5. Using Templates

As your vault grows, the necessity of *automating recurring processes* becomes apparent. Luckily, Obsidian is compatible with a template engine called Templater.

In this section, I will give you an overview of the Templater plugin1 and explain how the Note Generator note included in this vault works.

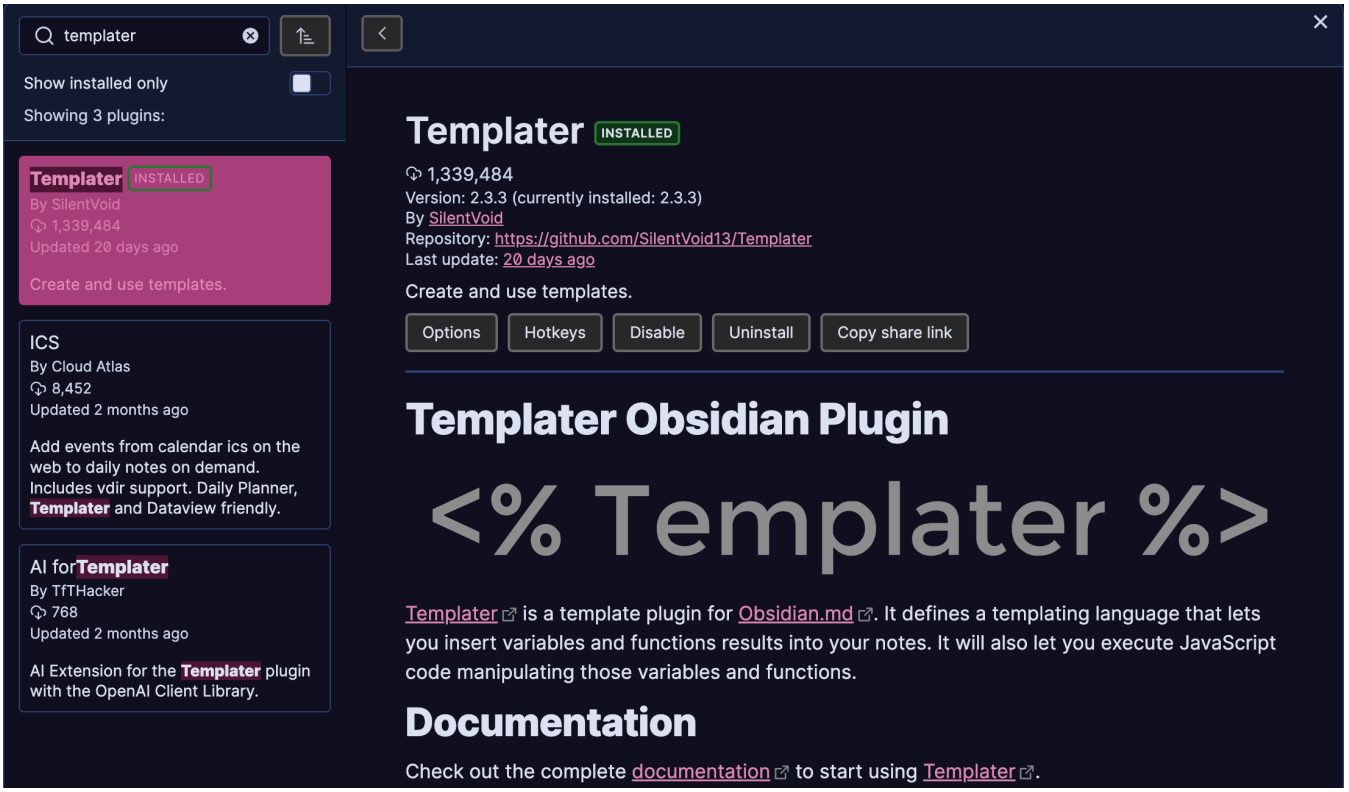
5.1 Plugins Needed for This Methodology

For this methodology, you will need the following plugin(s):

1. **Templater**: Templater is a powerful templates plugin that simplifies the creation of notes. This plugin features thorough documentation that you should familiarize yourself with. A basic understanding of JavaScript is necessary to get the most out of it.
2. **Admonition**: Admonition is an optional plugin used by the template notes featured in this tutorial. This plugin works by creating a "callout" when you add code blocks within your note and append `ad-[suffix]` to the first three single quotes. Documentation for this plugin is available on its GitHub page.

