Thunderbird

Christian Külker

2023-07-21

Contents

1 Introduction

While Thunderbird is primarily recognized as an email client, it also features a unique calendar mode. Distinctively, this calendar mode is designed as a web page and is seamlessly delivered through Chrome. Due to this tight integration, many users might not even realize that Thunderbird functions as a local web application. This document is intended to offer useful tips for maximizing the use and configuration of the Thunderbird calendar.

2 Change the Date Format

By default, Thunderbird's date formatting adheres to the locale setting of the operating system. For instance, if you're using the locale `en_us.UTF8`, you'll encounter dates in the format "2023/07/30". But if you're utilizing the German locale, it'll display as "30.07.2023". Now, what if you wish to employ the ISO format while retaining the German language setting? Contrary to the Evolution email client, Thunderbird's standard configuration dialog doesn't readily support this preference.

Solution 1: Opt for a locale that supports your desired short date format.

Solution 2: This is specifically for users with Thunderbird version 91 or later:

1. Open the `Config Editor` by navigating to Preferences > General. Scroll down to the bottom of the page.
2. Adjust the setting `intl.regional_prefs.use_os_locales` to `false`.

```
intl.regional_prefs.use_os_locales = false

```

3. Next, alter `intl.date_time.pattern_override.date_short` to the string value "`yyyy-MM-dd`".

```
intl.date_time.pattern_override.date_short = yyyy-MM-dd

```

For a comprehensive guide on this topic, please refer to Mozilla's official guide.

3 Change the Calendar Font Size

Although users can adjust the font and its size in the 'Preferences General' setting of the Thunderbird application, it doesn't affect the calendar. For instance, on Debian 12 Bookworm under MATE, the default font is Cantarell at a size of 17, relatively large. To display four calendar entries spaced 15 minutes apart within a single hour, users either need to significantly zoom in or the display splits into two columns. Notably, unlike in Evolution, setting the time division to 15 minutes doesn't alter this behavior.

As there's no direct configuration option to adjust the calendar font or size, users need to employ a workaround by modifying the application's underlying CSS.

To identify the current settings and values, initiate Thunderbird with web tools using: `thunderbird --devtools`

1. Enable a Custom CSS File

   Navigate to the `Config Editor` and configure `toolkit.legacyUserProfileCustomizations.sty` to `true`.

   ```
   toolkit.legacyUserProfileCustomizations.stylesheets = true

   ```

2. Set Up an Empty CSS File

   Create an empty file named `userChrome.css` inside the `chrome` directory.

   ```
   cd ~/.thunderbird/*.default-default
   mkdir chrome
   cd chrome
   touch useChrome.css
   ```

3. Test the CSS Integration

   To verify Thunderbird's use of the file, add a test CSS rule. For instance, adjusting the size of the calendar day header:

   ```
    .day-column-container {
       font-size: 12px;
    }
   ```

   It's advisable to utilize the development tools and the inspector to check **computed** values before and after modifications to discern any changes. However, be mindful that the application's HTML and CSS might evolve over time.

4. Address the Font Size Issue

   To remedy the specific font size concern, reduce the font size from 17.333px to 16px and eliminate some padding. The original `.day-column-container` setting can be discarded.

   ```
   .calendar-item-flex {
        padding-top: 0px !important;
        padding-bottom: 0px !important;
        row-gap: 2px !important;
    }
    .event-name-label {
   ```

```
    font-weight: 400 !important;
    font-size: 16px;
}
```

5. Restart Required

   Remember, any modifications necessitate a Thunderbird restart for changes to take effect.

For additional guidance, refer to:

- https://www.userchrome.org/how-create-userchrome-css.html

4 Add CalDAV Calendars via Ansible

For many users, Thunderbird's graphical interface is the go-to method for adding calendars. While efficient, this method doesn't lend itself to replicating configurations across multiple machines, especially if you need consistency in test environments. The solution? Automate this process using Ansible.

However, using Ansible (or any other configuration management tool) for Thunderbird introduces a challenge: Thunderbird uses a hashed location to store its configuration. For instance, the main configuration resides in `~/.thunderbird`. Within this directory, the file `profiles.ini` contains a section `Profile0` with a `Path` key pointing to the real configuration. For clarification:

```
...
[Install7DFCE75BB80C198C]
Default=oGao5Tha.default-default
Locked=1

[Profile0]
Name=default-default
IsRelative=1
Path=oGao5Tha.default-default
...
```

On Debian, the `Profile1` profile is recognized as the `default` profile. But it's the `default-default` profile that retains all the configuration.

