types of media files found in releases. In order to install it we simply need to add the MediaInfo `aptitude` repository:

```
sudo apt-get install -y python-software-properties
sudo add-apt-repository ppa:shiki/mediainfo
sudo apt-get update
sudo apt-get install -y mediainfo
```

Sphinx

Sphinx is used for full-text searching. It is insanely fast and if you really want your Newznab+ install to fly, it is highly recommended. Unfortunately the version in Ubuntu's `aptitude` repository is horribly old, so we'll need to build a newer version.

Let's download the source and extract it:

```
wget http://sphinxsearch.com/files/sphinx-2.0.2-beta.tar.gz
tar xvfz sphinx-2.0.2-beta.tar.gz
cd sphinx-2.0.2-beta
```

Download and extract `libstemmer_c`:

```
wet http://snowball.tartarus.org/dist/libstemmer_c.tgz
tar --strip-components=1 -zxf libstemmer_c.tgz -C libstemmer_c
```

Configure it:

```
./configure --prefix=/usr/local --with-libstemmer
```

Now we're ready to compile Sphinx. For this step you can speed up the compilation on a multi-core system. If you have a 4-core system, for example, you can do (replace `j4` with the number of cores your machine has):

```
make -j4
```

Once that finally finished, install it:

```
sudo checkinstall --pkgname=sphinx --pkgversion="2.0.2-beta" --backup=no \
                  --deldoc=yes --fstrans=no --default
```

Now we have a nice new version of Sphinx installed in `/usr/local`. Binaries are installed in `/usr/local/bin`.

Newznab Source

Finally, we can now begin installing Newznab! We'll be grabbing the latest and greatest version, so we'll need Subversion installed first:

```
sudo apt-get install -y subversion
```

Now we can check out the Newznab code:

```
svn co svn://svn.newznab.com/nn/branches/nnplus /var/www/newznab
```

At this point we might as well set the permissions on a couple of directories as well:

```
sudo chmod 777 /var/www/newznab/www/lib/smarty/templates_c
sudo chmod 777 /var/www/newznab/www/covers/movies
sudo chmod 777 /var/www/newznab/www/covers/anime
sudo chmod 777 /var/www/newznab/www/covers/music
sudo chmod 777 /var/www/newznab/www
sudo chmod 777 /var/www/newznab/www/install
sudo chmod 777 /var/www/newznab/nzbfiles/
```

Configure Newznab

Now that Newznab is installed, we need to configure it. Configuration is done via a web browser.

Run Installer

It is now time to configure Newznab. This is done via a web-based installer. Open up `http://localhost/install` in a web browser (or whatever the address/IP address is of your server) and follow the guided steps.

Enable Groups

Head over to `/admin/group-list.php` in your web browser and pick some groups to index by clicking "activate" on a few groups.

Set Paths and Options

Make sure to set your `newznab ID`.

We need to let Newznab know where all the extra software is that we installed earlier, so head over to `/admin/site-edit.php` in your browser, scroll down to the "3rd Party Application Paths" section and update the fields:

Unrar Path `/usr/bin/unrar`

Mediainfo Path `/usr/bin/mediainfo`

Ffmpeg Path `/usr/local/bin/ffmpeg`

Lame Path `/usr/bin/lame`

If you'd like to enable audio previews, check `Save audio preview`. It is worthwhile to do rar checking, so set `Check For Passworded Releases` to `Deep`.

That's it for configuration right now, but don't close your browser yet as we'll be coming back to the configuration page when configuring Sphinx.

Indexing

Back at the command line its time to fire up the binaries and releases update scripts:

```
cd /var/www/newznab/misc/update_scripts
```

The `update_scripts` folder contains a lot of scripts. The most important ones are `update_binaries.php` and `update_releases.php`. If you have any experience with `screen` or `tmux`, it is highly recommended that you use one of these to run the update scripts, as it will allow you to monitor the update process, observe and resolve issues; this is especially important for newcomers to Newznab. With that said, Newznab also ships with an init-sytle script that can be installed to make Newznab run more or less as a daemon that will start and stop with startup and shutdown, respectively.

Screen or tmux

If you want to go the `screen` or `tmux` route, you'll need to pick one and install it:

```
# Install screen...
sudo apt-get install -y screen

# ...or tmux
sudo apt-get install -y tmux
```

In the `nix_scripts` directory there is a useful script called `newznab_screen.sh` that runs `update_binaries.php` and `update_releases.php`, in addition to a few other scripts, continuously and automatically. First, we need to modify it however, so lets change dir and make a copy:

```
cd /var/www/newznab/misc/update_scripts/nix_scripts
cp newznab_screen.sh newznab_screen_local.sh
```

Now open `newznab_screen_local.sh` in a text editor and modify `NEWZNAB_PATH` near the top to point to our installation path:

```
nano newznab_screen_local.sh
```

Set NEWZNAB_PATH:

```
export NEWZNAB_PATH="/var/www/newznab/misc/update_scripts"
```

Now we can run the script via screen:

```
screen sh newznab_screen_local.sh
```

You should see the script download headers for the groups that you have enabled and then run various stages that will attempt to group and catalogue the headers. For now, just leave the script running and detach from screen by typing cntl a d.

Sphinx

As mentioned previously, Sphinx is a fast full-text indexer. By default, it is disabled in Newznab, so go ahead and enable it by visiting /admin/site-edit.php and setting "Use Sphinx" to "Yes". While there, you'll also notice that there are a few other configuration options for Sphinx. By default, Sphinx will index all of the release information, however, there are three other optional indexes: NZB contents, NFO contents and release files. Enabling these optional indexes will add increased processing time, so you will likely want to experiment to see what combination works best for your hardware. For now, you don't have to enable any of the optional indexes.

To get Sphinx running, we need to generate a sphinx.conf file. To do this we'll use the nnindexer.php script in misc/sphinx:

```
cd /var/www/newznab/misc/sphinx
./nnindexer generate
```

The script will print out the location of the sphinx.conf file, which by default will be /var/www/newznab/sphinx/sphinx.conf. This path needs to be entered into the "Sphinx Configuration Path" setting located at /admin/site-edit.php.

Now we need to start the search daemon and create the indexes and restart the daemon:

```
./nnindexer.php daemon
./nnindexer.php index full all
./nnindexer.php index delta all
./nnindexer.php daemon --stop
./nnindexer.php daemon
```

Summary of Installed Software

- PHP v5.3.6
- Pear v1.9.2
- MySQL v5.1.58 or
- Percona v5.5.17
- Apache v2.2.20 or
- Nginx 1.0.5
- FFmpeg v0.8.7

- MediaInfo v0.7.50
- Lame v3.98.4
- unrar v4.00-beta3
- Sphinx 2.0.2-beta

Newznab on Ubuntu 14.04.2 x64

This guide will help you install everything you need to get Newznab+ running on a fresh Ubuntu server. It will run through all the requirements for intalling: PHP, Apache2, phpmyadmin, Percona, unrar, Lame, Mediainfo, subversion, screen, tmux, Sphinxsearch, ffmpeg and of course Newznab+ itself.

It will then go on to help you get Newznab+ basically configured and background scripts running.

Note: This assumes you know linux in general, and know how to use CLI.

Note: This is a step-by-step guide. I am assuming that you are not tinkering with locations of installation et cetera. If you decide to install Newznab+ elsewhere or put unrar in your home directory, there will be other changes you will have to make.

Guide written by Stifler. Find me on synIRC #newznab

Version 0.1 - Initial Release

Version 0.2 - fixed typos; fixed incorrect character encoding; added install nano; fixed subversion install; added key_buffer change

Version 0.3 - sphinxsearch installation change;

Version 0.4 - removed references to tokudb

Step 1: Install Ubuntu 14.04.2

Download the iso image of Ubuntu Server 14.04.2 amd64.

On the start page (after you select your language) hit F4 and select "install a minimal system" (or if you are doing it in a VM, "Install a minimal virtual machine") and hit Enter.

Hit Enter again to start the install.

During the install choose region, keyboard, hostname, user name, timezone, partitions, etc and go get a beverage.

Note: When it comes to asking what software to install, select "OpenSSH server" only.

Step 2: Update Ubuntu

Login into your new server and run:

```
sudo apt-get -y update
sudo apt-get -y upgrade
sudo apt-get -y dist-upgrade
```

The Next step is optional, it will update the kernel to version 4.0.

```
cd /tmp/
wget http://kernel.ubuntu.com/~kernel-ppa/mainline/v4.0-vivid/linux-headers-4.0.0-
↪040000_4.0.0-040000.201504121935_all.deb
wget http://kernel.ubuntu.com/~kernel-ppa/mainline/v4.0-vivid/linux-headers-4.0.0-
↪040000-generic_4.0.0-040000.201504121935_amd64.deb
wget http://kernel.ubuntu.com/~kernel-ppa/mainline/v4.0-vivid/linux-image-4.0.0-
↪040000-generic_4.0.0-040000.201504121935_amd64.deb
sudo dpkg -i linux-headers-4.0.0-*.deb linux-image-4.0.0-*.deb
sudo reboot
```

At this point you have a basic Ubuntu x64 Linux kernel 4.0.0 with SSH. Which if you haven't already, can start using for the rest of the process if you need to.

Step 3: Install the necessities for NN+

Install php5 and some additional modules for it. At time of writing version 5.5.9

```
sudo apt-get install -y php5 php5-dev php-pear php5-gd php5-mysql php5-curl
```

Install Percona. At time of writing version 5.6

```
sudo apt-key adv --keyserver keys.gnupg.net --recv-keys 1C4CBDCDCD2EFD2A
    sudo gpg -a --export CD2EFD2A | sudo apt-key add -
sudo sh -c "echo \"deb http://repo.percona.com/apt trusty main \ndeb-src http://repo.
↪percona.com/apt trusty main\" >> /etc/apt/sources.list"
sudo apt-get -y update
sudo apt-get install percona-server-server-5.6 percona-server-client-5.6
```

Note: You will be asked to **create** a password for the DB server root user.

Install apache2

```
sudo apt-get install -y apache2
```

Install unrar

```
sudo apt-get install -y unrar
```

Install Lame

```
sudo apt-get install -y lame
```

Install MediaInfo

```
sudo apt-get install -y mediainfo
```

Install Subversion (SVN)

```
sudo apt-get install -y subversion
```

Install Screen and tmux

```
sudo apt-get install -y screen tmux
```

Install Sphinx search

Note: OMG! sphinxsearch is so far behind in Ubuntu!! Notice I even had to goto Precise to get one :-P

Lets get a daily build instead.:

```
sudo gpg --keyserver  hkp://keys.gnupg.net --recv-keys B9D8946B16932B16
sudo gpg -a --export 16932B16 | sudo apt-key add -
sudo sh -c "echo \"deb http://ppa.launchpad.net/builds/sphinxsearch-daily/ubuntu␣
↪precise main \ndeb-src http://ppa.launchpad.net/builds/sphinxsearch-daily/ubuntu␣
↪precise main\" >> /etc/apt/sources.list"
sudo apt-get -y update
sudo apt-get install -y sphinxsearch
```

Install nano

```
sudo apt-get install -y nano
```

Install ffmpeg

You have two options, either follow this https://trac.ffmpeg.org/wiki/CompilationGuide/Ubuntu or better yet, do this:

Copy the script from here: https://gist.github.com/xdamman/e4f713c8cd1a389a5917

Create a new script file:

```
sudo nano ~/install_ffmpeg_ubuntu.sh
```

Paste the script in and save the file. Then make it executable:

```
sudo chmod 777 ~/install_ffmpeg_ubuntu.sh
```

And run it:

```
sudo ~/install_ffmpeg_ubuntu.sh
```

Whichever way you choose, you will be spending a lot of time drinking the beverage of your choice waiting for ffmpeg...

Install phpmyadmin

```
sudo apt-get install -y phpmyadmin
```

Now let's improve phpmyadmin security a bit:

```
sudo a2disconf phpmyadmin
sudo nano /etc/apache2/ports.conf
```

Add `Listen 12322` directly under `Listen 80`. This is where we will make phpmyadmin available. Save the file. Now we will create the new site phpmyadmin:

```
sudo nano /etc/apache2/sites-available/phpmyadmin.conf
```

Add this to the new file:

```
<VirtualHost *:12322>
    ServerName phpmyadmin
    DocumentRoot /var/www/phpmyadmin
</VirtualHost>
```

Save the file.

Create a symlink to phpmyadmin:

```
sudo ln -s /usr/share/phpmyadmin /var/www/phpmyadmin
```

Enable the site:

```
sudo a2ensite phpmyadmin
sudo service apache2 restart
```

You will now be able to access phpmyadmin at `http://yourserver:12322/`

Step 4: Modifying various settings to how we want.

Editing php configuration (php.ini)

Note: There are two php.ini files, edit both of them and make the same changes to both files.

```
sudo nano /etc/php5/cli/php.ini
```

and:

```
sudo nano /etc/php5/apache2/php.ini
```

Find these settings (under [PHP] section) and check/add/modify as necessary:

```
ADD: register_globals = Off
MODIFY: max_execution_time = 120
CONFIRM: memory_limit = -1
```

Next set date.timezone (under [Date] section)

Remove the ';' from the start of the line. For example:

```
date.timezone = Europe/London
```

Note: The timezone can be whatever, but it will need to be set the same as the database later.