5.2 How to Implement This Methodology

Start by downloading and installing Templater.



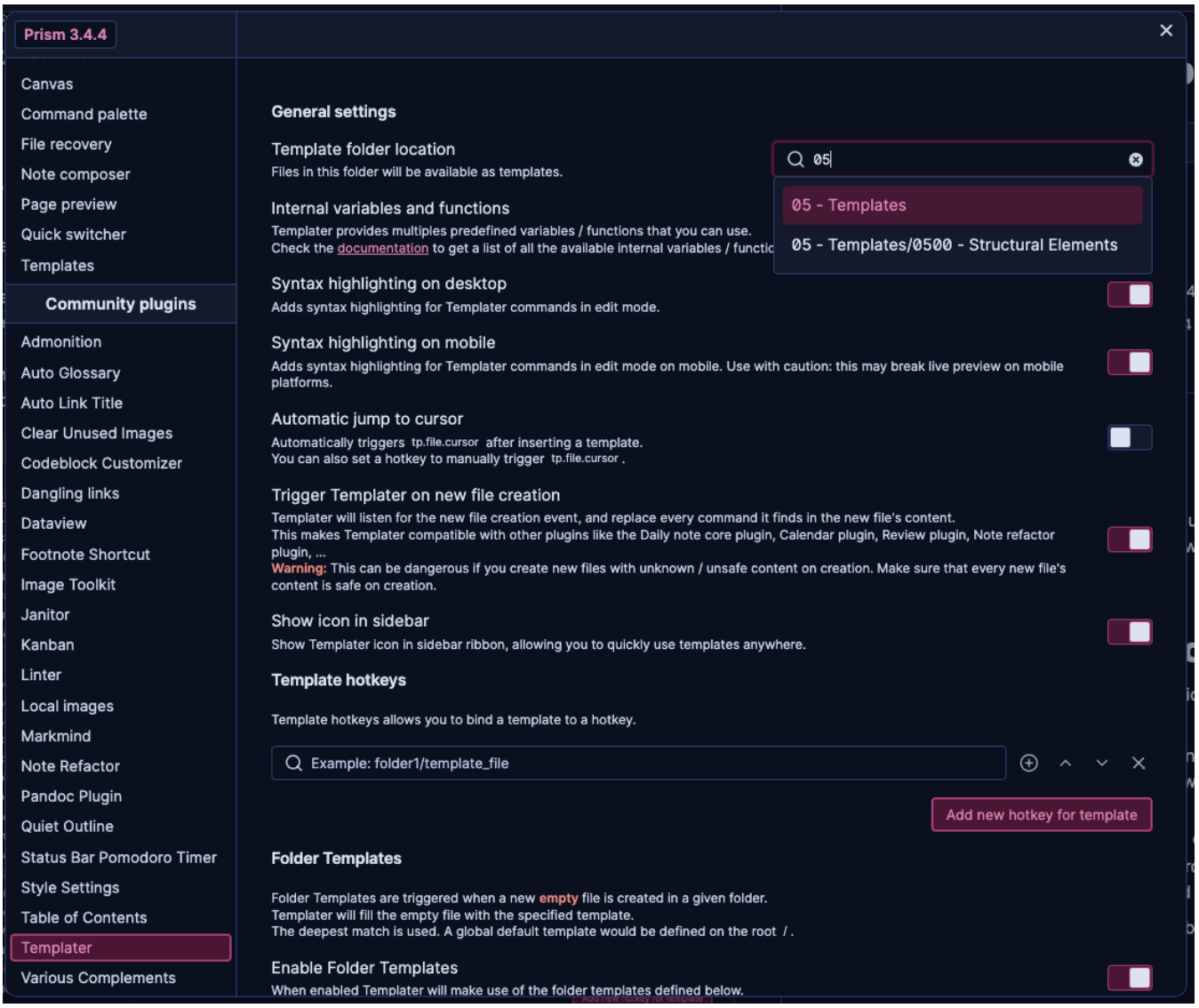
5.2.1 Templater Setup

We can now proceed to setting up Templater.

