
Exaile Documentation

Release 3.4.5+544bd5b

Adam Olsen <arolsen@gmail.com>, Johannes Sasongko <sasongko@gmail.com>

May 24, 2017

Contents

1	Exaile Users Guide	3
1.1	Installation dependencies	3
1.2	Installation	4
1.3	Windows Notes	5
1.4	OSX Notes	6
1.5	Frequently Asked Questions	8
1.6	Support	8
2	Exaile Developer Documentation	11
2.1	Contributing to Exaile	11
2.2	Exaile plugin development guide	11
2.3	Code guidelines	20
2.4	Release process	23
2.5	Debugging tips	25
2.6	Exaile API Docs	25
2.7	Index + search	55
	Python Module Index	57

Exaile is a music player with a simple interface and powerful music management capabilities. Features include automatic fetching of album art, lyrics fetching, streaming internet radio, tabbed playlists, smart playlists with extensive filtering/search capabilities, and much more.

Exaile is written using python and GTK+ and is easily extensible via plugins. There are over 50 plugins distributed with Exaile that include advanced track tagging, last.fm scrobbling, support for portable media players, podcasts, internet radio such as icecast and Soma.FM, ReplayGain, output via a secondary output device (great for DJs!), and much more.

The great thing about Exaile is that it's really simple to use, so we haven't written instructions on how to use it! ;)

Installation dependencies

The official installers for Windows and OSX should already come with/install the necessary dependencies for base functionality to work.

Note: Plugin dependencies should be listed in their description

Dependencies

Essential:

- python (≥ 2.6)
- pygst 0.10
- gstreamer-plugins-good 0.10
- mutagen (≥ 1.10)
- dbus-python
- pygtk (≥ 2.17)
- pygobject (≥ 2.18 , built with gio support)

Translation:

- gettext

Documentation:

- sphinx

- help2man

Device detection (one of the following):

- hal
- udisks
- udisks2

CD info:

- python-cddb

Scalable icons:

- librsvg2

Installation

Windows

On Windows, using the official Windows installation program is recommended. If you want to build your own installer, see *Step three: Linux + Windows*.

Install on OSX

On OSX, using the official OSX installation DMG is recommended. See *OSX Notes*

If you want to build your own installer, see *Step four: OSX*.

Linux/*nix

On *nix-based systems (but not OSX), run the following to install:

```
$ make
# make install
```

The “make” step is optional and only compiles the modules and translations.

There is also an `install_no_locale` target if you wish to install without translations.

Environment Variables

Environment variables that affect “make install”:

LIBINSTALLDIR The lib directory to be appended to PREFIX. Default: `/lib`

PREFIX The installation prefix. Default: `/usr/local` Note: If this default is changed, you may need to set the `XDG_DATA_DIRS`

environment variable to include the changed path. See [1].

XDGCONFDIR The directory to install system-wide config files in, following xdg spec. Default: `/etc/xdg` Note: If this default is changed, you may need to set the `XDG_CONFIG_DIRS`

environment variable to include the changed path. See [1].

DESTDIR

Destination directory to put the file structure under. Mainly of use for packagers.

Default: not set (/)

DEFAULTARGS Default arguments that the ‘exaile’ launcher script (installed to \$PREFIX/bin/exaile) will pass to exaile on startup

Additional Resources: [1] - XDG Base Directory Specification

<http://standards.freedesktop.org/basedir-spec/basedir-spec-0.6.html>

Windows Notes

We’re excited to announce that Exaile has official support for Windows platforms as of version 3.3.0, and is distributed in an official installer program.

Exaile (and its installer) has been tested on:

- Windows XP SP3
- Windows 7 x64 SP1
- Windows 8.1
- Python 2.7.8 (32-bit and 64-bit)
- GStreamer.com SDK 2013.6 (32-bit and 64-bit)

Exaile may work with the OSSBuild SDK and Python 2.6.6, but this is not a recommended configuration.

Running Exaile

If you installed exaile using the Windows installer, you can find icons to launch Exaile in the Start Menu under “All Programs” -> “Exaile”.

Alternatively, you can directly launch “exaile_win.py”.

Exaile DirectSound plugin (libgstexailedirectsoundsink.dll)

Exaile currently includes a custom DirectSound GST plugin that is compiled against the GStreamer.com SDK git repository as of September 2012.

The reason a custom plugin is included with Exaile is because GST mainline does not currently support selecting the audio output for the DirectSound plugin. Another project created a patch to add this functionality, and you can find the bug at https://bugzilla.gnome.org/show_bug.cgi?id=584980

The plugin is LGPL, and our modifications to the GST source are in `toolswin-installdirectsound.patch`

If you do not wish to use the Exaile directsound plugin, you can change the audiosink in Preferences->Playback

GStreamer.com SDK bugs

2012.9:

“libssp-0.dll is missing”

GStreamer.com SDK 2012.9 was not packaged properly, and is missing a file that comes with MinGW. Obtain the file from somewhere, and place it in the bin directory of your gstreamer.com SDK

See https://bugs.freedesktop.org/show_bug.cgi?id=54710

An x86 version of this file is distributed with Exaile in the directory toolswin-installer, and should be installed automatically by the Exaile installer. If it was not installed for you, please file a bug with Exaile.

2012.7:

2012.7 does not have complete python bindings shipped with it, and is missing libpyglib-2.0-python.pyd . You can download it at https://bugs.freedesktop.org/show_bug.cgi?id=52983

Possible Errors and Solutions

The GStreamer.com SDK environment is a bit fragile and can sometimes cause mysterious problems when launching exaile. Some of the known issues that have been reported are below:

“ImportError: could not import gobject (error was: ‘DLL load failed: The specified module could not be found.’)”

This has been observed when there are multiple versions of Glib/GTK on your system. Possible problems may include:

- You have a GTK living somewhere else that isn’t compatible with the GStreamer.com SDK. This happens if you have GStreamer SDK and PyGTK installed on the same machine.
- You may have MinGW installed somewhere else with older DLLs that are not compatible with the GStreamer.com SDK

What you can do to diagnose the problem:

- Get depends.exe: <http://www.dependencywalker.com/>
- Modify exaile.bat to launch depends instead of exaile (that way the environment is identical)
- Load %GSTREAMER_SDK_ROOT_X86%libpython2.7site-packagesgst-0.10gst_gst.pyd in depends.exe (the pyd file is really a DLL), and see what errors it reports. This should be able to tell you what DLL is conflicting, and what its path is

Bug Reporting

If exaile crashes, you should be able to find a logfile from the last time exaile was launched in the following directories:

```
Windows XP: %USERPROFILE%\Local Settings\Application Data\exaile\logs
Windows 7+: %USERPROFILE%\AppData\Local\exaile\logs
```

If you run into any bugs not mentioned in this document, please report them to us via github at <https://github.com/exaile/exaile/issues> . Please include any relevant logfile snippets.

OSX Notes

We’re excited to announce that Exaile has experimental yet official support for OSX as of version 3.4.0, and is distributed in an official installer program. Current support for OSX is still considered experimental, and probably will continue to be until we migrate to GTK3.

Exaile has only been tested extensively on OSX 10.8 Mountain Lion. It may work on other versions of OSX, but they have not been tested.

Requirements

On OSX, Exaile requires the GStreamer SDK Runtime to be installed, otherwise it will not function.

Download the SDK runtime for OSX here:

<http://docs.gstreamer.com/display/GstSDK/Installing+on+Mac+OS+X>

Or use this direct download link:

<http://cdn.gstreamer.com/osx/universal/gstreamer-sdk-2013.6-universal.pkg>

The default installation will allow the Exaile UI to work correctly, and should support playing many types of audio formats.

Install process

First, make sure that you have installed the GStreamer SDK.

Next, install Exaile like you would install any other OSX application.

- Open the DMG file
- Drag the Exaile application to the “Applications” folder

MP3 support

By default GStreamer SDK does not install support for MP3 files or certain other formats because of licensing issues. If you require support for these types of files, use the following procedure when installing GStreamer SDK.

- Run the GStreamer installation package
- Click continue
- Click “Agree” to agree to the license agreement
- Click “Install for all users of this computer”, and click “Continue”
- Click “Customize”
- **Ensure the following package names are checked in addition to the defaults:**
 - GStreamer codecs under the GPL license
 - GStreamer restricted codecs
 - GStreamer plugins for network protocols
- Click Install, and it should do the install for you

Known issues

The OSX version is about as functional as the Windows version, so most things will work without any problems. The CD plugin and other device plugins will not work on OSX.

Transparency may not work.

Keyboard shortcuts aren't mapped to the option/command keys, and are still mapped to CTRL.

GTK2 seems to have a lot of odd problems on OSX, such as combo boxes immediately switching when clicked. We anticipate that these will not be fixed until we migrate exaile to GTK3.

Frequently Asked Questions

Error “no suitable plugin found” when playing a (.mp3, .m4a, etc)

Exaile currently uses GStreamer 0.10 to decode/play audio files, and does not directly decode audio itself. For playback to work, you need to have the correct GStreamer plugins installed.

Note: For Linux users, you may find that other GStreamer programs can play specific file type, but Exaile cannot. Check to make sure that the correct plugins are installed for GStreamer 0.10, as other players may be using GStreamer 1.x instead.

File tags don't update when I change them using an external program

When setting up your collection, ensure that the 'monitored' and 'scan on startup' options are checked, otherwise Exaile may become out of sync with your collection if it is modified by external programs.

To detect that the file has changed, Exaile checks to see if the modification time of the file has changed. This makes rescans much quicker.

Some third-party taggers (notably EasyTag) have options where they do not update the modification time of the file when they change the contents of the file. In these cases, Exaile may not be able to detect that the file has changed. To remain compatible with Exaile (and other media players), you should configure your tagger to update the modification time.

Note: As of Exaile 3.4.2, there is a menu option called 'Rescan Collection (slow)' which will force a rescan of every file in your collections, regardless of whether the modification time has changed. This should detect any changes to your collection.

How do I enable output to a secondary soundcard?

A: Enable the 'preview device' plugin. You can change the secondary output device settings by editing the plugin's settings.

Support

If you believe you have found a bug, you can file bugs at [Exaile's github issue tracker](#). You will need to register for a github account before you can post a bug report.

The following are ways you can get support or questions to answers you might have about using or developing Exaile.

Mailing lists

We run mailing lists for various purposes.

- [exaile-users](#) - A list for people who use Exaile. Open to discussion between users to communicate and help each other. If you're having trouble using Exaile, ask here.
- [exaile-devel](#) - Our primary list for general development discussion. Questions about the code, plugin development, packaging, etc. should go here.

IRC

Exaile developers can be reached on [#exaile](#) channel on [Freenode](#). The channel is not very active, but if you stick around long enough someone will probably answer your question – think in terms of email response time)

Contributing to Exaile

The exaile team is always looking for others to help contribute to exaile in various ways:

- Bugfixes
- Documentation updates
- Translations on <https://hosted.weblate.org/projects/exaile/master/>
- New features + plugins

The best way to contribute is to submit patches/etc via pull request on github.

If you think your bug report/request is being ignored, it probably isn't. All of the Exaile developers work on this project in their spare time, and so we don't always have time to work on your problems. We *do* try to push good patches as soon as we can, however. Ping the bug report, or leave a message on #exaile if we haven't at least made an initial response, sometimes bug report emails can get lost in the noise.

Exaile plugin development guide

Note: these instructions always track current Exaile trunk, and may not be fully compatible with stable releases. It is recommended that you develop plugins against trunk, so that you can submit patches to trunk if need be during the creation of your plugin, and so that your plugin can easily be merged into trunk when it is ready.

Style

If you plan to submit your plugin for inclusion in Exaile, please read and follow the guidelines in the *Code guidelines*

Basic plugin structure

Plugins in Exaile 3.x+ are handled slightly differently than in the past. Each plugin has its own directory in `~/.local/share/exaile/plugins/`. In order for your plugin to be recognized as valid by Exaile, it needs to have at least two files in the plugin directory (`~/.local/share/exaile/plugins/myplugin/`):

- `__init__.py`
- `PLUGININFO`

The format of the `PLUGININFO` is as follows:

```
Version='0.0.1'
Authors=['Your Name <your@email.com>']
Name=_('Plugin Name')
Description=_('Something that describes your plugin. Also mention any extra_
↳dependencies.')
Category=_('Development')
```

The following two attributes are optional:

- *Platforms* - A list of the platforms your plugin works on. If you have no specific requirements, omitting this argument or using an empty list is fine. The values of the list are the `sys.platform` value.
- *RequiredModules* - If your plugin requires a module that may not be present on all installations of exaile (eg, modules like `gtk` are always going to be present, so don't need to specify those), then you should specify it here.

