AnyBlok / Dramatiq Documentation

Release 1.0.3

Jean-Sébastien Suzanne

Feb 24, 2018

Contents:

Contents

CHAPTER 1

Front Matter

Information about the AnyBlok / Dramatiq project.

1.1 Project Homepage

AnyBlok is hosted on github - the main project page is at https://github.com/AnyBlok/anyblok_dramatiq. Source code is tracked here using GIT.

Releases and project status are available on Pypi at http://pypi.python.org/pypi/anyblok_dramatiq.

The most recent published version of this documentation should be at http://doc.anyblok-dramatiq.anyblok.org.

1.2 Project Status

AnyBlok with Dramatiq is currently in beta status and is expected to be fairly stable. Users should take care to report bugs and missing features on an as-needed basis. It should be expected that the development version may be required for proper implementation of recently repaired issues in between releases;

1.3 Installation

Install released versions of AnyBlok from the Python package index with pip or a similar tool:

```
pip install anyblok_dramatiq
```

Installation via source distribution is via the `setup.py` script:

```
python setup.py install
```

Installation will add the `anyblok` commands to the environment.

1.4 Unit Test

Run the test with `nose`:

```
pip install nose
nosetests anyblok_dramatiq/tests
```

1.5 Dependencies

AnyBlok / Dramatiq works with **Python 3.6** and later. The install process will ensure that AnyBlok, dramatiq are installed, in addition to other dependencies. The latest version of them is strongly recommended.

1.6 Contributing (hackers needed!)

Anyblok / Dramatiq is at a very early stage, feel free to fork, talk with core dev, and spread the word!

1.7 Author

Jean-Sébastien Suzanne

1.8 Contributors

Anybox team:

- Jean-Sébastien Suzanne

Sensee team:

- Franck Bret

1.9 Bugs

Bugs and feature enhancements to AnyBlok should be reported on the Issue tracker.

Contents

CHAPTER 2

MEMENTO

2.1 How to use dramatiq and which context

The goal of dramatiq is to process some task on another system process. If your tasks will be done on the same process, you read the wrong soluce. But if your tasks can be execute in another process, and its take time to process them, your are welcome

The first thing to known is you will need to run to application:

- anyblok_dramatiq: to process the task.
- Another anyblok script to execute the main process, anyblok_pyramid if you have a web service.

Make attention that the both use the same broker, else they should not be communicate each other.

Warning: the blok `dramatiq` must be installed

To execute your task by dramatiq, you have to define **actor** or **actor_send** on your AnyBlok Model. Read the doc of the doc of `dramatiq` blok to know how declare it.

2.2 Add middleware on dramatiq

dramatiq allow to add middleware to improve the process, **anyblok_dramatiq** add one middleware for historize the messages and their status.

You can add in your project an existing **dramatiq** middleware or your own. read more to known existing middleware or how create your own.

anyblok_dramatiq add this own console script to run the workers, you need add the middleware in the entrypoint `anyblok_dramatiq.middleware`:

```
setup(
    ...
    entry_points={
        'anyblok_dramatiq.middleware': [
            'mymiddleware=module.path:ClassName',
        ],
    },
    ...
)
```

Contents

CHAPTER 3

Code

3.1 Actor

exception anyblok_dramatiq.actor.**AnyBlokActorException**
Bases: ValueError

A ValueError exception for anyblok_dramatiq

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

class anyblok_dramatiq.actor.**AnyBlokActor**(*fn*, *, *broker*, *actor_name*, *queue_name*, *priority*, *options*)
Bases: dramatiq.actor.Actor

Overload the dramatiq.actor.Actor class

the goal is to allowthe decorator actor_send, this decorator use directly the method send

send(**args*, ***kwargs*)
Send to the broker

anyblok_dramatiq.actor.**declare_actor_for**(*method*, ***kwargs*)
Method to add anyblok_dramatiq.actor.actor decorator on the class method

