MYRA

Myra EU CAPTCHA

Manual

Version History

Version	Date	Reason for Change
V1.6	08/2026	New sections: • Content Security Policy Updated sections: • How it works • Embedding the widget • Integration • HTML and JavaScript • TYPO3 • FAQ • Widget not detected • Glossary
V1.5	08/2026	New sections: • Traefik • Client from the specification • Managing sitekeys • Retrieving statistics • Glossary
V1.4	08/2026	New sections: • Managing additional secrets • Changing the personal data • Reset password • Password policy • Deleting passkeys • Delete account • Cookie and tracking settings Updated sections: • Integration

Version	Date	Reason for Change
		• Create an account • Log in • Create a passkey • Logins tab
V1.3	08/2026	New sections: • Getting access • Initial setup • Commit to a plan during the trial period Updated sections: • Getting started • Trial period
V1.2	08/2026	New sections: • Getting started • Trial period Updated sections: • Introduction • How EU CAPTCHA works • Testing the integration Sections deleted: • System architecture
V1.1	08/2026	Updated sections: • Introduction • How EU CAPTCHA works • System architecture • Create the first sitekey • Testing the integration • Dashboard • Sitekey List area

Version	Date	Reason for Change
		• Configuration • Integration • Permissions • Challenges • Profile tab • Plans tab • Subscription History tab • Logins tab • Delete Account tab • Help • Creating a sitekey • Configuring a sitekey • WordPress
V1.0	08/2026	Initial version

Contents

1. Introduction

Myra EU CAPTCHA protects websites and APIs against misuse such as form spam and credential stuffing. The service is operated by Myra Security GmbH and hosted in Europe. The verification runs invisibly in the background; visitors solve no puzzle.

> **Video:** Overview of Myra EU CAPTCHA and the web interface — playable in the online help

1.1 Characteristics

The service differs from classic CAPTCHA methods in the following points:

Characteristic	Description
Form and login protection	The service protects forms and login pages against spam and credential stuffing.
Bot protection	A fingerprint detection recognises requests from bots and blocks them.
Verification without user input	Users solve no puzzle and tick no checkbox. The verification runs automatically in the background, through cryptographic computations.
Intelligent risk detection	The risk analysis adjusts the difficulty of the challenges dynamically as soon as a behaviour becomes suspicious.
Signal-based	The detection is trained daily with more than 100 billion CDN signals.
No cookies and no data storage	The service works without HTTP cookies and without persistent browser storage. No cookie banner is required for it.
European operation	The service is operated in Europe and designed for the GDPR.
Self-service setup	The integration requires no DNS change and is completed in three steps.
Cross-platform integration	Libraries and scripts are available for CMS platforms, frontend frameworks, backend languages, mobile apps and infrastructure tools.
API support	A REST API is available for the programmatic connection.
Dedicated dashboard	A web interface shows the statistics and manages the sitekeys.
Sitekey creation	A new sitekey is created by entering the domain to be protected.

Characteristic	Description
Sitekey configuration	The protection can be activated or deactivated per sitekey. Difficulty, delay and parallel challenges are configurable.
Widget customisation	A dark mode is available for the widget.
Access management	Team members receive read or write access to a sitekey.

1.2 Limitations

EU CAPTCHA is at an early stage of development. Note the following limitations:

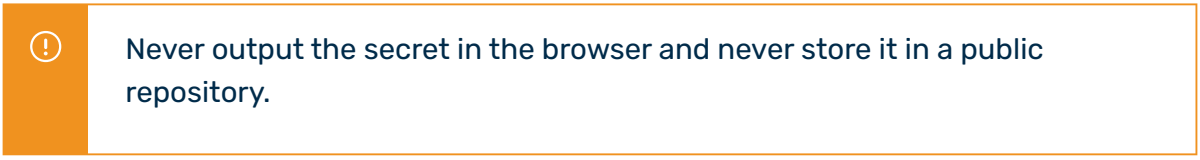
Limitation	Description
No integration with other Myra products	A connection to other products of Myra Security is currently not available.
No migration	A migration option from an existing CAPTCHA solution is currently not available.
JavaScript required	The verification is based on JavaScript and a client-side proof of work. Without activated JavaScript, it does not work.
Computing time on weak devices	On low-performance devices, the proof of work takes up a noticeable amount of computing time.
No absolute protection	Even a signal-based bot detection does not completely rule out highly developed bots with real browser environments.

Understand the risk assessment as one layer of protection, not as the only security measure.

1.3 Sitekey and secret

Every protected domain gets a key pair:

Key	Format	Use
Sitekey	UUID	Public. Appears in the HTML of the page and identifies the website.
Secret	Base64	Confidential. Used on the server only, for the verification.



Never output the secret in the browser and never store it in a public repository.

Both values appear in the **Public Sitekey and Secret(s)** area. You reach it through the **Integrations** button in the row of the sitekey, **Details** step. See Details view.

1.4 How EU CAPTCHA works

Two components provide the protection: the widget, which runs in a hidden iframe on the protected page, and the verification API of EU CAPTCHA. The verification runs in the background and requires no input from the visitors.

1.4.1 Components

Myra EU CAPTCHA consists of the following components:

Component	Address	Task
Dashboard	`https://app.eu-captcha.eu`	Manages the account, sitekeys, permissions and statistics.
Widget	`https://cdn.eu-captcha.eu/verify.js`	Runs the verification in the user's browser and creates the token.
Verification API	`https://api.eu-captcha.eu/v1`	Verifies the token on the server.
Documentation	`https://docs.eu-captcha.eu`	Contains this manual.

The protected page loads the widget from the CDN and speaks to the verification API. If your website sends a Content Security Policy, then it must permit the two addresses. See **Content Security Policy**.

1.4.2 How a verification runs

A verification runs in the following steps:

1. The protected page loads `verify.js` from the CDN.
2. `verify.js` creates a hidden iframe and loads `check.html` into it.
3. `check.html` loads `check.all.js`. This script performs the actual verification.
4. The iframe reports the created token back to `verify.js`.
5. `verify.js` writes the token into a hidden input field named `eu-captcha-response`.
6. When the form is submitted, the token reaches your own server.
7. Your own server verifies the token through the endpoint `POST /verify` of the verification API.