We have already discussed *template notes*. As you know, we are keeping them in the `05 - Templates` folder. Think of these as the skeletons for all your future notes.

Under Templater's *General Settings*, specify where these template notes are located within your vault. Enter "05 - Templates" at the Template folder location prompt.

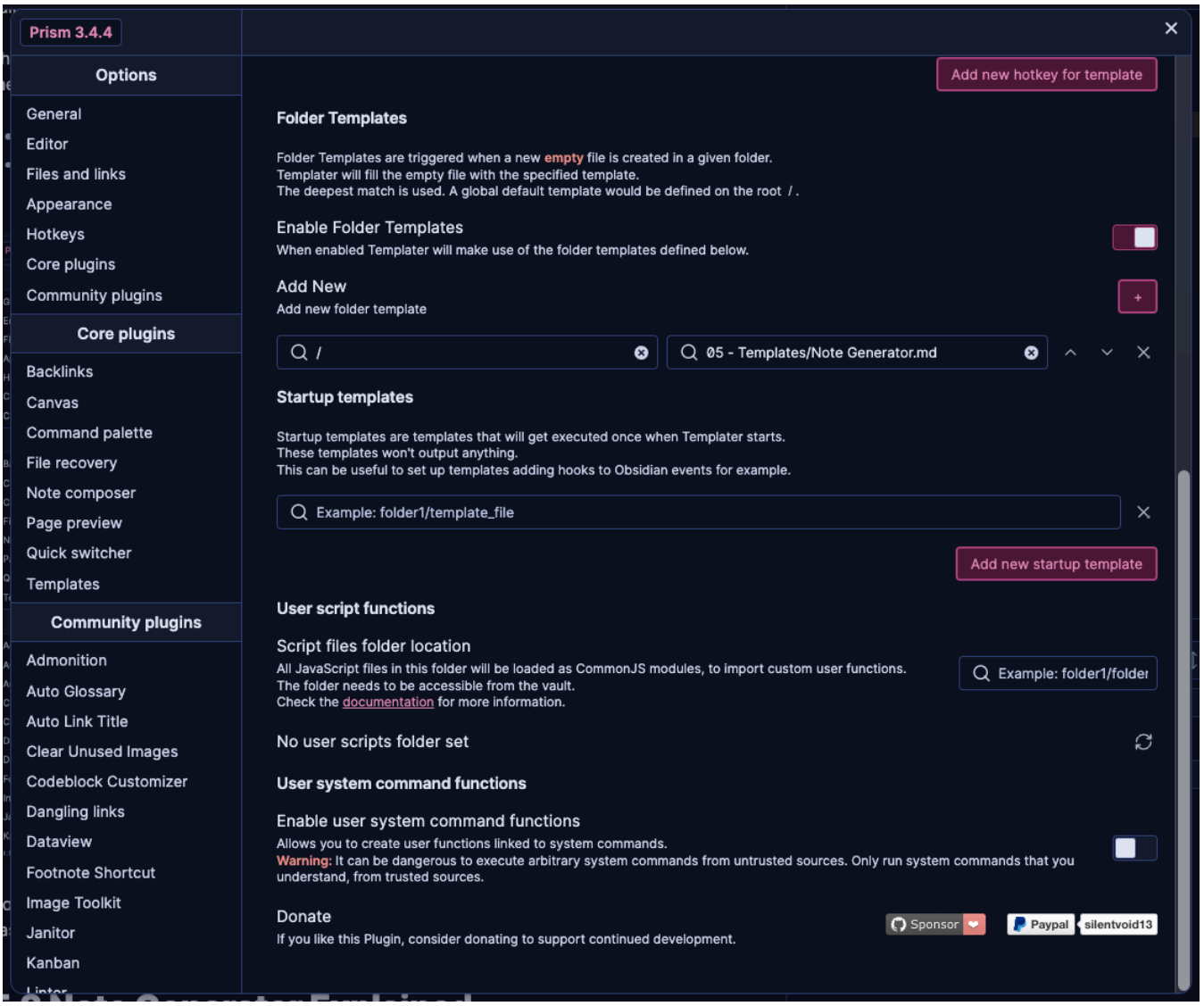
Make sure to enable "Trigger Templater on new file creation".