Parameters

- **method** – classmethod pointer
- **_**kwargs** – decorator kwargs

anyblok_dramatiq.actor.**declare_actor_send_for**(*method*, ***kwargs*)
Method to add anyblok_dramatiq.actor.actor_send decorator on the class method

Parameters

- **method** – classmethod pointer
- **_**kwargs** – decorator kwargs

anyblok_dramatiq.actor.**actor**(*queue_name='default'*, *priority=0*, ***options*)
Decorator to get an Actor

Parameters

- **queue_name** – name of the queue
- **priority** – priority of the actor
- **_**options** – options for actor

anyblok_dramatiq.actor.**actor_send**(*queue_name='default'*, *priority=0*, ***options*)
Decorator to get an AnyBlokActor

Parameters

- **queue_name** – name of the queue
- **priority** – priority of the actor
- **_**options** – options for actor

anyblok_dramatiq.actor.**call_directly_the_actor_send**()
Context manager to call directly without use dramatiq

class anyblok_dramatiq.actor.**ActorPlugin**(*registry*)
Bases: anyblok.model.plugins.ModelPluginBase

anyblok.model.plugin to allow the build of the anyblok_dramatiq.actor

after_model_construction(*base*, *namespace*, *transformation_properties*)
Do some action with the constructed Model

Parameters

- **base** – the Model class
- **namespace** – the namespace of the model
- **transformation_properties** – the properties of the model

initialisation_tranformation_properties(*properties*, *transformation_properties*)
Initialise the transform properties

Parameters

- **properties** – the properties declared in the model
- **new_type_properties** – param to add in a new base if need

insert_in_bases(*new_base*, *namespace*, *properties*, *transformation_properties*)
Insert in a base the overload

Parameters

- **new_base** – the base to be put on front of all bases
- **namespace** – the namespace of the model
- **properties** – the properties declared in the model
- **transformation_properties** – the properties of the model

transform_base_attribute(*attr*, *method*, *namespace*, *base*, *transformation_properties*, *new_type_properties*)
transform the attribute for the final Model

Parameters

- **attr** – attribute name
- **method** – method pointer of the attribute
- **namespace** – the namespace of the model
- **base** – One of the base of the model
- **transformation_properties** – the properties of the model
- **new_type_properties** – param to add in a new base if need

class anyblok_dramatiq.actor.**ActorSendPlugin**(*registry*)

Bases: anyblok.model.plugins.ModelPluginBase

anyblok.model.plugin to allow the build of the anyblok_dramatiq.actor_send

after_model_construction(*base*, *namespace*, *transformation_properties*)

Do some action with the constructed Model

Parameters

- **base** – the Model class
- **namespace** – the namespace of the model
- **transformation_properties** – the properties of the model

initialisation_tranformation_properties(*properties*, *transformation_properties*)

Initialise the transform properties

Parameters

- **properties** – the properties declared in the model
- **new_type_properties** – param to add in a new base if need

insert_in_bases(*new_base*, *namespace*, *properties*, *transformation_properties*)

Insert in a base the overload

Parameters

- **new_base** – the base to be put on front of all bases
- **namespace** – the namespace of the model
- **properties** – the properties declared in the model
- **transformation_properties** – the properties of the model

transform_base_attribute(*attr*, *method*, *namespace*, *base*, *transformation_properties*, *new_type_properties*)

transform the attribute for the final Model

Parameters

- **attr** – attribute name
- **method** – method pointer of the attribute
- **namespace** – the namespace of the model
- **base** – One of the base of the model
- **transformation_properties** – the properties of the model
- **new_type_properties** – param to add in a new base if need

3.2 Broker

anyblok_dramatiq.broker.**prepare_broker**(*withmiddleware=True*)
Configure the broker for send and workers

3.3 Dramatiq middleware

class anyblok_dramatiq.middleware.**DramatiqMessageMiddleware**
Bases: dramatiq.middleware.middleware.Middleware

Middleware for dramatiq, the goal is to detect if the the call was done by anyblok tools with the Model.Dramatiq.Message. This model stock the status of the message and the history of the status's change

after_process_message(*broker*, *message*, *, *result=None*, *exception=None*)
Called after process message

If the message is in the Model.Dramatiq.Message then the status will be change to **done** or **failed**.