The verification in the browser alone gives no protection. Only the server-side verification of the token decides whether a request is accepted or rejected. See **Embedding the widget**.

1.4.3 Risk assessment

When the page loads, the hidden iframe collects a fingerprint from dozens of browser signals. The visitors are not involved in this. The following groups of signals go into the assessment:

Signal group	Examples
Hardware	Characteristics of the device and the geometry of the screen
Rendering	Display through WebGL and audio
Behaviour	Mouse movement, scrolling behaviour and interactions
Network	Characteristics of the connection and of the origin of the request

From these signals, the service calculates a risk score between 0.0 and 1.0.

The service does not output the reasons for a block through the API. A disclosed detection logic would allow attackers to circumvent the protection in a targeted way. See **The cause of a block is not visible**.

1.4.4 Challenge and proof of work

The risk score determines the difficulty of the challenge: the higher the risk, the more elaborate the computation. The service knows eight levels of difficulty:

Difficulty	Computing effort	Applies to
0	around 600 milliseconds	Standard case for human visitors. The verification stays unnoticed.
1 to 6	rising	Suspicious signals raise the effort step by step.
7	around 30 minutes	Traffic with a very high risk.

The computation is a proof of work. Together with the task, the browser receives a target value and, in a background thread, tries values until the result reaches the target value. The algorithm used is Argon2id. It is memory-intensive and therefore resistant to attacks through graphics cards and botnets.

The widget reports the solution to the verification API:

- If the API answers with `wait`, the browser continues to compute.
- If the API answers with `success`, the challenge is passed and the widget receives the signed token.

> The service knows no failed challenge. A challenge is either solved or it takes longer. An unusually long duration indicates traffic with a high risk. See **A challenge takes an unusually long time**.

> The initial value that you set for each sitekey contains the levels 0 to 3. The risk detection sets the higher levels itself. See **Configuring a sitekey**.

1.4.5 Verification modes

The point in time at which the verification starts is adjustable. The following modes are available:

Mode	Point in time of the verification	Use
Solved Immediate (default)	The verification starts as soon as the page is loaded.	Best user experience: the verification is already complete when the users submit the form.
Solved Triggered	The verification only starts once the users touch the form.	Pages with many calls but few submissions, for example checkout or ticket pages.

The mode affects the consumption: in the **Solved Immediate** mode, every page call with a form counts as an assessment; in the **Solved Triggered** mode, only the submissions actually begun.

You change the mode in the settings of your own application, that is, where the widget is embedded. The **Sitekey - [Domain]** view contains no setting for it.

An assessment counts as soon as the widget begins the verification in the background. The verification in the background and the corresponding server-side verification together count as one assessment, regardless of the selected mode.

2. Getting started

This chapter leads from the empty account to the protected website: create an account, log in to the customer portal, create the first sitekey, embed the widget, add the server-side verification and test the finished integration in the customer portal.

The following sections explain the following topics:

- **Getting access** describes the registration of an account and the log-in to the customer portal.
- **Initial setup** describes the way from the first sitekey through the widget to the confirmed integration test.
- **Trial period** describes the 30 days of the free trial period and the time after it.

2.1 Getting access

Access to the customer portal of Myra EU CAPTCHA requires a user account. New users create it themselves at `https://app.eu-captcha.eu` , existing users log in with their credentials.

The following sections explain the following topics:

- **Create an account** describes the registration with an email address or through a sign-in provider.
- **Log in** describes the log-in with a password, with a second factor and with a passkey.

No payment details are necessary for the registration. Every new account starts with a trial period of 30 days. See **Trial period**.

How the password of an existing account is changed is described under **Change password**.

2.1.1 Create an account

To use Myra EU CAPTCHA, you must have an account in the customer portal. You create it yourself. A credit card is not necessary.

The following requirements must be met:

☑ You have access to an e-mail mailbox.

Proceed as follows to create an account:

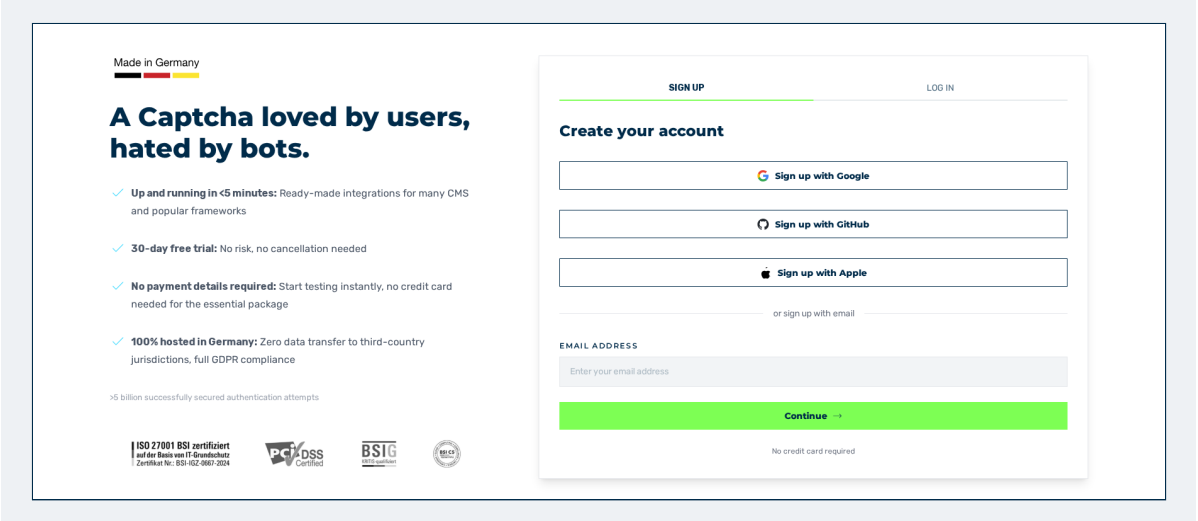


Figure 1: Create your account view

- ► Open the registration page of the customer portal.
- ► In the **Email address** field, enter your e-mail address.
- ► Click on the **Continue** button.
  - ↳ The **Complete your account** view is shown.

