CommunityPatch.com Documentation

Release 1.0

Bryson Tyrrell

Mar 07, 2020

Contributors

CommunityPatch.com is a free, open-source patch source for Jamf Pro administrators to publish patch definitions they maintain for the broader Jamf community to subscribe to.

Note: This documentation covers CommunityPatch `Beta 2`.

CHAPTER 1

Subscribing to Contributors

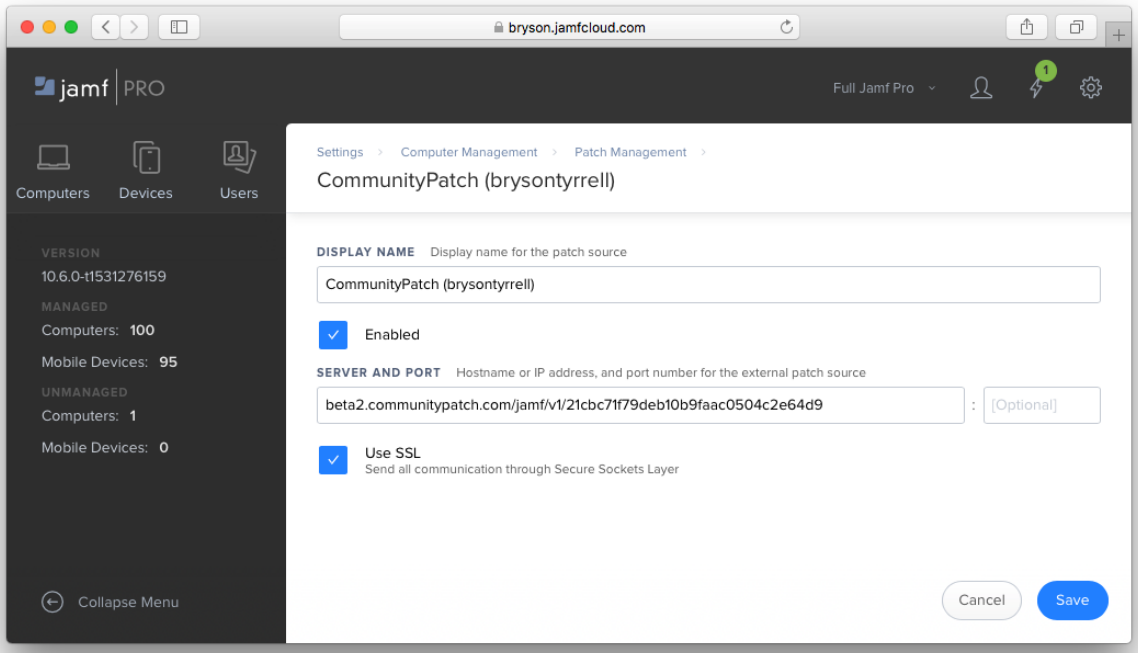
Each Contributor to CommunityPatch maintains their own set of software titles that any Jamf administrator can subscribe to. If you wish to use the definitions provided by a subscriber, you may do so using their unique ID.

Note: Each Contributor on this service acts as an independent external patch source. Refer to Jamf's documentation to learn more about external patch sources.