Note: Name and Description are what show up in the plugin manager. Category is used to list your plugin alongside other plugins. Platforms and RequiredModules are used to filter out the plugin on inappropriate platforms.

Before Exaile 3.4, `__init__.py` was required to define at least two methods, `enable()` and `disable()`. However, Exaile 3.4 introduced a new way to write plugins which will eliminate a lot of unnecessary boilerplate for plugin authors. We will use this model below:

```
class MyPlugin(object):

    def enable(self, exaile):
        print('You enabled me!')

    def disable(self, exaile):
        print('I am being disabled')

plugin_class = MyPlugin
```

For many types of plugins, this might be enough. However, there are other optional methods you can define in your plugin object.

- *on_gui_loaded* - This will be called when the GUI is ready, or immediately if already done
- *on_exaile_loaded* - This will be called when exaile has finished loading, or immediately if already done
- *teardown* - This will be called when exaile is unloading

These methods may be necessary for your plugin because plugins can only access Exaile's infrastructure when Exaile itself finishes loading. The first `enable()` method is called when Exaile is partway through loading. But since we can't do anything until Exaile finishes loading, we can add *on_exaile_loaded* to our object that is called when Exaile finishes loading. Some plugins need to modify state earlier in the startup process, hence the need for this separation.

The *exaile* object in the above example is an instance of a class called *Exaile*, which is defined in `xl/main.py`. This class is a base for everything in the program.

You can get a handle on various objects in *Exaile* by looking at the members of this class.

Something (slightly) more useful

Here is an example of a plugin that will, when a track is played, show the track information in a `MessageDialog`. It demonstrates a callback on an event, and getting the `gtk.Window` object of *Exaile* to use as a parent for a `MessageBox`.

The `PLUGININFO` is as follows:

```
Version='0.0.1'
Authors=['Me <me@internets.com>']
Name='Tutorial Plugin'
Description='Plugin to demonstrate how to make a plugin.'
```

and the `__init__.py` is as follows

```
'''
    This plugin will show an obnoxious gtk.MessageDialog that
    wont disappear, when a track is played. The MessageDialog
    will contain the information of the currently playing track.
'''

from xl import event
import gtk

# The main functionality of each plugin is generally defined in a class
# This is by convention, and also makes programming easier
class TutorialPlugin(object):

    def enable(self, exaile):
        '''This method is called when the plugin is loaded by exaile'''

        # We need a reference to the main Exaile object in order to set the
        # parent window for our obnoxious MessageDialog
        self.exaile = exaile

    def disable(self, exaile):
        '''This method is called when the plugin is disabled. Typically it is used for
        removing any GUI elements that we may have added in _enable()'''
        self.show_messagebox("Byebye!")

    def on_exaile_loaded(self):
        '''Called when exaile is ready for us to manipulate it'''

        #The reason why we dont use show_messagebox here is it hangs the GUI
        #which means it would hang Exaile as soon as you restart, because all
        #enabled plugins are loaded on start.
        print('You enabled the Tutorial plugin!')

        # Add a callback for the 'playback_track_start' event.
        # See xl/event.py for more details.
        event.add_callback(self.popup_message, 'playback_track_start')

    def popup_message(self, type, player, track):
```

```
# The Track object (defined in xl/track.py) stores its data in lists
# Convert the lists into strings for displaying
title = track.get_tag_display('title')
artist = track.get_tag_display('artist')
album = track.get_tag_display('album')
message = "Started playing %s by %s on %s" % (title, artist, album)
self.show_messagebox(message)

def show_messagebox(self, message):
    # This is the obnoxious MessageDialog. Due to (something to do with threading?
    ↪)
    # it will steal, and never relinquish, focus when it is displayed.
    dialog = gtk.MessageDialog(self.exaile.gui.main.window, 0, gtk.MESSAGE_INFO, ↪
    ↪gtk.BUTTONS_OK, message)
    dialog.run()
    dialog.destroy()

plugin_class = TutorialPlugin
```

Have a look in the comments for an explanation of what everything is doing.

Adding a track to the Playlist

This is relatively simple. A Playlist consists of the actual graphical representation of a playlist (see `xlgui/playlist.py`) and its underlying Playlist object (see `xl/playlist.py`). Any changes made to the underlying playlist object are shown in the graphical representation. We will be appending Track objects to this underlying playlist.

First you need to get a handle on the underlying Playlist:

```
playlist_handle = exaile.gui.main.get_selected_playlist().playlist
```

Then, you need to create a Track object (defined in `xl/track.py`). The method to do this from a local file versus a URL is slightly different.

For a local source:

```
from xl import trax
path = "/home/user/track.ogg" #basically, just specify an absolute path
myTrack = trax.Track(path)
```

For a url:

```
from xl import trax
url = "http://path/to/streaming/source"
myTrack = trax.get_tracks_from_uri(url)
```

You can set the track information like so:

```
myTrack.set_tag_raw('title', "Cool Track")
myTrack.set_tag_raw('artist', "Cool Artist")
myTrack.set_tag_raw('album', "Cool Album")
```

Once you have a Track object, and a handle on the Playlist you would like to add the track to, you can proceed to add the track:

```
playlist_handle.add(myTrack)
```

Note that `get_tracks_from_uri()` returns a list, so you will need to use the method for adding multiple tracks if your Track object was created this way. You can also create your own list of Track objects and add them all in one go like this too:

```
playlist_handle.add_tracks(myTrack)
```

This is pretty much all you need to do to add a track to the playlist. An example in a plugin might be:

```
from xl import event, trax

class PlaylistExample(object):

    def enable(self, exaile):
        self.exaile = exaile

    def disable(self, exaile):
        pass

    def on_gui_loaded(self):
        self.playlist_handle = self.exaile.gui.main.get_selected_playlist().playlist

        local_tr = self.create_track_from_path('/home/user/track.ogg')
        remote_tr = self.create_track_from_url('http://site.com/track.ogg')
        self.add_single_to_playlist(local_tr)
        self.add_multiple_to_playlist(remote_tr)

    def create_track_from_path(self, path):
        return trax.Track(path)

    def create_track_from_url(self, url):
        return trax.get_tracks_from_uri(url)

    def add_single_to_playlist(self, track):
        self.playlist_handle.add(track)

    def add_multiple_to_playlist(self, tracks):
        self.playlist_handle.add_tracks(tracks)

plugin_class = PlaylistExample
```

You can do more things when adding a track than simply specifying a track object to add, see the methods in the class `Playlist` (`xl/playlist.py`) for more details.

Adding another page to the left-hand Notebook

This is done pretty easily. Basically, you need to subclass `xlgui.panel.Panel` and register a provider advertising your panel.

The subclass needs to have two things:

- *ui_info* - This defines the location of the `.glade` file that will be loaded into the notebook page (This file must be in `Gtk.Builder` format, not `glade` format)
- *name* - This is the name that will show on the notebook page, such as “MyPlugin”

```
from xl import providers
from xlgui import panel

# Note: The following uses the exaile object from the enable() method. You
# might want to call this from the on_gui_loaded function of your plugin.
page = MyPanel(exaile.gui.main.window)
providers.register('main-panel', page)

# to remove later:
providers.unregister('main-panel', page)

class MyPanel(panel.Panel):

    #specifies the path to the glade file (must be in Gtk.Builder format) and the name_
    ↪ of the Root Element in the glade file
    ui_info = (os.path.dirname(__file__) + "mypanel_gladefile.glade",
    ↪ 'NameOfRootElement')

    def __init__(self, parent):
        panel.Panel.__init__(self, parent)

        #This is the name that will show up on the tab in Exaile
        self.name = "MyPlugin"

        #typically here you'd set up your gui further, eg connect methods to signals_
    ↪ etc
```

That's pretty much all there is to it. To see an actual implementation, have a look at `xlgui/panel/collection.py` or take a look at the Jamendo plugin.

Setting the cover art for a track

This is done by subclassing `CoverSearchMethod` and adding an instance of the subclass to the existing list. When Exaile plays a track with no cover, it uses all the methods in its `CoverSearchMethod` list to try and find a cover.

A `CoverSearchMethod` must define:

- *name* - The name of the `CoverSearchMethod`, used for removing it from the list once it's been added
- *type* - The type of the `CoverSearchMethod` (local, remote)
- *find_covers(self, track, limit=-1)* - This is the method that is called by Exaile when it utilises the `CoverSearchMethod`. This method must return an absolute path to the cover file on the user's harddrive.

Here is an example `CoverSearchMethod` (taken from the Jamendo plugin). It searches Jamendo for covers, downloads the cover to a local temp directory and returns the path to the downloaded cover.

```
import urllib
import hashlib
from xl.cover import CoverSearchMethod, NoCoverFoundException

class JamendoCoverSearch(CoverSearchMethod):
    name = 'jamendo'
    type = 'remote'

    def __init__(self):
        CoverSearchMethod.__init__(self)
```

```

def find_covers(self, track, limit=-1):
    jamendo_url = track.get_loc_for_io()

    cache_dir = self.manager.cache_dir
    if (not jamendo_url) or (not ('http://' and 'jamendo' in jamendo_url)):
        raise NoCoverFoundException

    #http://stream10.jamendo.com/stream/61541/ogg2/02%20-%20PieRref%20-
    ↪%20Hologram.ogg?u=0&h=f2b227d38d
    split=jamendo_url.split('/')
    track_num = split[4]
    image_url = jamapi.get_album_image_url_from_track(track_num)

    if not image_url:
        raise NoCoverFoundException

    local_name = hashlib.sha1(split[6]).hexdigest() + ".jpg"
    covername = os.path.join(cache_dir, local_name)
    urllib.urlretrieve(image_url, covername)

    return [covername]

```

You can then add it to the list of CoverSearchMethods for Exaile to try like so:

```
exaile.covers.add_search_method(JamendoCoverSearch())
```

And remove it like so:

```
exaile.covers.remove_search_method_by_name('jamendo')
```

Make strings translatable

Every message should be written in English and should be translatable. The following example shows how you can make a string translatable:

```

from xl.nls import gettext as _
print _('translatable string')

```

Saving/Loading arbitrary settings

This is quite easy. It's probably quicker to just show some code instead of trying to explain it:

```

from xl import settings

#to save a setting:
setting_value = 'I am the value for this setting!'
settings.set_option('plugin/pluginname/settingname', setting_value)

#to get a setting
default_value = 'If the setting doesnt exist, I am the default value.'
retrieved_setting = settings.get_option('plugin/pluginname/settingname', default_
↪value)

```

That's all there is to it. There is a few restrictions as to the datatypes you can save as settings, see xl/settings.py for more details.

Searching the collection

The following method returns an list of similiar tracks to the current playing track:

```
exaile.dynamic.find_similar_tracks(exaile.player.current, 5) #the second optional_  
↪argument is the limit
```

This method returns an list of tuples, which consist of the match rate and the artist's name:

```
exaile.dynamic.find_similar_artists(exaile.player.current)
```

If you would like to search the collection for a specific artist, album or genre, you can use the following code:

```
from xl.trax import search  
  
artist = 'Oasis'  
tracks = [x.track for x in search.search_tracks_from_string(  
    exaile.collection, ('artist=="%s"%artist))]  
  
genre = 'pop'  
tracks = [x.track for x in search.search_tracks_from_string(  
    exaile.collection, ('genre=="%s"%genre))]  
  
album = 'Hefty Fine'  
tracks = [x.track for x in search.search_tracks_from_string(  
    exaile.collection, ('album=="%s"%album))]
```

You can search the collection also for different assignments, like the last played tracks, the most recently added tracks or the tracks, which were played most often. Here you see an example to display the most recently added tracks:

```
from xl.trax import search  
from xl.trax.util import sort_tracks  
  
tracks = [x.track for x in search.search_tracks_from_string(exaile.collection, ('!  
↪%s==__null__' % '__last_played'))]  
tracks = sort_tracks(['__last_played'], tracks, True) #sort the tracks by the last_  
↪playing
```

The other keywords are `__date_added` and `__playcount`

Exaile D-Bus

Here is a simple example how to use the D-Bus object:

```
#!/usr/bin/env python  
  
import sys, dbus  
import Image  
from StringIO import StringIO  
  
def test_dbus():  
    bus = dbus.SessionBus()  
    try:  
        remote_object = bus.get_object("org.exaile.Exaile", "/org/exaile/Exaile")  
        iface = dbus.Interface(remote_object, "org.exaile.Exaile")  
        if iface.IsPlaying():  
            title = iface.GetTrackAttr("title")
```

```

print 'Title: %s' % title
album = iface.GetTrackAttr("album")
print 'Album: %s' % album
artist = iface.GetTrackAttr("artist")
print 'Artist: %s' % artist
genre = iface.GetTrackAttr("genre")
print 'Genre: %s' % genre
dbusArray = iface.GetCoverData()
coverdata = "".join(chr(byte) for byte in dbusArray)
if coverdata:
    im = Image.open(StringIO(coverdata))
    im.show()
else:
    print "Exaile is not playing."
except dbus.exceptions.DBusException:
    print "Exaile is not running."

if __name__ == "__main__":
    test_dbus()

```

Please check out `xl/xldb.py` for further method signatures.