Note: the status is failed if an exception is passed or a rollback is need

---

Before the end, the session is expired to release the Session pool thread

Parameters

- **broker** – the broker used
- **message** – the message send in the broker
- **result** – return by the process
- **exception** – any Exception raised by the process

after_skip_message(*broker*, *message*)
Called after skip message

If the message is in the Model.Dramatiq.Message then the status will be change to **skip**

Before the end, the session is expired to release the Session pool thread

Parameters

- **broker** – the broker used
- **message** – the message send in the broker

after_worker_shutdown(**args*, ***kwargs*)
Called before worker shutdown

Close the AnyBlok registry

before_consumer_thread_shutdown(**args*, ***kwargs*)
Called before consumer thread shutdown

remove the session instance to clean the Session pool

before_enqueue(*broker*, *message*, *delay*)
Called when a message is delayed or enqueued

If the message is in the Model.Dramatiq.Message then the status will be change to **delayed** or **enqueued**

Parameters

- **broker** – the broker used
- **message** – the message send in the broker
- **delay** – delay in milliseconds

before_process_message(*broker*, *message*)
Called before process message

Invalid the cache, this is mean that if a cache have to be invalidated then it will be invalidated else nothing is done

If the message is in the `Model.Dramatiq.Message` then the status will be change to **running**

Parameters

- **broker** – the broker used
- **message** – the message send in the broker

before_worker_thread_shutdown(**args*, ***kwargs*)
Called before worker thread shutdown

remove the session instance to clean the Session pool

3.4 Scripts

anyblok_dramatiq.scripts.**worker_process**(*worker_id*, *logging_fd*)
consume worker to process messages and execute the actor

Contents

CHAPTER 4

Bloks

4.1 Blok dramatiq

class anyblok_dramatiq.bloks.dramatiq.**DramatiqBlok**(*registry*)

Bases: anyblok.blok.Blok

Dramatiq's Blok class definition

author = 'jssuzanne'

conditional_by = []

conflicting_by = []

classmethod declare_actors(*registry*)

Actor declaration

```
from anyblok_dramatiq import (
    declare_actor_for,
    declare_actor_send_for,
)
declare_actor_for(Model.methode_name)
# or
declare_actor_send_for(Model.methode_name)
```

classmethod import_declaration_module()

Python module to import in the given order at start-up

load()

Load all the actor defined in all the installed bloks

name = 'dramatiq'

optional_by = []

classmethod reload_declaration_module(*reload*)

Python module to import while reloading server (ie when adding Blok at runtime

```
required = ['anyblok-core']
required_by = ['dramatiq-task']
version = '1.0.3'
```

4.1.1 Memento

An **actor** method is a classmethod who are executed by the dramatiq worker. AnyBlok define two differente actor decorator:

- `actor`: Works exactly as the dramatiq.actor decorator
- `actor_send`: This actor call the send broker method by default

The actor decorator from anyblok must decorate only an AnyBlok Model, Mixin or Core

actor

The more basic actor:

```
from anyblok_dramatiq import actor

@register(Model)
class MyModel:

    ...

    @actor()
    def actor_method(cls, *args, **kwargs)
        # do something
        ...
```

This use case is simple, you may:

- Call directly the actor_method and execute it:

```
registry.MyModel.actor_method(, *a, **kw)
```

- Use the dramatiq functionnality without `Model.Dramatiq.Message`:

```
message = registry.MyModel.actor_method.send(*a, **kw)
registry.Dramatiq.send2broker(message)
```