Next, we will set up the "Folder Templates" settings. Enabling this feature will trigger an API call to the Templater plugin and look for our "Note Generator".

The Note Generator is a special note written in JavaScript, which will be explained further in the next section.

- First, turn on the "Enable Folder Templates" option.
- Then, enter the "Note Generator" location `05 - Templates/Note Generator.md` in the rightmost prompt under "Add New".

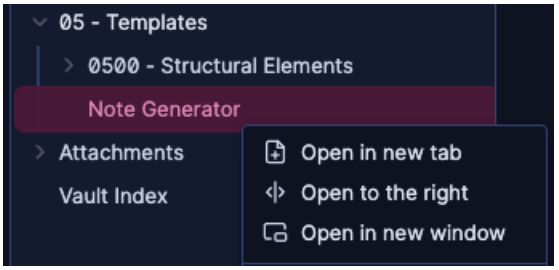


Now we are done setting up Templater. There are more advanced features, but for our use case, which is documenting our techniques, tools, and concepts, this is more than enough.

5.3 Note Generator Explained

Now, we will walk through our note generator.

Start by opening it next to this document to make it easier for you to follow along with this walkthrough.



> **Quick disclaimer**: I do not have a solid programming background, so expect this code to look rather unoptimized. I encourage you to collaborate in making it better. Simply make a pull request on this project's GitHub2, and we will review the code and merge it.

5.3.1 The Script

5.3.1.1 Title Prompt

The script starts by setting a new variable to hold the note's title. A prompt will appear in the user's window asking for input. After you enter a title, a quick check ensures no invalid title is set.

```
<%*
////// Prompt user for note title
// returns $title
var title = tp.file.title;
// No empty titles or untitled in file name allowed.
if (title === null || title.includes('Untitled') || title === "" ){
    var title = await tp.system.prompt("Note Title:");
    await tp.file.rename(title);
} else {
    console.log(`Title: "${title}" is valid`);
}
```

5.3.1.2 Note Type Prompt

Next, a prompt opens asking the user to select a *note type*.

```
////// Prompt user for note type
// returns $type
var type = await tp.system.suggester( (item) => item, [
    "Cheatsheet Note",
    "Documentation Note",
    "Technique Note",
    "Write Up Note"
] );
console.log(`Select a note type: "${type}" is valid`);
```

The script retrieves all template notes from the "0500 - Structural Elements" subfolder. These notes are saved into different variables for subsequent use.

```
//// Retrieve all necessary files (TO-DO, turn this into a function so only files that are needed get retrieved)
body = await tp.file.include("[[0505 - Body]]");
challenges = await tp.file.include("[[0506 - Body_Challenges]]");
mitre = await tp.file.include("[[0507 - Body_MITRE]]");
steps = await tp.file.include("[[0508 - Body_Steps]]");
opsec = await tp.file.include("[[0509 - Body_OPSEC]]");
code = await tp.file.include("[[0510 - Body_Code]]");
examples = await tp.file.include("[[0511 - Body_Examples]]");
install = await tp.file.include("[[0512 - Body_Install]]");
resources = await tp.file.include("[[0513 - Resources]]");
```