Playback events

Since playback events can occur far before the main GUI object or even the *exaile* object is loaded, connecting to them in advance is required. To do this, in your `__init__` method:

```
event.add_callback(self.on_playback_player_start, 'playback_player_start')
```

Distributing the Plugin

Create a Plugin Archive

Basically, you just need to tar up your plugin's directory, and rename the tarfile to `<name_of_plugin_directory>.exz`

You will need to develop your plugin with a similar hierarchy to the following:

```

root --
  \ -- __init__.py
  \ -- PLUGININFO
  \ -- data
    \ -- somefile.glade
    \ -- somefile.dat
  \ -- images
    \ -- somefile.png

```

The archive should be named with the extension `.exz`. The name of the `plugin.exz` file needs to match the name of the plugin directory.

So in the above example, you would need to call your plugin `root.exz` in order for it to be accepted by Exaile.

`exz` files can optionally be compressed, using either `gzip` or `bzip2`. the extension remains the same.

This is all you need to do to make a plugin archive.

Exaile API

Now you know the basics about programming plugins for Exaile, but there are many more useful classes you may need. You can get an overview about the classes and their use by going through the *Exaile API Docs*.

Building your own version of this documentation

In Ubuntu you have to install the package *python-sphinx*. Then you can run the following command in a terminal:

```
$ cd doc && make html
```

You'll find the documentation in `doc/_build/html`.

Code guidelines

Page to hold style and practice guidelines for contributions to Exaile. Patches to make the existing core codebase follow these guidelines are always welcome and a good way to start learning about the internal workings of Exaile.

Note that this document will generally reflect the 'trunk' version of Exaile, and may not be fully applicable to stable releases. If in doubt, ask!

Basic Style

- Use 4 spaces for indents, no tabs.
- Avoid lines >80 characters. Try to insert sensible line breaks to accomplish this
- In general, PEP 8 applies: <http://www.python.org/dev/peps/pep-0008/>
- Keep imports on one line each to make sure imports cannot be missed:

```
# Not recommended
import gobject, gtk, threading

# Preferred
import gobject
import gtk
import threading
```

- The same goes for module imports, here parentheses can be used:

```
# Not recommended
from threading import Event, Thread, Timer

# Preferred
from threading import (
    Event,
    Thread,
    Timer
)
```

- Always write out variable names to keep them descriptive. Thus *notebook_page* is to be preferred over *nb*.
 - Exceptions:

- * Names which are prone to spelling mistakes like *miscellaneous* and *utilities*. Here *misc* and *util* are perfectly fine.
 - * If a very-long-named (like `foooooo.bar_baz_biz_boz`) variable or function needs to be accessed by a large percentage of lines in a small space, it may be shortened as long as 1) the name it is shortened to is consistent across all uses of this shortcut, and 2) the shortcut is limited in scope to just the area where it is used repeatedly. If in doubt, do NOT use this exception.
- Try to group related methods within a class, this makes it easier to debug. If it's a particularly significant group of methods, mark them with a triple-comment at the beginning and end, like so:

```
### Methods for FOOBAR ###
## more-detailed description (if needed)
def meth1(self):
    ...

### End FOOBAR ###
```

- The closing triple-comment may be omitted if at the end of a class or if another triple-comment starter comes after it.
- If you need a collection of constants for some purpose, it is recommended to use the `enum` function from `xl.common` to construct one. The constant type should be UpperCamelCase, the possible values UPPERCASE:

```
from xl.common import enum

ActionType = enum(ADD='add', EDIT='edit', ...)

# ...

if action.type == ActionType.EDIT:
    # ...
```

Documentation

- Always add docstrings to your public classes, methods and functions.
- Follow the [Sphinx](#) format for documentation within docstrings.

Events and Signals

- Items internal to Exaile (ie. anything under `xl/`) should generally prefer `xl.event` over `gobject` signals. Items that tie deeply into the (GTK) UI should prefer `gobject` signals over `xl.event`.
- Keep in mind all events are synchronous - if your callback might take a while, run it in a separate thread.
- - * Make sure that every access to GTK UI components is run in the GTK main thread. Otherwise unpredictable issues can occur including crashes due to cross-thread access.** This can be accomplished by running the specific code through the `glib.idle_add` function. A typical mistake:

```
def __init__(self):
    """
        Set up a label in the GTK main thread and
        connect to the playback_track_start event
    """
    self.label = gtk.Label()
    event.add_callback(self.on_playback_track_start, 'playback_track_start
    ...)
```

```
def on_playback_track_start(event, player, track):
    """
        Serious problem: this event is run in a
        different thread, a crash is likely to occur
    """
    self.label.set_text(track.get_tag_display('title'))
```

- Event names should be all lower-case, using underscores to separate words.
 - Names should be prefixed by the general name indicating the category or sender of the event. For example, events sent from *xl.player* start with a *playback_* prefix.
 - The remainder of the name should indicate what action just happened. eg. *playback_player_pause*.
 - The data sent in an event should be whatever piece (or pieces) of data are most relevant to the event. For example, if the event is signaling that a state has changed, the new state should be sent, or if the event indicates that an item was added, the new item should be sent.
- Callbacks for *gobject* and *xl.event* should always be named “*on_*” + the name of the event. This avoids confusion and draws a line between regular methods and signal/event callbacks.
- If you need to handle the same signal/event for multiple objects but differently (as in: different callbacks), include the name of the object in the callback name. Thus the event “*clicked*” for the *gtk.Button* “*play_button*” would become “*on_play_button_clicked*”. A small exception to this rule is when a word would be repeated. Thus “*on_play_button_press_event*” should be preferred over “*on_play_button_button_press_event*” for the “*button-press-event*” signal of the button.
- If you use [[<http://www.pygtk.org/docs/pygtk/class-gtkbuilder.html>]] for UI descriptions, apply the rules above, make the callbacks methods of your class and simply call *gtk.Builder.connect_signals(self)*

Managed object access

- To keep classes interchangeable, try to make use of existing signals/events wherever possible. Avoid reaching deeply into property hierarchies under all circumstances. This is bound to break sooner than later.
- If you need access to the main *exaile* object, call *xl.main.exaile()*, if you need access to the main GUI object, call *xlgui.get_controller()*, for the main window *xlgui.main.mainwindow()*
- Many systems are already ported to singleton managers. Examples are *xl.covers* and *xlgui.icons*. Simply use their *MANAGER* property to access them.

GUI

- Use *.ui* files to define most widgets - reduces code clutter. A lot of basic structure can be easily prepared with the *Glade* interface designer, especially objects where cell renderers and models are involved.
- Try to avoid dialogs, as they are intrusive and users generally don't read them anyway. Inline alternatives like *gtk.InfoBar* and its convenience wrapper *xlgui.widgets.dialogs.MessageBar* are much more effective.

Logging

- Messages should
 - Be short but descriptive.
 - Be proper English sentences, minus the period.

- Happen after the thing they are logging, UNLESS the thing might take a while, in which case it may be printed before, with a confirmation after the action completes.
 - * The tense of the message should match when it's sent - if after the action, use the past tense ("Logged into Audioscrobbler"), if before, use the present(?) tense ("Logging into audioscrobbler...").
 - * Messages which are present tense may use an ellipsis ("...") to indicate the different state more clearly than by tense alone.
- Not be given prefixes to identify module, as `-debug` will automatically add module names. It is acceptable to use related names in the message to increase clarity however. For example, "Logged into Audioscrobbler" is much clearer than "Logged in", but "Audioscrobbler: Logged in" is not acceptable.
- There are 4 standard logging levels built into Exaile, their names and purpose are as follows:
 - DEBUG - A significant internal event happened. Not shown by default.
 - INFO - A major but expected event happened.
 - WARNING - Something suboptimal happened. Exaile will continue to work properly but some features may be unavailable.
 - ERROR - A critical error occurred. Exaile was unable to perform a requested action and may be in an inconsistent state if the error was not fully handled.
- When writing messages, please run both with and without `-debug` to ensure it looks correct and does not duplicate the information provided by any other message.
- Be sparing in the use of logging messages, particularly non-DEBUG messages. Logging messages are not an alternative to inserting print statements when debugging!

Other

- If you create a new on-disk format, add a version flag to it. This makes forwards and backwards compatibility MUCH easier should the format ever need to change.

Release process

This is an attempt to document what needs to be done in order to create a release for Exaile.

TODO: Update these instructions for github releases! Translations are now on <https://hosted.weblate.org/projects/exaile/master/>

Step one: Translations

Merge the translations from <https://hosted.weblate.org/projects/exaile/master/>

If you are a member of <https://github.com/exaile> you should be able to import all translations from some github web UI into the git repository.

If you are no member clone [git://git.weblate.org/exaile.git](https://git.weblate.org/exaile.git) and add the translations into your git repository.

Step two: Version bumping

First, adjust the version in your local working tree to reflect the version you want to make a release for. We should *never* do releases with `-dev` in them.

The file to adjust is `xl/version.py`. You should do a commit, and then tag the release.:

```
$ git tag RELEASE_VERSION
```

Step three: Linux + Windows

The ‘make dist’ command will build both the source distribution and the Windows version using NSIS running on Wine. You must install NSIS and the inetc plugin.

- Install NSIS 2 (http://nsis.sourceforge.net/Main_Page)
- **Install the inetc plugin (http://nsis.sourceforge.net/Inetc_plugin)**
 - Unzip it to `~/.wine/drive_c/Program Files (x86)/NSIS`

Once everything is installed, you can just run the following:

```
$ make dist
```

Step four: OSX

You need `py2app` installed to create an OSX `dmg` file. Once you have that installed, then you can do the following:

```
$ cd tools/osx
$ ./create_dmg.sh
```

If everything succeeded, you should find a file called “`exaile-VERSION.dmg`” at `dist/exaile-VERSION.dmg`.

Step five: Upload everything to launchpad

- Linux: `exaile-VERSION.tar.gz` + `exaile-VERSION.tar.gz.asc`
- Windows: `exaile-VERSION.exe` + `exaile-VERSION.exe.asc`
- OSX: `exaile-VERSION.dmg` + `exaile-VERSION.dmg.asc`

Step five: clean any relevant bug reports

Next, any bugs on launchpad for the release should be marked as ‘Fix released’. There is an easy way to do this via email... TODO

Step six: bump the version again

The version in trunk should reflect the upcoming release with a `-dev` in it.

TODO: Except after a beta/RC? What’s the right transition?

Step seven: send release notices

- Update download links on `exaile.org`
- Add article to `exaile.org`
- Send email to mailing lists

Debugging tips

TODO: developing with exaile

- **Setup pydev properly**
 - Add glib, gtk, pygst, gst, gobject to builtins
 - Add correct source directory

GST issues

When tracking down GST issues, a useful thing to do is the following:

```
$ GST_DEBUG=3 ./exaile
$ GST_DEBUG="cat:5;cat2:3" .. etc.

$ GST_DEBUG="GST_STATES:4" ./exaile
```

`GST_DEBUG_NO_COLOR=1` is good if you're running exaile inside of pydev on eclipse.