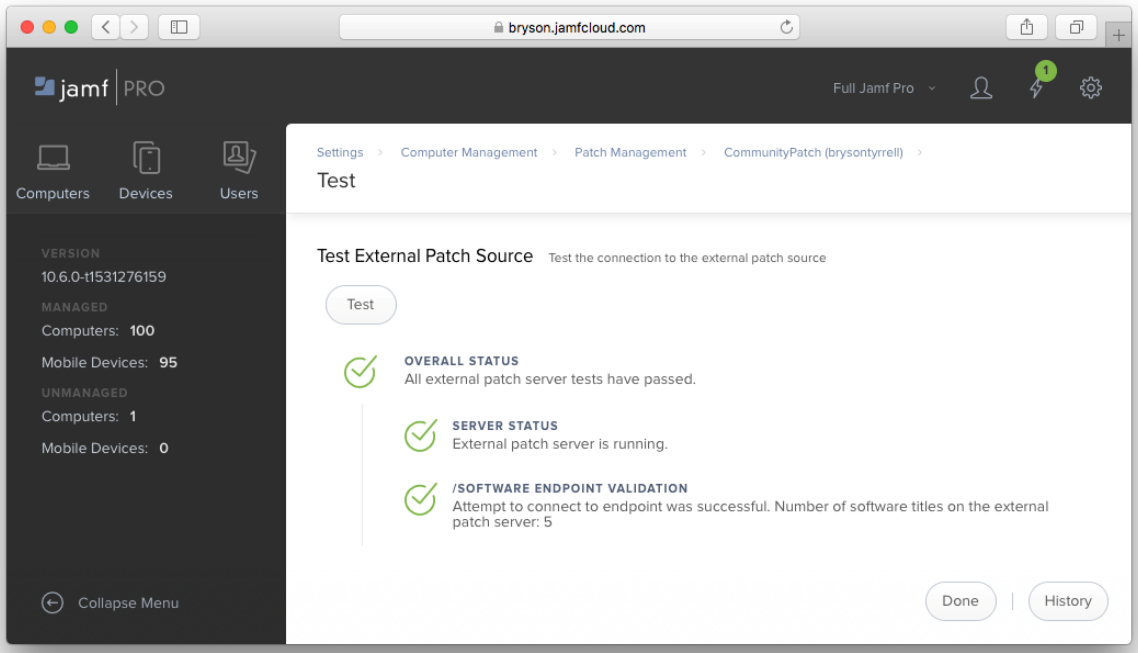
In Jamf Pro, navigate to **Settings > Computer Management > Patch Management**. Click the **+ New** button to add a new external patch source.

Provide a `Display Name` and enter the address for the Contributor's source into the `Server and Port` field as follows:

```
beta2.communitypatch.com/jamf/v1/{CONTRIBUTOR_ID}
```



After saving, use the **Test** option to verify Jamf Pro can successfully connect to the