Creating the NN+ Website

```
sudo nano /etc/apache2/sites-available/newznab.conf
```

Add this to the new file:

```
<VirtualHost *:80>
        <Directory /var/www/newznab/www/>
            Options FollowSymLinks
            AllowOverride All
            Order allow,deny
            allow from all
        </Directory>
    ServerAdmin admin@example.com
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot /var/www/newznab/www
    LogLevel warn
    ServerSignature Off
</VirtualHost>
```

Save the file.

Now to enable the new site, disable the default site and enable modrewrite.

```
sudo service apache2 stop
sudo a2dissite 000-default.conf
sudo a2ensite newznab
sudo a2enmod rewrite
sudo service apache2 start
```

Editing Percona configuration (mysql.cnf)

```
sudo nano /etc/mysql/my.cnf
```

Find these settings (under [mysqld] section) and check/add/modify as necessary:

```
CONFIRM: max_allowed_packet = 16M
ADD: group_concat_max_len = 8192
ADD: default_time_zone='Europe/London'
```

```
    ADD: sql_mode = ''
MODIFY: key_buffer = 16M to key_buffer_size = 16M
```

Save the file.

Note: `key_buffer` is deprecated, and now becomes `key_buffer_size`

Note: `sql_mode` removing only_full_group_by which became default in mysql 5.7

Note: **BEFORE** you restart mysql import the timezones from Ubuntu into Percona database.

Run the following:

```
mysql_tzinfo_to_sql /usr/share/zoneinfo | mysql -u root -p mysql
```

Note: This will ask for your database password.

Now restart the database service:

```
sudo service mysql restart
```

Note: The timezone can be whatever, but it will need to be set the same as PHP. See the PHP section prior.

Step 5: The one you have been waiting for! Installing Newznab.

```
cd /var/www
sudo mkdir newznab
sudo chmod 777 newznab
sudo svn co svn://svn.newznab.com/nn/branches/nnplus /var/www/newznab
```

Note: You will need your username and password from newznab to access the svn. When you run this command first time, it will ask you for the password of the user you are logged in with. Ignore that and just hit Enter, it will then ask for your svn username.

```
sudo chmod 777 /var/www/newznab/www/lib/smarty/templates_c
sudo chmod 777 /var/www/newznab/www/covers/movies
sudo chmod 777 /var/www/newznab/www/covers/anime
sudo chmod 777 /var/www/newznab/www/covers/music
sudo chmod 777 /var/www/newznab/www
sudo chmod 777 /var/www/newznab/www/install
sudo chmod 777 /var/www/newznab/nzbfiles/
```

Once you have completed all that you should now be able to login at `http://yourserver/`

Step 6: Make Newznab+ do what you want it to do.

Here we are going to setup some basics to get Newznab+ running. There are many more things can can and will need to be done. This is not going to cover everything, but you will be running after this.

Configuration from the home page.

Login to your server `http://yourserver/`

```
Click on "Go to step one: Pre flight check"
Confirm that the preflight checklist is all **OK**. If not, you will need to fix
↪those problems first.
Click on "Go to step two: Set up the database"
Enter your database username ``root`` and your password. Leave the other settings as
↪is.
Click on "Setup Database"
Click on "Step three: Setup news server connection"
Enter your details for your news server.
Click on "Test Connection"
Click on "Step four: Cache Settings"
Leave Caching Type on "None"
Click on "Save configuration file"
Click on "Step five: Setup admin user"
Enter the details you want for the administrator of Newznab+
Click on "Create Admin User"
Click on "Step Seven: Set NZB File Path"
Leave the location as ``/var/www/newznab/nzbfiles/``
Click on "Set NZB File Path"
```

Done! Click on the link to the admin page. Leave the browser open.

Configure Sphinx

Log into your Newznab server shell.

First generate a config for sphinxsearch.:

```
cd /var/www/newznab/misc/sphinx
sudo ./nnindexer.php generate
```

Next we need create the indexes.:

```
sudo ./nnindexer.php daemon
sudo ./nnindexer.php index full all
sudo ./nnindexer.php index delta all
sudo ./nnindexer.php daemon --stop
sudo ./nnindexer.php daemon
```

Configure Site Settings

Back in your browser at the `Admin Home`. Click on the “hide this welcome message” so you can see the `Status Alerts`. There will probably be a couple of them.

Note: Most obvious and important is the one that is telling you that your Newznab ID is missing.

Now click on `Edit Site` and we will set a couple of basic items up.

Scroll to the bottom of `Main Site Settings, HTML Layout, Tags` and you will see `newznab ID:`. Paste your ID in there.

Scroll down to `3rd Party Application Paths` and change the following:

```
Unrar Path to    : /usr/bin/unrar
Mediainfo Path to: /usr/bin/mediainfo
ffmpeg Path to   : /usr/local/bin/ffmpeg
Lame Path to     : /usr/bin/lame
```

Now scroll down further to `Password Settings` and change `Check For Passworded Releases:` to `Deep (requires unrar)`

Now scroll further to `Sphinx Settings` and change `Use Sphinx` to `Yes`. Then change `Index ReleaseFiles` to `Yes`.

Now scroll all the way to the bottom and hit `Save Site Settings`.

Configure Groups

Under the Admin Functions click on `View` in `View Add BulkAdd Groups`. Click on `Activate` for a couple of the groups you want to index.

You will probably want to add more groups later and you can do that by clicking on `Add` or `BulkAdd` under Admin Functions, `View Add BulkAdd Groups`.

Step 7: Running the scripts.

To run the scripts we need to do it in either `screen` or `tmux`. The choice is yours, if you dont know either then I suggest screen first for its simplicity. I'll use `screen` for this guide.

Go back to your shell login and change to ''nix_scripts" directory. We will create our own script based on the default and modify it:

```
cd /var/www/newznab/misc/update_scripts/nix_scripts
sudo cp newznab_screen.sh mynewznab.sh
sudo chmod 744 mynewznab.sh
sudo nano mynewznab.sh
```

Now modify the `export NEWZNAB_PATH` to:

```
export NEWZNAB_PATH="/var/www/newznab/misc/update_scripts"
```

Save the file and start screen.:

```
screen -S newznab
```

Run the script.:

```
sudo ./mynewznab.sh
```

You should now see the script start to run. To exit the `screen` and leave the script running press [CTRL]-[a] and then [d].

At anytime you wish to go back into the `screen` to see what is happening run:

```
screen -r newznab
```

Newznab on Ubuntu 16.04

This guide will help you install everything you need to get Newznab+ running on a fresh Ubuntu 16.04 server. At the end are optional sections for installing sphinx search and to configure scripts to allow newznab to continually run.

Step 1: Install required packages

Enter the following:

```
sudo -s
apt update && apt upgrade
apt install ssh screen apache2 php mariadb-server libapache2-mod-php -y
apt install php-pear php-gd php-mysql php-curl php-json unrar lame mediainfo␣
↪subversion ffmpeg memcached -y
```

Step 2: Configure maria-mysql

Enter the following:

```
mysql_secure_installation
```

Note: You will be asked to **create** a password for the DB server root user.

Create a mysql user for the application:

```
mysql -u root -p
<enter password>
CREATE USER 'newznab'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON newznab.* TO 'newznab'@'%' WITH GRANT OPTION;
exit
```

Edit the mysql configuration file as follows:

```
pico /etc/mysql/conf.d/mysql.cnf
```

Add a new section as follows:

```
[mysqld]
group_concat_max_len=8192
innodb_flush_log_at_trx_commit = 2
```

Step 3: Create web directories

Enter the following:

```
mkdir /var/www/newznab
mkdir /var/www/newznab/htdocs
mkdir /var/www/newznab/logs
```

Step 4: Create a script for keeping newznab up to date

Enter the following:

```
pico /var/www/newznab/svn.sh
```

Paste the following into the file, **entering the username and password provided after purchasing newznab** inplace of the asterix:

```
svn export --no-auth-cache --force --username **** --password **** svn://svn.newznab.
↪com/nn/branches/nnplus /var/www/newznab/htdocs/

#rm /var/www/newznab/htdocs/*.txt
#rm -rf /var/www/newznab/htdocs/www/install
#rm -rf /var/www/newznab/htdocs/www/lib/smarty/templates_c/*

cd /var/www/newznab/htdocs/misc/update_scripts
php update_database_version.php
cd /var/www/newznab/htdocs

service memcached restart
service apache2 restart
```

Make the file executable and run it:

```
chmod +x /var/www/newznab/svn.sh
/var/www/newznab/svn.sh 2> /dev/null
```

Step 5: Setup website

Ensure web directories are writeable by the update scripts and installer:

```
chmod 777 /var/www/newznab/htdocs/www/lib/smarty/templates_c
chmod 777 /var/www/newznab/htdocs/www/covers/movies
chmod 777 /var/www/newznab/htdocs/www/covers/anime
chmod 777 /var/www/newznab/htdocs/www/covers/music
chmod 777 /var/www/newznab/htdocs/www/covers/tv
chmod 777 /var/www/newznab/htdocs/www
chmod 777 /var/www/newznab/htdocs/www/install
chmod -R 777 /var/www/newznab/htdocs/nzbfiles/
```

Replace the default apache conf to point to newznab directory:

```
pico /etc/apache2/sites-available/000-default.conf
```

Remove everything from the file and replace with:

```
<VirtualHost *:80>
    <Directory /var/www/newznab/htdocs/www/>
        Options FollowSymLinks
        AllowOverride All
        Order allow,deny
        allow from all
    </Directory>

    DocumentRoot /var/www/newznab/htdocs/www
    ErrorLog /var/www/newznab/logs/error.log
    CustomLog /var/www/newznab/logs/access.log combined
</VirtualHost>
```

Note: A more practical alternative to this is to create a vhost specifically for the newznab installation.

Update a few defaults in the php.ini files:

```
pico /etc/php/7.0/apache2/php.ini
```

Find and edit the following settings:

```
date.timezone = 'Europe/London'
max_execution_time = 120
memory_limit = -1
```

Enable apache mod_rewrite and restart services:

```
a2enmod rewrite
service apache2 restart
service mysql restart
```

Step 5: Install newznab

Open a browser pointing to the IP address of the server and follow the instructions. Use the newznab mysql user created earlier to connect to the database. Follow the installation wizard to the end.

After completing the installation open the svn update script, uncomment the commented out lines and re-run it. This will remove any installation files and be the mechanism you use to update newznab to the latest version:

```
pico /var/www/newznab/svn.sh
<uncomment commented out lines prefixed with #>
```

Save, close and run the file:

```
/var/www/newznab/svn.sh
```

Step 6: Configure newznab

Open a browser, go to the admin home page http://<your server IP>/admin/site-edit.php and set up some paths and config options:

```
default home page : /browse
newznabID : <provided in signup email>
unrar path : /usr/bin/unrar
mediainfo path : /usr/bin/mediainfo
ffmpeg path : /usr/bin/ffmpeg
lame path : /usr/bin/lame
integrated cleanup : yes
save audio preview : yes
check for password : deep
delete passworded releases : yes
```

Activate one group to use as a test such as alt.binaries.teevee http://<your server IP>/admin/group-list.php:

```
Filter for the group and press go
Click the Activate link
```

Step 7: Run update scripts

A couple of scripts are used to populate newznab - update_binaries and update_releases. Now newznab is installed and configured it needs to be populated:

```
cd /var/www/newznab/htdocs/misc/update_scripts
php update_binaries.php
php update_releases.php
```

Complete

newznab is now installed. Navigate to the home page to observe indexed content

Optional Step 1: Configure newznab to run continually unattended inside screen

Take a copy of the run script so it can be edited locally without having your local changes overwritten later:

```
cd /var/www/newznab/htdocs/misc/update_scripts/nix_scripts
cp newznab_screen.sh newznab_local.sh
pico newznab_local.sh
```

Ensure the following variables are set appropriately. Use of the threaded run script is significantly faster:

```
export NEWZNAB_PATH="/var/www/newznab/htdocs/misc/update_scripts"
export NEWZNAB_SLEEP_TIME="30" # in seconds
/usr/bin/php ${NEWZNAB_PATH}/update_binaries_threaded.php
```

Make the script executable:

```
chmod +x /var/www/newznab/htdocs/misc/update_scripts/nix_scripts/newznab_local.sh
```

The run script is best ran in a screen, so you can reattach later and monitor it. Screen launches a new bash shell that you can later reattach to. Name the screen so you know which one to re-attach to:

```
screen -S newznab
cd /var/www/newznab/htdocs/misc/update_scripts/nix_scripts
```

```
./newznab_local.sh
ctrl-a d
```

Optional Step 2: Configure newznab to use sphinxsearch

Open a browse to the admin site-edit page http://<your server IP>/admin/site-edit.php Set the following values:

```
use sphinx : yes
index releasefiles : yes
index predb : yes
```

Open a shell and enter the following:

```
sudo -s
apt install sphinxsearch -y
cd /var/www/newznab/htdocs/misc/sphinx
php nnindexer.php generate
php nnindexer.php index full all
php nnindexer.php index delta all
php nnindexer.php daemon
```

Once the indexes have been built running update_releases will populate them:: cd /var/www/newznab/htdocs/misc/update_scripts php update_releases.php

CHAPTER 4

FAQ

Below is a collection of common questions and errors and their answers and solutions. Please read through this carefully before asking for help.

