Set up self-administered reader options

Last Modified on 04/28/2026 3:37 pm EDT

If you allow readers to administer their own passwords, you should review the **Self-Administered Reader Options** in **Account > Readers > Settings**.

Use these options to control:

- How frequently passwords expire
- If readers are allowed to reuse passwords
- If passwords need to meet specific validation/complexity rules
- If reader signups are allowed, and what the process looks like if they are

Review and change settings

To review and update these settings:

1. Go to **Account > Readers**. The **Readers** page opens to the **Readers** tab.
2. Open the **Settings** tab.
3. If the **Reader Password Security** section's **Password Management** option to **Allow readers to administer their own passwords** is selected, scroll down to the **Self-Administered Reader Options** section. Use the settings here to control the following password behavior:
    - **Password Expiration Interval:** Use this control to set whether reader passwords should expire and how frequently they should expire. When a reader's password expires, they're prompted to create a new password the next time they log in. Choose from these options:
        - Never (default)
        - Every Month
        - Every 2 Months
        - Every 3 Months
        - Every 6 Months
        - Every Year
    - **Repeat Password Limitations:** Use this control to set whether readers can reuse an existing password

and/or how many previous passwords they can't reuse. Choose from these options:

- None (default)
- Cannot use previous password
- Cannot use previous 2 passwords
- Cannot use previous 3 passwords
- Cannot use previous 4 passwords
- Cannot use previous 5 passwords

4. **Custom Validation Rule:** Use this setting to enter regex to enforce your company's password requirements for complexity or format. You can find prewritten validation rules using your favorite search engine. Refer to Regex for custom validation rules for more help.
5. **Custom Validation Description:** This message will be displayed on the password reset screen if you have a custom validation rule. Use it to tell your reader about the rule so they can create a password that works.
6. **Auto-assign Group Rules:** If you are using group rules to automatically assign your readers to groups, check this box to ensure that reader groups will update based on the rules each time a reader logs in. This allows you to create new rules and have it automatically applied to existing readers, but it will override any groups you might have set manually. Don't turn this option on if you're manually setting reader groups.
7. **Allow Google Sign In:** You can allow readers to sign up for and log in to your knowledge base with their Google account. Refer to Allow Google log in for readers for the additional steps to get Google Sign-in set up on your knowledge base.
8. **Reader Signups:** Add a reader signup link to your login page so readers can sign up on their own. By default, new readers will be added and a welcome email will be sent with a temporary password.
    a. You can choose to require an admin approval before the welcome email is sent.
    b. You can also set up notification emails to inform you of new reader signups or signup requests.
9. **Signup Notification Recipients:** If you're using reader signups, add the email address(es) you'd like to be notified when a new reader signs up or requests access.
    a. To add multiple addresses, use a comma-separated list, such as: `linus@knowledgeowl.com,owlbert@knowledgeowl.com`.
10. Be sure you **Save** your changes.

Regex for custom validation rules

KnowledgeOwl doesn't automatically enforce any password validation, but you can use the **Custom Validation Rule** field to force readers to use more complex passwords (for example, enforce a mixture of upper and lower case, numbers, and symbols).

What is regex?

Regex is a common abbreviation of 'regular expressions'. Regular expressions are "a sequence of characters that specifies a *search pattern*. Usually such patterns are used by string-searching algorithms for "find" or "find and replace" operations on strings, or for input validation" (from Wikipedia - Regular expression).

KnowledgeOwl takes the regex you provide and uses it to check that the password the reader creates matches your requirements.