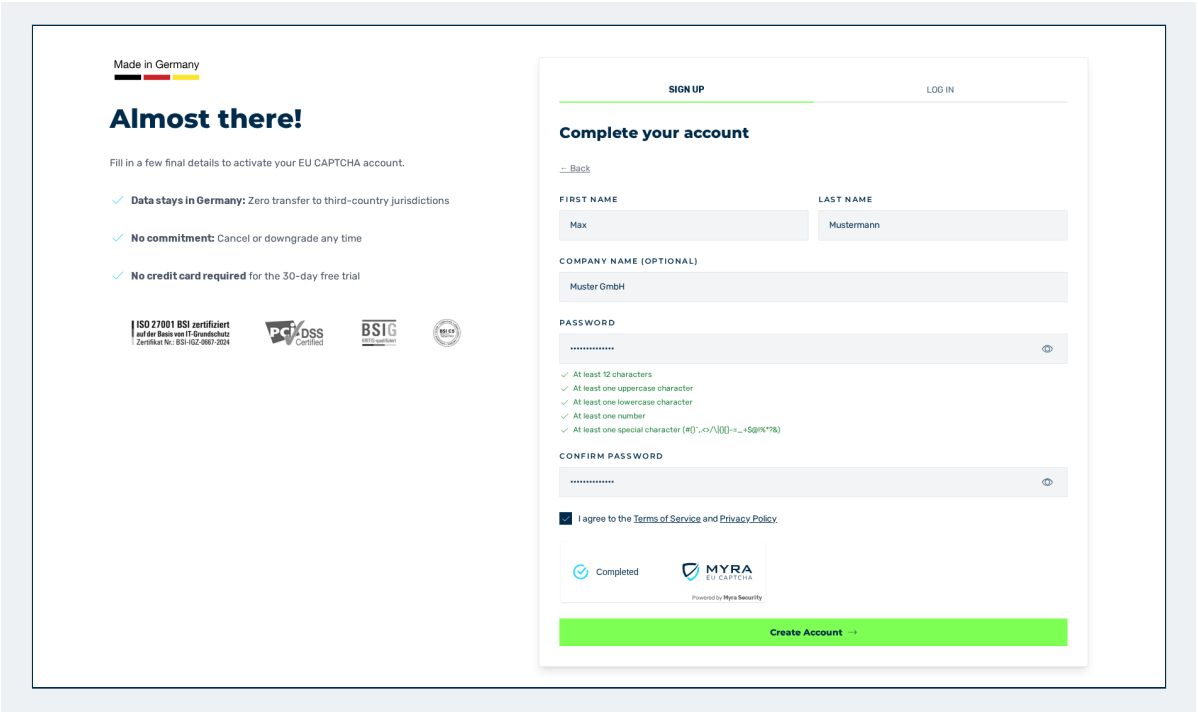


Figure 2: Complete your account view

- In the **First name** and **Last name** fields, enter your first name and your last name.
- If required, enter the name of your company in the **Company name (optional)** field.
- In the **Password** and **Confirm password** fields, enter the same password.
  - ↳ Below the **Password** field, a checklist shows which rules the password obeys.
- Select the check box in front of **I agree to the Terms of Service and Privacy Policy**.
- Solve the **EU CAPTCHA** widget below the check box.
- Click on the **Create Account** button.
  - ↳ The **Verify your email** view is shown. The system sends a six-digit confirmation code to the given e-mail address.

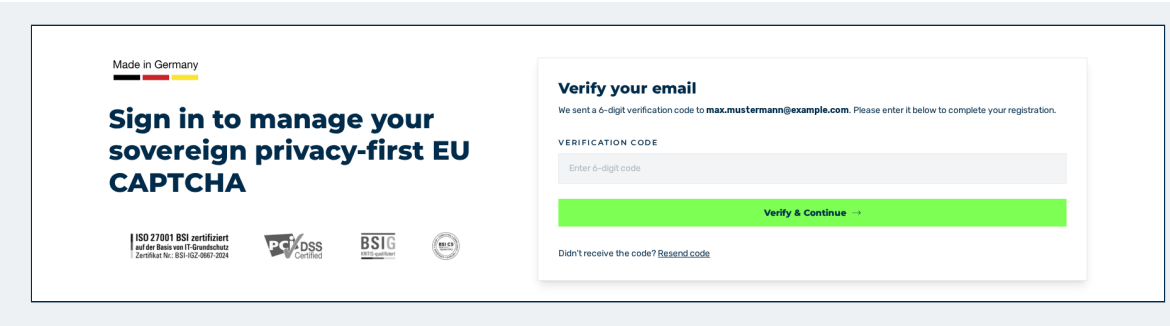


Figure 3: Verify your email view

- ► In the **Verification code** field, enter the code from the e-mail.
- ► Click on the **Verify & Continue** button.
- ➔ The account is created. You are logged in. The system shows the **Dashboard** view.