Authorization rejected from nntp server

Check you have enough available connections not currently in use

White screen instead of web page

This is probably a php error not being displayed to browser or session timed out and 403 being throw.

Database logging

Having issues with performance or want to debug database hits, then define define('DB_LOG', '/tmp/nndblog.log'); to output all ran queries to text file.

Lots of binary headers processed but few releases created

The binary headers downloaded do not match the *regex*s used to create a release. The message headers must follow popular formats in order for releases to be created.

Search and raws earch requests lose page CSS styling

Use the provided *Apache VirtualHost*.

Error: Server did not return article numbers 1234567

This isn't necessarily a bad thing, see section on *missing parts*.

Error: Connection timed out

If you're seeing errors like:

```
Connection timed out. Reconnecting...
Cannot connect to server *******
Already connected, disconnect first!
```

Disabling compressed headers should solve the issue.

Error: Session error during install step1

Set `register_globals` to off in `php.ini`.

Error in TMDb.php with `strstr`

If you're seeing an error like:

```
Warning: Wrong parameter count for ``strstr()`` in ``newznab\www\lib\TMDb.php`` on␣
↪line 354
```

You're most likely using the wrong php version; upgrade to version 5.3+.

Error: `PEAR::isError()`

Error like:

> Strict Standards: Non-static method PEAR::isError() should not be called statically

Set `error_reporting = E_ALL ^ E_STRICT` in `php.ini`.

Error: 502 Bad Gateway

Error at `$cfg->pearCheck = @include('System.php');` solved by adding in *open_basedir* path to pear.

Error: `curl_init()`

Call to undefined function `curl_init()`. Make sure you are using the right `php.ini` file. If you are using WAMP, then the `php.ini` file that Apache uses is in the Apache `/bin` folder (not the `php.ini` in `wamp/php`). The php cli will use the first `php.ini` it can find in the Windows `path` environmental variable. In my case, this was an old version in another php directory I set up. Once I deleted that, it used the version in the `/wamp/php` directory.

Error: "MySQL server has gone away..."

MySQL is dropping your connection. Adjust `max_allowed_packet` in `my.cnf`, which you should have already set according to the *install docs*, and if that doesn't fix it, you can try adjusting `mysql.connect_timeout` in `php.ini`.

Movies with a large number of releases are not listing the releases properly

Read the *install docs* (see `group_concat_max_len` under MySQL).

No previews or media info

Check that you version of unrar > 3.8.

Php cli not seeing that curl is installed on a wamp server

When you use the php cli windows uses php.exe inside the php wamp directory, that php is for web only really. Point Newznab to the `php-win.exe` in the same directory. The only script having problem was `updatereleases.bat` so in that script now looks like `C:\wamp\bin\php\php5.3.5\php-win.exe update_releases.php`

Updating releases is taking forever

If updating releases appears to have frozen at `Stage 8`. You have the header retention set too high, or during a large import have allowed too many parts/binaries to be inserted, and mysql takes a long time to prune the table. It will finish eventually, just be patient.

Error: `Parts failed to insert`

If you're getting the `WARNING: xxx Parts failed to insert` error it is likely that something went a little crazy with your parts table. In order to fix this you need to truncate (empty) your `parts`, `partrepair` and `binaries` tables and "reset" your `groups`.

First, truncate your tables (SQL):

```
TRUNCATE `parts`;
TRUNCATE `partrepair`;
TRUNCATE `binaries`;
```

Now, if the error occurred during normal `update_binaries` operation, then you don't actually need to completely reset your `groups` table and instead you can just reset its knowledge of how to go "forward" (i.e. what to consider as new posts). You can do this like so:

```
UPDATE groups SET last_record = 0;
UPDATE groups SET last_record_postdate = NULL;
```

If the error occurred while backfilling, you'll most likely want to update `first_record` and `first_record_postdate` in the same manner.

Additionally, you'll probably want to set the "Where to start new groups" setting to the appropriate value (like 1 day, for example).

Error: `stream_socket_client` or "Failed to write to socket"

If you see errors like `stream_socket_client` or "Failed to write to socket" while trying to connect to your news server, it is possible that the standard NNTP ports (119 and 563) are being blocked (either by your ISP or a firewall). Try setting your NNTP port to `80` (for non-ssl) or `443` (for ssl), if you're news server supports it (many do).

Error: PHP Warning: `mysql_fetch_assoc()` expects parameter 1 to be resource, boolean given

Corrupted mysql tables. Run `misc\update_scripts\optimise_db.php true` to force an optimise and repair of all mysql tables.

Error: You must have POSIX and PCNTL functions to use PowerSpawn

The threaded update scripts can only be run on Linux systems with posix installed. Recompile php with –enable-pcntl

No releases appear in audio or console view

Check you are not using cover view and using the shared Amazon api key. Get your own key or switch to list view.

Error: "Notice: Trying to get property of non-object in C:xampphtdocsnnpluswwwinstallindex.php on line 50"

Reinstall xampp.

Sphinx not updating Delta index. New releases not visible

There is known issues with Windows. Solution is to change "preopen_indexes" from 1 to 0 in sphinx.conf

Sphinx error - PHP Notice: Undefined index: total-documents in /var/www/newznab/htdocs/www/lib/sphinx.php on line 331

Indexes have failed to be rotated from <indexname>.new.sph to <indexname>.sph. Sphinx expects the old files to be there. Solution is to goto the sphinxdata dir and...

rm releases* touch releases.{spa,spd,sph,spi,spk,spm,spp,sps} touch releases_delta.{spa,spd,sph,spi,spk,spm,spp,sps} then run a full + delta index for releases.

See also http://sphinxsearch.com/forum/view.html?id=9859

Script terminating early when using freebsd

Ensure all the php libraries are included

extension=gd.so extension=session.so extension=mysql.so extension=curl.so extension=xml.so extension=ctype.so extension=openssl.so extension=iconv.so extension=mysqli.so extension=hash.so extension=zlib.so extension=pcntl.so extension=posix.so extension=simplexml.so

If you are seeing this in your error log: Fatal error: Class 'COM' not found

You require this in php.ini:

[PHP_COM_DOTNET] extension=php_com_dotnet.dll

Previously it was compiled as built-in on the Windows build.

PHP Warning: mysqli::mysqli(): Headers and client library minor version mismatch. Headers:50532 Library:50614 in /var/www/newznab/www/lib/framework/db.php on line 15

Fix = apt-get install php5-mysqlnd (note: actual version numbers may vary from install to install)

PHP Fatal error: Call to undefined function gzopen() in /var/www/newznab/www/lib/nzb.php on line 22

Related to known issue http://ubuntuforums.org/showthread.php?t=2217927 Fix = update all references to gzopen to be gzopen64

CHAPTER 5

Settings

Newznab has a large number of variable settings which can be defined via the `Admin` interface.

When you click on `Admin` you will be taken to the `Admin Hangout`. On the left hand side you will see a list of option under the title `Admin Functions`.

This page is a work-in-progress and incomplete

Admin Hangout

The Admin Hangout by default will show you a welcome page. The actual intention of the Admin Hangout is to show you any `Status Alerts`. You can enable this by clicking on `hide this welcome message`.

The status messages can identify several issues:

```
* If there are any database patches that need to be run.
* Bad database version.
* Shared keys being used.
* High binary header retention
* Newznab ID missing.
* Mysql configuration issues.
```

Admin Functions

Home

Will take you home, as defined by the `Default Home Page` in the *Edit Site* page.

Admin Home

Will bring you back to the *Admin Hangout*.

Edit Site

Add Edit Content Page

View Add Menu Items

Edit Categories

View Add BulkAdd Groups

View Add Test Send Regex

View Add Blacklist

View Releases

View Previews

View Add TVRage List

View TheTVDB List

View Add Movie List

View AniDB List

View Music List

View Console List

View Book List

Import Export Nzb's

Optimise Tables

View Comments

View Add Spotnab Sources

View Add Users

View Add Roles

Site Stats Debug

CHAPTER 6

Config

Config variables are stored in the `config.php` file at the root of the `www` directory. Unlike *Settings*, config options cannot be updated via the admin page and instead must be edited directly by modifying the contents of `config.php`.Below you will find detailed information on each option.

Config.php

The file config.php file is located in `newznab/www` looks like this:

```
<?php
//==========================
// Config you must change - updated by installer.
//==========================
define('DB_TYPE', 'mysql');
define('DB_HOST', 'localhost');
define('DB_PORT', 3306);
define('DB_USER', 'username');
define('DB_PASSWORD', 'password');
define('DB_NAME', 'newznab');
define('DB_INNODB', false);
define('DB_PCONNECT', true);
define('DB_ERRORMODE', PDO::ERRMODE_SILENT);
define('NNTP_USERNAME', 'username');
define('NNTP_PASSWORD', 'password');
define('NNTP_SERVER', 'my.newsserver.com');
define('NNTP_PORT', '563');
define('NNTP_SSLENABLED', true);
define('CACHEOPT_METHOD', 'none');
define('CACHEOPT_TTLFAST', '120');
define('CACHEOPT_TTLMEDIUM', '600');
define('CACHEOPT_TTLSLOW', '1800');
define('CACHEOPT_MEMCACHE_SERVER', '127.0.0.1');
define('CACHEOPT_MEMCACHE_PORT', '11211');
// define('EXTERNAL_PROXY_IP', ''); //Internal address of public facing server
```

```
// define('EXTERNAL_HOST_NAME', ''); //The external hostname that should be used
require("automated.config.php");
```

DB_TYPE

Currently the only option for this is `mysql` and is not used.

The default setting for this option is `mysql`.

DB_HOST

The database server which has the Newznab database on it. `localhost` if the database is on the same webserver as Newznab. This can be fully qualified name, IP address, 'localhost' or '127.0.0.1'.

The default setting for this option is `localhost`.

DB_PORT

The port used to access the database.

The default setting for this option is `3306`.

DB_USER

The username to access the database.

DB_PASSWORD

The password for `DB_USER`.

DB_NAME

The name of the newznab database.

The default setting for this option is `newznab`.

DB_INNODB

This changes the behaviour of how deletes are done on the database. If you are using InnoDB tables, set this to `True`.

The default for this option is `false`.

DB_PCONNECT

Forces a persistent connection to the database server.

The default setting for this option is `True`.

DB_ERRORMODE

This attribute controls how the PHP Data Object (PDO) error reporting mode operates.

The options for this are:

```
PDO::ERRMODE_SILENT
PDO::ERRMODE_WARNING
PDO::ERRMODE_EXCEPTION
```

The default setting for this option is `PDO::ERRMODE_SILENT`.

NNTP_USERNAME

Your username for your news server.

NNTP_PASSWORD

Your password for `NNTP_USERNAME`

NNTP_SERVER

The fully qualified domain name of your news server.

NNTP_PORT

The port to connect to your news server. Typically this is port `119` or port `563` if using SSL. If you are going to use SSL, you will need to set `NNTP_SSLENABLED`.

The default setting for this option is `119`.

NNTP_SSLENABLED

Enable SSL for communications to your news server. Set it to `true` to enable SSL.

The default setting for this option is `false`.

CACHEOPT_METHOD

This setting enables the use of memcached. To use it, set it to `memcache`. The other option is to have query results saved to files. They will be located in "newznab/db/cache".

The options for this are:

```
none
memcache
file
```

The default setting for this option is `none`.

CACHEOPT_TTLFAST

The default is `120` seconds.

CACHEOPT_TTLMEDIUM

The default is `600` seconds.

CACHEOPT_TTLSLOW

The default is `1800` seconds.

CACHEOPT_MEMCACHE_SERVER

The address of the memcached server. The default assumes memcached is installed on the same server as Newznab. This can be fully qualified name, IP address, 'localhost' or '127.0.0.1'.

The default setting for this option is `127.0.0.1`.

CACHEOPT_MEMCACHE_PORT

The port used to connect to the memcache server.

The default setting for this option is `11211`.

EXTERNAL_PROXY_IP

If you are behind a proxy, enable this setting and set it to the **Internal Address** of your proxy server. This can be fully qualified name or IP address.

The default setting for this option is not enabled.

EXTERNAL_HOST_NAME

If you are behind a proxy, enable this setting and set it to the fully qualified external hostname.

The default setting for this option is not enabled.

CHAPTER 7

Miscellaneous

Web API

Introduction

This document describes the NEWZNAB Usenet Searching Web API. The API is designed to be implemented by Usenet indexing sites, i.e. sites that index Usenet newsgroups through some means, typically by downloading and inspecting the NTTP headers. The API is aimed for NZB aware client applications to allow them to perform Usenet searches against Newznab servers and receive NZB information in order to facilitate direct downloading from Usenet without having to download any NTTP headers.

