Captain Shove Documentation

Release 0.1

Mozilla Foundation

Sep 27, 2017

Contents

Captain Shove is a system for remote command execution for multiple projects via a central frontend.

Code https://github.com/mozilla/captain

Documentation http://captain.readthedocs.org/

Issue tracker https://github.com/mozilla/captain/issues

IRC `#capshove` on irc.mozilla.org

License Mozilla Public License v2

Contents:

CHAPTER 1

Overview

Captain Shove is a system for remote command execution for multiple projects via a central frontend. Projects define a whitelist of acceptable commands. Users use the Captain web frontend to execute commands on a remote machine running a Shove daemon. Captain then shows info about the result of executing that command including the return code, console log and its current favorite ice cream flavor.

Captain is the frontend portion of the system, and is a Django project. Shove is a process that executes commands on the remote machine. The two systems communicate via RabbitMQ; Captain sends messages to Shove processes via queues, and Shove processes send the results and logs of command executions back to Captain via queues.

Server Architecture

Since we (Mozilla) built it for our own setup, Captain Shove was designed with a few things in mind:

- Servers are organized into clusters, and each cluster may host one or more projects.
- Each cluster has at least one "admin" node, which performs tasks like minifying CSS and JS or pulling translation files during a deployment. Once these tasks are done, the admin node syncs code on the other servers in the cluster that actually serve the site.
- Admin nodes perform commands for all the projects in the cluster.
- The commands we want to execute should be executed on the admin node; executing commands on each individual application server is handled by another system (in our case, commander).
- There is a single RabbitMQ cluster that most admin nodes already connect to for other reasons.

In our setup, Captain lives in its own cluster, and Shove processes are installed on any admin node that we want to run commands on. The *project setup documentation* has more info on how to register an individual project with Captain.

Captain

Captain is a Django-based site that presents an interface for sending commands to Shove and showing the results of those commands. Users log into Captain using Persona and are given permission by an administrator to run commands on certain projects registered with Captain.

Note: The command history and logs are visible to all users who can access the site; we run Captain behind a VPN so the logs aren't visible to the public.

Shove

The Shove process runs on the admin nodes for each cluster and is responsible for executing commands it receives from Captain. The Shove configuration includes a dictionary that maps project names to directories on the filesystem of the admin node. These directories are usually a checkout of the project's repository, and all commands are run with this directory as the working directory.

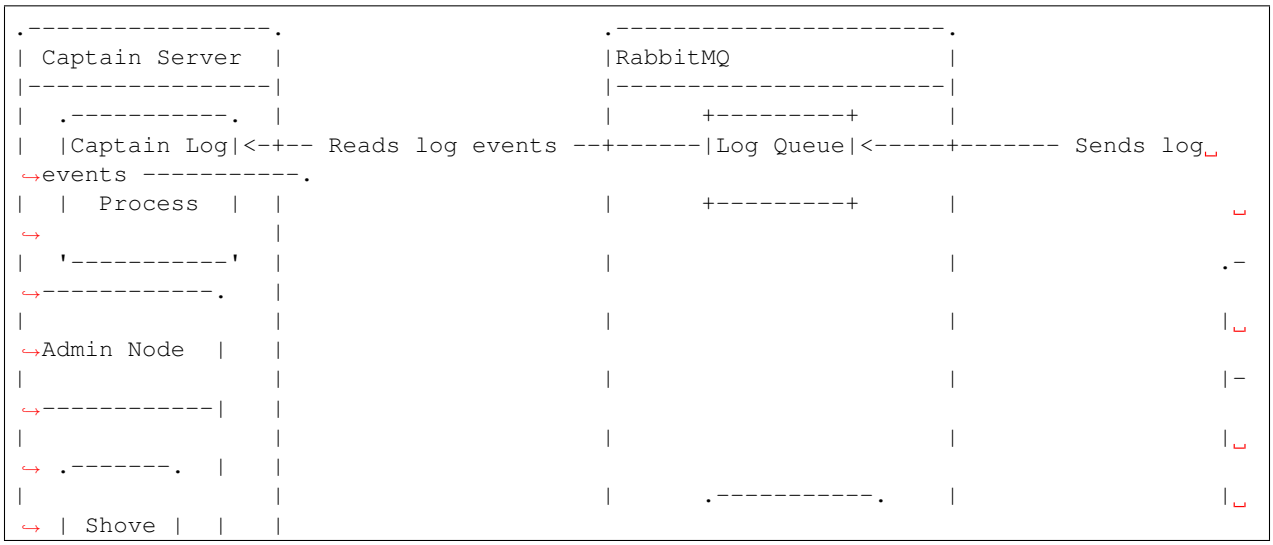
Shove will only run whitelisted commands; each project should have a whitelist stored in the `bin/commands.procfile` relative to the project's directory in the Shove configuration. A procfile maps a command name to an actual shell command. When Shove receives an order, it looks up the procfile for the requested project, and looks for a command matching the command name in the order. If it finds one, it executes that command and sends the logs back to Captain.