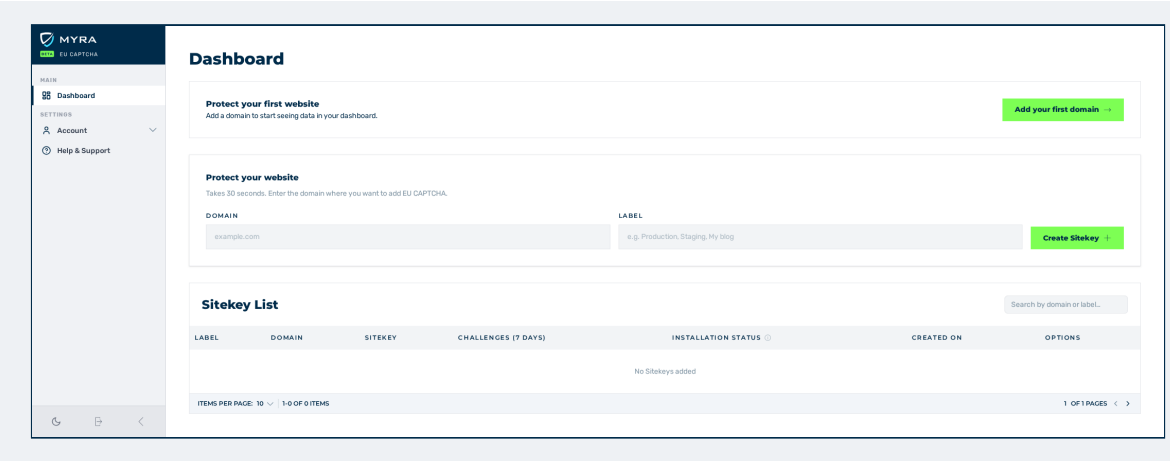


Figure 4: Dashboard view

The ← **Back** button takes you from the **Complete your account** view back to the entry of the e-mail address.

The password must obey the rules that follow:

Rule	Condition
Length	Minimum 12 characters
Uppercase character	Minimum one character A to Z
Lowercase character	Minimum one character a to z
Number	Minimum one number 0 to 9
Special character	Minimum one character from `#()^,.<>/|{}[]-=_+$@!%*?&`

If an account for the given e-mail address is already available, the view shows the **An account with this email already exists.** message. The view also gives the **Log in** and **Forgot password?** links.

If the confirmation code does not arrive, request it again with the **Resend code** link.

Instead of an e-mail address and a password, you can create the account with these buttons:

- **Sign up with Google**
- **Sign up with GitHub**
- **Sign up with Apple**

The view contains no button for passkeys. A passkey can be used for the login only. See **Create a passkey**.

See **Log in**.

2.1.2 Log in

You log in to the customer portal with your e-mail address and your password. If two-factor authentication is active for your account, the portal also asks for a code.

The following requirements must be met:

☑ You created an account and confirmed your e-mail address. See **Create an account**.

Proceed as follows to log in:

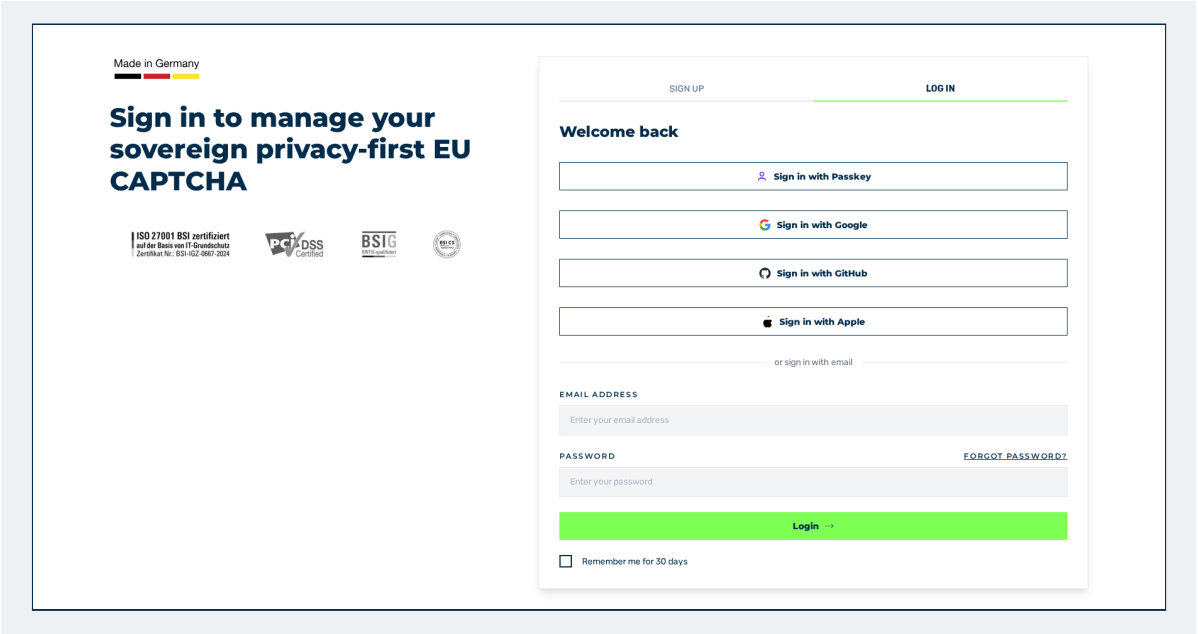


Figure 5: Welcome back view

- ► Open the login page of the customer portal.
- ► In the **Email address** field, enter your e-mail address.
- ► In the **Password** field, enter your password.
- ► If required, select the **Remember me for 30 days** check box.
- ► Click on the **Login** button.
  - ↳ If two-factor authentication is active, the **Two-Factor Authentication** view is shown.

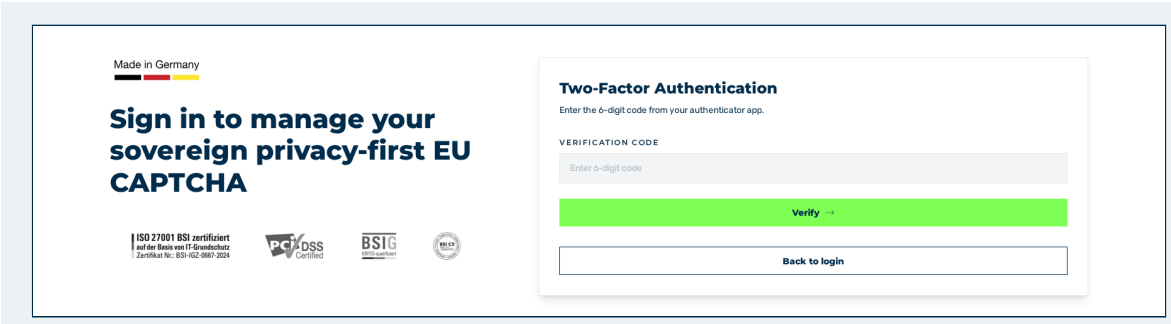


Figure 6: Two-Factor Authentication view

- ► Enter the six-digit code from your authenticator app.
- ► Click on the **Verify** button.
- ➔ The **Dashboard** view is shown.