- Use the dramatiq functionnality with `Model.Dramatiq.Message` to get status and history:

```
registry.Dramatiq.create_message(registry.MyModel.actor_method, *a, **kw)
```

> **Note:** In this case the message will be send to dramatiq worker by the `postcommit_hook` of AnyBlok

actor_send

The more basic actor:

```
from anyblok_dramatiq import actor_send

@register(Model)
class MyModel:

    ...

    @actor_send()
    def actor_method(cls, *args, **kwargs)
        # do something
        ...
```

By default this decorator allow one case, use the dramatiq functionnality with `Model.Dramatiq.Messag`:

```
registry.MyModel.actor_method(*a, **kw)
```

The inheritance of AnyBlok allow to overwrite all classmethod to transform them by an actor easily.

In the case where you want execute directly the actor you have to use the context manager `call_directly_the_actor_send`:

```
from anyblok_dramatiq import call_directly_the_actor_send

with call_directly_the_actor_send():
    registry.MyModel.actor_method(*a, **kw)
```

4.1.2 API doc

Message

class anyblok_dramatiq.bloks.dramatiq.message.**Dramatiq**
Bases: object

No SQL Model, use to get tools for dramatiq messaging

Declaration type Model

Registry name Model.Dramatiq

Tablename dramatiq

Inherit model or mixin

classmethod create_message(*actor*, **args*, ***kwargs*)
Prepare a message and add an entry in the Message Model :param actor: an Actor instance :param delay: use for postcommit hook send2broker :param _*args: args of the actor :param _*_*kwargs: kwargs of the actor :rtype: dramatiq message instance

classmethod send2broker(**messages*, *delay=None*, *run_at=None*)
Send all the messages with the delay

Parameters

- **_*messages** – message instance list
- **delay** – delay before send
- **run_at** – datetime when the process must be executed

class anyblok_dramatiq.bloks.dramatiq.message.**Message**
Bases: anyblok.mixin.DramatiqMessageStatus

Message model for dramatiq

Declaration type Model

Registry name Model.Dramatiq.Message

Tablename dramatiq_message

Inherit model or mixin

- <class 'anyblok.mixin.DramatiqMessageStatus'>

field name	Description
id	• primary_key - True • is crypted - False • DB column name - None • nullable - False • Context: • Field type - <class 'anyblok.column.UUID'> • default - <class 'anyblok.column.NoDefaultValue'> • foreign_key - None • Label - None
message	• DB column name - None • is crypted - False • nullable - False • Context: • Field type - <class 'anyblok.column.Json'> • default - <class 'anyblok.column.NoDefaultValue'> • foreign_key - None • Label - None
updated_at	• Context: • Field type - <class 'anyblok.column.DateTime'> • is auto updated - False • Label - None

classmethod get_instance_of(*message*)
Called by the middleware to get the model instance of the message

classmethod insert(**args*, ***kwargs*)
Over write the insert to add the first history line

update_status(*status*, *error=None*)
Called by the middleware to change the status and history

4.2 Blok dramatiq task

class anyblok_dramatiq.bloks.task.**DramatiqTaskBlok**(*registry*)
Bases: anyblok.blok.Blok

Dramatiq's task definition

author = 'jssuzanne'

conditional_by = []

conflicting_by = []

classmethod import_declaration_module()
Python module to import in the given order at start-up

name = 'dramatiq-task'

optional_by = []

classmethod reload_declaration_module(*reload*)
Python module to import while reloading server (ie when adding Blok at runtime

required = ['anyblok-core', 'dramatiq']

required_by = []

version = '1.0.3'

4.2.1 Memento

The tasks is based on dramatiq, The instance of the model:

- Task: define what the task have to do
- Job: historize the execution of an instance of Task with specific data

Model.Dramatiq.Task

This model is not directly useable, you have to use polymosphic model:

- Model.Dramatiq.Task.CallMethod: call a classmethod on a model, defined by the task
- Model.Dramatiq.Task.StepByStep: call each sub task one by one on function of the order
- Model.Dramatiq.Task.Parallel: call all sub tasks on one shot