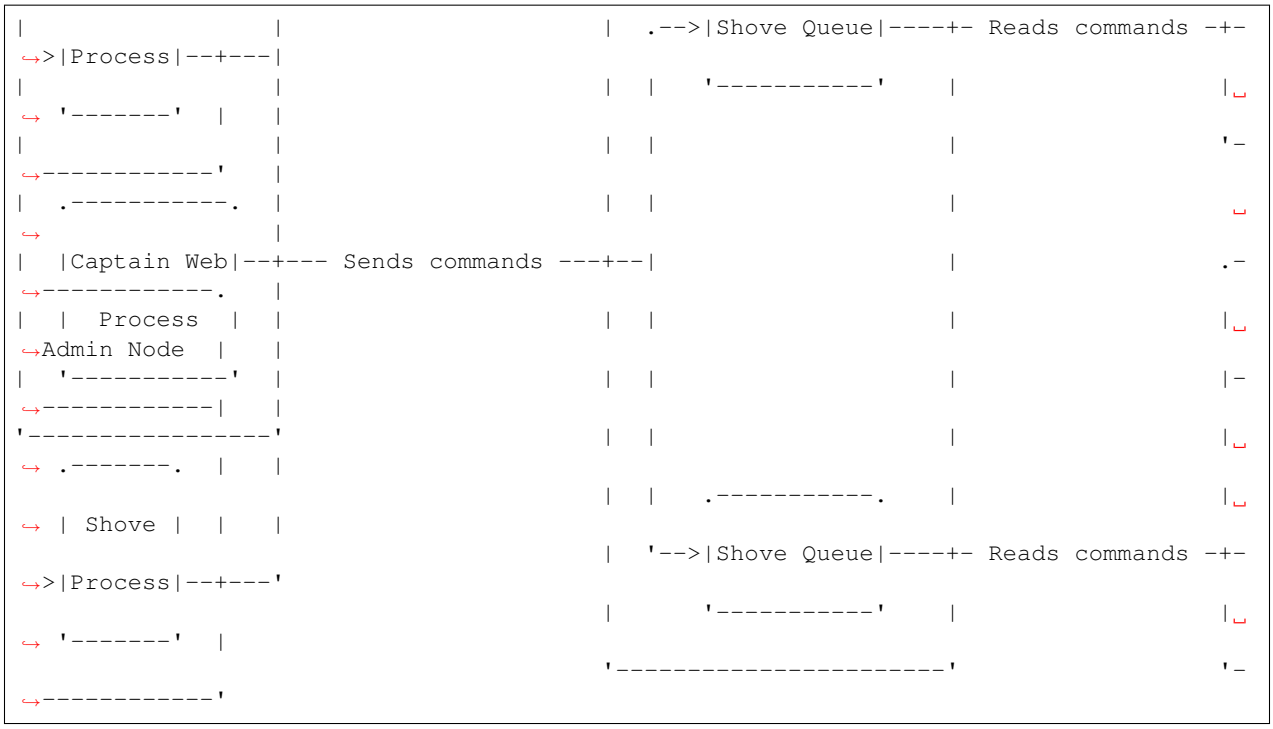
Communication Flow

Captain and Shove communicate via RabbitMQ. Each instance of Shove creates a queue that it listens to for commands. When you create a project in Captain, you give it the name of this queue, and Captain will send commands for that project to the queue.