Additional help about GStreamer debugging variables can be found at <http://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/html/section-checklist-debug.html>

Exaile API Docs

Core:

Collection

Classes representing collections and libraries

A collection is a database of tracks. It is based on `TrackDB` but has the ability to be linked with libraries.

A library finds tracks in a specified directory and adds them to an associated collection.

Collections

`xl.collection.get_collection_by_loc` (*loc*)
gets the collection by a location.

Parameters `loc` – Location of the collection

Returns collection at location or None

Return type *Collection*

class `xl.collection.Collection` (*name*, *location=None*, *pickle_attrs=[]*)
Manages a persistent track database.

Parameters `args` – see `xl.trax.trackdb.TrackDB`

Simple usage:

```
>>> from xl.collection import *
>>> from xl.trax import search
>>> collection = Collection("Test Collection")
>>> collection.add_library(Library("./tests/data"))
>>> collection.rescan_libraries()
>>> tracks = [i.track for i in search.search_tracks_from_string(
...     collection, ('artist==TestArtist'))]
>>> print len(tracks)
5
>>>
```

add_library (*library*)

Add this library to the collection

Parameters **library** (*Library*) – the library to add

close ()

close the collection. does any work like saving to disk, closing network connections, etc.

freeze_libraries ()

Prevents “libraries_modified” events from being sent from individual add and remove library calls.

Call this before making bulk changes to the libraries. Call `thaw_libraries` when you are done; this sends a single event if the libraries were modified.

get_libraries ()

Gets a list of all the Libraries associated with this Collection

Return type list of *Library*

remove_library (*library*)

Remove a library from the collection

Parameters **library** (*Library*) – the library to remove

rescan_libraries (*startup_only=False, force_update=False*)

Rescans all libraries associated with this Collection

serialize_libraries ()

Save information about libraries

Called whenever the library’s settings are changed

stop_scan ()

Stops the library scan

thaw_libraries ()

Re-allow “libraries_modified” events from being sent from individual add and remove library calls. Also sends a “libraries_modified” event if the libraries have ben modified since the last call to `freeze_libraries`.

unserialize_libraries (*_serial_libraries*)

restores libraries from their serialized state.

Should only be called once, from the constructor.

class `xl.collection.CollectionScanThread` (*collection*, *startup_scan=False*, *force_update=False*)

Scans the collection

Initializes the thread

Parameters

- **collection** – the collection to scan

- **startup_scan** – Only scan libraries scanned at startup
- **force_update** – Update files regardless whether they've changed

on_scan_progress_update (*type, collection, progress*)

Notifies about progress changes

run ()

Runs the thread

stop ()

Stops the thread

Libraries

class `xl.collection.Library` (*location, monitored=False, scan_interval=0, startup_scan=False*)

Scans and watches a folder for tracks, and adds them to a Collection.

Simple usage:

```
>>> from xl.collection import *
>>> c = Collection("TestCollection")
>>> l = Library("./tests/data")
>>> c.add_library(l)
>>> l.rescan()
True
>>> print c.get_libraries()[0].location
./tests/data
>>> print len(list(c.search('artist="TestArtist"')))
5
>>>
```

Sets up the Library

Parameters

- **location** (*string*) – the directory this library will scan
- **monitored** (*bool*) – whether the library should update its collection at changes within the library's path
- **scan_interval** (*int*) – the interval for automatic rescanning

add (*loc, move=False*)

Copies (or moves) a file into the library and adds it to the collection

delete (*loc*)

Deletes a file from the disk

Warning: This permanently deletes the file from the hard disk.

get_location ()

Gets the current location associated with this Library

Returns the current location

Return type string

get_monitored ()

Whether the library should be monitored for changes

get_rescan_interval ()

Returns the scan interval in seconds

rescan (*notify_interval=None, force_update=False*)

Rescan the associated folder and add the contained files to the Collection

set_location (*location*)

Changes the location of this Library

Parameters **location** (*string*) – the new location to use

set_monitored (*monitored*)

Enables or disables monitoring of the library

Parameters **monitored** (*bool*) – Whether to monitor the library

set_rescan_interval (*interval*)

Sets the scan interval in seconds. If the interval is 0 seconds, the scan interval is stopped

Parameters **interval** (*int*) – scan interval in seconds

update_track (*gloc, force_update=False*)

Rescan the track at a given location

Parameters

- **gloc** (*gio.File*) – the location
- **force_update** – Force update of file (default only updates file when mtime has changed)

returns: the Track object, None if it could not be updated

monitored

Whether the library should be monitored for changes

class `xl.collection.LibraryMonitor` (*library*)

Monitors library locations for changes

Parameters **library** (*Library*) – the library to monitor

Common utilities

General functions and classes shared in the codebase

General functions

`xl.common.log_exception` (*log=<logging.Logger object>, message='Exception caught!'*)

Convenience function to log an exception + traceback

Parameters

- **log** – the logger object to use. important to specify so that it will be logged under the right module name.
- **message** – a message describing the error condition.

`xl.common.to_unicode` (*x, default_encoding=None*)

Force getting a unicode string from any object.

`xl.common.order_poset` (*items*)

Parameters `items` (list of *PosetItem*) – poset to order

Filesystem

`xl.common.open_file` (*path*)

Opens a file or folder using the system configured program

`xl.common.open_file_directory` (*path*)

Opens the parent directory of a file, selecting the file if possible.

`xl.common.walk` (*root*)

Walk through a Gio directory, yielding each file

Files are enumerated in the following order: first the directory, then the files in that directory. Once one directory's files have all been listed, it moves on to the next directory. Order of files within a directory and order of directory traversal is not specified.

Parameters `root` – a `gio.File` representing the directory to walk through

Returns a generator object

Return type `gio.File`

`xl.common.walk_directories` (*root*)

Walk through a Gio directory, yielding each subdirectory

Parameters `root` – a `gio.File` representing the directory to walk through

Returns a generator object

Return type `gio.File`

Decorators

`xl.common.threaded` (*func*)

A decorator that will make any function run in a new thread

Parameters `func` – the function to run threaded

`xl.common.synchronized` (*func*)

A decorator to make a function synchronized - which means only one thread is allowed to access it at a time.

This only works on class functions, and creates a variable in the instance called `_sync_lock`.

If this function is used on multiple functions in an object, they will be locked with respect to each other. The lock is re-entrant.

`xl.common.profileit` (*func*)

Decorator to profile a function

`xl.common.classproperty` (*function*)

Decorator allowing for class property access

`xl.common.cached` (*limit*)

Decorator to make a function's results cached does not cache if there is an exception.

Note: This probably breaks on functions that modify their arguments

Classes

exception `xl.common.VersionError` (*message*)

class `xl.common.LimitedCache` (*limit*)

Bases: `UserDict.DictMixin`

Simple cache that acts much like a dict, but has a maximum # of items

class `xl.common.TimeSpan` (*span*)

Calculates the number of days, hours, minutes, and seconds in a time span

Parameters `span` (*float*) – Time span in seconds

days = 0

number of days

hours = 0

number of hours

minutes = 0

number of minutes

seconds = 0

number of seconds

class `xl.common.MetadataList` (*iterable=[]*, *metadata=[]*)

Like a list, but also associates an object of metadata with each entry.

`(get|set|del)_meta_key` are the metadata interface - they allow the metadata to act much like a dictionary, with a few optimizations.

List aspects that are not supported:

- `sort`
- comparisons other than equality
- `multiply`

class `xl.common.ProgressThread`

A basic thread with progress updates

run ()

Override and make sure that the ‘progress-update’ signal is emitted regularly with the progress

stop ()

Stops the thread

class `xl.common.PosetItem` (*name*, *after*, *priority*, *value=None*)

Parameters

- **name** (*string*) – unique identifier for this item
- **after** (*list of string*) – which items this item comes after
- **priority** (*int*) – tiebreaker, higher values come later
- **value** – arbitrary data associated with the item

Cover

Provides the base for obtaining and storing covers, also known as album art.

Cover Manager

`xl.covers.MANAGER = <xl.covers.CoverManager object>`

The singleton *CoverManager* instance

class `xl.covers.CoverManager` (*location*)

Handles finding covers from various sources.

Parameters `location` – The directory to load and store data in.

find_covers (**__args, **__kw*)

Find all covers for a track

Parameters

- **track** – The track to find covers for
- **limit** – maximum number of covers to return. -1=unlimited.
- **local_only** – If True, will only return results from local sources.

get_cover (*track, save_cover=True, set_only=False, use_default=False*)

get the cover for a given track. if the track has no set cover, backends are searched until a cover is found or we run out of backends.

Parameters

- **track** – the Track to get the cover for.
- **save_cover** – if True, a `set_cover` call will be made to store the cover for later use.
- **set_only** – Only retrieve covers that have been set in the db.
- **use_default** – If True, returns the default cover instead of None when no covers are found.

get_cover_data (*db_string, use_default=False*)

Get the raw image data for a cover.

Parameters

- **db_string** – The `db_string` identifying the cover to get.
- **use_default** – If True, returns the default cover instead of None when no covers are found.

get_db_string (*track*)

Returns the internal string used to map the cover to a track

Parameters `track` (*xl.trax.Track*) – the track to retrieve the string for

Returns the internal identifier string

get_default_cover ()

Get the raw image data for the cover to show if there is no cover to display.

load ()

Load the saved db

remove_cover (*track*)

Remove the saved cover entry for a track, if it exists.

save ()

Save the db

set_cover (*track*, *db_string*, *data=None*)

Sets the cover for a track. This will overwrite any existing entry.

Parameters

- **track** – The track to set the cover for
- **db_string** – the string identifying the source of the cover, in “method:key” format.
- **data** – The raw cover data to store for the track. Will only be stored if the method has `use_cache=True`

set_preferred_order (*order*)

Sets the preferred search order

Parameters **order** – a list containing the order you’d like to search first

Cover Search Methods

class `xl.covers.CoverSearchMethod`

Base class for creating cover search methods.

Search methods do not have to inherit from this class, it’s intended more as a template to demonstrate the needed interface.

find_covers (*track*, *limit=-1*)

Find the covers for a given track.

Parameters

- **track** – The track to find covers for.
- **limit** – Maximal number of covers to return.

Returns A list of strings that can be passed to `get_cover_data`.

get_cover_data (*db_string*)

Get the image data for a cover

Parameters **db_string** – A method-dependent string that identifies the cover to get.

fixed = False

Whether the backend should have a fixed priority instead of being

fixed_priority = 50

Priority for fixed-position backends. Lower is earlier, non-fixed

name = ‘base’

A name uniquely identifying the search method.

use_cache = True

If true, cover results will be cached for faster lookup

class `xl.covers.TagCoverFetcher`

Cover source that looks for images embedded in tags.

class `xl.covers.LocalFileCoverFetcher`

Cover source that looks for images in the same directory as the Track.

Events

Provides a signals-like system for sending and listening for ‘events’

Events are kind of like signals, except they may be listened for on a global scale, rather than connected on a per-object basis like signals are. This means that ANY object can emit ANY event, and these events may be listened for by ANY object.

Events should be emitted AFTER the given event has taken place. Often the most appropriate spot is immediately before a return statement.

```
xl.event.log_event(type, obj, data)
```

Sends an event.

Parameters

- **type** (*string*) – the *type* or *name* of the event.
- **obj** (*object*) – the object sending the event.
- **data** (*object*) – some data about the event, None if not required

```
xl.event.add_callback(function, type=None, obj=None, *args, **kwargs)
```

Adds a callback to an event

You should ALWAYS specify one of the two options on what to listen for. While not forbidden to listen to all events, doing so will cause your callback to be called very frequently, and possibly may cause slowness within the player itself.

Parameters

- **function** (*callable*) – the function to call when the event happens
- **type** (*string*) – the *type* or *name* of the event to listen for, eg *tracks_added*, *cover_changed*. Defaults to any event if not specified.
- **obj** (*object*) – the object to listen to events from, e.g. *exaile.collection* or *xl.covers.MANAGER*. Defaults to any object if not specified.

Any additional parameters will be passed to the callback.

Returns a convenience function that you can call to remove the callback.

```
xl.event.remove_callback(function, type=None, obj=None)
```

Removes a callback

The parameters passed should match those that were passed when adding the callback

Formatter

Provides an extensible framework for processing and preparation of data for display in various contexts.