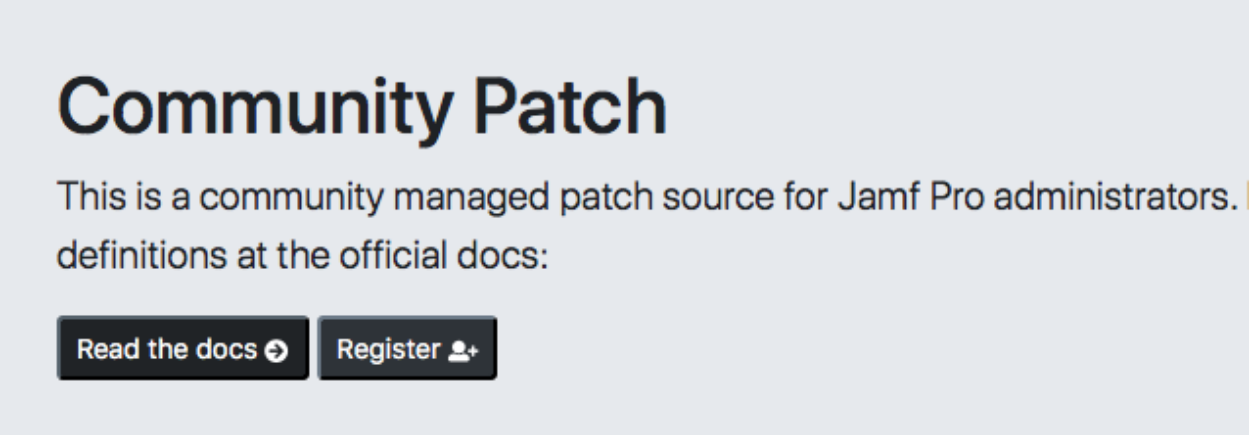
CHAPTER 2

Managing Software Titles as a Contributor

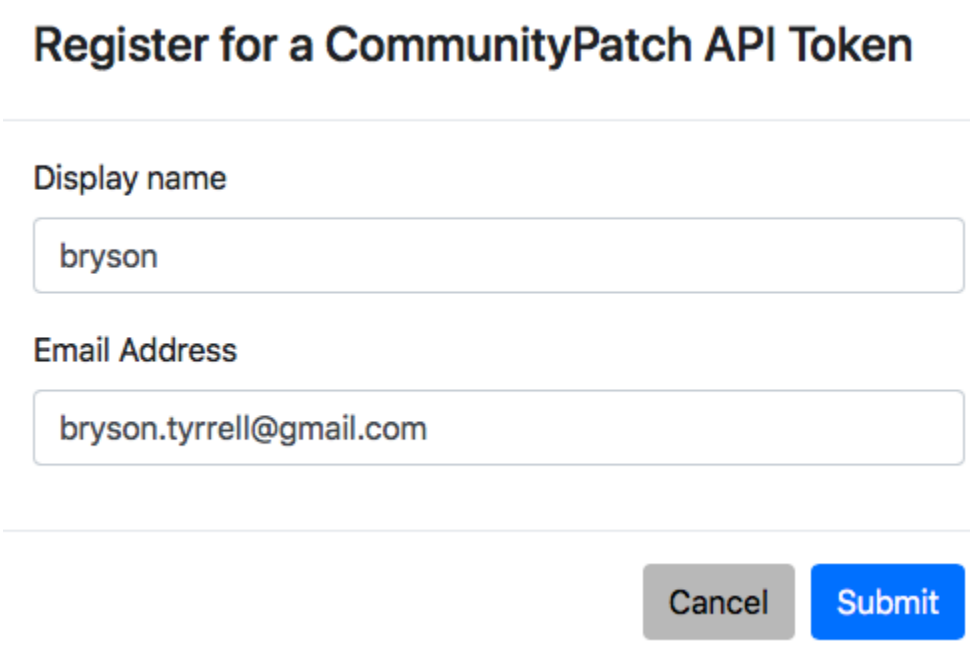
Become a Contributor and manage patch definitions for the community by registering.

2.1 Registration

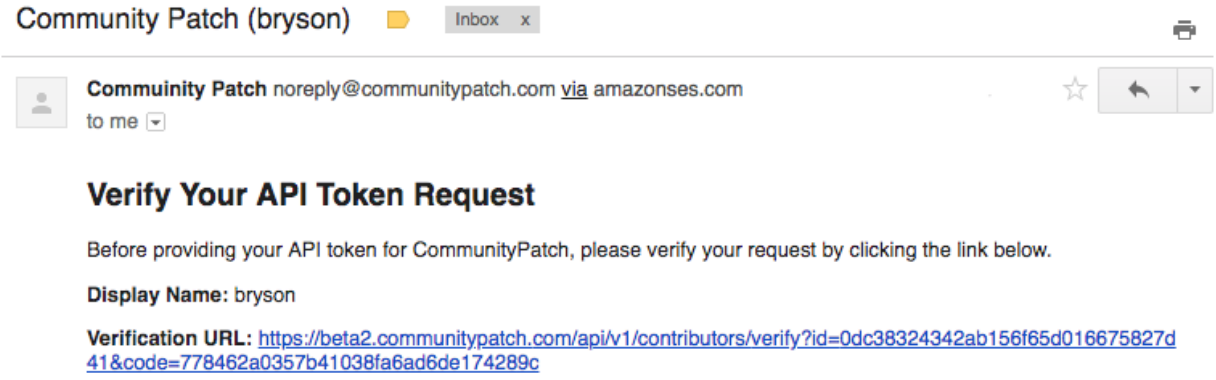
- To use CommunityPatch you must register for an API token. You may start the registration process from the home page by clicking the `Register` button.



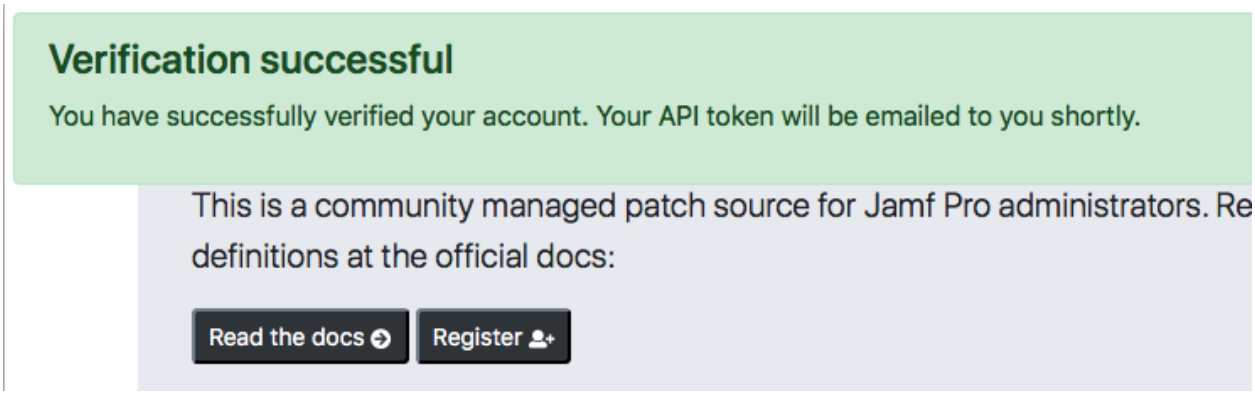
- Enter your desired `Display name` and a valid `Email address`.