Once Shove has executed a command, it sends the return code and logs back to a log event queue specified in the command it received. A process included in the Captain codebase listens to this queue and updates Captain's database with the results of the command when it receives these events.

The following is a diagram of the processes and queues involved with the system:





Security

There are a few features to point out when evaluating the security of Captain Shove:

- Commands are whitelisted by the procfile for each project, so only developers with commit access to a project's repository can specify a command to run.
- RabbitMQ includes access control features that allow you to restrict certain users to only be able to read or write to certain queues. For example, Shove users should only be able to read their own queues and write to the log queue, and the Captain user should only be able to read the log queue and write to the Shove queues.
- Captain uses standard Django username/password authentication for the admin interface, and Persona authentication for the user-facing side. Admins can create projects and grant permissions (using the django-guardian library) to certain users to allow them to run commands on a project.

Example Flow

The following is an example from start to finish of executing a command with Captain Shove:

1. User logs into Captain via Persona.
2. User enters a command named "pwd" into a form for the "Firefox Flicks" project and submits.
3. Captain creates a log entry in it's database for this submission.
4. Knowing that Flicks is on the "generic" cluster, Captain sends a message to the queue for the generic cluster that contains an order to run the "pwd" command on the "Firefox Flicks" project as well as the ID of the log entry it created and the name of the queue it is listening for log events on.
    - The user sees a message confirming the command has been sent and will have to revisit the page after the results are saved to be able to view the output.

5. The Shove process on the generic cluster admin node, which has been listening on the generic cluster queue, receives the message and looks up the directory for “Firefox Flicks” in it’s configuration.
6. Once it finds the directory, Shove reads in the procfile for Flicks and looks for a command named “pwd”.
7. When it finds the command, it takes the shell command listed in the procfile and spins off a subprocess to execute the command.
8. Shove waits for the command to finish and captures the output of the command, including any errors, and the return code.
9. Shove combines the output of the command, return code, and the ID of the log entry in Captain into a log event message and sends it to the logging queue specified in the command from Captain.
10. The Captain logging process, which is listening on the logging queue, receives the logging event and saves the output and return code to the log in the database specified by the log entry ID.

CHAPTER 2

Installing Captain and Shove

These instructions are for installing Captain and Shove in a production setup. If you want to work on Captain or Shove as a developer, see the *Contributing* page.

Prerequisites

- Python 2.6 or 2.7
- A database (or use Django's SQLite support)
- A running instance of RabbitMQ

Setting up Shove

Shove installs as a normal Python package. It's not on PyPI yet, but you can install it using pip:

```
pip install git+https://github.com/mozilla/shove.git#egg=shove
```

This should install the `shove` executable into your environment, which is used to start the Shove daemon.

Shove requires a settings file; an example settings file can be found in the Shove source code. The `SHOVE_SETTINGS_FILE` environment variable should contain an absolute file path to the settings file you want to use.

The settings file contains details for connecting to RabbitMQ, as well as a mapping of project IDs to directories that projects are contained in. You must edit this dictionary to include file paths to any projects that you want Shove to be able to run commands for.

Setting up Captain

Captain is a Django project. It's intended to be run as a WSGI application. The WSGI file for Captain is located at `captain/wsgi.py` under the repository root.

You can retrieve the code for Captain by cloning `https://github.com/mozilla/captain.git` using git.

Dependencies

Captain comes with almost all of its dependencies included in the `vendor` directory, and `wsgi.py` automatically alters the Python import path to include them. There are a few compiled dependencies that aren't included: They are specified in `requirements/compiled.txt` and can be installed on your target system using pip:

```
pip install -r requirements/compiled.txt
```

Note: Alternatively, you can create system packages for the compiled requirements and have them installed via a server automation framework like Puppet.

Settings

Once you've installed the dependencies, you need to create a settings file by copying `captain/settings/local.py-dist` to `captain/settings/local.py` and editing the contents:

```
cp captain/settings/local.py-dist captain/settings/local.py
vi captain/settings/local.py
```

The comments in the file and the Django settings documentation will help explain how to configure the settings for your setup.

Database

Next, you must initialize the database using the `syncdb` and `migrate` commands:

```
python manage.py syncdb
python manage.py migrate
```

Static Content

There are two directories that need to be served up by a static webserver alongside Captain: the `static` directory and the `media` directory. `static` contains all the static CSS, JavaScript, and images for the site, while `media` contains the raw logs sent back from Shove.