Formatters

```
class xl.formatter.Formatter(format)
```

A generic text formatter based on a format string

By default the following parameters are provided to each identifier:

- **prefix, suffix:** a string to put before or after the formatted string if that string is not empty
 - Whitespace will be not be touched and transferred as is

– The characters `,`, `}` and `=` need to be escaped like `\,`, `\}` and `\=` respectively

• `pad`: desired length the formatted string should have, will be achieved using the `padding`

• **`padding`**: a string to use for padding, will be repeated as often as possible to achieve the desired length specified

– Example: `${identifier:pad=4, padding=XY}` for *identifier* having the value *a* will become *XYXa*

Parameters format (*string*) – the initial format, see the documentation of `string`.
Template for details

extract ()

Retrieves the identifiers and their optional parameters

Example of the returned dictionary:

```
extractions = {
    'identifier1': (
        'identifier1', {}),
    'identifier2:parameter': (
        'identifier2', {'parameter': True}),
    'identifier3:parameter=argument': (
        'identifier3', {'parameter': 'argument'})
}
```

Returns the extractions

Return type dict

format (*args)

Returns a string by formatting the passed data

Parameters args – data to base the formatting on

Returns the formatted text

Return type string

class `xl.formatter.ProgressTextFormatter` (*format, player*)

A text formatter for progress indicators

format (*current_time=None, total_time=None*)

Returns a string suitable for progress indicators

Parameters

- **`current_time`** (*float*) – the current progress, taken from the current playback if not set
- **`total_time`** (*float*) – the total length of a track, taken from the current playback if not set

Returns The formatted text

Return type string

class `xl.formatter.TrackFormatter` (*format*)

A formatter for track data

Parameters format (*string*) – the initial format, see the documentation of `string`.
Template for details

format (*track*, *markup_escape=False*)

Returns a string for places where track data is presented to the user

Parameters

- **track** (*xl.trax.Track*) – a single track to take data from
- **markup_escape** (*bool*) – whether to escape markup-like characters in tag values

Returns the formatted text

Return type string

class *xl.formatter.TagFormatter* (*name*)

A formatter provider for a tag of a track

Parameters *name* (*string*) – the name of the tag

format (*track*, *parameters*)

Formats a raw tag value. Accepts optional parameters to manipulate the formatting process.

Parameters

- **track** (*xl.trax.Track*) – the track to get the tag from
- **parameters** (*dictionary*) – optionally passed parameters

Returns the formatted value

Return type string

class *xl.formatter.TrackNumberTagFormatter*

Bases: *xl.formatter.NumberTagFormatter*

A formatter for the tracknumber of a track

class *xl.formatter.DiscNumberTagFormatter*

Bases: *xl.formatter.NumberTagFormatter*

A formatter for the discnumber of a track

class *xl.formatter.ArtistTagFormatter*

Bases: *xl.formatter.TagFormatter*

A formatter for the artist of a track

format (*track*, *parameters*)

Formats a raw tag value

Parameters

- **track** (*xl.trax.Track*) – the track to get the tag from
- **parameters** – optionally passed parameters Possible values are:
 - *compile*: Allows for proper handling of compilations, either via *albumartist* tag, a fallback value, or simply all artists

Returns the formatted value

Return type string

class *xl.formatter.LengthTagFormatter*

Bases: *xl.formatter.TimeTagFormatter*

A formatter for the length of a track

format (*track, parameters*)
Formats a raw tag value

Parameters

- **track** (*xl.trax.Track*) – the track to get the tag from
- **parameters** (*dictionary*) – Verbosity of the output, possible values for “format” are:
 - short: “1:02:42”
 - long: “1h, 2m, 42s”
 - verbose: “1 hour, 2 minutes, 42 seconds”

Returns the formatted value

Return type string

format_value (*value, format='short'*)
Formats a length value

Parameters

- **value** (*float*) – the length in seconds
- **format** (*string*) – verbosity of the output, possible values are:
 - short: “1:02:42”
 - long: “1h, 2m, 42s”
 - verbose: “1 hour, 2 minutes, 42 seconds”

Returns the formatted value

Return type string

class `xl.formatter.RatingTagFormatter`
Bases: `xl.formatter.TagFormatter`

A formatter for the rating of a track

Will return glyphs representing the rating like

class `xl.formatter.LastPlayedTagFormatter`
Bases: `xl.formatter.DateTagFormatter`

A formatter for the last time a track was played

Templates

class `xl.formatter.ParameterTemplate` (*template*)

An extended template class which additionally accepts parameters assigned to identifiers.

This introduces another pattern group named “parameters” in addition to the groups created by `string.Template`

Examples:

- `${foo:parameter1}`
- `${bar:parameter1, parameter2}`
- `${qux:parameter1=argument1, parameter2}`

Parameters `template` – the template string

Metadata

`xl.metadata.formats = {'669': <class 'xl.metadata.mod.ModFormat'>, 'xm': <class 'xl.metadata.mod.ModFormat'>, 'aif': <class 'xl.metadata.mod.ModFormat'>}`
 dictionary mapping extensions to Format classes.

`xl.metadata.get_format(loc)`
 get a Format object appropriate for the file at loc. if no suitable object can be found, None is returned.

Parameters `loc` – The location to read from. can be any gio-parseable path or uri.

Format Objects

exception `xl.metadata.NotWritable`

exception `xl.metadata.NotReadable`

class `xl.metadata.BaseFormat(loc)`
 Base class for handling loading of metadata from files.

subclasses using mutagen should set MutagenType and overload the `_get_tag`, `_set_tag`, and `_del_tag` methods as needed.

subclasses not using mutagen should leave MutagenType as None

Raises `NotReadable` if the file cannot be opened for some reason.

Parameters `loc` – absolute path to the file to read (note - this may change to accept gio uris in the future)

read_all()
 Reads all non-blacklisted tags from the file.

Blacklisted tags include lyrics, covers, and any field starting with `__`. If you need to read these, call `read_tags` directly.

read_tags(tags)
 get the values for the specified tags.

returns a dict of the found values. if no value was found for a requested tag it will not exist in the returned dict.

Parameters `tags` – a list of tag names to read

Returns a dictionary of tag/value pairs.

write_tags(tagdict)
 Write a set of tags to the file. Raises a `NotWritable` exception if the format does not support writing tags.

Parameters `tagdict` – A dictionary of tag/value pairs to write.

Player

Allows for playback and queue control

`xl.player.PLAYER = None`

The player singleton of `ExailePlayer` for playback control

`ExailePlayer.play(track, **kwargs)`

Starts the playback with the provided track or stops the playback it immediately if none

Parameters `track` (`xl.trax.Track`) – the track to play

Note: The following *events* will be emitted by this method:

- *playback_player_start*: indicates the start of playback overall
 - *playback_track_start*: indicates playback start of a track
-

`ExailePlayer.stop(_fire=True, **kwargs)`

Stops the playback

Parameters `fire` – Send the ‘playback_player_end’ event. Used by engines to avoid spurious playback_end events. Not public API.

Note: The following *events* will be emitted by this method:

- *playback_player_end*: indicates the end of playback overall
 - *playback_track_end*: indicates playback end of a track
-

`ExailePlayer.pause()`

Pauses the playback, does not toggle it

Note: The following *events* will be emitted by this method:

- *playback_player_pause*: indicates that the playback has been paused
-

`ExailePlayer.unpause()`

Resumes the playback, does not toggle it

Note: The following *events* will be emitted by this method:

- *playback_player_resume*: indicates that the playback has been resumed
-

`ExailePlayer.toggle_pause()`

Toggles between playing and paused state

Note: The following *events* will be emitted by this method:

- *playback_toggle_pause*: indicates that the playback has been paused or resumed
-

`ExailePlayer.seek(value)`

Seek to a position in the currently playing stream

Parameters `value` (`int`) – the position in seconds

`ExailePlayer.get_position()`

Gets the current playback position of the playing track

Returns the playback position in nanoseconds

Return type `int`

`ExailePlayer.get_time()`

Gets the current playback time

Returns the playback time in seconds

Return type int

`ExailePlayer.get_progress()`

Gets the current playback progress

Returns the playback progress as [0..1]

Return type float

`ExailePlayer.set_progress(progress)`

Seeks to the progress position

Parameters `progress` (*float*) – value ranged at [0..1]

`ExailePlayer.get_volume()`

Gets the current volume

Returns the volume percentage

Type int

`ExailePlayer.set_volume(volume)`

Sets the current volume

Parameters `volume` (*int*) – the volume percentage

`ExailePlayer.get_state()`

Gets the player state

Returns one of *playing*, *paused* or *stopped*

Return type string

`ExailePlayer.is_playing()`

Convenience method to find out if the player is currently playing

Returns whether the player is currently playing

Return type bool

`ExailePlayer.is_paused()`

Convenience method to find out if the player is currently paused

Returns whether the player is currently paused

Return type bool

`ExailePlayer.is_stopped()`

Convenience method to find out if the player is currently stopped

Returns whether the player is currently stopped

Return type bool

`xl.player.QUEUE = <xl.player.queue.PlayQueue object>`

Manages the queue of songs to be played

The content of the queue are processed before processing the content of the assigned playlist.

When the `remove_item_when_played` option is enabled, the queue removes items from itself as they are played.

When not enabled, the queue acts like a regular playlist, and moves the position as tracks are played.

In this mode, when a new track is queued, the position is set to play that track, and play will continue with that track until the queue is exhausted, and then the assigned playlist will be continued.

The queue singleton of `PlayQueue`

class `xl.player.queue.PlayQueue` (*player, name, location=None*)

Bases: `xl.playlist.Playlist`

Manages the queue of songs to be played

The content of the queue are processed before processing the content of the assigned playlist.

When the `remove_item_when_played` option is enabled, the queue removes items from itself as they are played.

When not enabled, the queue acts like a regular playlist, and moves the position as tracks are played.

In this mode, when a new track is queued, the position is set to play that track, and play will continue with that track until the queue is exhausted, and then the assigned playlist will be continued.

get_current ()

Gets the current track

Returns the current track

Type `xl.trax.Track`

get_next ()

Retrieves the next track that will be played. Does not actually set the position. When you call `next()`, it should return the same track.

This exists to support retrieving a track before it actually needs to be played, such as for pre-buffering.

Returns the next track to be played

Return type `xl.trax.Track` or None

is_play_enabled ()

Returns True when calling `play()` will have no effect

next (*autoplay=True, track=None*)

Goes to the next track, either in the queue, or in the current playlist. If a track is passed in, that track is played

Parameters

- **autoplay** (*bool*) – play the track in addition to returning it
- **track** (`xl.trax.Track`) – if passed, play this track

Note: The following *events* will be emitted by this method:

- *playback_playlist_end*: indicates that the end of the queue has been reached
-

play (*track=None*)

Starts queue processing with the given track preceding the queue content

Parameters **track** (`xl.trax.Track`) – the track to play

prev ()

Goes to the previous track

queue_length ()

Returns the number of tracks left to play in the queue's internal playlist.

set_current_playlist (*playlist*)

Sets the playlist to be processed in the queue

Parameters **playlist** (`xl.playlist.Playlist`) – the playlist to process

Note: The following *events* will be emitted by this method:

- *queue_current_playlist_changed*: indicates that the queue playlist has been changed
-

current_playlist

The playlist currently processed in the queue

Playlist

Provides the fundamental objects for handling a list of tracks contained in playlists as well as methods to import and export from various file formats.