This document does not describe the actual implementation of either the client or the server but just describes the HTTP(S) interface and request/response sequences.

Intended readers are server and client implementers.

Notation

This document uses the following notations:

Parameters:

- `t=c` denotes a required HTTP query parameter.
- `[o=json | o=xml]` denotes optional parameters with possible values.

Functions

All functions are executed as HTTP(S) requests over TCP. All parameters are to be passed as query parameters unless otherwise indicated. All returned XML/JSON data is UTF-8 encoded unless otherwise specified. All query parameters should be UTF-8 and URL encoded, i.e.:

```
query-param = URL-ENCODE(UTF8-ENCODE(param=value)).
```

The functions are divided into two categories. Functions specific to searching and retrieving of items and the their information such as SEARCH and TV-SEARCH and functions that are for site/user account management such as CAPS and REGISTER.

Any conforming implementation should support the CAPS and SEARCH functions. Other functions are optional and if not supported will return the "203 Function Not Available" when invoked.

CAPS

The `CAPS` function is used to query the server for supported features and the protocol version and other meta data relevant to the implementation. This function doesn't require the client to provide any login information but can be executed out of "login session".

Returned Fields

server/version	The version of the protocol implemented by the server. All implementations should be backwards compatible.
limits	The limit and defaults to the number of search results returned.
retention	Server retention (how many days NZB information is stored before being purged).
category	Defines a searchable category which might have any number of subcategories.
category/id	Unique category ID, can be either one of the standard category IDs or a site specific ID.
category/name	Any descriptive name for the category. Can be site/language specific.
category/description	A description of the contents of the category.
category/subcat	A subcategory.
category/subcat/id	Unique category ID, can be either one of the standard category IDs or a site specific ID.
category/subcat/name	Any descriptive name for the category. Can be site/language specific.
category/subcat/description	A description of the contents of the category.
groups	Defines a list of active, indexed usenet groups.
group/name	Name of usenet group.
group/description	Description of usenet group.
group/lastupdate	Date and time usenet group was last updated.
genres	Defines a list of active genres.
genre/id	Id of genre.
genre/name	Name of genre.
genre/categoryid	The category the genre is associate with.

HTTP Method

GET

HTTP Response

200 OK

Parameters

`t=caps`	Caps function, must always be "caps".

Optional parameters

`o=xxx`	Output format, either "JSON" or "XML. Default is "XML".

Examples

1. Normal behavior

Request:

```
GET http://servername.com/api?t=caps
```

Response:

```
200 OK
```

Content:

```
<?xml version="1.0" encoding="UTF-8"?>
  <caps>
    <!-- server information -->
    <server version="1.0" title="Newznab" strapline="A great usenet indexer"
            email="info@newznab.com" url="http://servername.com/"
            image="http://servername.com/theme/black/images/banner.jpg" />

    <!-- limit parameter range -->
    <limits max="100" default="50" />

    <!-- the server NZB retention -->
    <retention days="400" />

    <!-- registration available or not -->
    <registration available="yes" open="yes" />

    <!--
         The search functions available at the server
         The only currently defined search functions are SEARCH and TV-SEARCH.
         Any conforming implementation should at least support the basic search.
         Other search functions are optional.
    -->
    <searching>
        <search available="yes" />
        <tv-search available="yes" />
        <movie-search available="no" />
    </searching>

    <!-- supported categories -->
    <categories>
        <category id="1000" name="Console">
          <subcat id="1010" name="NDS" />
          <subcat id="1020" name="PSP" />
        </category>
        <category id="2000" name="Movies">
          <subcat id="2010" name="Foreign" />
        </category>

        <!-- site specific categories -->
        <category id="1000001" name="Debian"           description="Latest Debian stuff" />
        <category id="1000002" name="Mandrake 2010"    description="Mandrake 2010">
          <subcat id="1000003" name="Mandrake 2010 HD" description="Mandrake HD stuff" />
          <subcat id="1000004" name="Mandrake 2010 SD" description="Mandrake SD stuff" />
```

```
        </category>
        <!-- etc.. -->
    </categories>
  </caps>
</xml>
```

REGISTER

The `REGISTER` function is used for automatically creating and registering user account. This is an optional function and may or may not be available at a site. It is also possible that function is available but currently registrations at the site are closed.

The only prerequisite for registering an account is a valid email address and any server policies. It is at the server administration discretion to allow or deny registrations based on for example the validity of the email address or the the current client host address.

On successful registration a valid username, password and api key are returned to the caller. On error an appropriate error code is returned.

HTTP Method

GET

HTTP Response

200 OK

Parameters

`t=register`	Register function, must always be "register"
`email=xxx`	A valid email address to be used for registration. (URL/UTF-8 encoded).

Examples

1. Normal behavior

   Request:

   ```
   GET HTTP://servername.com/api?t=register&email=john.
   joe%40acme.com
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <register username="user123" password="pass123" apikey="abcabcd11234abc" /
   ↪>
   ```

2. Denial

   Request:

   ```
   GET HTTP://servername.com/api?t=register&email=john.
   joe%40acme.com
   ```

   Response:

   ```
   200 OK
   ```

   Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<error code="103" description="Registration denied"/>
```

3. Registration limit imposed

   Request:

   ```
   GET HTTP://servername.com/api?t=register&email=john.
   joe%40acme.com
   ```

   Response:

   ```
   200 OK
   ```

   Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<error code="104" description="No more registrations allowed"/>
```

4. Registration disabled

   Request:

   ```
   GET HTTP://servername.com/api?t=register&email=john.
   joe%40acme.com
   ```

   Response:

   ```
   200 OK
   ```

   Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<error code="203" description="Function not available"/>
```

SEARCH

The `SEARCH` function searches the index for items matching the search criteria. On successful search the response contains a list of found items. Even if search matched nothing an empty response set is created and returned. This function requires passing the user credentials.

Searches that include categories that are not supported by the server are still executed but the non-supported categories are simply skipped. This basically treats such a search simply as a "no match" but allows the same query to be ran simultaneously against several servers.

The list of search categories specifies a logical OR condition. I.e. an item matching the search input in any of the specified categories is considered a match and is returned. E.g. a search searching for "linux" in "computer" and "ebook" categories searches for matching items in "computer" and "ebook" but does not search for example the "movies" category. Items found in either group are then combined into a single result set. If the input string for search is empty all items (within the server/query limits) are returned for the matching categories.

When performing the query the categories to be searched are concatenated into a single query parameter by `,` (comma). For example `cat=200,300,400`, which is then URL encoded.

The returned XML data stream is RSS 2.0 compliant and also contains additional information in the extra namespace.

Response-offset field identifies the current subset of all the matches that are being transmitted in the response. In other words, if a search for "disco" finds more matches than the server is capable of transmitting in a single response, the response needs to be split into several responses. Then it is's the clients responsibility to repeat the same query with same parameters but specify an increased offset in order to return the next set of results.

If offset query parameter is not used response data contains items between 0 offset - limit. If offset query parameter is out of bounds an empty result set is returned.

Attrs parameter provides a comma ”,” separated list of additional (extended) attributes that the search should return if they are applicable to the current item. If attrs is not specified a set of default parameters is returned.

Important fields of the returned data (RSS)

title	Title of the found item.
guid	A globally unique (GUID) item identifier.
pubdate	The publishing date in RSS date object as specified by RFC822/2822. (not the Usenet date)
category	The category the NZB belongs to. (This is human readable for RSS. More precise category is found in additional data)
enclosure	The NZB url

HTTP Method

GET

HTTP Response

200 OK

Parameters

`t=search`	Search function, must always be “search”
`apikey=xxxx`	User’s key as provided by the service provider.

Optional parameters

Examples

1. Normal behavior

   Request:

   ```
   GET http://servername.com/api?t=search&apikey=xxxxx&q=a%20tv%20show
   ```

   Response:

   ```
   200 OK
   ```

   Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">

<channel>
<title>example.com</tile>
<description>example.com API results</description>
<!--
   More RSS content
 -->

<!-- offset is the current offset of the response
     total is the total number of items found by the query
 -->
<newznab:response offset="0" total="2344"/>

<item>
  <!-- Standard RSS 2.0 Data -->
  <title>A.Public.Domain.Tv.Show.S06E05</title>
  <guid isPermaLink="true">http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c</guid>
```

```
  <link>http://servername.com/rss/nzb/e9c515e02346086e3a477a5436d7bc8c&
↪amp;i=1&amp;r=18cf9f0a736041465e3bd521d00a90b9</link>
  <comments>http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c#comments</comments>
  <pubDate>Sun, 06 Jun 2010 17:29:23 +0100</pubDate>
  <category>TV > XviD</category>
  <description>Some TV show</description>
  <enclosure url="http://servername.com/rss/nzb/
↪e9c515e02346086e3a477a5436d7bc8c&amp;i=1&amp;
↪r=18cf9f0a736041465e3bd521d00a90b9" length="154653309" type=
↪"application/x-nzb" />

  <!-- Additional attributes -->
  <newznab:attr name="category" value="2000"/>
  <newznab:attr name="category" value="2030"/>
  <newznab:attr name="size"     value="4294967295"/>
</item>
</channel>
</rss>
```

2. No items matched the search criteria.

   Request:

   ```
   GET http://servername.com/api?t=search&apikey=xxxxx&q=linux%20image
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <rss>
   <channel>
       <newznab:response offset="0" total="0"/>
   </channel>
   </rss>
   ```

3. Query could not be completed because user credentials are broken

   Request:

   ```
   GET http://servername.com/api?t=search&apikey=xxxxx&q=linux%20image
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <error code="100" description="Incorrect user credentials"/>
   ```

4. Query could not be completed because it was malformed

   Request:

   ```
   GET http://servername.com/api?t=search&apikey=xxxxx&q=linux%20image
   ```

   Response:

   ```
   200 OK
   ```

Content:

```
<?xml version="1.0" encoding="UTF-8"/>
<error code="200" description="Missing parameter: key"/>
```

TV-SEARCH

The `TV-SEARCH` function searches the index in the TV category for items matching the search criteria. The criteria includes query string and in addition information about season and episode. On successful search the response contains a list of items that matched the query. Even if the search matched nothing an empty but valid response is created and returned. This function requires passing the user credentials.

The returned XML data stream is RSS 2.0 compliant and also contains additional information in the extra namespace and optionally TV specific information.

HTTP Method

GET

HTTP Response

200 OK

Parameters

`t=tvsearch`	TV-Search function, must always be "tvsearch".
`apikey=xxx`	User's key as provided by the service provider.

Optional parameters

Examples

1. Normal behavior

   Request: `GET http://servername.com/api?t=tvsearch&apikey=xxxq=The%20Beverly%20Hillbi`

   Response:

   ```
   200 OK
   ```

   Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
<title>example.com</title>
<description>example.com API results</description>
<!--
  More RSS content
-->

<!-- offset is the current offset of the response
     total is the total number of items found by the query
-->
<newznab:response offset="0" total="1234"/>

<item>
  <!-- Standard RSS 2.0 data -->
  <title>A.Public.Domain.Tv.Show.S06E05</title>
  <guid isPermaLink="true">http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c</guid>
  <link>http://servername.com/rss/nzb/e9c515e02346086e3a477a5436d7bc8c&
↪amp;i=1&amp;r=18ef9f0a736041465e3bd521d00a90b9</link>
```

```
  <comments>http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c#comments</comments>
  <pubDate>Sun, 06 Jun 2010 17:29:23 +0100</pubDate>
  <category>TV > XviD</category>
  <description>Some TV show</description>
  <enclosure url="http://servername.com/rss/nzb/
↪e9c515e02346086e3a477a5436d7bc8c&amp;i=1&amp;
↪r=18cf9f0a736041465e3bd521d00a90b9" length="154653309" type=
↪"application/x-nzb" />

  <!-- Additional attributes -->
  <newznab:attr name="category" value="5030"/>
  <newznab:attr name="size"     value="154653309"/>
  <newznab:attr name="season"   value="3"/>
  <newznab:attr name="episode"  value="2"/>
</item>

<item>
  <!-- Standard RSS 2.0 data -->
  <title>A.Public.Domain.Tv.Show.S06E05</title>
  <guid isPermaLink="true">http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c</guid>
  <link>http://servername.com/rss/nzb/e9c515e02346086e3a477a5436d7bc8c&
↪amp;i=1&amp;r=18cf9f0a736041465e3bd521d00a90b9</link>
  <comments>http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c#comments</comments>
  <pubDate>Sun, 06 Jun 2010 17:29:23 +0100</pubDate>
  <category>TV > XviD</category>
  <description>Some TV show</description>
  <enclosure url="http://servername.com/rss/nzb/
↪e9c515e02346086e3a477a5436d7bc8c&amp;i=1&amp;
↪r=18cf9f0a736041465e3bd521d00a90b9" length="154653309" type=
↪"application/x-nzb" />

  <!-- Additional attributes -->
  <newznab:attr name="category" value="5000" />
  <newznab:attr name="category" value="5030" />
  <newznab:attr name="size"     value="4294967295" />
  <newznab:attr name="season"   value="3"/>
  <newznab:attr name="episode"  value="1"/>
</item>