Regex password rule examples

Regex can be very powerful, and can look very complicated. Don't panic! If you're stuck, check if any of these examples meet your requirements. You can always contact us for more help:

Password rules	Regex	Modifications
Password must: • Be eight characters or more • Include at least one each of: number, symbol, lowercase letter, uppercase letter • Not contain whitespace	`^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&+=])(?=\S+$).{8,}$`	• If you want to remove or add permitted symbols, change the contents of `[@#$%^&+=]` • If you want to change the character limit (for example, you want 12 as a minimum length, rather than eight), change the number in `{8,}`
Password must be eight characters or more. It can contain any characters apart from whitespace	`^(?=\S+$).{8,}$`	Change the `8` to any other number to alter the length restriction
Password must be between 12 and 24 characters long. It can contain any characters apart from whitespace	`^(?=\S+$).{12,24}$`	• Change the `12` to another number to alter the minimum length • Change the `24` to another number to alter the maximum length

Password rules	Regex	Modifications
Password must: • Be 16 characters or more • Include at least one each of: lowercase letter, uppercase letter • Not contain whitespace	`^(?=.*[a-z])(?=.*[A-Z])(?=\S+$).{16,}$`	Change the `16` to any other number to alter the length restriction
Password must: • Be between 12 and 128 characters. • Contain three out of four of: number, symbol, lowercase letter, uppercase letter • Have no more than two of the same character in a row	`^(?:(?=.*\d)(?=.*[A-Z])(?=.*[a-z])|(?=.*\d)(?=.*[^A-Za-z0-9])(?=.*[a-z])|(?=.*[^A-Za-z0-9])(?=.*[A-Z])(?=.*[a-z])|(?=.*\d)(?=.*[A-Z])(?=.*[^A-Za-z0-9]))(?!.*(.)\1{2,})[A-Za-z0-9!~<>,;:_=?*+#."&§%°()|\[\]\-\$\^\@\/]{12,128}$`	

A detailed example

It's fine to just use any of the examples from the list above, but if you want to learn a bit more about what they are doing and how regex works, here is a detailed explanation of one of the examples.

```
^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&+=])(?=\S+$).{8,}$
```

- `^` tells us that we must match the pattern from the start of the line. For example, `^a` means to look for 'a' at the start of the line. It matches the 'a' in 'abc', but not in 'bca'.
- `>(?=)` creates a positive lookahead. This means that it matches something followed by something else. For example, `>h(?=o)` matches an 'h' followed by an 'o'. So it matches the 'h' in 'hoot' but not in 'hype'. In our example it's a bit more complicated, as we have chained multiple positive lookaheads together, and added some special characters.
- `.` matches any single character, and `*` tells us to match the previous token any number for times (from zero up). So `.*` means match any number of single characters. By itself this is meaningless: `.*` would match any phrase. In the context of our example, it means that the restriction that follows it (the bit in square brackets) can be preceded by any number of any other characters. For example, given the regex `.*[0-9]`, we'll match 'owl23', 'lotsOfOwls36', 'owlsWithSomeSymbols$48', and so on.
- `[]` ranges in square brackets match a single character in that range. For example, by itself `[0-9]` matches each number in '123owl456'. In our example, `[@#$%^&+=]` provides a list of symbols readers can use in their password.
- `\S` matches any non-whitespace character, and `+` tells us to match the previous character any number of times (from one up - so there has to be at least one character). `$` indicates the end of a line. To take one of our previous example phrases, 'abc', `c$` matches the 'c' at the end, in the same way that `^a` matches the 'a' at the beginning. In the context of our example, `\S+$` is there to ensure there are no whitespace characters in the password.
- `{8,}` tells us to match the preceding character (in this case, `.`, which matches any single character), eight

or more times. In other words, there must be at least eight characters in the line for it to match. This means if you want a 12 character minimum length, you can change the 8 to 12.

Tips for creating your own regex rule

- Start the rule with `^`. This ensures we look for a password that matches your rule right from the start of the phrase that the reader enters.
- Include `(?=\S+$)` to ensure readers can't create passwords containing whitespace.
- Be aware that KnowledgeOwl uses PCRE2 (PHP >=7.3) regex.

Learning more

If you want to dive in and really learn regex, here are a few tips to get you started:

- Be aware different programming languages can have slightly different flavors of regex. If you're already familiar with/using a particular language, it's worth looking for regex tutorials specific to that language.
- regex101 is a handy site that allows you to try out Regex rules against example words and phrases. When testing regex for use in KnowledgeOwl, select **PCRE2 (PHP >= 7.3)** under **FLAVOR**.