The script checks which *note type* was selected, selects the correct header note, and saves it to a variable. Then it selects the other parts of the note and saves them to a variable.

```
////// Select correct correct header and options based on $type
// returns $header
switch (type) {
    case "Cheatsheet Note":
        header = await tp.file.include("[[0501 - Header_CheatSheet]]");
        options = (mitre + examples + resources);
        break;
    case "Documentation Note":
        header = await tp.file.include("[[0503 - Header_Documentation]]");
        options = (install + resources);
        break;
    case "Technique Note":
        header = await tp.file.include("[[0502 - Header_Technique]]");
        options = (mitre + challenges + steps + opsec + code + examples + resources);
        break;
```

5.3.1.3 Write Up Note Type

If your note type selection is a "Write Up" note, the script follows a different logic. It will ask you to select a sub-type. This feature is helpful for auditing a single machine or a more complex Active Directory set.

I will briefly explain each "Write Up" note:

- *Progress Tracker Note*: Used when attacking Active Directory sets. It helps track compromised machines, stolen credentials, trusts, MSSQL database links, and hashed passwords for cracking.
- *External Discovery Template*: Useful when starting to enumerate an IP address. It contains common commands for auditing different protocols. The user is prompted to select a machine IP, then all commands are populated using this IP.
- *Linux Methodology Template*: Used to track discovery, credential access, privilege escalation, persistence, port redirection settings, and lateral movement efforts on compromised Linux machines.
- *Windows Methodology Template*: Similar to the Linux template, but for Windows machines where a foothold has been established.
- *Active Directory Methodology Template*: Designed to track Active Directory exploitation progress.
- *Payload Delivery Commands Template*: A bonus note with useful Windows exploitation commands. Customize it with your most used commands for maximum utility.

```
// write up note (more complex)
case "Write Up Note":
    header = await tp.file.include("[[0504 - Header_Writeup]]");
    var type = await tp.system.suggester( (item) => item, [
        "Progress Tracker Note",
        "Discovery Note",
        "Host Note",
        "Domain Note",
        "Payload Delivery Note"
    ] );
```

Upon selecting your *Write Up* note, a nested switch statement checks to select your desired note.

```
    switch (type) {
        case "Progress Tracker Note":
            progressTracker= await tp.file.include("[[0614 - Progress Tracker Template]]");
            options = progressTracker;
            var note = (header + options);
            console.log(`Note defined. All ready`);
            return note;
            break;
        case "Discovery Note":
            discovery = await tp.file.include("[[0615 - External Discovery Template]]");
            options = discovery;
            var note = (header + options);
            console.log(`Note defined. All ready`);
            return note;
            break;
            break;
        case "Host Note":
            host = await tp.file.include("[[0616 - Host Discovery + Privilege Escalation Template]]");
            options = host;
            var note = (header + options);
            console.log(`Note defined. All ready`);
            return note;
            break;
        case "Domain Note":
            domain = await tp.file.include("[[0617 - Domain Discovery + Privilege Escalation + Lateral
            Movement Template]]");
            options = domain;
            var note = (header + options);
            console.log(`Note defined. All ready`);
            return note;
            break;
        case "Payload Delivery Note":
            payloadDelivery = await tp.file.include("[[0618 - Payload Delivery Commands Template]]");
            options = payloadDelivery;
            var note = (header + options);
            console.log(`Note defined. All ready`);
            return note;
            break;
        default:
            console.log("Invalid write up note type.");
```