<!-- more items to follow -->

</channel>
</rss>
```

MOVIE-SEARCH

The `MOVIE-SEARCH` function searches the index for items matching an IMDb ID or search query. On successful search the response contains a list of items that matched the query. Even if the search matched nothing an empty but valid response is created and returned. This function requires passing the user credentials.

The returned XML data stream is RSS 2.0 compliant and also contains additional information in the extra namespace and optionally movie specific information.

HTTP Method

GET

HTTP Response

200 OK

Parameters

t=movie	Movie-Search function, must always be "movie".
apikey=xxx	User's key as provided by the service provider.

Optional parameters

limit=123	Upper limit for the number of items to be returned, e.g. 123.
imdbid=xxxx	IMDB id of the item being queried e.g. 0058935.
cat=xxx	List of categories to search delimited by ","
genre=xxx	A genre string i.e. 'Romance' would match '(Comedy, Drama, Indie, Romance)'
q=xxxx	Search input (URL/UTF-8 encoded). Case insensitive.
o=xml	Output format, either "JSON" or "XML". Default is "XML".
attrs=xxx	List of requested extended attributes delimeted by ","
extended=1	List all extended attributes (attrs ignored)
del=1	Delete the item from a users cart on download.
maxage=123	Only return results which were posted to usenet in the last x days.
offset=50	The 0 based query offset defining which part of the response we want.

Examples

1. Normal behavior

   Request:

   ```
   GET http://servername.com/api?t=movie&apikey=xxx&imdbid=0019798
   ```

   Response:

   ```
   200 OK
   ```

   Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
    <title>example.com</title>
    <description>example.com API results</description>
    <!--
      More RSS content
    -->

    <!-- offset is the current offset of the response
         total is the total number of items found by the query
    -->
    <newznab:response offset="0" total="1234"/>

    <item>
      <!-- Standard RSS 2.0 data -->
      <title>A.Public.Domain.Movie.720p.DTS.x264</title>
      <guid isPermaLink="true">http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c</guid>
      <link>http://servername.com/rss/nzb/
↪e9c515e02346086e3a477a5436d7bc8c&amp;i=1&amp;
↪r=18cf9f0a736041465e3bd521d00a90b9</link>
      <comments>http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c#comments</comments>
```

```
      <pubDate>Sun, 06 Jun 2010 17:29:23 +0100</pubDate>
      <category>Movie > XviD</category>
      <description>Some movie</description>
      <enclosure url="http://servername.com/rss/nzb/
↪e9c515e02346086e3a477a5436d7bc8c&amp;i=1&amp;
↪r=18cf9f0a736041465e3bd521d00a90b9" length="154653309" type=
↪"application/x-nzb" />

      <!-- Additional attributes -->
      <newznab:attr name="category" value="2000" />
      <newznab:attr name="category" value="2030" />
      <newznab:attr name="size"     value="4294967295" />
    </item>

</channel>
</rss>
```

MUSIC-SEARCH

The `MUSIC-SEARCH` function searches the index for items matching music properties. On successful search the response contains a list of items that matched the query. Even if the search matched nothing an empty but valid response is created and returned. This function requires passing the user credentials.

The returned XML data stream is RSS 2.0 compliant and also contains additional information in the extra namespace and optionally music specific information.

HTTP Method

GET

HTTP Response

200 OK

Parameters

`t=music`	Music-Search function, must always be "music".
`apikey=xxx`	User's key as provided by the service provider.

Optional Parameters

`limit=123`	Upper limit for the number of items to be returned, e.g. 123.
`album=xxxx`	Album title (URL/UTF-8 encoded). Case insensitive.
`artist=xxxx`	Artist name (URL/UTF-8 encoded). Case insensitive.
`label=xxxx`	Publisher/Label name (URL/UTF-8 encoded). Case insensitive.
`track=xxxx`	Track name (URL/UTF-8 encoded). Case insensitive.
`year=xxxx`	Four digit year of release.
`genre=123`	List of music genre id's to search delimited by ",". See CAPS for available genres.
`cat=xxx`	List of categories to search delimited by ","
`o=xml`	Output format, either "JSON" or "XML". Default is "XML".
`attrs=xxx`	List of requested extended attributes delimited by ","
`extended=1`	List all extended attributes (attrs ignored)
`del=1`	Delete the item from a users cart on download.
`maxage=123`	Only return results which were posted to usenet in the last x days.
`offset=50`	The 0 based query offset defining which part of the response we want.

Examples

1. Normal behavior

Request:

```
GET http://servername.com/api?t=music&apikey=xxx&album=Groovy&extended=1
```

Response:

```
200 OK
```

Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
    <title>example.com</title>
    <description>example.com API results</description>
    <!--
      More RSS content
    -->

    <!-- offset is the current offset of the response
         total is the total number of items found by the query
    -->
    <newznab:response offset="0" total="1234"/>

    <item>
      <!-- Standard RSS 2.0 data -->
      <title>A.Public.Domain.Album.Name</title>
      <guid isPermaLink="true">http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c</guid>
      <link>http://servername.com/rss/nzb/
↪e9c515e02346086e3a477a5436d7bc8c&amp;i=1&amp;
↪r=18cf9f0a736041465e3bd521d00a90b9</link>
      <comments>http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c#comments</comments>
      <pubDate>Sun, 06 Jun 2010 17:29:23 +0100</pubDate>
      <category>Music > MP3</category>
      <description>Some music</description>
      <enclosure url="http://servername.com/rss/nzb/
↪e9c515e02346086e3a477a5436d7bc8c&amp;i=1&amp;
↪r=18cf9f0a736041465e3bd521d00a90b9" length="154653309" type=
↪"application/x-nzb" />

      <!-- Additional attributes -->
      <newznab:attr name="category" value="3000" />
      <newznab:attr name="category" value="3010" />
      <newznab:attr name="size"     value="144967295" />
      <newznab:attr name="artist"   value="Bob Smith" />
      <newznab:attr name="album"    value="Groovy Tunes" />
      <newznab:attr name="publisher" value="Epic Music" />
      <newznab:attr name="year"     value="2011" />
      <newznab:attr name="tracks"   value="track one|track two|track three
↪" />
      <newznab:attr name="coverurl" value="http://servername.com/covers/
↪music/12345.jpg" />
      <newznab:attr name="review"   value="This album is great" />
    </item>

</channel>
</rss>
```

BOOK-SEARCH

The `BOOK-SEARCH` function searches the index for items matching book properties. On successful search the response contains a list of items that matched the query. Even if the search matched nothing an empty but valid response is created and returned. This function requires passing the user credentials.

The returned XML data stream is RSS 2.0 compliant and also contains additional information in the extra namespace and optionally music specific information.

HTTP Method

GET

HTTP Response

200 OK

Parameters

`t=book`	Book-Search function, must always be "book".
`apikey=xxx`	User's key as provided by the service provider.

Optional Parameters

`limit=123`	Upper limit for the number of items to be returned, e.g. 123.
`title=xxxx`	Book title (URL/UTF-8 encoded). Case insensitive.
`author=xxxx`	Author name (URL/UTF-8 encoded). Case insensitive.
`o=xml`	Output format, either "JSON" or "XML". Default is "XML".
`attrs=xxx`	List of requested extended attributes delimited by ","
`extended=1`	List all extended attributes (attrs ignored)
`del=1`	Delete the item from a users cart on download.
`maxage=123`	Only return results which were posted to usenet in the last x days.
`offset=50`	The 0 based query offset defining which part of the response we want.

Examples

1. Normal behavior

   Request:

   ```
   GET http://servername.com/api?t=book&apikey=xxx&author=Charles%20Dack&extended=1
   ```

   Response:

   ```
   200 OK
   ```

   Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
    <title>example.com</title>
    <description>example.com API results</description>
    <!--
      More RSS content
    -->

    <!-- offset is the current offset of the response
         total is the total number of items found by the query
    -->
    <newznab:response offset="0" total="1234"/>

    <item>
```

```
      <!-- Standard RSS 2.0 data -->
      <title>A.Public.Domain.Book.Name</title>
      <guid isPermaLink="true">http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c</guid>
      <link>http://servername.com/rss/nzb/
↪e9c515e02346086e3a477a5436d7bc8c&amp;i=1&amp;
↪r=18cf9f0a736041465e3bd521d00a90b9</link>
      <comments>http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c#comments</comments>
      <pubDate>Sun, 06 Jun 2010 17:29:23 +0100</pubDate>
      <category>Book > Ebook</category>
      <description>Some book</description>
      <enclosure url="http://servername.com/rss/nzb/
↪e9c515e02346086e3a477a5436d7bc8c&amp;i=1&amp;
↪r=18cf9f0a736041465e3bd521d00a90b9" length="154653309" type=
↪"application/x-nzb" />

      <!-- Additional attributes -->
      <newznab:attr name="category" value="7020" />
      <newznab:attr name="size"     value="144967295" />
      <newznab:attr name="author"   value="Charles Dack" />
      <newznab:attr name="title"    value="Weather and Folk Lore of␣
↪Peterborough and District" />
      <newznab:attr name="review"   value="This book is a classic" />
    </item>

</channel>
</rss>
```

DETAILS

The `DETAILS` function returns all information for a particular Usenet (NZB) item. The response contains the generic RSS part + full extra information + full type/category specific information.

HTTP Method

GET

HTTP Response

200 OK

Parameters

`t=details`	Details function, must always be "details".
`id=xxxx`	The GUID of the item being queried.
`apikey=xxxx`	User's key as provided by the service provider.

Optional Parameters

`o=xxx`	Output format, either "JSON" or "XML". Default is "XML".

Examples

1. Normal behavior

   Request:

   ```
   GET http://servername.com/api?t=details&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

200 OK

Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
  <item>
    <!-- Standard RSS 2.0 Data -->
    <title>A.Public.Domain.Tv.Show.S06E05</title>
    <guid isPermaLink="true">http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c</guid>
    <link>http://servername.com/rss/nzb/e9c515e02346086e3a477a5436d7bc8c&
↪amp;i=1&amp;r=18cf9f0a736041465e3bd521d00a90b9</link>
    <comments>http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c#comments</comments>
    <pubDate>Sun, 06 Jun 2010 17:29:23 +0100</pubDate>
    <category>TV > XviD</category>
    <description>Some TV show</description>
    <enclosure url="http://servername.com/rss/nzb/
↪e9c515e02346086e3a477a5436d7bc8c&amp;i=1&amp;
↪r=18cf9f0a736041465e3bd521d00a90b9" length="154653309" type=
↪"application/x-nzb" />

    <!--
         Additional attributes
         Details function returns all possible attributes that are 1)␣
↪known and 2) applicable
         for the item requested.
    -->
    <newznab:attr name="category"   value="2000" />
    <newznab:attr name="category"   value="2030" />
    <newznab:attr name="size"       value="4294967295" />
    <newznab:attr name="files"      value="107" />
    <newznab:attr name="poster"     value="example@example.net (example)"␣
↪/>
    <newznab:attr name="grabs"      value="1" />
    <newznab:attr name="comments"   value="0" />
    <newznab:attr name="usenetdate" value="Tue, 22 Jun 2010 06:54:22 +0100
↪" />
    <newznab:attr name="group"      value="alt.binaries.tv" />

  </item>
</channel>
</rss>
```

2. Query could not be completed because it was malformed

Request:

GET http://servername.com/api?t=details&apikey=xxxxx&guid=xxxxxxxxx

Response:

200 OK

Content:

```
<?xml version="1.0" encoding="UTF-8"/>
<error code="200" description="Missing parameter: key"/>
```

3. Query could not be completed because no such item was available

   Request:

   ```
   GET http://servername.com/api?t=details&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

```
<?xml version="1.0" encoding="UTF-8"/>
<error code="300" description="No such GUID"/>
```

4. Query could not be completed because user credentials are broken

   Request:

   ```
   GET http://servername.com/api?t=details&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

```
<?xml version="1.0" encoding="UTF-8"/>
<error code="100" description="Incorrect user credentials"/>
```

GETNFO

The `GETNFO` function returns an nfo file for a particular Usenet (NZB) item.

HTTP Method

GET

HTTP Response

200 OK

Parameters t=getnfo Details function, must always be "getnfo". id=xxxx The GUID of the item being queried. apikey=xxxx User's key as provided by the service provider.

Optional parameters raw=1 If provided returns just the nfo file without the rss container o=xxx Output format, either "JSON" or "XML". Default is "XML".