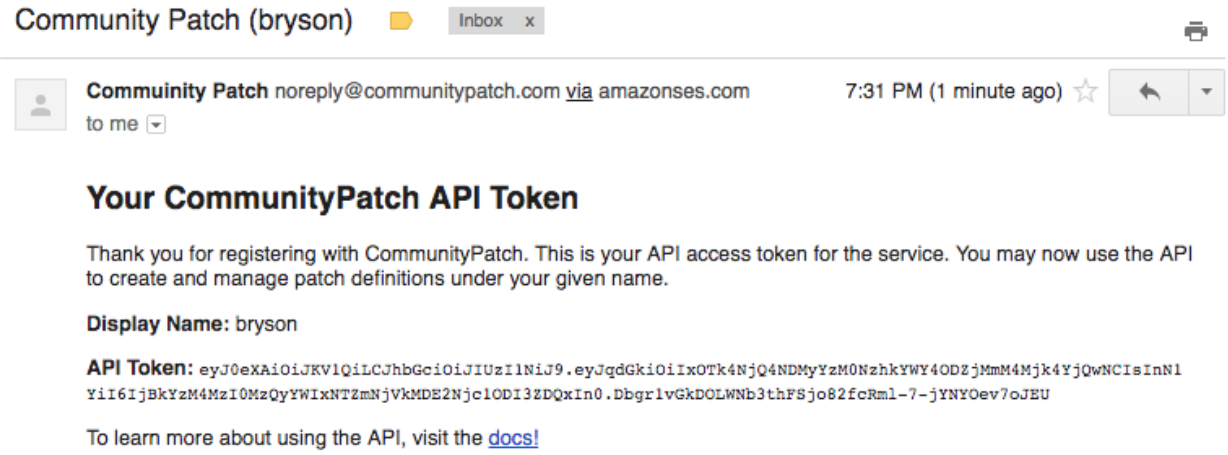
- After clicking `Submit`, CommunityPatch will send a verification email to the provided address.



- Click the link provided to be taken back to the CommunityPatch home page where a banner will display showing you have successfully verified your account.



- Your API token will be emailed to you after seeing this message.



- This token can be used to manage patch definitions on CommunityPatch. Check out the API documentation to learn more on how to create, update, and delete patch definitions.

Note: Your email address is saved using encryption. Your email address is not used for any purposes other than system generated notifications. This includes the registration process, resetting your API token, and error notifications when using repository syncing features.

2.2 API

2.2.1 GET /api/v1/contributors

Return a list of contributors sorted by their `title_count`.

Note: This endpoint is used for rendering the Web UI.

```
GET /api/v1/contributors
```

2.2.2 Response

On success you will receive a message stating the new title has been created.

```
200 OK
Content-Type: application/json
```

```
[
    {
        "id": "",
        "display_name": "",
        "title_count": 0,
        "urn": "jamf/v1/{ID}/software",
```

(continues on next page)

(continued from previous page)

```
        "url": "https://{DOMAIN}/jamf/v1/{ID}/software"
    }
]
```

Use your API token to manage patch definitions for your and the community's use.

2.3 Add a New Software Title Definition

Note: If you are creating a definition file for the first time, you can use the Patch-Starter-Script available on GitHub. The examples provided below will reference this script.

2.3.1 POST /api/v1/titles

Create a new patch definition. The JSON payload should contain the data for the full patch definition.

```
POST /api/v1/titles
Content-Type: application/json
```

```
{
    "id": "",
    "name": "",
    "publisher": "",
    "appName": "",
    "bundleId": "",
    "requirements": [],
    "patches": [],
    "extensionAttributes": []
}
```

2.3.2 Response

On success you will receive a message stating the new title has been created.

```
201 Created
Content-Type: application/json
```

```
{
    "message": "Title '{ID}' created"
}
```

2.3.3 Examples

An example using `curl` and `Patch-Starter-Script`:

```
curl https://beta2.communitypatch.com/api/v1/titles \
    -X POST \
    -d "$(python patchstarter.py '/Applications/{APP}' -p '{PUBLISHER}')" \
    -H 'Content-Type: application/json' \
    -H 'Authorization: Bearer {API-KEY}'
```