The filesystem paths for these directories are configured by the `MEDIA_ROOT` and `STATIC_ROOT` settings in the settings file, and default to being located at the root of the repository. The public-facing URLs for them are controlled by the `MEDIA_URL` and `STATIC_URL` settings, and default to `/static` and `/media`.

Once you've configured these settings (if necessary), you must populate the `static` directory by running the following command:

```
python manage.py collectstatic
```

This should fill `static` with files. Then you must use the web server of your choice to serve these files alongside the rest of the Captain interface.

Finished!

After that, you should be ready to run the site via whatever WSGI-compliant web server you prefer.

Log Event Listener

Captain includes a command that listens for log events from Shove. After configuring Captain using the steps above, you should be able to start the process with this command:

```
python manage.py monitor_shove_logs
```

Note: You should probably use a process control system like supervisord to manage this process as well as the Shove process.

CHAPTER 3

Deploying a Project with Captain

So you want to execute commands for your project using Captain? Great! Assuming there's an instance of Captain running that you want to use, here's how you add your project to it:

1. Add your commands to your project.

Captain works on the assumption that commands that projects want to run (such as deploying, downloading new translations, etc.) are specified in the code for the project itself in a file called `bin/commands.procfile`.

The file is in the same format as a Heroku Procfile, which specifies one command per line in the following format:

```
mycommand: python myscript.py
anothercommand: python manage.py some_management_command
git_yolo: git commit -am "DEAL WITH IT" && git push -f origin master
```

Warning: Any syntax errors in the format will cause the command in question to not be available.

Note: Commands from the procfile are executed in the environment that the Shove process is running in. The current working directory for the command is set to the root of your project as specified in the Shove configuration.

2. Setup and configure Shove to recognize your project.

If you haven't already, set up an instance of Shove on the machine you want to run your commands on using the directions in the *Installation documentation*.

In the Shove configuration file, add an entry to the `PROJECTS` setting with a name for your project and the path to the directory where your project's code is stored:

```
PROJECTS = {
    'myproject': '/data/www/myproject-web'
}
```

3. Create a project entry in Captain and grant permissions.

Next, a user with admin access to Captain should create a new Project entry. The project will need the queue name for the Shove instance that will be running the command (found in the Shove configuration) and the project name used as the key in the `PROJECTS` setting in Shove.

It's a good practice to also create a user group using the admin interface and grant permission to run commands on the project to that group. That way, you can just add users to the group instead of granting permission to each individual user.

If you set up a group, you'll need to add any users that want to run commands to that group. Otherwise, grant permission directly to the users that need it. In either case, the link for managing object permissions can be found on the detail page for the project in the admin interface.

4. Test running a command on the project.

Lastly, you'll need to test running a command on the project by sending a command via Captain and inspecting the output when the result returns. If no result is returned, this may indicate a problem with how Shove was configured, and you should check the Shove output for any errors or warnings.

It may be useful to add a test command like `pwd` to the procfile to test for errors in Shove as opposed to errors in the command itself.

Controlling Permissions

Captain controls who can run commands on projects using project-level permissions. The interface for these permissions is a link titled "Object Permissions" on the detail page of a project in the admin interface.

Permissions can be assigned to individual users, or groups. It is recommended that you use groups, as it's easier to add a user to a group than to give permission to a user. Permissions can also be revoked, or you can remove a user from a group if you're using groups to manage permissions.

CHAPTER 4

Contributing

Developer Setup

Prerequisites:

- Python 2.6 or 2.7
- pip
- virtualenv
- RabbitMQ

Note: While it is technically possible to work on Captain or Shove without RabbitMQ installed for very small changes, it is highly recommended to install it anyway.