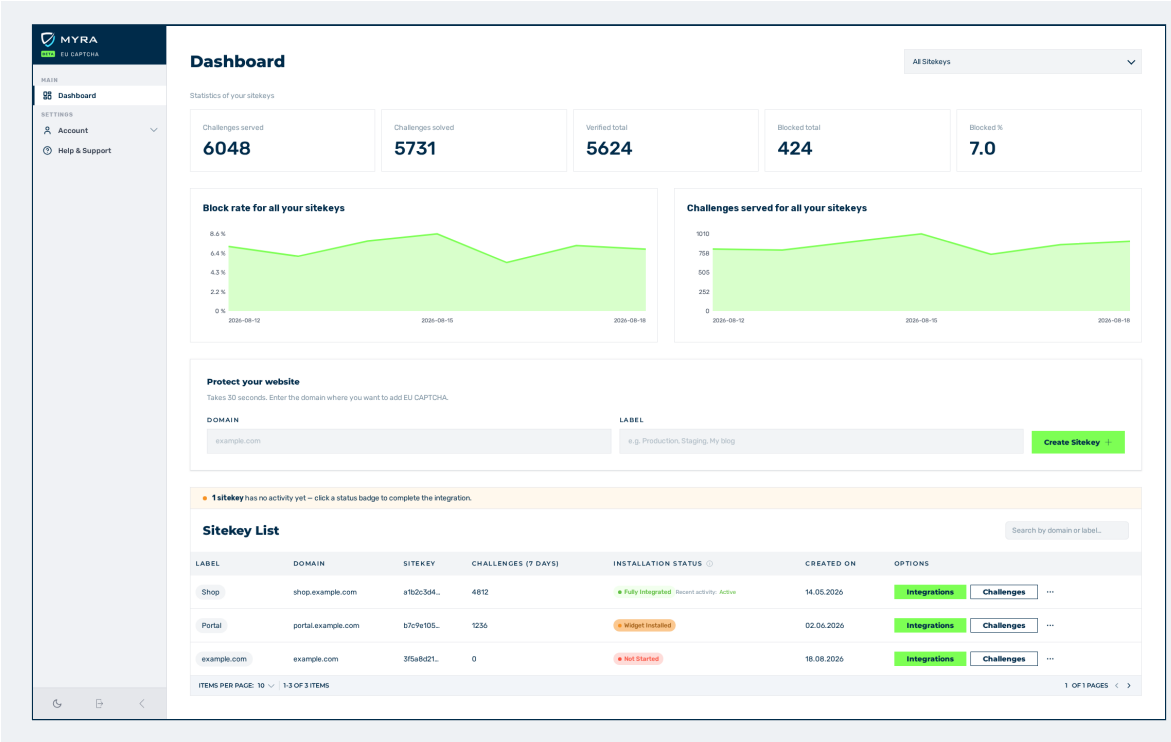


Figure 7: Dashboard view

The **Back to login** button takes you from the **Two-Factor Authentication** view back to the login page.

Instead of an e-mail address and a password, you can log in with these buttons:

- ■ **Sign in with Passkey**
- ■ **Sign in with Google**

- **Sign in with GitHub**
- **Sign in with Apple**

> The sequence of the buttons depends on your device. On Apple devices, **Apple** is in the first position; on Android devices, **Google** is in the first position. On all other devices, **Passkey** is in the first position. On Android devices, the **Sign in with Apple** button does not apply.

If you forgot your password, click on the **Forgot Password?** link on the login page. The system sends you an e-mail with a link to the **Reset Password** view.

See Dashboard view.

2.2 Initial setup

This section leads from the empty account to the confirmed integration: create the first sitekey, retrieve the key pair, embed the widget, verify the token on the server and have the result confirmed in the customer portal.

The customer portal accompanies this way with a wizard. It leads in three steps through the views **Details**, **Integration** and **Integration Test** and shows the indication **Step ⅓** to **Step 3/3** above them. See **Sitekey**.

The following sections explain the following topics:

- **Create the first sitekey** describes how the first sitekey for your own domain is created and where the key pair is shown.
- **Embedding the widget** describes the widget in the form and the verification of the token on your own server.
- **Testing the integration** describes the test that confirms both parts of the integration.

The following requirements must be met:

☑ You have an account at `https://app.eu-captcha.eu` and are logged in. See **Getting access**.

☑ You know the domain that must be protected.

☑ You have access to the source code of the page with the form and to the server that receives the form.

Complete examples for common combinations of front end and back end are given under **Integration**.

2.2.1 Create the first sitekey

A sitekey connects a domain with Myra EU CAPTCHA. For each domain that you want to protect, create a related sitekey. The sitekey has a public key for the widget and a secret for the server-side verification.

> **Video:** Creation of the first sitekey in the customer portal — playable in the online help

The following requirements must be met:

☑ You are logged in to the customer portal. See **Log in**.

☑ You know the domain that you want to protect.