2.4 Add a new Version to a Software Title

2.4.1 POST /api/v1/titles/{ID}/versions

Update a software title's definition with a new version. The JSON payload should only contain the data for the version.

```
POST /api/v1/titles/{ID}/version
Content-Type: application/json
```

```
{
    "version": "",
    "releaseDate": "",
     "standalone": true,
    "minimumOperatingSystem": "",
    "reboot": false,
    "killApps": [],
    "components": [],
    "capabilities": [],
    "dependencies": []
}
```

2.4.2 Response

On success you will receive a message stating the new version has been added to the title.

```
201 Created
Content-Type: application/json
```

```
{
    "message": "Version '{VERSION}' added to title '{ID}'"
}
```

2.4.3 Examples

An example using `curl` and `Patch-Starter-Script`:

```
curl https://beta2.communitypatch.com/api/v1/titles/{ID}/versions \
    -X POST \
    -d "$(python patchstarter.py '/Applications/{APP}' --patch-only)" \
    -H 'Content-Type: application/json' \
    -H 'Authorization: Bearer {API-KEY}'
```

The default behavior for this request is to add the new version as the `latest` version of this definition.

2.4.4 Options

Note: If you make a request using the `insert_after` or `insert_before` options and the placement of the new version is not at the `latest` position, the definition's `currentVersion` will not be updated, but the `lastModified` timestamp will be.

insert_after

To specify the position of the new version in the `patches` array of the definition, use the `insert_after={VERSION}` or `insert_before={VERSION}` parameters where `VERSION` is an existing version in the definition.

```
POST /api/v1/titles/{ID}/version?insert_after={VERSION}
Content-Type: application/json
```

```
curl https://beta2.communitypatch.com/api/v1/titles/{ID}/versions?insert_after=
↪{VERSION} \
    -X POST \
    -d "$(python patchstarter.py '/Applications/{APP}' --patch-only)" \
    -H 'Content-Type: application/json' \
    -H 'Authorization: Bearer {API-KEY}'
```

insert_before

```
POST /api/v1/titles/{ID}/version?insert_before={VERSION}
Content-Type: application/json
```

```
curl https://beta2.communitypatch.com/api/v1/titles/{ID}/versions?insert_before=
↪{VERSION} \
    -X POST \
    -d "$(python patchstarter.py '/Applications/{APP}' --patch-only)" \
    -H 'Content-Type: application/json' \
    -H 'Authorization: Bearer {API-KEY}'
```

2.5 Delete a Version from a Software Title

2.5.1 DELETE /api/v1/titles/{ID}/versions/{VERSION}

Delete a version from a software title's definition. The advertised `currentVersion` will be updated to reflect whatever the most 'recent" version is after the operation.

Note: You must have at least one (1) version defined for a software title. The API will respond with a 400 error if you attempt to delete the last remaining version.

Warning: Use caution and judgement when deleting a version from a software title. You may negatively impact other admins using the definition for their own patch reports and policies!

```
DELETE /api/v1/titles/{ID}/versions/{VERSION}
```

2.5.2 Response

On success you will receive a message stating the version has been removed from the title.

```
200 OK
Content-Type: application/json
```

```
{
    "message": "Version '{VERSION}' deleted from title"
}
```

2.5.3 Examples

An example using `curl`:

```
curl https://beta2.communitypatch.com/api/v1/titles/{ID}/versions/{VERSION} \
    -X DELETE \
    -H 'Authorization: Bearer {API-KEY}'
```

2.6 Delete a Software Title

2.6.1 DELETE /api/v1/titles/{ID}

Delete a software title from your submitted titles.

> **Warning:** This does not remove the patch definition from any Jamf Pro instance that has subscribed to it. Those objects will continue to exist until the admin deletes them, but will no longer be updated.

```
DELETE /api/v1/titles/{ID}
```

2.6.2 Response

On success you will recieve a message stating the title has been deleted.

```
200 OK
Content-Type: application/json
```

```
{
    "message": "Title '{ID}' has been deleted"
}
```

2.6.3 Examples

An example using `curl`:

```
curl https://beta2.communitypatch.com/api/v1/titles/{ID} \
    -X DELETE \
    -H 'Authorization: Bearer {API-KEY}'
```

Join the discussion on the MacAdmins Slack.