Model.Dramatiq.Job

The job is the execution of the task with dramatiq, The job historize also action done and action to do. To create a job, you must get a task create the job with the method do_the_job:

```
task = registry.Dramatiq.Task.query().first()
task.do_the_job(with_args=tuple(), with_kwargs=dict(), run_at=a_datetime)
# this job will not be traited now and by the process but by another process
```

Note: In the case where the task will be an StepByStep or Parallel task, then the task create one or more job, one for the job and one for sub jobs

4.2.2 API doc

Task

class anyblok_dramatiq.bloks.task.task.**Task**

Bases: object

Main Task, define the main table

Declaration type Model

Registry name Model.Dramatiq.Task

Tablename dramatiq_task

Inherit model or mixin

field name	Description
id	• DB column name - None • is crypted - False • primary_key - True • Context: • Field type - <class 'anyblok.column.Integer'> • autoincrement - True • default - <class 'anyblok.column.NoDefaultValue'> • foreign_key - None • Label - None
label	• DB column name - None • is crypted - False • nullable - False • Context: • Field type - <class 'anyblok.column.String'> • size - 64 • default - <class 'anyblok.column.NoDefaultValue'> • foreign_key - None • Label - None
update_at	• nullable - False • Context: • Field type - <class 'anyblok.column.DateTime'> • is auto updated - True • Label - None
main_task	• unique - False • _column_names - None • backref - sub_tasks • info: • primary_key - False • unique - False • remote_model - Model.Dramatiq.Task • remote_name - sub_tasks • index - False • model - Model.Dramatiq.Task • Context: • Field type - <class 'anyblok.relationship.Many2One'> • primary_key - False • _remote_columns - None • index - False • Label - None
create_at	• nullable - False • Context: • Field type - <class 'any-

classmethod define_mapper_args()
Polymorphism configuration

do_the_job(*main_job=None*, *run_at=None*, *with_args=None*, *with_kwargs=None*)
Create a job for this tash and add send it to dramatiq

Parameters

- **main_job** – parent job if exist
- **run_at** – datetime to execute the job
- **with_args** – tuple of the argument to pass at the job
- **with_kwargs** – dict of the argument to pass at the job

classmethod get_task_type()
List the task type possible

run(*job*)
Execute the task for one job

Parameters **job** – job executed

run_next(*job*)
next action to execute when a sub job finish this task for one job

Parameters **job** – job executed

Job

class anyblok_dramatiq.bloks.task.job.**Job**
Bases: object

The job is an execution of an instance of task

Declaration type Model

Registry name Model.Dramatiq.Job

Tablename dramatiq_job

Inherit model or mixin

field name	Description
uuid	• primary_key - True • is crypted - False • DB column name - None • nullable - False • Context: • Field type - <class 'anyblok.column.UUID'> • default - <function uuid1 at 0x7fa1edcf5510> • foreign_key - None • Label - None
status	• DB column name - None • is crypted - False • selections: • ('new', 'New') • ('waiting', 'Waiting') • ('running', 'Running') • ('failed', 'Failed') • ('done', 'Done') • nullable - False • Context: • Field type - <class 'anyblok.column.Selection'> • size - 64 • default - new • foreign_key - None • Label - None
task	• unique - False • _column_names - None • info: • nullable - False • unique - False • primary_key - False • remote_model - Model.Dramatiq.Task • index - False • model - Model.Dramatiq.Task • Context: • Field type - <class 'anyblok.relationship.Many2One'> • primary_key - False • _remote_columns - None • index - False • Label - None
update_at	• nullable - False • Context: • Field type - <class 'anyblok.column.DateTime'> • is auto updated - True • Label - None

call_main_job()
Call the main job if exist to do the next action of the main job

lock()
lock the job to be sure that only one thread execute the run_next

classmethod run(*job_uuid=None*)
dramatiq actor to execute a specific task

actor_send event call with positionnal argument {'priority': 0, 'queue_name': 'default'}