Proceed as follows to create the first sitekey:

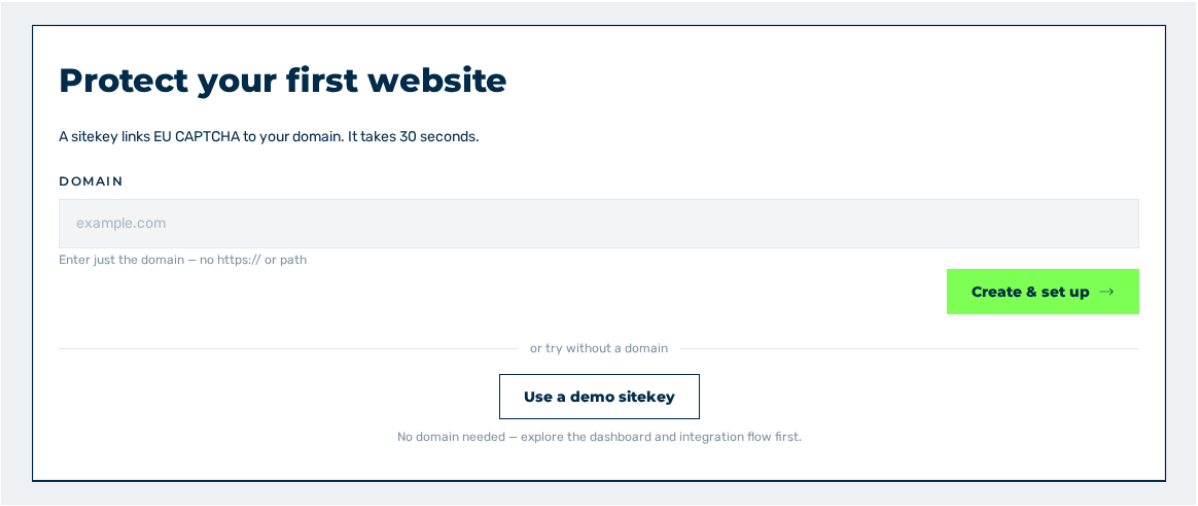


Figure 8: Protect your first website dialog

► Open the **Dashboard** view.

↳ While no sitekey is available, the **Protect your first website** dialog is shown. The description is **A sitekey links EU CAPTCHA to your domain. It takes 30 seconds.**

► In the **Domain** field, enter the domain to protect, for example `example.com`. The note below the field is **Enter just the domain — no https:// or path**.

► Click on the **Create & set up** button.

↳ The system creates the sitekey and shows the **Success** message with this text: **You have successfully created your first Sitekey! Now it is time to integrate the Sitekey on your website. We will show you how.**

➔ The **Integration** view of the new sitekey is shown.

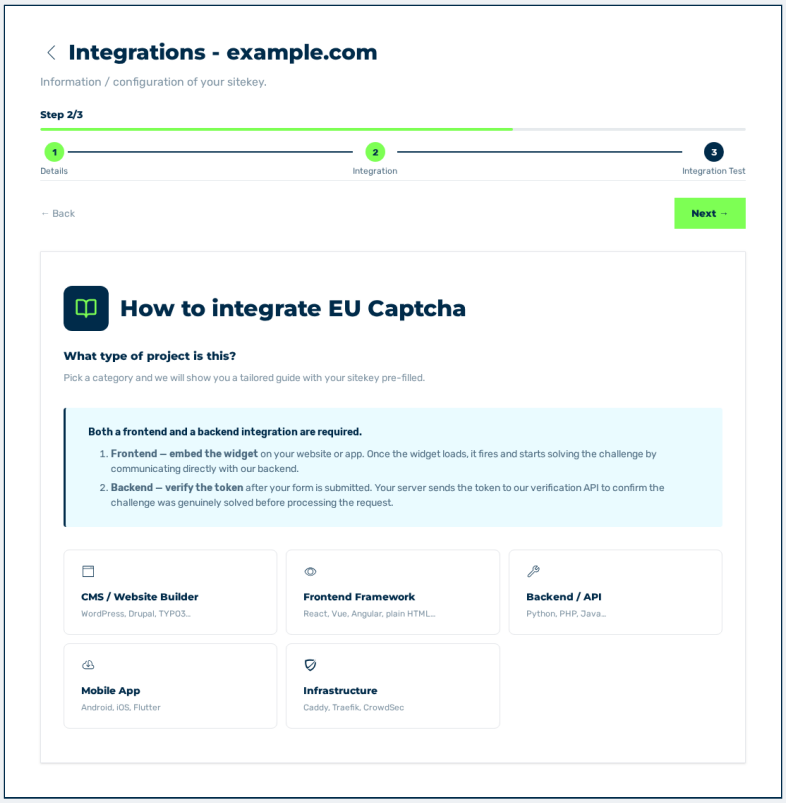


Figure 9: Integration view

> At this time, the system does not prevent more than one sitekey for the same domain. Examine the list of the sitekeys and delete the double entries. See Two sitekeys for the same domain.

2.2.1.1 A sitekey without your own domain

Below the **or try without a domain** divider, the **Use a demo sitekey** button creates a sitekey with a generated demo domain. The description is **No domain needed – explore the dashboard and integration flow first.**

The system stays in the **Dashboard** view and shows the **Sitekey created** message with this text: **Your sitekey is ready. Use the integration guide below to add EU CAPTCHA to your project.** The new entry is highlighted in the **Sitekey List** list.

2.2.1.2 Open the dialog again

If you close the dialog, the **Dashboard** view shows the **Protect your first website** note. The description is **Add a domain to start seeing data in your dashboard.**