To determine the current configuration path, you can use the script `ansible-thunderbird-cfg-path-get`:

```
#!/usr/bin/bash
export DIR=/home/USER/.thunderbird
```

```
cat $DIR/p*.ini|grep 'Pa.*-default'|sed -e "s%Path=%$DIR/%"
```

When executed, the output might resemble:

```
/home/USER/.thunderbird/oGao5Tha.default-default
```

4.1 How the Ansible Playbook Works

The provided Ansible playbook performs the following functions:

1. **Installation**: It ensures Thunderbird is installed and up-to-date.
2. **Configuration Retrieval**: It fetches the current Thunderbird configuration path.
3. **Calendar Configuration**: It creates individual JavaScript files for each CalDAV calendar from a predefined template.
4. **File Assembly**: It compiles these individual files into a consolidated `user.js` configuration file.

For this playbook to function correctly:

- Ensure you have the correct ID, in this case 'id1' from `prefs.js`.
- Modify the `USER` placeholder with the appropriate user name.
- Adjust the `ansible_cfg_path` to the location where your playbooks, templates, and scripts are stored. The key directories are: `bin` (scripts), `pb` (playbooks), and `tpl` (templates).
- Use this playbook cautiously if you already possess a `user.js` as it might overwrite your existing configurations.

Below is the enhanced playbook:

```
hosts: localhost
  gather_facts: no
  vars:
    ns: thunderbird
    ansible_cfg_path: /srv/ansible
    packages:
      - thunderbird
    caldav_calendars:
      - uuid: work
        color: "#1c71d8"
        refresh: 5
        name: Work
        uri: https://example.org/rad/USER/work
        username: USER
        imip:
          identity: id1
  tasks:
    - name: "{{ ns }}: Install and update Thunderbird"
      package:
        name: "{{ packages }}"
        state: latest
    - name: "{{ ns }}: Create a temporary directory for calendar files"
      tempfile:
        state: directory
      register: tempdir
    - name: "{{ ns }}: Retrieve the Thunderbird configuration path"
      command: "{{ ansible_cfg_path
↪   }}/bin/ansible-thunderbird-cfg-path-get"
      register: config_path
      changed_when: False
    - name: "{{ ns }}: Display the configuration path (optional)"
      debug:
        var: config_path.stdout
    - name: "{{ ns }}: Generate calendar configuration files"
      template:
        src: "{{ ansible_cfg_path
↪   }}/tpl/HOME/thunderbird/caldav_calendar.js"
        dest: "{{ tempdir.path }}/caldav_calendar_{{ calendar.uuid }}.js"
        mode: 'a'
      loop: "{{ caldav_calendars }}"
      loop_control:
        loop_var: calendar
      when: config_path.stdout is defined and config_path.stdout != ''
    - name: "{{ ns }}: Combine individual files into user.js"
      assemble:
        src: "{{ tempdir.path }}/"
        dest: "{{ config_path.stdout }}/user.js"
      when: config_path.stdout is defined and config_path.stdout != ''
```

The is the template `tpl/HOME/thunderbird/caldav_calendar.js`:

```
// {{ calendar.name }}
user_pref("calendar.registry.{{ calendar.uuid }}.cache.enabled", true);
user_pref("calendar.registry.{{ calendar.uuid }}.calendar-main-default",
↪   true);
```

```
user_pref("calendar.registry.{{ calendar.uuid }}.calendar-main-in-composite", true);
user_pref("calendar.registry.{{ calendar.uuid }}.color", "{{ calendar.color }}");
user_pref("calendar.registry.{{ calendar.uuid }}.disabled", false);
user_pref("calendar.registry.{{ calendar.uuid }}.imip.identity.key", "{{ calendar.imip.identity }}");
user_pref("calendar.registry.{{ calendar.uuid }}.name", "{{ calendar.name }}");
user_pref("calendar.registry.{{ calendar.uuid }}.notifications.times", "");
user_pref("calendar.registry.{{ calendar.uuid }}.readOnly", false);
user_pref("calendar.registry.{{ calendar.uuid }}.refreshInterval", "{{ calendar.refresh }}");
user_pref("calendar.registry.{{ calendar.uuid }}.suppressAlarms", false);
user_pref("calendar.registry.{{ calendar.uuid }}.type", "caldav");
user_pref("calendar.registry.{{ calendar.uuid }}.uri", "{{ calendar.uri }}");
user_pref("calendar.registry.{{ calendar.uuid }}.username", "{{ calendar.username }}");
```

By following this playbook, you can seamlessly integrate multiple CalDAV calendars into Thunderbird across various machines for one user.

5 Add ICS Calendars via Ansible

Adding ICS calendars to Thunderbird through Ansible closely mirrors the process for adding CalDAV calendars, albeit with some variations in parameters. This section builds upon the Ansible playbook discussed in the previous section. If you haven't reviewed it yet, it's recommended to do so before proceeding.