Examples

1. Normal behavior

   Request:

   ```
   GET http://servername.com/api?t=getnfo&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
  <item>
    <!-- Standard RSS 2.0 Data -->
    <title>A.Public.Domain.Tv.Show.S06E05</title>
    <guid isPermaLink="true">http://servername.com/details/
↪e9c515e02346086e3a477a5436d7bc8c</guid>
    <link>http://servername.com/nfo/e9c515e02346086e3a477a5436d7bc8c</
↪link>
    <pubDate>Sun, 06 Jun 2010 17:29:23 +0100</pubDate>
    <description>This is the nfo file</description>
    <enclosure url="http://servername.com/nfo/
↪e9c515e02346086e3a477a5436d7bc8c&ampenclosure=1&amp;i=1&amp;
↪r=18cf9f0a736041465e3bd521d00a90b9" length="154653309" type=
↪"application/x-nzb" />
  </item>
</channel>
</rss>
```

2. Query could not be completed because it was malformed

   Request:

   ```
   GET http://servername.com/api?t=getnfo&apikey=xxxxx&id=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"/>
   <error code="200" description="Missing parameter: id"/>
   ```

3. Query could not be completed because no such item was available

   Request:

   ```
   GET http://servername.com/api?t=getnfo&apikey=xxxxx&id=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"/>
   <error code="300" description="No such GUID"/>
   ```

4. Query could not be completed because user credentials are broken

   Request:

   ```
   GET http://servername.com/api?t=getnfo&apikey=xxxxx&id=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

```
<?xml version="1.0" encoding="UTF-8"/>
<error code="100" description="Incorrect user credentials"/>
```

GET

The `GET` function returns an nzb for a guid.

HTTP Method

GET

HTTP Response

200 OK

Parameters t=get Details function, must always be "get". id=xxxx The GUID of the item being queried. apikey=xxxx User's key as provided by the service provider.

Optional parameters del=1 If provided removes the nzb from the users cart (if present)

Examples

1. Normal behavior

   Request:

   ```
   GET http://servername.com/api?t=get&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <!DOCTYPE nzb PUBLIC "-//newzBin//DTD NZB 1.1//EN" "http://www.newzbin.
   ↪com/DTD/nzb/nzb-1.1.dtd">
   <nzb xmlns="http://www.newzbin.com/DTD/2003/nzb">
   ...
   ```

2. Query could not be completed because it was malformed

   Request:

   ```
   GET http://servername.com/api?t=get&apikey=xxxxx&id=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"/>
   <error code="200" description="Missing parameter: id"/>
   ```

3. Query could not be completed because no such item was available

   Request:

   ```
   GET http://servername.com/api?t=get&apikey=xxxxx&id=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

Content:

```
<?xml version="1.0" encoding="UTF-8"/>
<error code="300" description="No such GUID"/>
```

4. Query could not be completed because user credentials are broken

   Request:

   ```
   GET http://servername.com/api?t=get&apikey=xxxxx&id=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

```
<?xml version="1.0" encoding="UTF-8"/>
<error code="100" description="Incorrect user credentials"/>
```

CART-ADD

The `CART-ADD` function adds an item to the users cart.

HTTP Method

GET

HTTP Response

200 OK

Parameters

`t=cartadd`	Cart add function, must always be "cartadd".
`id=xxxx`	The GUID of the item being added to the cart.
`apikey=xxxx`	User's key as provided by the service provider.

Optional Parameters

`o=xxx`	Output format, either "JSON" or "XML". Default is "XML".

Examples

1. Default behavior

   Request:

   ```
   GET http://servername.com/api?t=cartadd&id=12344234234234&apikey=xxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<cartadd id="123" />
```

2. Query could not be completed because it was malformed

   Request:

   ```
   GET http://servername.com/api?t=cartadd&apikey=xxxxx&guid=xxxxxxxxx
   ```

Response:

```
200 OK
```

Content:

```
<?xml version="1.0" encoding="UTF-8"/>
<error code="200" description="Missing parameter: id"/>
```

3. Query could not be completed because no such item was available

   Request:

   ```
   GET http://servername.com/api?t=cartadd&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"/>
   <error code="300" description="No such GUID"/>
   ```

4. Query could not be completed because user credentials are broken

   Request:

   ```
   GET http://servername.com/api?t=cartadd&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"/>
   <error code="100" description="Incorrect user credentials"/>
   ```

5. Query could not be completed because item already exists

   Request:

   ```
   GET http://servername.com/api?t=cartadd&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"/>
   <error code="310" description="Item already exists"/>
   ```

CART-DEL

The `CART-DEL` function deletes an item from the users cart.

HTTP Method

GET

HTTP Response

200 OK

Parameters

t=cartdel	Cart del function, must always be "cartdel".
id=xxxx	The GUID of the item being delete from the cart.
apikey=xxxx	User's key as provided by the service provider.

Optional Parameters

o=xxx	Output format, either "JSON" or "XML". Default is "XML".

Examples

1. Default behavior

   Request:

   ```
   GET http://servername.com/api?t=cartdel&id=12344234234234&apikey=xxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <cartdel id="123" />
   ```

2. Query could not be completed because it was malformed

   Request:

   ```
   GET http://servername.com/api?t=cartdel&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"/>
   <error code="200" description="Missing parameter: id"/>
   ```

3. Query could not be completed because no such item was available

   Request:

   ```
   GET http://servername.com/api?t=cartdel&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"/>
   <error code="300" description="No such GUID"/>
   ```

4. Query could not be completed because user credentials are broken

   Request:

   ```
   GET http://servername.com/api?t=cartdel&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

Content:

```
<?xml version="1.0" encoding="UTF-8"/>
<error code="100" description="Incorrect user credentials"/>
```

COMMENTS

The `COMMENTS` function returns all comments known about a release.

HTTP Method

GET

HTTP Response

200 OK

Parameters

`t=comments`	Comments function, must always be "comments".
`guid=xxxx`	The GUID of the item being queried.
`apikey=xxxx`	User's key as provided by the service provider.

Optional Parameters

`o=xxx`	Output format, either "JSON" or "XML". Default is "XML".

Examples

1. Default behavior

   Request:

   ```
   GET http://servername.com/api?t=comments&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
  <item>
    <!-- Standard RSS 2.0 Data -->
    <title>username_of_poster</title>
    <guid isPermaLink="true">http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c</guid>
    <link>http://servername.com/rss/viewnzb/
↪e9c515e02346086e3a477a5436d7bc8c</link>
    <pubDate>Sun, 06 Jun 2010 17:29:23 +0100</pubDate>
    <description>Comment about the item</description>
  </item>
</channel>
</rss>
```

2. Query could not be completed because it was malformed

   Request:

   ```
   GET http://servername.com/api?t=comments&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

```
200 OK
```

Content:

```
<?xml version="1.0" encoding="UTF-8"/>
<error code="200" description="Missing parameter: key"/>
```

3. Query could not be completed because no such item was available

   Request:

   ```
   GET http://servername.com/api?t=comments&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"/>
   <error code="300" description="No such GUID"/>
   ```

4. Query could not be completed because user credentials are broken

   Request:

   ```
   GET http://servername.com/api?t=comments&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"/>
   <error code="100" description="Incorrect user credentials"/>
   ```

COMMENTS-ADD

The `COMMENTS-ADD` function adds a comment to a release.

HTTP Method

GET

HTTP Response

200 OK

Parameters

`t=commentadd`	Comments-add function, must always be "commentadd".
`guid=xxxx`	The GUID of the item being queried.
`apikey=xxxx`	User's key as provided by the service provider.
`text=xxxx`	The comment being added (URL/UTF-8 encoded).

Optional Parameters

`o=xxx`	Output format, either "JSON" or "XML". Default is "XML".

Examples

1. **Request**:

   ```
   GET HTTP://servername.com/api?t=commentadd&apikey=xxxxx&guid=xxxxxxxxx&text=comment
   ```

Response:

```
200 OK
```

Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<commentadd id="123" />
```

2. **Request**:

```
GET HTTP://servername.com/api?t=commentadd&apikey=xxxxx&guid=xxxxxxxxx&text=comment
```

Response:

```
200 OK
```

Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<error code="300" description="No such item"/>
```

3. **Request**:

```
GET HTTP://servername.com/api?t=commentadd&apikey=xxxxx&guid=xxxxxxxxx&text=comment
```

Response:

```
200 OK
```

Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<error code="203" description="Function not available"/>
```

USER

The `USER` function is used for retrieving information about a user account

HTTP Method

GET

HTTP Response

200 OK

Parameters

`t=user`	User function, must always be "user"
`username=xxx`	A valid username (URL/UTF-8 encoded).

Examples

1. **Request**:

```
GET HTTP://servername.com/api?t=user&username=user123
```

Response:

```
200 OK
```

Content:

```
<?xml version="1.0" encoding="UTF-8"?>
<user username="user123" grabs="123" role="User" apirequests="100"␣
↪downloadrequests="100"
movieview="1" musicview="1" consoleview="1" createddate="2011-08-23 12:31:47" />
```

2. **Request**:

   ```
   GET HTTP://servername.com/api?t=user&username=user123
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <error code="300" description="No such item"/>
   ```

3. **Request**:

   ```
   GET HTTP://servername.com/api?t=user&username=user123
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <error code="203" description="Function not available"/>
   ```

Predefined Categories

In order to facilitate operation that does not rely on a particular natural language, e.g. english a set of predefined category IDs have been defined. It is possible to define custom categories in the custom category range. Each category is given a range for a set of subcategories. It is possible for an item to belong to several categories at the same time.

Category Ranges

Category List

Categories	Category Name
0000	Reserved
1000	Console
1010	Console/NDS
1020	Console/PSP
1030	Console/Wii
1040	Console/XBox
1050	Console/XBox 360
1060	Console/Wiiware
1070	Console/XBox 360 DLC
2000	Movies
2010	Movies/Foreign
2020	Movies/Other
2030	Movies/SD
2040	Movies/HD

Continued on next page

Table 7.1 – continued from previous page

Categories	Category Name
2045	Movies/UHD
2050	Movies/BluRay
2060	Movies/3D
3000	Audio
3010	Audio/MP3
3020	Audio/Video
3030	Audio/Audiobook
3040	Audio/Lossless
4000	PC
4010	PC/0day
4020	PC/ISO
4030	PC/Mac
4040	PC/Mobile-Other
4050	PC/Games
4060	PC/Mobile-iOS
4070	PC/Mobile-Android
5000	TV
5020	TV/Foreign
5030	TV/SD
5040	TV/HD
5045	TV/UHD
5050	TV/Other
5060	TV/Sport
5070	TV/Anime
5080	TV/Documentary
6000	XXX
6010	XXX/DVD
6020	XXX/WMV
6030	XXX/XviD
6040	XXX/x264
6050	XXX/Pack
6060	XXX/ImgSet
6070	XXX/Other
7000	Books
7010	Books/Mags
7020	Books/EBook
7030	Books/Comics
8000	Other
8010	Other/Misc
100000-	Custom

Predefined Attributes

A set of known attributes for items in different categories has been defined. Its possible that not all attributes are available at all times. Therefore a client application should not rely on any particular attributes being in the returned data but should take this list as an optional extra information. However attributes marked with * are always available.

Additionally, not all attributes are applicable to all items. The category information can be used to check which attributes area available for which category items.

All attributes are defined using XML namespace syntax. e.g. xmlns:newznab=”http://www.newznab.com/DTD/2010/feeds/attributes/”

List of Attributes

Attribute	Category	Description	Example value
size *	ALL	Size in bytes	“252322”
category *	ALL	Item’s category	“5004”
guid	ALL	Unique release guid	“6c6734da3e92a7b0e494e896b58081
files	ALL	Number of files	“4”
poster	ALL	NNTP Poster	“yenc@power-post“
group	ALL	NNTP Group(s)	“a.b.group, a.b.tv”
team	ALL	Team doing the release	“Team”
grabs	ALL	Number of times item downloaded	“1”
password	ALL	Whether the archive is passworded	“0” no, “1” rar pass, “2” contains inne
comments	ALL	Number of comments	“2”
usenetdate	ALL	Date posted to usenet	“Tue, 22 Jun 2010 06:54:22 +0100”
nfo	ALL	Has an nfo or not	“1”
info	ALL	Info (.nfo) file URL	“http://site/api?t=getnfo&id=492296e
year	ALL	Release year	“2009”
season	TV	Numeric season	“1”
episode	TV	Numeric episode within the season	“1”
rageid	TV	TVRage ID. (www.tvrage.com)	“2322”
tvtitle	TV	TVRage Show Title. (www.tvrage.com)	“The Show Title”
tvairdate	TV	TVRage Show Air date. (www.tvrage.com)	“Tue, 22 Jun 2010 06:54:22 +0100”
video	TV, Movies	Video codec	“x264”
audio	TV, Movies, Audio	Audio codec	“AC3 2.0 @ 384 kbs”
resolution	TV, Movies	Video resolution	“1280x716 1.78:1”
framerate	TV, Movies	Video fps	“23.976 fps”
language	TV, Movies, Audio	Natural languages	“English”
subs	TV, Movies	Subtitles	“English, Spanish”
imdb	Movies	IMDb ID (www.imdb.com)	“0104409”
imdbscore	Movies	IMDb score	“5/10”
imdbtitle	Movies	IMDb score	“Bobs Movie”
imdbtagline	Movies	IMDb tagline	“Bobs new adventure”
imdbplot	Movies	IMDb plot	“All about the movie”
imdbyear	Movies	IMDb year	“1971”
imdbdirector	Movies	IMDb director	“Bob Smith”
imdbactors	Movies	IMDb actors	“Bob Smith, Kate Smith”
genre	TV, Movies	Genre	“Horror, Comedy”
artist	Music	Artist name	“Bob Smith”
album	Music	Album name	“Groovy Tunes”
publisher	Music, Book	Publisher name	“Epic Music”
tracks	Music	Track listing	“track one|track two|track three”
coverurl	TV, Movies, Music, Book	URL to cover image	“http://servername.com/covers/music/
backdropcoverurl	TV, Movies, Music	URL to backdrop image	“http://servername.com/covers/movie
review	Movies, Music, Book	Critics review	“This media is great”
booktitle	Book	Book title	“Weather and Folk Lore of Peterborou
publishdate	Book	Date book published	“Tue, 22 Jun 2010 06:54:22 +0100”
author	Book	Book author	“Charles Dack”
pages	Book	Number of pages	“123”

Attribute Example

Example attribute declarations within <item> element:

```
<newznab:attr name="category" value="2000" />
<newznab:attr name="category" value="2030" />
<newznab:attr name="size"     value="4294967295" />
```

Newznab Error Codes

Under normal circumstances i.e. when the HTTP request/response sequence is successfully completed Newznab implementations always respond with `HTTP 200 OK`. However this doesn't mean that the query was semantically correct. It simply means that the HTTP part of the sequence was successful. One then must check the actual response body/data to see if the request was completed without errors.