3,796 characters

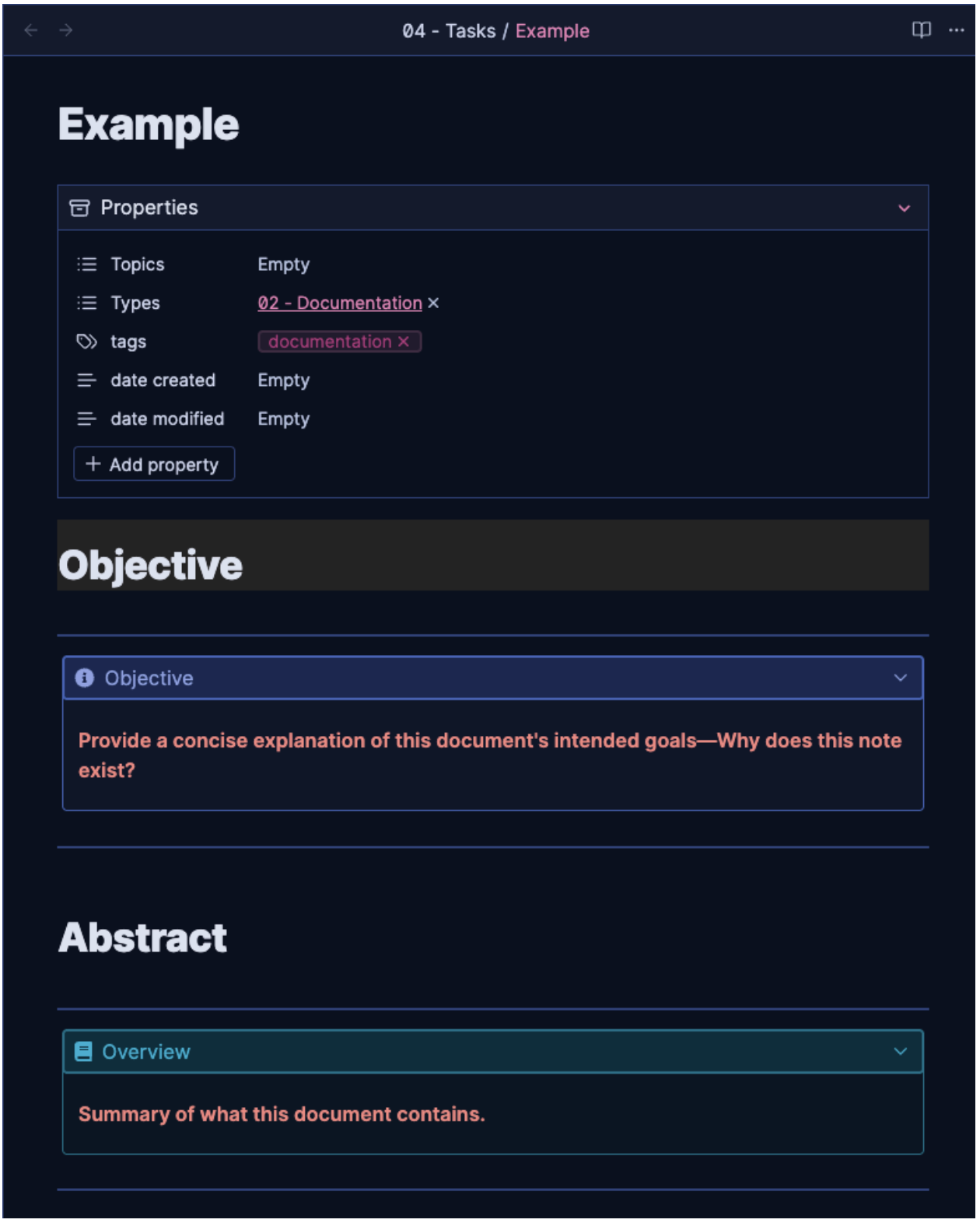
5.3.1.4 Finishing Up

The program then appends all previous selections into a note, and the note is created.

```
////// Builds note using $header, main body and options
var note = (header + body + options);
console.log(`Note defined. All ready`);
return note;
%>
```

Let's try creating a new note. I will select "Example" as the *note title* and "Documentation" as the *note type*.

As you can see, a new note titled "Example" was created.



> Try experimenting with other combinations and feel free to edit the template notes to fit your own note-taking methodology. I personally like using callouts as much as possible with Admonition.