Contents

CHAPTER 5

CHANGELOG

5.1 1.0.3 (2018-02-24)

- [REF] Anyblok 0.17.0 changed setter to add application and application groups, So I had to adapt the existing to use new setter

5.2 1.0.2 (2018-02-12)

- [FIX] multi process lock AnyBlok seem lock the data base during the migration, the dramatiq process don't migrate the data base, the migration is now forbidden

5.3 1.0.1 (2018-01-10)

- [FIX] put the configuration `dramatiq-broker` on the default application

5.4 1.0.0 (2017-12-23)

- [IMP] dramatiq console script to execute workers process
- [IMP] actor and actor_send decorator to define dramatiq actor
- [IMP] dramatiq middleware to modify `Model.Dramatiq.Message` status
- [IMP] dramatiq blok to historize the message and status
- [IMP] dramatiq-task to add a back task with dramatiq

CHAPTER 6

Mozilla Public License Version 2.0

6.1 1. Definitions

6.1.1 1.1. “Contributor”

Means each individual or legal entity that creates, contributes to the creation of, or owns Covered Software.

6.1.2 1.2. “Contributor Version”

Means the combination of the Contributions of others (if any) used by a Contributor and that particular Contributor’s Contribution.

6.1.3 1.3. “Contribution”

Means Covered Software of a particular Contributor.

6.1.4 1.4. “Covered Software”

Means Source Code Form to which the initial Contributor has attached the notice in Exhibit A, the Executable Form of such Source Code Form, and Modifications of such Source Code Form, in each case including portions thereof.

6.1.5 1.5. “Incompatible With Secondary Licenses”

Means:

- **That the initial Contributor has attached the notice described in Exhibit B** to the Covered Software; or
- **That the Covered Software was made available under the terms of version 1.1** or earlier of the License, but not also under the terms of a Secondary License.

6.1.6 1.6. “Executable Form”

Means any form of the work other than Source Code Form.

6.1.7 1.7. “Larger Work”

Means a work that combines Covered Software with other material, in a separate file or files, that is not Covered Software.

6.1.8 1.8. “License”

Means this document.

6.1.9 1.9. “Licensable”

Means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently, any and all of the rights conveyed by this License.

6.1.10 1.10. “Modifications”

Means any of the following:

- **Any file in Source Code Form that results from an addition to, deletion from,** or modification of the contents of Covered Software; or
- Any new file in Source Code Form that contains any Covered Software.

6.1.11 1.11. “Patent Claims” of a Contributor

Means any patent claim(s), including without limitation, method, process, and apparatus claims, in any patent Licensable by such Contributor that would be infringed, but for the grant of the License, by the making, using, selling, offering for sale, having made, import, or transfer of either its Contributions or its Contributor Version.

6.1.12 1.12. “Secondary License”

Means either the GNU General Public License, Version 2.0, the GNU Lesser General Public License, Version 2.1, the GNU Affero General Public License, Version 3.0, or any later versions of those licenses.

6.1.13 1.13. “Source Code Form”

Means the form of the work preferred for making modifications.

6.1.14 1.14. “You” (or “Your”)

Means an individual or a legal entity exercising rights under this License. For legal entities, “You” includes any entity that controls, is controlled by, or is under common control with You. For purposes of this definition, “control” means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

6.2 2. License Grants and Conditions

6.2.1 2.1. Grants

Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

- **Under intellectual property rights (other than patent or trademark)** Licensable by such Contributor to use, reproduce, make available, modify, display, perform, distribute, and otherwise exploit its Contributions, either on an unmodified basis, with Modifications, or as part of a Larger Work; and
- **Under Patent Claims of such Contributor to make, use, sell, offer for sale,** have made, import, and otherwise transfer either its Contributions or its Contributor Version.