In case of a Newznab error the response contains an error code and an a description of the error.

The error codes have been defined into different ranges. 100-199 Account/user credentials specific error codes, 200-299 API call specific error codes, 300-399 content specific error codes and finally 900-999 Other error codes.

Error code	Description
100	Incorrect user credentials
101	Account suspended
102	Insufficient privileges/not authorized
103	Registration denied
104	Registrations are closed
105	Invalid registration (Email Address Taken)
106	Invalid registration (Email Address Bad Format)
107	Registration Failed (Data error)
200	Missing parameter
201	Incorrect parameter
202	No such function. (Function not defined in this specification).
203	Function not available. (Optional function is not implemented).
300	No such item.
300	Item already exists.
900	Unknown error
910	API Disabled

Error code example

1. Query could not be completed because user credentials are broken

   Request:

   ```
   GET http://servername.com/api?t=details&apikey=xxxxx&guid=xxxxxxxxx
   ```

   Response:

   ```
   200 OK
   ```

   Content:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <error code="100" description="Incorrect user credentials"/>
   ```

Sphinx

This is the documentation for the full-text search extension for Newznab. It is built on top of the very powerful Sphinx full-text indexer. To learn more about Sphinx go to: http://sphinxsearch.com/

Install

To install the search indexing system for Newznab, follow the steps below. If you follow these directions carefully you shouldn't have any issues. Read through them at least once before actually doing anything to make sure you know what's going to happen.

1. Download and/or install Sphinx. Make sure to get version 2.0.2 or higher: http://sphinxsearch.com/
2. Create the necessary directories. By default the directory `sphinxdata` in newznab's db directory will be used. If you don't need to specify a different location, you can skip this step and go to step 3.

   The first directory is where Sphinx will write the indexes to. Make sure it exists and is writeable:

   ```
   mkdir /path/to/index/storage/dir
   chmod 755 /path/to/index/storage/dir
   ```

   The second directory is where Sphinx will hold it's binary log files. This directory needs to exist inside of the first directory. Again, make sure the directory exists and is writeable:

   ```
   mkdir /path/to/index/storage/dir/binlog
   chmod 755 /path/to/index/storage/dir/binlog
   ```

3. Generate a 'sphinx.conf' file. The `nnindexer.php generate` command takes a single optional argument: the path to the first directory you created in Step 2 (the indexes storage directory). If you didn't create a custom directory in step 3, then type:

   ```
   ./nnindexer.php generate
   ```

   If you created a custom storage directory in step 2, pass the **first** directory you created as the first argument to the `generate` command:

   ```
   ./nnindexer.php generate /path/to/index/storage/dir
   ```

   The command will generate a `sphinx.conf` for you and will print out where it saved it to. It will also update the "Sphinx Configuration Path" setting in the database.

4. Login to the admin area of your Newznab install and set the Sphinx settings as desired. Two very important options that you **must** have filled in correctly before proceeding are:

   (a) **"Sphinx Configuration Path" - The full path to the `sphinx.conf` file** that you just generated in Step 3. Verify that this matches the value printed out by the `nnindexer.php generate` command that you ran in step 3.

   (b) "Sphinx Binaries Path" - The full path to the location where you installed the Sphinx binaries to in Step 1. If you're not sure where this is, you want the **directory** returned by the command `which searchd` or equivalent. If you leave this blank, then it is imperative that the Sphinx binaries be installed to a location within your system's `$PATH` variable (or the Windows equivalent if not on a *nix platform).

   It's worth mentioning that if you want the following commands to work, you need to make sure that "Use Sphinx" is set to "Yes".

   This is also a good time to decide which indexes to enable. The default `releases` index is enabled by default and cannot be disabled (unless you disable Sphinx entirely). For more information on the indexes and how much

"effort" it takes if they are enabled, see the section "Available Indexes" below. As a general rule, if you just want to speed up searching releases, leave the extra indexes disabled. You can always enable them later if you want.

5. Start the Sphinx search daemon (`searchd`):

```
./nnindexer.php daemon
```

Don't worry about any errors mentioning missing indexes, preload or "No such file or directory; NOT SERVING"–this is normal because we haven't indexed anything yet (that's the next step).

Important: You **must** run the daemon this as the same user that you run `update_releases.php`. If you don't do this, things will almost certainly not work correctly!

---

6. Generate the initial indexes:

```
./nnindexer.php index full all
./nnindexer.php index delta all
```

Depending on which items you've enabled for indexing, this step could take a while.

7. Restart the search daemon now that we have created all of the indexes. Note that future updates will not require a restart of the search daemon. The only reason that we have to restart it this time is because the initial indexes didn't exist. However, for future updates the indexes will be updated without any need to restart and with zero downtime because we take advantage of Sphinx's ability to "rotate" indexes:

```
# Stop the search daemon...
./nnindexer.php daemon --stop

# ...and restart it
./nnindexer.php daemon
```

8. You're done!

Overview

Below you'll find some useful information for understanding how Sphinx works and how it is integrated into Newznab.

Full vs. Delta

Sphinx is designed in such a way that every time you "index", you have to actually "re-index"; you can't just simply update the index with only the new data. However, we obviously don't want to have to re-index such a large dataset every time a new record is added. So, to get around this issue, we use "delta index" update scheme. The way this works is fairly simple; for every index we actually have two indexes: a "main" or "full" index and a "delta" index. The "main" index holds most of the indexed data, while the "delta" index only hold the data that has been added/modified since we last updated the "main" index. Fortunately, Sphinx also provides a way to merge indexes, so every so often (say once a day) we merge the "delta" into the "main". You can control this merge frequency via the "Merge Frequency" setting from the site settings page.

For more information about how this works see the Sphinx website: http://sphinxsearch.com/docs/2.0.2/delta-updates.html

Fields vs. Attributes

An important concept in Sphinx is the difference between "fields" and "attributes". "fields" store data that is directly retrievable from a search string; this is the data that make the index a "full-text" search. "attributes" contain data that gets attached to each record in the full-text index. While not directly searchable, "attributes" can be used to filter, group and sort the results returned from the search.

Deleting Releases

The situation is further complicated by the fact that removing items from the index is somewhat complicated. As a simple remedy to this, there is the "Rebuild Frequency" setting on the site settings page. This setting controls how often we do a full rebuild of the main index. When the main index is rebuilt, all of the deleted items will no longer be present in the index. It is also worth mentioning here that even though your index may contain items that have subsequently been deleted from MySQL this won't have any visible effect on the search results on Newznab's frontend. The reason that we rebuild is so that performance and integrity of the main index doesn't degrade over time.

Available Indexes

Currently there are 5 supported indexes. You can enable or disable any of them except for the main "releases" index. They are, listed in order of difficulty to index:

- releases (main)
- releasefiles
- releasenfo
- nzbs
- predb

As stated, you can choose to enable or disable any of the indexes except for "releases". In order to decide which ones to enable/disable, below you will find some information about each index which might help you make your decision.

Index: releases

This is the main index. It indexes nearly all of the data contained within the "releases", "bookinfo", "consoleinfo", "episodeinfo", "musicinfo" and "movieinfo" tables in Newznab. The "delta" index contains all releases that have been added or modified since the last time the "main" index was updated. This ensures that not just new releases are indexed, but also ones that were updated as well.

The searchable fields are:

- name
- searchname
- fromname
- tvtitle
- season
- episode
- bookinfo_title
- bookinfo_author
- bookinfo_publisher

- bookinfo_review
- consoleinfo_title
- consoleinfo_publisher
- consoleinfo_review
- episodeinfo_showtitle
- episodeinfo_eptitle
- episodeinfo_director
- episodeinfo_writer
- episodeinfo_gueststars
- episodeinfo_overview
- musicinfo_title
- musicinfo_review
- musicinfo_artist
- musicinfo_publisher
- musicinfo_tracks
- movieinfo_title
- movieinfo_tagline
- movieinfo_plot
- movieinfo_director
- movieinfo_actors
- movieinfo_genre
- predb_dirname
- predb_filename

The attributes are:

size	bigint
groupID	uint
categoryID	uint
totalpart	uint
grabs	uint
completion	uint
regexID	uint
rageID	uint
tvdbID	uint
imdbID	uint
episodeinfoID	uint
musicinfoID	uint
consoleinfoID	uint
bookinfoID	uint
preID	uint
anidbID	uint

Continued on next page

Table 7.3 – continued from previous page

reqID	uint
comments	uint
passwordstatus	uint
rarinnerfilecount	uint
haspreview	uint
guid	string
seriesfull	string
postdate	timestamp
adddate	timestamp
tvairdate	timestamp
bookinfo_genreID	uint
bookinfo_pages	uint
bookinfo_cover	uint
bookinfo_asin	string
bookinfo_url	string
bookinfo_dewey	string
bookinfo_ean	string
bookinfo_isbn	string
bookinfo_publishdate	timestamp
consoleinfo_asin	string
consoleinfo_url	string
consoleinfo_salesrank	uint
consoleinfo_platform	string
consoleinfo_genreID	uint
consoleinfo_esrb	string
consoleinfo_releasedate	timestamp
consoleinfo_cover	uint
episodeinfo_rageID	uint
episodeinfo_tvdbID	uint
episodeinfo_imdbID	uint
episodeinfo_epabsolute	uint
episodeinfo_rating	float
episodeinfo_fullep	string
episodeinfo_link	string
episodeinfo_airdate	timestamp
musicinfo_salesrank	uint
musicinfo_genreID	uint
musicinfo_cover	uint
musicinfo_asin	string
musicinfo_year	string
musicinfo_releasedate	timestamp
movieinfo_imdbID	uint
movieinfo_tmdbID	uint
movieinfo_year	uint
movieinfo_cover	uint
movieinfo_backdrop	uint
movieinfo_rating	float
movieinfo_language	string
predb_ctime	uint
predb_nuketime	uint

Continued on next page

Table 7.3 – continued from previous page

predb_filesize	`float`
predb_filecount	`uint`
predb_category	`string`
predb_nuketype	`string`
predb_nukereason	`string`

Index: releasefiles

Optional

This indexes everything in the "releasefiles" table within Newznab. An important thing to note here is that due to the nature of the query needed for this index, all the results need to be obtained in a single query. As a result, you're "releasefiles" table might become locked for an extended period of time as this index is built. However, depending on your database and hardware, this might be a non-issue for you, so it is best to test it and see what works. A solution for this might be implemented in future versions.

The searchable fields are:

- name (a concatenated list of all the file names for a given release)

There are no attributes associated with this index.

Index: releasenfo

Optional

This indexes everything in the "releasenfo" table within Newznab. Since the NFOs can be fairly large documents of text, this index take considerably longer to index than the others listed above and also requires more disk space.

The searchable fields are:

- nfo

There are no attributes associated with this index.

Index: nzbs

Optional

This indexes the contents of all the NZBs. You should think very carefully about whether or not your machine is capable of dealing with this index as it requires 2-3 orders of magnitude more disk space and processing time than all of the other indexes **combined**. With that said, this index also uses Sphinx's "real-time" indexing functionality. What that really means for you is that once you have the data indexed, you won't ever really have to re-index it (unlike the other indexes which **do not** work this way).

The searchable fields are:

- file_names (a space-concatenated string of the file names)

The attributes are:

- file_count (int)

Index: predb

Optional

If you use the `nzpre` feature and you frequently search PreDB, then this might be a worthwhile index for you. Since the `predb` table can contain many rows (3-5x as many as `releases`), this might strain your system a bit.

The searchable fields are:

- dirname
- category
- nuketype

The attributes are:

- ctime (uint)
- guid (string)
- nfoID (uint)

API

misc/sphinx/nnindexer.php

Could not open input file:

Update Scripts

This directory (`misc/update_scripts`) contains a collection of command-line utilities for updating Newznab. Whilst you can run them stand-alone for testing things out, it is intended that you should run the calling script win_scriptsrunme.bat or nix_scriptsnewznab_screen.sh which runs each of these scripts in the right order.

Updating

These scripts should be run on a frequent basis in order to stay current with the newest posts to usenet.

update_binaries.php

This script downloads new headers from the news server and puts them in the database (binaries and parts tables).

update_binaries_threaded.php

This script runs on linux only and calls the update_binaries script in 10 separate threads.

update_releases.php

This script creates releases from downloaded headers. It includes all the additional post processing which is performed as a release is formed.

update_theaters.php

This script updates the 'whats on in theaters' data from rotten tomatoes if a rotten tomatoes api key is present.

update_tvschedule.php

This script updates the tv schedule data from thetvdb.

import.php

This script is used for importing .nzb files from a path into newznab.

Maintenance

These scripts should be run occasionally.

optimise_db.php

Optimises and repairs mysql tables if necessary. Pass in the true argument to force an optimise and repair regardless of whether its necessary.

Backfilling

backfill.php

The equivalent of `update_binaries.php` but for going forwards from the group.backfilldays to the latest post. Downloads headers from usenet and puts them in the database (binaries and parts tables).

backfill_date.php

The same as `backfill.php` but goes back to a specific date passed as an argument.

backfill_threaded.php

Calls `backfill.php` with a thread for each group requiring backfilling.

CHAPTER 8

Software license for Newznab+

Summary

Personal use only (cannot be resold or distributed) Commercial use allowed Can modify source-code but cannot distribute modifications (derivative works) Software trademarks are included in the license Parts of the software are provided under separate licenses, as follows:

- jquery.js is under the MIT license
- PEAR/NNTP is under the W3C license
- rarinfo.php is under the modified BSD license
- Smarty.class.php is under the LGPL license
- Tmdb.php is under the BSD license
- powerspawn.php is under the BSD license

Terms and Conditions

1. Preamble: This Agreement governs the relationship between licensee, a private person, (hereinafter: Licensee) and newznab.com, whose principal place of business is United Kingdom (Hereinafter: Licensor). This Agreement sets the terms, rights, restrictions and obligations on using [newznab] (hereinafter: The Software) created and owned by Licensor, as detailed herein
2. License Grant: Licensor hereby grants Licensee a Personal, Non-assignable & non-transferable, Commercial, Royalty free, Including the rights to create but not distribute derivative works, Non-exclusive license, all with accordance with the terms set forth and other legal restrictions set forth in 3rd party software used while running Software.
3. Limited: Licensee may use Software for the purpose of:
   (a) Running Software on Licensee‘s Website[s] and Server[s];
   (b) Allowing 3rd Parties to run Software on Licensee‘s Website[s] and Server[s];

(c) Publishing Software‘s output to Licensee and 3rd Parties;

(d) Distribute Verbatim Copites of Software‘s output;

(e) Modify Software to suit Licensee‘s needs and specifications.

4. Personal: Licensee may not sublicense, lease, rent or otherwise allow 3rd parties to use Software, or any portions thereof, apart from being executed in any form apart from being run as a server script on Licensee's Website[s] or Server[s]
5. Non Assignable & Non-Transferable: Licensee may not assign or transfer his rights and duties under this license.
6. Commercial, Royalty Free: Licensee may use Software for any purpose, including paid-services, without any royalties
7. Including the Right to Create Derivative Works: Licensee may create derivative works based on Software, including amending Software's source code, modifying it, integrating it into a larger work or removing portions of Software, as long as no distribution of the derivative works is made
8. [Multi-]Site: Licensee may use Software on unlimited Server[s] and unlimited Website[s], for Licensee's Websites only
9. Licensor shall retain full title in Trademarks, and any trademarks and tradenames contained, including Software's names, logos, and all other intellectual property. Unless specifically stated in this license, no license shall be made to use, associate or affiliate Software with Licensee in any manner. Licensee may not use Software‘s name, tradename, trademarks or logo when distributing derivative works of software to 3rd parties.
10. Additional licenses: Portions of the The Software are based on source-code licensed under different licenses, as follows:
    - jquery.js is under the MIT license
    - PEAR/NNTP is under the W3C license
    - rarinfo.php is under the modified BSD license
    - Smarty.class.php is under the LGPL license
    - Tmdb.php is under the BSD license
11. Term & Termination: The Term of this license shall be until terminated. Licensor may terminate this Agreement, including Licensee‘s license in the case where Licensee:

    (a) became insolvent or otherwise entered into any liquidation process; or

    (b) exported The Software to any jurisdiction where licensor may not enforce his rights under this agreements in; or

    (c) Licenee was in breach of any of this license's terms and conditions and such breach was not cured, immediately upon notification; or

    (d) Licensee in breach of any of the terms of clause 2 to this license; or

    (e) Licensee otherwise entered into any arrangement which caused Licensor to be unable to enforce his rights under this License.
12. Payment: In consideration of the License granted under clause 2, Licensee shall pay Licensor a {fee}, via PayPal or any other mean which Licensor may deem adequate, and through Binpress‘ clearing system. Failure to perform payment shall construe as material breach of this Agreement.
13. Upgrades, Updates and Fixes: Licensor may provide Licensee, from time to time, with Upgrades, Updates or Fixes, as detailed herein and according to his sole discretion. Licensee hereby warrants to keep The Software up-to-date and install all relevant updates and fixes, and may, at his sole discretion, purchase upgrades, according to the rates set by Licensor. Licensor shall provide any update or Fix free of charge; however, nothing in this Agreement shall require Licensor to provide Updates or Fixes.

(a) Upgrades: for the purpose of this license, an Upgrade shall be a material amendment in The Software, which contains new features and or major performance improvements and shall be marked as a new version number. For example, should Licensee purchase The Software under version 1.X.X, an upgrade shall commence under number 2.0.0.

(b) Updates: for the purpose of this license, an update shall be a minor amendment in The Software, which may contain new features or minor improvements and shall be marked as a new sub-version number. For example, should Licensee purchase The Software under version 1.1.X, an upgrade shall commence under number 1.2.0.

(c) Fix: for the purpose of this license, a fix shall be a minor amendment in The Software, intended to remove bugs or alter minor features which impair the The Software's functionality. A fix shall be marked as a new sub-sub-version number. For example, should Licensee purchase Software under version 1.1.1, an upgrade shall commence under number 1.1.2.

14. Support: Software is provided under an AS-IS basis and without any support, updates or maintenance. Nothing in this Agreement shall require Licensor to provide Licensee with support or fixes to any bug, failure, mis-performance or other defect in The Software.

15. Bug Notification: Licensee may provide Licensor of details regarding any bug, defect or failure in The Software promptly and with no delay from such event; Licensee shall comply with Licensor's request for information regarding bugs, defects or failures and furnish him with information, screenshots and try to reproduce such bugs, defects or failures.

16. Feature Request: Licensee may request additional features in Software, provided, however, that (i) Licesee shall waive any claim or right in such feature should feature be developed by Licensor; (ii) Licensee shall be prohibited from developing the feature, or disclose such feature request, or feature, to any 3rd party directly competing with Licensor or any 3rd party which may be, following the development of such feature, in direct competition with Licensor; (iii) Licensee warrants that feature does not infringe any 3rd party patent, trademark, trade-secret or any other intellectual property right; and (iv) Licensee developed, envisioned or created the feature solely by himself.

17. Liability: To the extent permitted under Law, The Software is provided under an AS-IS basis. Licensor shall never, and without any limit, be liable for any damage, cost, expense or any other payment incurred by Licesee as a result of Software‘s actions, failure, bugs and/or any other interaction between The Software and Licesee‘s end-equipment, computers, other software or any 3rd party, end-equipment, computer or services. Moreover, Licensor shall never be liable for any defect in source code written by Licensee when relying on The Software or using The Software‘s source code.

18. Warranty:

(a) Intellectual Property: Licensor hereby warrants that The Software does not violate or infringe any 3rd party claims in regards to intellectual property, patents and/or trademarks and that to the best of its knowledge no legal action has been taken against it for any infringement or violation of any 3rd party intellectual property rights.

(b) No-Warranty: The Software is provided without any warranty; Licensor hereby disclaims any warranty that The Software shall be error free, without defects or code which may cause damage to Licensee‘s computers or to Licensee, and that Software shall be functional. Licensee shall be solely liable to any damage, defect or loss incurred as a result of operating software and undertake the risks contained in running The Software on License‘s Server[s] and Website[s].

(c) Prior Inspection: Licensee hereby states that he inspected The Software thoroughly and found it satisfactory and adequate to his needs, that it does not interfere with his regular operation and that it does meet the standards and scope of his computer systems and architecture. Licensee found that The Software interacts with his development, website and server environment and that it does not infringe any of End User License Agreement of any software Licensee may use in performing his services. Licensee hereby waives any claims regarding The Software's incompatibility, performance, results and features, and warrants that he inspected the The Software.

19. No Refunds: Licensee warrants that he inspected The Software according to clause 7(c) and that it is adequate to his needs. Accordingly, as The Software is intangible goods, Licensee shall not be, ever, entitled to any refund, rebate, compensation or restitution for any reason whatsoever, even if The Software contains material flaws.
20. Indemnification: Licensee hereby warrants to hold Licensor harmless and indemnify Licensor for any lawsuit brought against it in regards to Licensee‘s use of The Software in means that violate, breach or otherwise circumvent this license, Licensor’s intellectual property rights or Licensor’s title in The Software. Licensor shall promptly notify Licensee in case of such legal action and request Licensee‘s consent prior to any settlement in relation to such lawsuit or claim.
21. Governing Law, Jurisdiction: Licensee hereby agrees not to initiate class-action lawsuits against Licensor in relation to this license and to compensate Licensor for any legal fees, cots or attorney fees should any claim brought by Licensee against Licensor be denied, in part or in full.

CHAPTER 9

Todo

This page will list of items that are yet to be included or completed.

CHAPTER 10

Glossary

Amazon A provider of metadata for releases. You need an API key to use this service.

imdb A provider of metadata for movie releases. This information is obtained via scraping.

tmdb A provider of metadata for movie releases. A shared API key is used for this service.

nfo A text file containing information about a release.

par A parity file for completing usenet downloads which contain missing parts.

rotten tomatoes A provider of metadata for movie releases. You need an API key to use this service.

NZB An NZB is an XML-based file format for retrieving posts from NNTP (Usenet) servers. For more information about NZB files, see NZB.

regex Short for "regular expression", which provides a concise and flexible means for matching and extracting strings of text. "Regexs" are a central concept in Newznab as they provide the foundation for matching and grouping releases. More information can be found REGEX.

TvRage A provider of metadata for releases. The website is www.tvrage.com.

CHAPTER 11

Developer Docs

These are the low-level developer docs for Newznab's source code. They are aimed at developers or other users who want to get their hands dirty and modify Newznab's source code or just simply learn more about the Newznab's internals.

lib/TMDb.php

Could not open input file:

lib/adminpage.php

Could not open input file:

lib/amazon.php

Could not open input file:

lib/anidb.php

Could not open input file:

lib/backfill.php

Could not open input file:

lib/binaries.php

Could not open input file:

lib/book.php

Could not open input file:

lib/category.php

Could not open input file:

lib/console.php

Could not open input file:

lib/content.php

Could not open input file:

lib/flist.php

Could not open input file:

lib/forum.php

Could not open input file:

lib/genres.php

Could not open input file:

lib/groups.php

Could not open input file:

lib/install.php

Could not open input file:

lib/installpage.php

Could not open input file:

lib/menu.php

Could not open input file:

lib/movie.php

Could not open input file:

lib/music.php

Could not open input file:

lib/nfo.php

Could not open input file:

lib/nntp.php

Could not open input file:

lib/nzb.php

Could not open input file:

lib/nzbget.php

Could not open input file:

lib/nzbinfo.php

Could not open input file:

lib/nzbvortex.php

Could not open input file:

lib/page.php

Could not open input file:

lib/parsing.php

Could not open input file:

lib/postprocess.php

Could not open input file:

lib/powerprocess.php

Could not open input file:

lib/predb.php

Could not open input file:

lib/recaptchalib.php

Could not open input file:

lib/releasecomments.php

Could not open input file:

lib/releaseextra.php

Could not open input file:

lib/releasefiles.php

Could not open input file:

lib/releaseimage.php

Could not open input file:

lib/releaseregex.php

Could not open input file:

lib/releases.php

Could not open input file:

lib/sabnzbd.php

Could not open input file:

lib/site.php

Could not open input file:

lib/sitemaps.php

Could not open input file:

lib/sphinx.php

Could not open input file:

lib/spotnab.php

Could not open input file:

lib/thetvdb.php

Could not open input file:

lib/tvinfo.php

Could not open input file:

lib/tvmaze.php

Could not open input file:

lib/usermovies.php

Could not open input file:

lib/users.php

Could not open input file:

lib/userseries.php

Could not open input file:

lib/util.php

Could not open input file:

lib/zipfile.php

Could not open input file:

lib/framework/basepage.php

Could not open input file:

lib/framework/cache.php

Could not open input file:

lib/framework/db.php

Could not open input file:

CHAPTER 12

Indices and tables

- genindex
- modindex
- *Glossary*
- search

Index