Playlists

class `xl.playlist.Playlist` (*name*, *initial_tracks=[]*)
 Basic class for handling a list of tracks

EVENTS: (all events are synchronous)

- **playlist_tracks_added**
 - fired: after tracks are added
 - data: list of tuples of (index, track)
- **playlist_tracks_removed**
 - fired: after tracks are removed
 - data: list of tuples of (index, track)
- **playlist_current_position_changed**
- **playlist_shuffle_mode_changed**
- **playlist_random_mode_changed**
- **playlist_dynamic_mode_changed**

Parameters

- **name** (*string*) – the initial name of the playlist
- **initial_tracks** (list of *xl.trax.Track*) – the tracks which shall populate the playlist initially

append (*other*)

Appends a single track to the playlist

Parameters *other* – list of *xl.trax.Track*

clear ()

Removes all contained tracks

clear_shuffle_history ()

Clear the history of played tracks from a shuffle run

count (*other*)

Returns the count of contained tracks

Returns the count

Return type int

extend (*other*)

Extends the playlist by another playlist

Parameters *other* – list of *xl.trax.Track*

get_current ()

Retrieves the track at the current position

Returns the track

Return type *xl.trax.Track* or None

get_current_position ()

Retrieves the current position within the playlist

Returns the position

Return type int

get_dynamic_mode ()

Retrieves the current dynamic mode

Returns the dynamic mode

Return type string

get_repeat_mode ()

Retrieves the current repeat mode

Returns the repeat mode

Return type string

get_shuffle_history ()

Retrieves the history of played tracks from a shuffle run

Returns the tracks

Return type list

get_shuffle_mode ()

Retrieves the current shuffle mode

Returns the shuffle mode

Return type string

get_spat_position ()

Retrieves the current position within the playlist after which progressing shall be stopped

Returns the position

Return type int

index (*item*, *start=0*, *end=None*)

Retrieves the index of a track within the playlist

Returns the index

Return type int

load_from_location (*location*)

Loads the content of the playlist from a given location

Parameters *location* (*string*) – the location to load from

next ()

Progresses to the next track within the playlist and takes shuffle and repeat modes into account

Returns the new current track

Return type `xl.trax.Track` or `None`

pop (*i=-1*)

Pops a track from the playlist

Parameters **i** (*int*) – the index

Returns the track

Return type `xl.trax.Track`

prev ()

Progresses to the previous track within the playlist and takes shuffle and repeat modes into account

Returns the new current track

Return type `xl.trax.Track` or `None`

randomize (*positions=None*)

Randomizes the content of the playlist contrary to shuffle which affects only the progressing order

By default all tracks in the playlist are randomized, but a list of positions can be passed. The tracks on these positions will be randomized, all other tracks will keep their positions.

Parameters **positions** (*iterable*) – list of track positions to randomize

save_to_location (*location*)

Writes the content of the playlist to a given location

Parameters **location** (*string*) – the location to save to

set_current_position (*position*)

Sets the current position within the playlist

Parameters **position** (*int*) – the new position

set_dynamic_mode (*mode*)

Sets the current dynamic mode

Parameters **mode** (*string*) – the new dynamic mode

set_repeat_mode (*mode*)

Sets the current repeat mode

Parameters **mode** (*string*) – the new repeat mode

set_shuffle_mode (*mode*)

Sets the current shuffle mode

Parameters **mode** (*string*) – the new shuffle mode

set_spat_position (*position*)

Sets the current position within the playlist after which progressing shall be stopped

Parameters **position** (*int*) – the new position

sort (*tags, reverse=False*)

Sorts the content of the playlist

Parameters

- **tags** (*list of strings*) – tags to sort by

- **reverse** (*boolean*) – whether the sorting shall be reversed

current_position

The position within the playlist (int)

dirty

Whether the playlist was changed or not (boolean)

dynamic_mode

The current dynamic mode (string)

name

The playlist name (string)

repeat_mode

The current repeat mode (string)

shuffle_mode

The current shuffle mode (string)

shuffle_mode_names = [**u'Shuffle_Off', u'Shuffle_Tracks', u'Shuffle_Albums'**]

Titles of the valid shuffle modes (list of string)

shuffle_modes = [**'disabled', 'track', 'album'**]

Valid shuffle modes (list of string)

spat_position

The position within the playlist after which to stop progressing (int)

Playlist Converters

class `xl.playlist.FormatConverter` (*name*)

Base class for all converters allowing to import from and export to a specific format

export_to_file (*playlist, path, options=None*)

Export a playlist to a given path

Parameters

- **playlist** (*Playlist*) – the playlist
- **path** (*string*) – the target path
- **options** (*PlaylistExportOptions*) – exporting options

import_from_file (*path*)

Import a playlist from a given path

Parameters **path** (*string*) – the source path

Returns the playlist

Return type *Playlist*

name_from_path (*path*)

Convenience method to retrieve a sane name from a path

Parameters **path** (*string*) – the source path

Returns a name

Return type string


```

class xl.playlist.M3UConverter
    Bases: xl.playlist.FormatConverter

    Import from and export to M3U format

class xl.playlist.PLSConverter
    Bases: xl.playlist.FormatConverter

    Import from and export to PLS format

class xl.playlist.ASXConverter
    Bases: xl.playlist.FormatConverter

    Import from and export to ASX format

class xl.playlist.XSPFConverter
    Bases: xl.playlist.FormatConverter

    Import from and export to XSPF format

```

Providers & Services

A generic framework for service providers, recommended to be used whenever there are multiple ways of accomplishing a task or multiple sources can offer the required data.

```

xl.providers.MANAGER
    Singleton instance of the ProviderManager

```

```

xl.providers.register(self, servicename, provider, target=None)
    Registers a provider for a service. The provider object is used by consumers of the service.

```

Services can be targeted for a specific use. For example, if you have a widget that uses a service ‘foo’, if your object can perform a service only for a specific type of widget, then target would be set to the widget type.

If you had a service that could perform ‘foo’ for all widgets, then target would be set to None, and all widgets could use your service.

It is intended that most services should set target to None, with some narrow exceptions.

Parameters

- **servicename** (*string*) – the name of the service [string]
- **provider** (*object*) – the object that is the provider [object]
- **target** (*object*) – a specific target for the service [object]

```

xl.providers.unregister(self, servicename, provider, target=None)
    Unregisters a provider.

```

Parameters

- **servicename** (*string*) – the name of the service
- **provider** (*object*) – the provider to be removed
- **target** (*object*) – a specific target for the service [object]

```

xl.providers.get(self, servicename, target=None)
    Returns a list of providers for the specified servicename.

```

This will return providers targeted for a specific target AND providers not targeted towards any particular target.

Parameters

- **servicename** (*string*) – the service name to get providers for

- **target** (*object*) – the target of the service

Returns list of providers

Return type list of objects

`xl.providers.get_provider(self, servicename, providername, target=None)`

Returns a single identified provider

This will return a provider either targeted for the specific target or a provider not targeted towards any particular target.

Parameters

- **servicename** (*string*) – The service name to get the provider for
- **providername** (*string*) – The provider name to identify the provider
- **target** (*object*) – the target of the service

Returns a provider or None

Return type object

class `xl.providers.ProviderManager`

The overall manager for services and providers for them

class `xl.providers.ProviderHandler(servicename, target=None, simple_init=False)`

Base class to handle providers for one specific service including notification about (un)registration

Target is the object that the service is being performed for. Often, if the service is truly global and it doesn't make sense to target a service at a particular consumer, it can be None.

Parameters

- **servicename** (*string*) – the name of the service to handle
- **target** (*string*) – the target for a provided service. Generally, this will be the object that uses the service
- **simple_init** – call `on_provider_added` for every element already registered on instantiation.

Settings

Central storage of application and user settings

`xl.settings.MANAGER`

Singleton instance of the `SettingsManager`

`xl.settings.get_option(name, default)`

`xl.settings.set_option(self, option, value)`

Set an option (in `section/key` syntax) to the specified value

Parameters

- **option** (*string*) – the full path to an option
- **value** (*any*) – the value the option should be assigned

class `xl.settings.SettingsManager(location=None, default_location=None)`

Bases: `ConfigParser.RawConfigParser`

Manages Exaile's settings

Sets up the settings manager. Expects a location to a file where settings will be stored. Also sets up periodic saves to disk.

Parameters

- **location** (*str* or *None*) – the location to save the settings to, settings will never be stored if this is *None*
- **default_location** – the default location to initialize settings from

get_option (*option*, *default=None*)

Get the value of an option (in *section/key* syntax), returning *default* if the key does not exist yet

Parameters

- **option** (*string*) – the full path to an option
- **default** (*any*) – a default value to use as fallback

Returns the option value or *default*

Return type *any*

has_option (*option*)

Returns information about the existence of a particular option

Parameters **option** (*string*) – the option path

Returns whether the option exists or not

Return type *bool*

remove_option (*option*)

Removes an option (in *section/key* syntax), thus will not be saved anymore

Parameters **option** (*string*) – the option path

save ()

Save the settings to disk

set_option (*option*, *value*)

Set an option (in *section/key* syntax) to the specified value

Parameters

- **option** (*string*) – the full path to an option
- **value** (*any*) – the value the option should be assigned

Trax

Provides the base for creating and managing Track objects.

Tracks

class `xl.trax.Track` (*uri=None*, *scan=True*, *_unpickles=None*)

Represents a single track.

Parameters

- **uri** – The path to the track.
- **scan** – Whether to try to read tags from the given uri. Use only if the tags need to be set by a different source.

- **_unpickles** – used internally to restore from a pickled state. not for normal use.

exists()

Returns whether the file exists This can be very slow, use with caution!

get_loc_for_io()

Gets the location as a full uri.

Safe for IO operations via gio, not suitable for display to users as it may be in non-utf-8 encodings.

get_rating()

Returns the current track rating as an integer, as determined by the `rating/maximum` setting.

get_tag_display(*tag*, *join=True*, *artist_compilations=False*, *extend_title=True*)

Get a tag value in a form suitable for display.

Parameters

- **tag** – The name of the tag to get
- **join** – If True, joins lists of values into a single value.
- **artist_compilations** – If True, automatically handle albumartist and other compilations detections when `tag=="albumartist"`.
- **extend_title** – If the title tag is unknown, try to add some identifying information to it.

get_tag_raw(*tag*, *join=False*)

Get the raw value of a tag. For non-internal tags, the result will always be a list of unicode strings.

Parameters

- **tag** – The name of the tag to get
- **join** – If True, joins lists of values into a single value.

get_tag_sort(*tag*, *join=True*, *artist_compilations=False*, *extend_title=True*)

Get a tag value in a form suitable for sorting.

Parameters

- **tag** – The name of the tag to get
- **join** – If True, joins lists of values into a single value.
- **artist_compilations** – If True, automatically handle albumartist and other compilations detections when `tag=="albumartist"`.
- **extend_title** – If the title tag is unknown, try to add some identifying information to it.

get_type()

Get the URI schema the file uses, e.g. file, http, smb.

list_tags()

Returns a list of the names of all tags present in this Track.

local_file_name()

If the file is accessible on the local filesystem, returns a standard path to it (e.g. `"/home/foo/bar"`), otherwise, returns None.

If a path is returned, it is safe to use for IO operations. Existence of a path does *not* guarantee file existence.

read_tags ()

Reads tags from the file for this Track.

Returns False if unsuccessful, and a Format object from *xl.metadata* otherwise.

set_loc (loc)

Sets the location.

Parameters **loc** – the location, as either a uri or a file path.

set_rating (rating)

Sets the current track rating from an integer, on the scale determined by the `rating/maximum` setting.

Returns the scaled rating

set_tag_raw (tag, values, notify_changed=True)

Set the raw value of a tag.

Parameters

- **tag** – The name of the tag to set.
- **values** – The value or values to set the tag to.
- **notify_changed** – whether to send a signal to let other parts of Exaile know there has been an update. Only set this to False if you know that no other parts of Exaile need to be updated.

write_tags ()

Writes tags to the file for this Track.

Returns False if unsuccessful, and a Format object from *xl.metadata* otherwise.

xl.trax.is_valid_track (location)

Returns whether the file at the given location is a valid track

Parameters **location** (*string*) – the location to check

Returns whether the file is a valid track

Return type boolean

xl.trax.get_uris_from_tracks (tracks)

Returns all URIs for tracks

Parameters **tracks** (list of *xl.trax.Track*) – the tracks to retrieve the URIs from

Returns the uris

Return type list of string

xl.trax.get_tracks_from_uri (uri)

Returns all valid tracks located at uri

Parameters **uri** (*string*) – the uri to retrieve the tracks from

Returns the retrieved tracks

Return type list of *xl.trax.Track*

xl.trax.sort_tracks (fields, iter, trackfunc=None, reverse=False, artist_compilations=False)

Sorts tracks.

Parameters

- **fields** (*iterable*) – tag names to sort by
- **iter** (*iterable*) – the tracks to sort, alternatively use *trackfunc*

- **trackfunc** (*function or None*) – function to get a *Track* from an item in the iterable
- **reverse** (*boolean*) – whether to sort in reversed order

`xl.trax.sort_result_tracks` (*fields, trackiter, reverse=False, artist_compilations=False*)
Sorts *SearchResultTracks*, ie. the output from a search.