The primary distinctions between ICS and CalDAV calendars are as follows:

```
ICS calendars are read-only. (In this example)
They don't have an associated identity. (In this example)
They come without alarms. (In this example)
They possess the distinct type 'ics'
```

For this configuration, we employ a template named `ics_calendar.js`.

Below is the comprehensive playbook:

```yaml
# pb/thunderbird.yaml
---
- hosts: localhost
  gather_facts: no
  vars:
    ns: thunderbird
    ansible_cfg_path: /srv/ansible
    packages:
      - thunderbird
    caldav_calendars:
      - uuid: 03-work
        color: "#1c71d8"
        refresh: 5
        name: Work
        uri: https://example.org/rad/USER/work
        username: USER
        imip:
          identity: id1
    ics_calendars:
      - uuid: 10-german-holidays
        color: "#f5c211"
        name: Feiertage
        uri: URL.ics
  tasks:
    - name: "{{ ns }}: Install and update packages"
      package:
        name: "{{ packages }}"
        state: latest
    - name: "{{ ns }}: Get Thunderbird configuration path"
      command: "{{ ansible_cfg_path
↪   }}/bin/ansible-thunderbird-cfg-path-get"
      register: config_path
      changed_when: False
    - name: "{{ ns }}: Create temporary directory for calendar files"
      file:
        path: "{{ config_path.stdout }}/ansible/calendars"
        state: directory
        owner: USER
        group: USER
        mode: '0750'
        recurse: yes
      register: tempdir
    - name: "{{ ns }}: Show the configuration path (debugging, can be
↪   removed)"
      debug:
        var: config_path.stdout
    - name: "{{ ns }}: Generate individual CalDAV calendar files"
      template:
        src: "{{ ansible_cfg_path
↪   }}/tpl/HOME/thunderbird/caldav_calendar.js"
        dest: "{{ tempdir.path }}/caldav_calendar_{{ calendar.uuid }}.js"
      loop: "{{ caldav_calendars }}"
      loop_control:
        loop_var: calendar
      when: config_path.stdout is defined and config_path.stdout != ''
    - name: "{{ ns }}: Generate individual ICS calendar files"
      template:
        src: "{{ ansible_cfg_path }}/tpl/HOME/thunderbird/ics_calendar.js"
        dest: "{{ tempdir.path }}/ics_calendar_{{ calendar.uuid }}.js"
      loop: "{{ ics_calendars }}"
      loop_control:
        loop_var: calendar
      when: config_path.stdout is defined and config_path.stdout != ''
    - name: "{{ ns }}: Assemble user.js from individual calendar files"
      assemble:
        src: "{{ tempdir.path }}/"
        dest: "{{ config_path.stdout }}/user.js"
      when: config_path.stdout is defined and config_path.stdout != ''
```

The is the template `tpl/HOME/thunderbird/ics_calendar.js`:

```
// {{ calendar.name }}
user_pref("calendar.registry.{{ calendar.uuid }}.cache.enabled", true);
user_pref("calendar.registry.{{ calendar.uuid
↪   }}.calendar-main-in-composite", true);
user_pref("calendar.registry.{{ calendar.uuid }}.color", "{{
↪   calendar.color }}");
user_pref("calendar.registry.{{ calendar.uuid }}.disabled", false);
user_pref("calendar.registry.{{ calendar.uuid }}.imip.identity.key", "");
user_pref("calendar.registry.{{ calendar.uuid }}.name", "{{ calendar.name
↪   }}");
user_pref("calendar.registry.{{ calendar.uuid }}.notifications.times",
↪   "");
user_pref("calendar.registry.{{ calendar.uuid }}.readOnly", true);
user_pref("calendar.registry.{{ calendar.uuid }}.refreshInterval", "600");
user_pref("calendar.registry.{{ calendar.uuid }}.suppressAlarms", true);
user_pref("calendar.registry.{{ calendar.uuid }}.type", "ics");
user_pref("calendar.registry.{{ calendar.uuid }}.uri", "{{ calendar.uri
↪   }}");
```

Ensure that you update placeholders such as USER, URL accordingly. If you alter the UUID, also remember to delete files from the cache directory located at `{{ config_path.stdout }}/ansible/calendars`.

A notable modification from the previous section's playbook is the replacement of `tempdir` with a manually selected cache directory. This adjustment was made because with multiple Ansible runs would label the result as 'ok' instead of 'changed'. The latter would have been the case with a `tempdir` that is constantly changing.

By following this playbook, you can seamlessly integrate multiple CalDAV and ICS calendars into Thunderbird across various machines for one user.