6.2.2 2.2. Effective Date

The licenses granted in Section 2.1 with respect to any Contribution become effective for each Contribution on the date the Contributor first distributes such Contribution.

6.2.3 2.3. Limitations on Grant Scope

The licenses granted in this Section 2 are the only rights granted under this License. No additional rights or licenses will be implied from the distribution or licensing of Covered Software under this License. Notwithstanding Section 2.1(b) above, no patent license is granted by a Contributor:

- For any code that a Contributor has removed from Covered Software; or
- **For infringements caused by: (i) Your and any other third party's** modifications of Covered Software, or (ii) the combination of its Contributions with other software (except as part of its Contributor Version); or
- **Under Patent Claims infringed by Covered Software in the absence of its** Contributions.

This License does not grant any rights in the trademarks, service marks, or logos of any Contributor (except as may be necessary to comply with the notice requirements in Section 3.4).

6.2.4 2.4. Subsequent Licenses

No Contributor makes additional grants as a result of Your choice to distribute the Covered Software under a subsequent version of this License (see Section 10.2) or under the terms of a Secondary License (if permitted under the terms of Section 3.3).

6.2.5 2.5. Representation

Each Contributor represents that the Contributor believes its Contributions are its original creation(s) or it has sufficient rights to grant the rights to its Contributions conveyed by this License.

6.2.6 2.6. Fair Use

This License is not intended to limit any rights You have under applicable copyright doctrines of fair use, fair dealing, or other equivalents.

6.2.7 2.7. Conditions

Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in Section 2.1.

6.3 3. Responsibilities

6.3.1 3.1. Distribution of Source Form

All distribution of Covered Software in Source Code Form, including any Modifications that You create or to which You contribute, must be under the terms of this License. You must inform recipients that the Source Code Form of the Covered Software is governed by the terms of this License, and how they can obtain a copy of this License. You may not attempt to alter or restrict the recipients' rights in the Source Code Form.

6.3.2 3.2. Distribution of Executable Form

If You distribute Covered Software in Executable Form then:

- **Such Covered Software must also be made available in Source Code Form, as** described in Section 3.1, and You must inform recipients of the Executable Form how they can obtain a copy of such Source Code Form by reasonable means in a timely manner, at a charge no more than the cost of distribution to the recipient; and
- **You may distribute such Executable Form under the terms of this License, or** sublicense it under different terms, provided that the license for the Executable Form does not attempt to limit or alter the recipients' rights in the Source Code Form under this License.

6.3.3 3.3. Distribution of a Larger Work

You may create and distribute a Larger Work under terms of Your choice, provided that You also comply with the requirements of this License for the Covered Software. If the Larger Work is a combination of Covered Software with a work governed by one or more Secondary Licenses, and the Covered Software is not Incompatible With Secondary Licenses, this License permits You to additionally distribute such Covered Software under the terms of such Secondary License(s), so that the recipient of the Larger Work may, at their option, further distribute the Covered Software under the terms of either this License or such Secondary License(s).

6.3.4 3.4. Notices

You may not remove or alter the substance of any license notices (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the Source Code Form of the Covered Software, except that You may alter any license notices to the extent required to remedy known factual inaccuracies.

6.3.5 3.5. Application of Additional Terms

You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, You may do so only on Your own behalf, and not on behalf of any Contributor. You must make it absolutely clear that any such warranty, support, indemnity, or liability obligation is offered by You alone, and You hereby agree to indemnify every Contributor for any liability incurred by such Contributor as a result of warranty, support, indemnity or liability terms You offer. You may include additional disclaimers of warranty and limitations of liability specific to any jurisdiction.

6.4 4. Inability to Comply Due to Statute or Regulation

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Software due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be placed in a text file included with all distributions of the Covered Software under this License. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

6.5 5. Termination

6.5.1 5.1.