Same params as `sort_tracks`.

`xl.trax.get_rating_from_tracks` (*tracks*)
Returns the common rating for all tracks or simply 0 if not all tracks have the same rating. Same goes if the amount of tracks is 0 or more than the internal limit.

Parameters `tracks` (*iterable*) – the tracks to retrieve the rating from

Track Database

Track databases are a simple persistence layer to hold collections of *Track* objects.

class `xl.trax.TrackDB` (*name='', location='', pickle_attrs=[], loadfirst=False*)
Manages a track database.

Allows you to add, remove, retrieve, search, save and load *Track* objects.

Parameters

- **name** – The name of this *TrackDB*.
- **location** – Path to a file where this *TrackDB* should be stored.
- **pickle_attrs** – A list of attributes to store in the pickled representation of this object. All attributes listed must be built-in types, with one exception: If the object contains the phrase ‘tracks’ in its name it may be a list or dict of *Track* objects.
- **load_first** – Set to True if this collection should be loaded before any tracks are created.

Sets up the trackDB.

add (*track*)
Adds a track to the database of tracks

Parameters `track` – The *xl.trax.Track* to add

add_tracks (**__args, **__kw*)
Like `add()`, but takes a list of *xl.trax.Track*

load_from_location (**__args, **__kw*)
Restores *TrackDB* state from the pickled representation stored at the specified location.

Parameters `location` (*string*) – the location to load the data from

remove (*track*)
Removes a track from the database

Parameters `track` – the *xl.trax.Track* to remove

remove_tracks (**__args, **__kw*)
Like `remove()`, but takes a list of *xl.trax.Track*

save_to_location (**__args, **__kw*)
Saves a pickled representation of this *TrackDB* to the specified location.

Parameters `location` (*string*) – the location to save the data to

Searching

class `xl.trax.TracksMatcher` (*search_string*, *case_sensitive=True*, *keyword_tags=None*)
 Holds criteria and determines whether a given track matches those criteria.

Parameters

- **search_string** – a string describing the match conditions
- **case_sensitive** – whether to search in a case-sensitive manner.
- **keyword_tags** – a list of tags to match search keywords in.

`xl.trax.search_tracks` (*trackiter*, *trackmatchers*)
 Search a set of tracks for those that match specified conditions.

Parameters

- **trackiter** – An iterable object returning Track objects
- **trackmatchers** – A list of TrackMatcher objects

`xl.trax.search_tracks_from_string` (*trackiter*, *search_string*, *case_sensitive=True*, *keyword_tags=None*)

Convenience wrapper around `search_tracks` that builds matchers automatically from the search string.

Arguments have the same meaning as the corresponding arguments on `search_tracks` and `TracksMatcher`.

D-Bus

D-Bus interface for playback control, data query and others

Access through the `/org/exaile/Exaile` object which implements the `org.exaile.Exaile` interface

org.exaile.Exaile Interface

class `xl.xldbus.DbusManager` (*exaile*)
 The dbus interface object for Exaile

Initializes the interface

GUI:

Icons & Images

Provides methods for convenient icons and image handling

Icon management

`xlgui.icons.MANAGER`
 Singleton instance of the `IconManager`

class `xlgui.icons.IconManager`
 Provides convenience functions for managing icons and images in general

add_icon_name_from_directory (*icon_name*, *directory*)
 Registers an icon name from files found in a directory

Parameters

- **icon_name** (*string*) – the name for the icon
- **directory** (*string*) – the location to search for icons

add_icon_name_from_file (*icon_name, filename, size=None*)

Registers an icon name from a filename

Parameters

- **icon_name** (*string*) – the name for the icon
- **filename** (*string*) – the filename of an image
- **size** (*int*) – the size the icon shall be registered for

add_icon_name_from_pixbuf (*icon_name, pixbuf, size=None*)

Registers an icon name from a pixbuf

Parameters

- **icon_name** (*string*) – the name for the icon
- **pixbuf** (*gtk.gdk.Pixbuf*) – the pixbuf of an image
- **size** (*int*) – the size the icon shall be registered for

add_stock_from_directory (*stock_id, directory*)

Registers a stock icon from files found in a directory

Parameters

- **stock_id** (*string*) – the stock id for the icon
- **directory** (*string*) – the location to search for icons

add_stock_from_file (*stock_id, filename*)

Registers a stock icon from a filename

Parameters

- **stock_id** (*string*) – the stock id for the icon
- **filename** (*string*) – the filename of an image

add_stock_from_files (*stock_id, filenames*)

Registers a stock icon from filenames

Parameters

- **stock_id** (*string*) – the stock id for the icon
- **filenames** (*list of string*) – the filenames of images

add_stock_from_pixbuf (*stock_id, pixbuf*)

Registers a stock icon from a pixbuf

Parameters

- **stock_id** (*string*) – the stock id for the icon
- **pixbuf** (*gtk.gdk.Pixbuf*) – the pixbuf of an image

add_stock_from_pixbufs (*stock_id, pixbufs*)

Registers a stock icon from pixbufs

Parameters

- **stock_id** (*string*) – the stock id for the icon
- **pixbuf** (list of `gtk.gdk.Pixbuf`) – the pixbufs of images

pixbuf_from_data (*data*, *size=None*, *keep_ratio=True*, *upscale=False*)

Generates a pixbuf from arbitrary image data

Parameters

- **data** (*byte*) – The raw image data
- **size** (*tuple of int*) – Size to scale to; if not specified, the image will render to its native size
- **keep_ratio** (*bool*) – Whether to keep the original image ratio on resizing operations
- **upscale** (*bool*) – Whether to upscale if the requested size exceeds the native size

Returns the generated pixbuf

Return type `gtk.gdk.Pixbuf` or `None`

pixbuf_from_icon_name (*icon_name*, *size=<class 'mocks.ICON_SIZE_BUTTON'>*)

Generates a pixbuf from an icon name

Parameters

- **stock_id** (*string*) – an icon name
- **size** (*int or GtkIconSize*) – the size of the icon, will be tried to converted to a GTK icon size

Returns the generated pixbuf

Return type `gtk.gdk.Pixbuf` or `None`

pixbuf_from_rating (**args*, ***kwargs*)

Returns a pixbuf representing a rating

Parameters **rating** (*int*) – the rating

Returns the rating pixbuf

Return type `gtk.gdk.Pixbuf`

pixbuf_from_stock (*stock_id*, *size=<class 'mocks.ICON_SIZE_BUTTON'>*)

Generates a pixbuf from a stock id

Parameters

- **stock_id** (*string*) – a stock id
- **size** (*GtkIconSize*) – the size of the icon

Returns the generated pixbuf

Return type `gtk.gdk.Pixbuf` or `None`

pixbuf_from_text (*text*, *size*, *background_color='#456eac'*, *border_color='#000'*,
text_color='#fff')

Generates a pixbuf based on a text, width and height

Parameters

- **size** (*tuple of int*) – A tuple describing width and height
- **background_color** (*string*) – The color of the background as hexadecimal value
- **border_color** (*string*) – The color of the border as hexadecimal value

- **text_color** (*string*) – The color of the text as hexadecimal value

Utilities

class `xlgui.icons.ExtendedPixbuf` (*pixbuf*)

A `gtk.gdk.Pixbuf` wrapper class allowing for interaction using standard operators

Thus you can do the following:

- `pixbuf1 + pixbuf2` (horizontally appends `pixbuf2` to `pixbuf1`)
- `pixbuf * 5` (multiplies the content of `pixbuf`)
- `pixbuf1 & pixbuf2` (simple composition of `pixbuf2` on `pixbuf1`, the desired alpha value has to be included in the `pixbufs` themselves)
- `pixbuf1 < pixbuf2`, `pixbuf1 > pixbuf2` (compares the `pixbuf` dimensions)
- `pixbuf1 == pixbuf2` (compares the pixel data, use the *is* operator to check for identity)

Even more is possible with the provided verbose methods

add_horizontal (*other*, *spacing=0*)

Horizontally appends a `pixbuf` to the current

Parameters

- **other** (`gtk.gdk.Pixbuf`) – the `pixbuf` to append
- **spacing** (*int*) – amount of pixels between the `pixbufs`

Returns a new `pixbuf`

Return type *ExtendedPixbuf*

add_vertical (*other*, *spacing=0*)

Vertically appends a `pixbuf` to the current

Parameters

- **other** (`gtk.gdk.Pixbuf`) – the `pixbuf` to append
- **spacing** (*int*) – amount of pixels between the `pixbufs`

Returns a new `pixbuf`

Return type *ExtendedPixbuf*

composite_simple (*other*)

Composites a `pixbuf` on the current `pixbuf` at the location (0, 0)

Parameters **other** (`gtk.gdk.Pixbuf`) – the `pixbuf` to composite

Returns a new `pixbuf`

Return type *ExtendedPixbuf*

move (*offset_x*, *offset_y*, *resize=False*)

Moves the content of the current `pixbuf` within its boundaries (clips overlapping data) and optionally resizes the `pixbuf` to contain the movement

Parameters

- **offset_x** (*int*) – the amount of pixels to move in horizontal direction
- **offset_y** (*int*) – the amount of pixels to move in vertical direction

- **resize** (*bool*) – whether to resize the pixbuf on movement

Returns a new pixbuf

Return type *ExtendedPixbuf*

multiply_horizontal (*multiplier, spacing=0*)

Horizontally multiplies the current pixbuf content

Parameters

- **multiplier** (*int*) – How often the pixbuf shall be multiplied
- **spacing** (*int*) – amount of pixels between the pixbufs

Returns a new pixbuf

Return type *ExtendedPixbuf*

multiply_vertical (*multiplier, spacing=0*)

Vertically multiplies the current pixbuf content

Parameters

- **multiplier** (*int*) – How often the pixbuf shall be multiplied
- **spacing** (*int*) – amount of pixels between the pixbufs

Returns a new pixbuf

Return type *ExtendedPixbuf*

`xlgui.icons.extended_pixbuf_new_from_file` (*filename*)

Returns a new *ExtendedPixbuf* containing an image loaded from the specified file

Parameters **filename** (*string*) – the name of the file containing the image to load

Returns a new pixbuf

Return type *ExtendedPixbuf*

Index + search

- [modindex](#)
- [search](#)
- [Exaile Users Guide](#)
- [Exaile Developer Documentation](#)
- [search](#)

X

`xl.collection`, 25
`xl.common`, 28
`xl.covers`, 30
`xl.event`, 33
`xl.formatter`, 33
`xl.metadata`, 37
`xl.player`, 37
`xl.playlist`, 41
`xl.providers`, 45
`xl.settings`, 46
`xl.trax`, 47
`xl.xldbus`, 51
`xlgui.icons`, 51

A

add() (xl.collection.Library method), 27
 add() (xl.trax.TrackDB method), 50
 add_callback() (in module xl.event), 33
 add_horizontal() (xlgui.icons.ExtendedPixbuf method), 54
 add_icon_name_from_directory() (xlgui.icons.IconManager method), 51
 add_icon_name_from_file() (xlgui.icons.IconManager method), 52
 add_icon_name_from_pixbuf() (xlgui.icons.IconManager method), 52
 add_library() (xl.collection.Collection method), 26
 add_stock_from_directory() (xlgui.icons.IconManager method), 52
 add_stock_from_file() (xlgui.icons.IconManager method), 52
 add_stock_from_files() (xlgui.icons.IconManager method), 52
 add_stock_from_pixbuf() (xlgui.icons.IconManager method), 52
 add_stock_from_pixbufs() (xlgui.icons.IconManager method), 52
 add_tracks() (xl.trax.TrackDB method), 50
 add_vertical() (xlgui.icons.ExtendedPixbuf method), 54
 append() (xl.playlist.Playlist method), 41
 ArtistTagFormatter (class in xl.formatter), 35
 ASXConverter (class in xl.playlist), 45

B

BaseFormat (class in xl.metadata), 37

C

cached() (in module xl.common), 29
 classproperty() (in module xl.common), 29
 clear() (xl.playlist.Playlist method), 41
 clear_shuffle_history() (xl.playlist.Playlist method), 41
 close() (xl.collection.Collection method), 26
 Collection (class in xl.collection), 25