6 Change the Calendar Font Size with Ansible

To modify the calendar font size in Thunderbird using Ansible, follow the steps outlined below:

1. With the Ansible playbook from the previous section, Ansible first copies the `userChrome.css` file to the chrome subdirectory.
2. Next, `toolkit.js` is added to the ansible directory.
3. Subsequently, this file, along with others, is appended to `user.js`.

Here's a snippet illustrating the process:

```
---
- hosts: localhost
  gather_facts: no
  vars:
    ns: thunderbird
    user_cal: 'c'
    user_sys: 'c'
    ansible_cfg_path: /srv/g/g.c8i.org/an
    thunderbird_src: "HOME/thunderbird/UUID.default-default"
    fls_src: "{{ ansible_cfg_path }}/fls/{{ thunderbird_src }}"
  tasks:
    - name: "{{ ns }}: Get Thunderbird configuration path"
      command: "{{ ansible_cfg_path
↪   }}/bin/ansible-thunderbird-cfg-path-get"
```

```
    register: config_path
    changed_when: False
  - name: "{{ ns }}: Ensure chrome directory exists in configuration"
    file:
      path: "{{ config_path.stdout }}/chrome"
      state: directory
      owner: "{{ user_sys }}"
      group: "{{ user_sys }}"
      mode: '0755'
  - name: "{{ ns }}: Copy userChrome.css to thunderbird configuration"
    copy:
      src: "{{ fls_src }}/chrome/userChrome.css"
      dest: "{{ config_path.stdout }}/chrome/userChrome.css"
      owner: "{{ user_sys }}"
      group: "{{ user_sys }}"
      mode: '0644'
  - name: "{{ ns }}: Create temporary directory for ansible files"
    file:
      path: "{{ config_path.stdout }}/ansible"
      state: directory
      owner: "{{ user_sys }}"
      group: "{{ user_sys }}"
      mode: '0755'
    register: tempdir
  - name: "{{ ns }}: Generate toolkit configuration for userChrome.js"
    copy:
      src: "{{ fls_src }}/ansible/toolkit.js"
      dest: "{{ tempdir.path }}/toolkit.js"
      owner: "{{ user_sys }}"
      group: "{{ user_sys }}"
      mode: '0644'
  - name: "{{ ns }}: Assemble user.js from individual calendar files"
    assemble:
      src: "{{ tempdir.path }}/"
      dest: "{{ config_path.stdout }}/user.js"
    when: config_path.stdout is defined and config_path.stdout != ''
```

The `{{ fls_src }}/ansible/toolkit.js` file as the following content:

```
// Enable ~/.thunderbird/UUID.default-default/chrome/userChrome.css
user_pref("toolkit.legacyUserProfileCustomizations.stylesheets", true);
```

7 Change the Date Format with Ansible

The **Change the Date Format** section elucidates how to utilize `user.js` to configure Thunderbird. This section leverages the Ansible playbook from the preceding section to automate this process.

```
    - name: "{{ ns }}: Generate custome date configuration"
      copy:
        src: "{{ fls_src }}/ansible/date.js"
        dest: "{{ tempdir.path }}/date.js"
        owner: "{{ user_sys }}"
        group: "{{ user_sys }}"
        mode: '0644'
```

The file `{{ fls_src }}/ansible/date.js` contains the following settings:

```
// Setting custom date format: yyyy-MM-dd
user_pref("intl.regional_prefs.use_os_locales", false)
user_pref("intl.date_time.pattern_override.date_short", "yyyy-MM-dd")
```

8 History

Version	Date	Notes
0.1.8	2023-07-21	Add section Change the Date Format with Ansible
0.1.7	2023-07-20	Add Ansible playbook snippet for userChrome.css
0.1.6	2023-07-19	Improve playbooks (fixes, line width)
0.1.5	2023-07-15	Fix missing color in CalDAV template
0.1.4	2023-07-14	Add section about adding a ICS calendar (Ansible)
0.1.3	2023-07-13	Add section about adding a CalDAV calendar (Ansible)
0.1.2	2023-07-12	Fix formatting
0.1.1	2023-07-10	Add section about changing calendar font size
0.1.0	2023-07-07	Initial release

9 Disclaimer of Warranty

UMENT OR THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE INFORMATION, DOCUMENTS AND PROGRAMS IS WITH YOU. SHOULD THE INFORMATION, DOCUMENTS OR PROGRAMS PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

10 Limitation of Liability

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE INFORMATION, DOCUMENTS OR PROGRAMS AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE INFORMATION, DOCUMENTS OR PROGRAMS (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE INFORMATION, DOCUMENTS OR PROGRAMS TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.