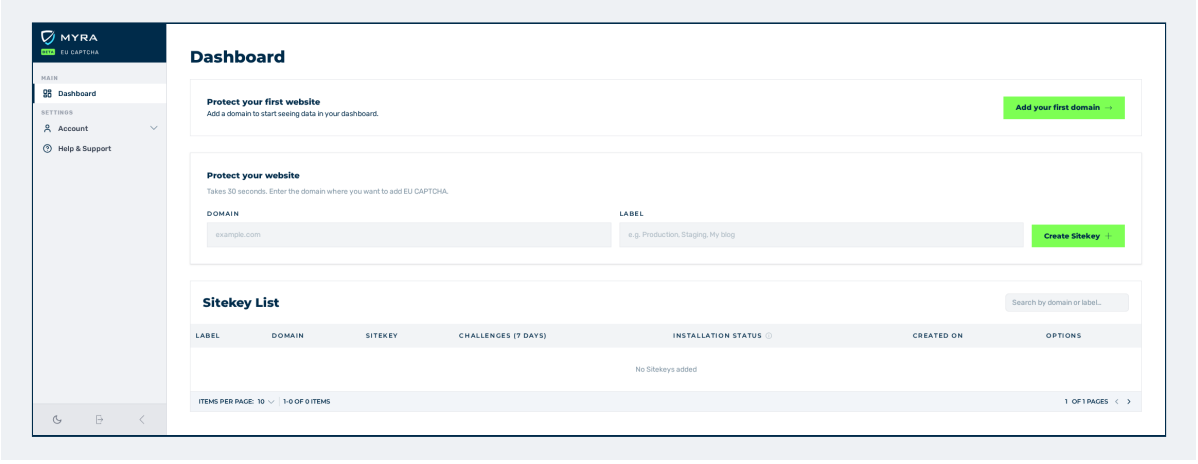


Figure 10: Dashboard view without a sitekey

- ► Click on the **Add your first domain** button.
- ➔ The **Protect your first website** dialog is shown again.

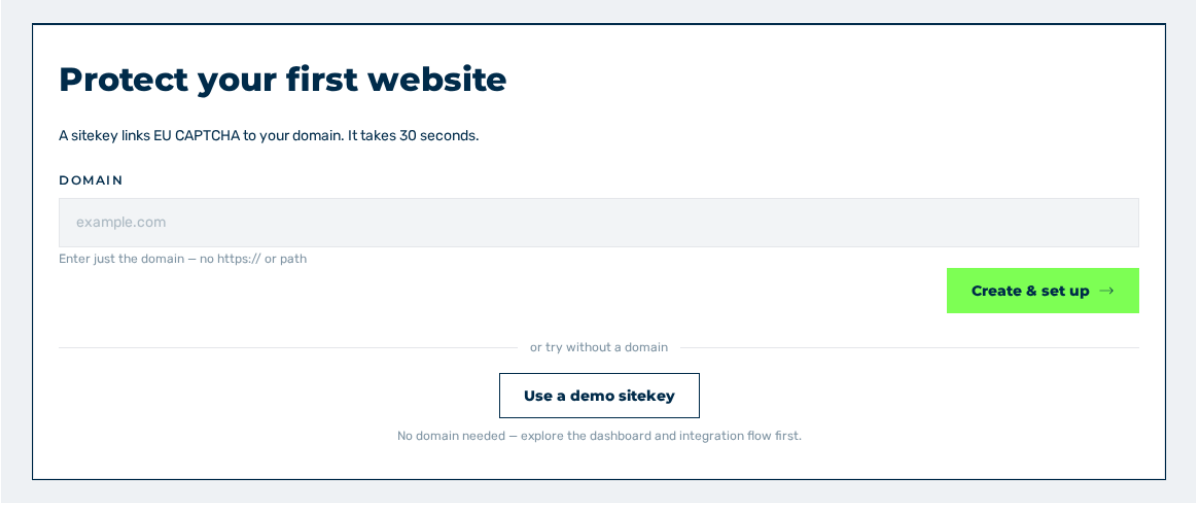


Figure 11: Protect your first website dialog

See **Embedding the widget**.

2.2.2 Embedding the widget

You embed the widget in three steps:

- a script in the head area of the page
- an element at the position of the logo
- a verification of the token on your server

Changes to the DNS are not necessary.

The following requirements must be met:

☑ You created a sitekey. See **Create the first sitekey**.

☑ You know the public sitekey and the secret. Both are shown in the **Details** view of the sitekey. See **Details view**.

☑ You can edit the HTML and the server-side source code of your website.

2.2.2.1 Step 1: Embed the script

Add this element to the `<head>` area of your page:

```
<script src="https://cdn.eu-captcha.eu/verify.js" async defer></script>
```

If your website sends a Content Security Policy, then it must permit the `https://cdn.eu-captcha.eu` and `https://api.eu-captcha.eu` addresses. Without this permission, the browser blocks the widget. See **Content Security Policy**.

2.2.2.2 Step 2: Add the widget element

Add this element at the position where the Myra logo must be shown, usually in the form to protect:

```
<div class="eu-captcha" data-sitekey="a1b2c3d4-0000-0000-0000-000000000000"></
div>
```

Replace the value of `data-sitekey` with your public sitekey.

2.2.2.3 Step 3: Verify the token on the server

Verify the token on your server before you process the form:

```
POST https://api.eu-captcha.eu/v1/verify
Content-Type: application/json

{
  "sitekey": "a1b2c3d4-0000-0000-0000-000000000000",
  "secret": "a1b2c3d4••••",
  "token": "TOKEN_FROM_THE_WIDGET"
}
```

If the response confirms the success, process the transmission. If it does not, refuse the transmission.

The secret is only for your server. Never show it in the HTML or in JavaScript.

2.2.2.4 Platform-specific instructions

For the usual platforms, there are libraries and extensions that do these three steps. See **Integration**.

See **Testing the integration**.

2.2.3 Testing the integration

The **Integration Test** view examines if the widget loads on your page, and if your server verifies the token. The test names the two parts separately. Thus you can see which part is missing.

> **Video:** Verification of the integration in the customer portal – playable in the online help

The following requirements must be met:

☑ You embedded the widget in your website. See **Embedding the widget**.

☑ You opened the protected page a minimum of one time and sent the form.