The rights granted under this License will terminate automatically if You fail to comply with any of its terms. However, if You become compliant, then the rights granted under this License from a particular Contributor are reinstated (a) provisionally, unless and until such Contributor explicitly and finally terminates Your grants, and (b) on an ongoing basis, if such Contributor fails to notify You of the non-compliance by some reasonable means prior to 60 days after You have come back into compliance. Moreover, Your grants from a particular Contributor are reinstated on an ongoing basis if such Contributor notifies You of the non-compliance by some reasonable means, this is the first time You have received notice of non-compliance with this License from such Contributor, and You become compliant prior to 30 days after Your receipt of the notice.

6.5.2 5.2.

If You initiate litigation against any entity by asserting a patent infringement claim (excluding declaratory judgment actions, counter-claims, and cross-claims) alleging that a Contributor Version directly or indirectly infringes any patent, then the rights granted to You by any and all Contributors for the Covered Software under Section 2.1 of this License shall terminate.

6.5.3 5.3.

In the event of termination under Sections 5.1 or 5.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or Your distributors under this License prior to termination shall survive termination.

6.6 6. Disclaimer of Warranty

Warning: Covered Software is provided under this License on an "as is" basis, without warranty of any kind, either expressed, implied, or statutory, including, without limitation, warranties that the Covered Software is free of defects, merchantable, fit for a particular purpose or non-infringing. The entire risk as to the quality and performance of the Covered Software is with You. Should any Covered Software prove defective in any respect, You (not any Contributor) assume the cost of any necessary servicing, repair, or correction. This disclaimer of warranty constitutes an essential part of this License. No use of any Covered Software is authorized under this License except under this disclaimer.

6.7 7. Limitation of Liability

Warning: Under no circumstances and under no legal theory, whether tort (including negligence), contract, or otherwise, shall any Contributor, or anyone who distributes Covered Software as permitted above, be liable to You for any direct, indirect, special, incidental, or consequential damages of any character including, without limitation, damages for lost profits, loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses, even if such party shall have been informed of the possibility of such damages. This limitation of liability shall not apply to liability for death or personal injury resulting from such party's negligence to the extent applicable law prohibits such limitation. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so this exclusion and limitation may not apply to You.

6.8 8. Litigation

Any litigation relating to this License may be brought only in the courts of a jurisdiction where the defendant maintains its principal place of business and such litigation shall be governed by laws of that jurisdiction, without reference to its conflict-of-law provisions. Nothing in this Section shall prevent a party's ability to bring cross-claims or counter-claims.

6.9 9. Miscellaneous

This License represents the complete agreement concerning the subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not be used to construe this License against a Contributor.

6.10 10. Versions of the License

6.10.1 10.1. New Versions

Mozilla Foundation is the license steward. Except as provided in Section 10.3, no one other than the license steward has the right to modify or publish new versions of this License. Each version will be given a distinguishing version number.

6.10.2 10.2. Effect of New Versions

You may distribute the Covered Software under the terms of the version of the License under which You originally received the Covered Software, or under the terms of any subsequent version published by the license steward.

6.10.3 10.3. Modified Versions

If you create software not governed by this License, and you want to create a new license for such software, you may create and use a modified version of this License if you rename the license and remove any references to the name of the license steward (except to note that such modified license differs from this License).

6.10.4 10.4. Distributing Source Code Form that is Incompatible With Secondary Licenses

If You choose to distribute Source Code Form that is Incompatible With Secondary Licenses under the terms of this version of the License, the notice described in Exhibit B of this License must be attached.

6.11 Exhibit A - Source Code Form License Notice

```
This Source Code Form is subject to the terms of the Mozilla Public
License, v. 2.0. If a copy of the MPL was not distributed with this file,
You can obtain one at http://mozilla.org/MPL/2.0/.
```

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

Note: You may add additional accurate notices of copyright ownership.

6.12 Exhibit B - “Incompatible With Secondary Licenses” Notice

```
This Source Code Form is “Incompatible With Secondary Licenses”, as defined
by the Mozilla Public License, v. 2.0.
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

A