Once you have the prerequisites installed, you must set up the Shove daemon:

```
# Clone the repository
git clone https://github.com/mozilla/shove.git
cd shove

# Create a virtualenv and activate it
# You should consider using virtualenvwrapper instead: http://virtualenvwrapper.
↪readthedocs.org/
virtualenv venv
source venv/bin/activate

# Install shove in development mode
python setup.py develop

# Copy the settings file
cp settings.py-dist settings.py
# You must edit settings.py with the settings for your setup! It is commented with␣
↪info on what
```

```
# you need to change.

# Start the shove daemon.
shove
```

Once Shove is running, you must set up the Captain frontend:

```
# Clone the repository
git clone https://github.com/mozilla/captain.git
cd shove

# Create a virtualenv and activate it
# You should consider using virtualenvwrapper instead: http://virtualenvwrapper.
↪readthedocs.org/
virtualenv venv
source venv/bin/activate

# Install libraries needed for development
pip install -r requirements/dev.txt

# Copy the settings file
cp captain/settings/local.py-dist captain/settings/local.py
# You must edit local.py with the settings for your setup! It is commented with info␣
↪on what
# you need to change.

# Initialize the database
python manage.py sync
python manage.py migrate

# Start the development server
python manage.py runserver
```

You should now have both Captain and Shove running and connected to RabbitMQ.

The last step is to start the Captain logging event process. The process listens for messages on the logging queue and saves them to the database to update Captain with the results of a command. To run it, run the following in a new terminal:

```
# Enter the captain directory.
cd captain

# Activate the virtualenv.
source venv/bin/activate

# Run the logging daemon.
python manage.py monitor_shove_logs
```

Running the Tests

```
# Enter the captain directory.
cd captain

# Activate the virtualenv.
source venv/bin/activate
```

```
# Run the tests.
python manage.py test
```

Changing the Database

Captain uses South to generate and run migrations for the database. The South documentation has more information on how to generate and run migrations when the models change.

Make sure to check for new migrations whenever you pull new code!

Third-party Libraries

Third-party libraries for Captain are listed in pip requirements files in the `requirements` directory. There are three files:

- `prod.txt`: Non-compiled libraries required for production.
- `compiled.txt`: Compiled libraries required for production.
- `dev.txt`: Libraries that are required for development (e.g. for running the tests). This also pulls in the requirements from `prod.txt` and `compiled.txt`.

In addition, the libraries from `prod.txt` are also included in a directory called `vendor`. This is used to import the libraries in a production environment where there isn't a PyPI mirror to install the libraries from.

If you add a new third-party library to Captain, make sure to add it to the appropriate requirements file. If you add to or update `prod.txt`, you'll also need to update vendor. This can be done with using pip like so:

```
# Executed from the repository root.
pip install -I --install-option="--home=`pwd`/vendor" library-name==1.2
```

Note: Make sure that any requirements in `prod.txt` are pinned to a specific version or commit.

Where to Find Us

We hang out on IRC on irc.mozilla.org in `#capshove`.

Additionally, we'll respond to issues in both the captain and shove projects.

CHAPTER 5

Mozilla Public License Version 2.0

1. Definitions

1.1. "Contributor" means each individual or legal entity that creates, contributes to the creation of, or owns Covered Software.

1.2. "Contributor Version" means the combination of the Contributions of others (if any) used by a Contributor and that particular Contributor's Contribution.

1.3. "Contribution" means Covered Software of a particular Contributor.

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.

1.5. "Incompatible With Secondary Licenses" means

1. that the initial Contributor has attached the notice described in Exhibit B to the Covered Software; or
2. 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.

1.6. "Executable Form" means any form of the work other than Source Code Form.

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.

1.8. "License" means this document.

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.

1.10. "Modifications" means any of the following:

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

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.

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.

1.13. "Source Code Form" means the form of the work preferred for making modifications.

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.

2. License Grants and Conditions

2.1. Grants

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

1. 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
2. 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.

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.

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:

1. for any code that a Contributor has removed from Covered Software; or
2. 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
3. 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).

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).

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.

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.

2.7. Conditions

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

3. Responsibilities

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.

3.2. Distribution of Executable Form

If You distribute Covered Software in Executable Form then:

1. 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
2. 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.

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).

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.

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.

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.

5. Termination

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.

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.

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.

- ———————————- *
- –
- 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. *
- –

- ———————————— *
- –
- 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. *
- –

---

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.

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.

10. Versions of the License

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.

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.

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).

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.

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.

You may add additional accurate notices of copyright ownership.

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 6

Indices and tables

- genindex
- modindex
- search