CollectionScanThread (class in xl.collection), 26
 composite_simple() (xlgui.icons.ExtendedPixbuf method), 54
 count() (xl.playlist.Playlist method), 41
 CoverManager (class in xl.covers), 31
 CoverSearchMethod (class in xl.covers), 32
 current_playlist (xl.player.queue.PlayQueue attribute), 41
 current_position (xl.playlist.Playlist attribute), 44

D

days (xl.common.TimeSpan attribute), 30
 DbusManager (class in xl.xldbus), 51
 delete() (xl.collection.Library method), 27
 dirty (xl.playlist.Playlist attribute), 44
 DiscNumberTagFormatter (class in xl.formatter), 35
 dynamic_mode (xl.playlist.Playlist attribute), 44

E

exists() (xl.trax.Track method), 48
 export_to_file() (xl.playlist.FormatConverter method), 44
 extend() (xl.playlist.Playlist method), 42
 extended_pixbuf_new_from_file() (in module xlgui.icons), 55
 ExtendedPixbuf (class in xlgui.icons), 54
 extract() (xl.formatter.Formatter method), 34

F

find_covers() (xl.covers.CoverManager method), 31
 find_covers() (xl.covers.CoverSearchMethod method), 32
 fixed (xl.covers.CoverSearchMethod attribute), 32
 fixed_priority (xl.covers.CoverSearchMethod attribute), 32
 format() (xl.formatter.ArtistTagFormatter method), 35
 format() (xl.formatter.Formatter method), 34
 format() (xl.formatter.LengthTagFormatter method), 35
 format() (xl.formatter.ProgressTextFormatter method), 34
 format() (xl.formatter.TagFormatter method), 35
 format() (xl.formatter.TrackFormatter method), 34
 format_value() (xl.formatter.LengthTagFormatter method), 36

FormatConverter (class in xl.playlist), 44
formats (in module xl.metadata), 37
Formatter (class in xl.formatter), 33
freeze_libraries() (xl.collection.Collection method), 26

G

get() (in module xl.providers), 45
get_collection_by_loc() (in module xl.collection), 25
get_cover() (xl.covers.CoverManager method), 31
get_cover_data() (xl.covers.CoverManager method), 31
get_cover_data() (xl.covers.CoverSearchMethod method), 32
get_current() (xl.player.queue.PlayQueue method), 40
get_current() (xl.playlist.Playlist method), 42
get_current_position() (xl.playlist.Playlist method), 42
get_db_string() (xl.covers.CoverManager method), 31
get_default_cover() (xl.covers.CoverManager method), 31
get_dynamic_mode() (xl.playlist.Playlist method), 42
get_format() (in module xl.metadata), 37
get_libraries() (xl.collection.Collection method), 26
get_loc_for_io() (xl.trax.Track method), 48
get_location() (xl.collection.Library method), 27
get_monitored() (xl.collection.Library method), 27
get_next() (xl.player.queue.PlayQueue method), 40
get_option() (in module xl.settings), 46
get_option() (xl.settings.SettingsManager method), 47
get_position() (xl.player._base.ExailePlayer method), 38
get_progress() (xl.player._base.ExailePlayer method), 39
get_provider() (in module xl.providers), 46
get_rating() (xl.trax.Track method), 48
get_rating_from_tracks() (in module xl.trax), 50
get_repeat_mode() (xl.playlist.Playlist method), 42
get_rescan_interval() (xl.collection.Library method), 27
get_shuffle_history() (xl.playlist.Playlist method), 42
get_shuffle_mode() (xl.playlist.Playlist method), 42
get_spat_position() (xl.playlist.Playlist method), 42
get_state() (xl.player._base.ExailePlayer method), 39
get_tag_display() (xl.trax.Track method), 48
get_tag_raw() (xl.trax.Track method), 48
get_tag_sort() (xl.trax.Track method), 48
get_time() (xl.player._base.ExailePlayer method), 38
get_tracks_from_uri() (in module xl.trax), 49
get_type() (xl.trax.Track method), 48
get_uris_from_tracks() (in module xl.trax), 49
get_volume() (xl.player._base.ExailePlayer method), 39

H

has_option() (xl.settings.SettingsManager method), 47
hours (xl.common.TimeSpan attribute), 30

I

IconManager (class in xlgui.icons), 51

import_from_file() (xl.playlist.FormatConverter method), 44
index() (xl.playlist.Playlist method), 42
is_paused() (xl.player._base.ExailePlayer method), 39
is_play_enabled() (xl.player.queue.PlayQueue method), 40
is_playing() (xl.player._base.ExailePlayer method), 39
is_stopped() (xl.player._base.ExailePlayer method), 39
is_valid_track() (in module xl.trax), 49

L

LastPlayedTagFormatter (class in xl.formatter), 36
LengthTagFormatter (class in xl.formatter), 35
Library (class in xl.collection), 27
LibraryMonitor (class in xl.collection), 28
LimitedCache (class in xl.common), 30
list_tags() (xl.trax.Track method), 48
load() (xl.covers.CoverManager method), 31
load_from_location() (xl.playlist.Playlist method), 42
load_from_location() (xl.trax.TrackDB method), 50
local_file_name() (xl.trax.Track method), 48
LocalFileCoverFetcher (class in xl.covers), 32
log_event() (in module xl.event), 33
log_exception() (in module xl.common), 28

M

M3UConverter (class in xl.playlist), 44
MANAGER (in module xl.covers), 31
MANAGER (in module xl.providers), 45
MANAGER (in module xl.settings), 46
MetadataList (class in xl.common), 30
minutes (xl.common.TimeSpan attribute), 30
monitored (xl.collection.Library attribute), 28
move() (xlgui.icons.ExtendedPixbuf method), 54
multiply_horizontal() (xlgui.icons.ExtendedPixbuf method), 55
multiply_vertical() (xlgui.icons.ExtendedPixbuf method), 55

N

name (xl.covers.CoverSearchMethod attribute), 32
name (xl.playlist.Playlist attribute), 44
name_from_path() (xl.playlist.FormatConverter method), 44
next() (xl.player.queue.PlayQueue method), 40
next() (xl.playlist.Playlist method), 42
NotReadable, 37
NotWritable, 37

O

on_scan_progress_update() (xl.collection.CollectionScanThread method), 27
open_file() (in module xl.common), 29

open_file_directory() (in module xl.common), 29
 order_poset() (in module xl.common), 28

P

ParameterTemplate (class in xl.formatter), 36
 pause() (xl.player._base.ExailePlayer method), 38
 pixbuf_from_data() (xlgui.icons.IconManager method), 53
 pixbuf_from_icon_name() (xlgui.icons.IconManager method), 53
 pixbuf_from_rating() (xlgui.icons.IconManager method), 53
 pixbuf_from_stock() (xlgui.icons.IconManager method), 53
 pixbuf_from_text() (xlgui.icons.IconManager method), 53
 play() (xl.player._base.ExailePlayer method), 38
 play() (xl.player.queue.PlayQueue method), 40
 PLAYER (in module xl.player), 37
 Playlist (class in xl.playlist), 41
 PlayQueue (class in xl.player.queue), 40
 PLSCConverter (class in xl.playlist), 45
 pop() (xl.playlist.Playlist method), 43
 PosetItem (class in xl.common), 30
 prev() (xl.player.queue.PlayQueue method), 40
 prev() (xl.playlist.Playlist method), 43
 profileit() (in module xl.common), 29
 ProgressTextFormatter (class in xl.formatter), 34
 ProgressThread (class in xl.common), 30
 ProviderHandler (class in xl.providers), 46
 ProviderManager (class in xl.providers), 46

Q

QUEUE (in module xl.player), 39
 queue_length() (xl.player.queue.PlayQueue method), 40

R

randomize() (xl.playlist.Playlist method), 43
 RatingTagFormatter (class in xl.formatter), 36
 read_all() (xl.metadata.BaseFormat method), 37
 read_tags() (xl.metadata.BaseFormat method), 37
 read_tags() (xl.trax.Track method), 48
 register() (in module xl.providers), 45
 remove() (xl.trax.TrackDB method), 50
 remove_callback() (in module xl.event), 33
 remove_cover() (xl.covers.CoverManager method), 31
 remove_library() (xl.collection.Collection method), 26
 remove_option() (xl.settings.SettingsManager method), 47
 remove_tracks() (xl.trax.TrackDB method), 50
 repeat_mode (xl.playlist.Playlist attribute), 44
 rescan() (xl.collection.Library method), 28
 rescan_libraries() (xl.collection.Collection method), 26
 run() (xl.collection.CollectionScanThread method), 27

run() (xl.common.ProgressThread method), 30

S

save() (xl.covers.CoverManager method), 31
 save() (xl.settings.SettingsManager method), 47
 save_to_location() (xl.playlist.Playlist method), 43
 save_to_location() (xl.trax.TrackDB method), 50
 search_tracks() (in module xl.trax), 51
 search_tracks_from_string() (in module xl.trax), 51
 seconds (xl.common.TimeSpan attribute), 30
 seek() (xl.player._base.ExailePlayer method), 38
 serialize_libraries() (xl.collection.Collection method), 26
 set_cover() (xl.covers.CoverManager method), 31
 set_current_playlist() (xl.player.queue.PlayQueue method), 40
 set_current_position() (xl.playlist.Playlist method), 43
 set_dynamic_mode() (xl.playlist.Playlist method), 43
 set_loc() (xl.trax.Track method), 49
 set_location() (xl.collection.Library method), 28
 set_monitored() (xl.collection.Library method), 28
 set_option() (in module xl.settings), 46
 set_option() (xl.settings.SettingsManager method), 47
 set_preferred_order() (xl.covers.CoverManager method), 32
 set_progress() (xl.player._base.ExailePlayer method), 39
 set_rating() (xl.trax.Track method), 49
 set_repeat_mode() (xl.playlist.Playlist method), 43
 set_rescan_interval() (xl.collection.Library method), 28
 set_shuffle_mode() (xl.playlist.Playlist method), 43
 set_spat_position() (xl.playlist.Playlist method), 43
 set_tag_raw() (xl.trax.Track method), 49
 set_volume() (xl.player._base.ExailePlayer method), 39
 SettingsManager (class in xl.settings), 46
 shuffle_mode (xl.playlist.Playlist attribute), 44
 shuffle_mode_names (xl.playlist.Playlist attribute), 44
 shuffle_modes (xl.playlist.Playlist attribute), 44
 sort() (xl.playlist.Playlist method), 43
 sort_result_tracks() (in module xl.trax), 50
 sort_tracks() (in module xl.trax), 49
 spat_position (xl.playlist.Playlist attribute), 44
 stop() (xl.collection.CollectionScanThread method), 27
 stop() (xl.common.ProgressThread method), 30
 stop() (xl.player._base.ExailePlayer method), 38
 stop_scan() (xl.collection.Collection method), 26
 synchronized() (in module xl.common), 29

T

TagCoverFetcher (class in xl.covers), 32
 TagFormatter (class in xl.formatter), 35
 thaw_libraries() (xl.collection.Collection method), 26
 threaded() (in module xl.common), 29
 TimeSpan (class in xl.common), 30
 to_unicode() (in module xl.common), 28
 toggle_pause() (xl.player._base.ExailePlayer method), 38

Track (class in xl.trax), 47
TrackDB (class in xl.trax), 50
TrackFormatter (class in xl.formatter), 34
TrackNumberTagFormatter (class in xl.formatter), 35
TracksMatcher (class in xl.trax), 51

U

unpause() (xl.player_base.ExailePlayer method), 38
unregister() (in module xl.providers), 45
unserialize_libraries() (xl.collection.Collection method),
26
update_track() (xl.collection.Library method), 28
use_cache (xl.covers.CoverSearchMethod attribute), 32

V

VersionError, 30

W

walk() (in module xl.common), 29
walk_directories() (in module xl.common), 29
write_tags() (xl.metadata.BaseFormat method), 37
write_tags() (xl.trax.Track method), 49

X

xl.collection (module), 25
xl.common (module), 28
xl.covers (module), 30
xl.event (module), 33
xl.formatter (module), 33
xl.metadata (module), 37
xl.player (module), 37
xl.playlist (module), 41
xl.providers (module), 45
xl.settings (module), 46
xl.trax (module), 47
xl.xldbus (module), 51
xlgui.icons (module), 51
xlgui.icons.MANAGER (in module xlgui.icons), 51
XSPFConverter (class in xl.playlist), 45