Proceed as follows to test the integration:

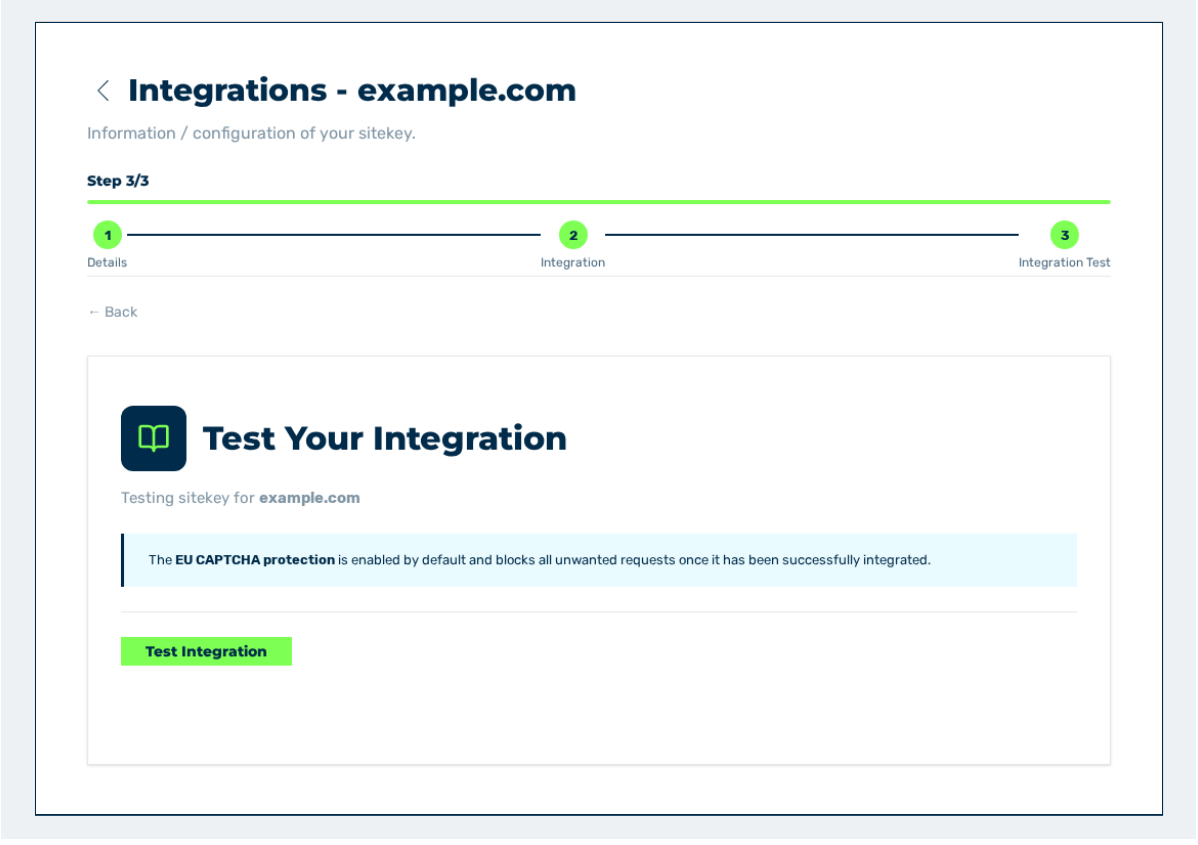


Figure 12: Integration Test view

- ▸ Open the sitekey and go to the **Integration Test** view.
- ▸ Click on the **Test Integration** button.
- ➔ The system shows the result of the test.

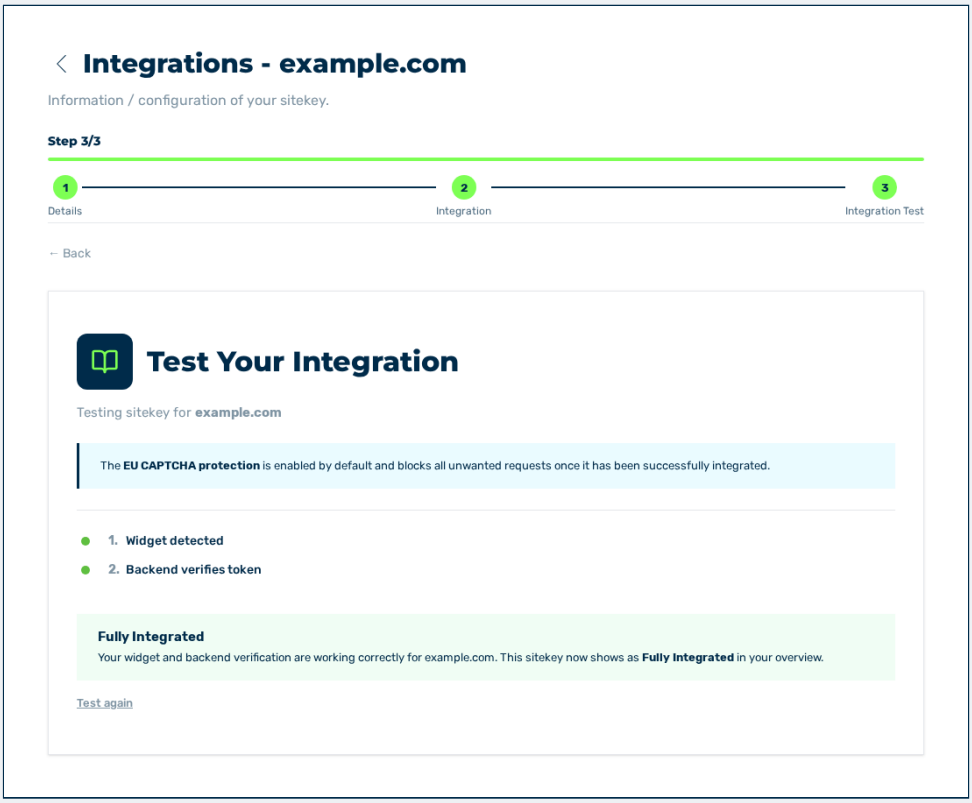


Figure 13: Integration Test view with the result Fully Integrated

The result has one of these values:

Result	Meaning	Measure
Fully Integrated	The widget loads, and your server verifies the token.	None. The integration is complete.
Widget is working but your server didn't verify the token.	The widget loads, but the server-side verification is missing.	Add the call of the `/verify` endpoint. See **Embedding the widget**.
Widget not detected.	The widget does not load.	Examine the script element and the element with the `eu-captcha` class.

2.2.3.1 Functional test in operation

Also examine the sequence from the view of a visitor:

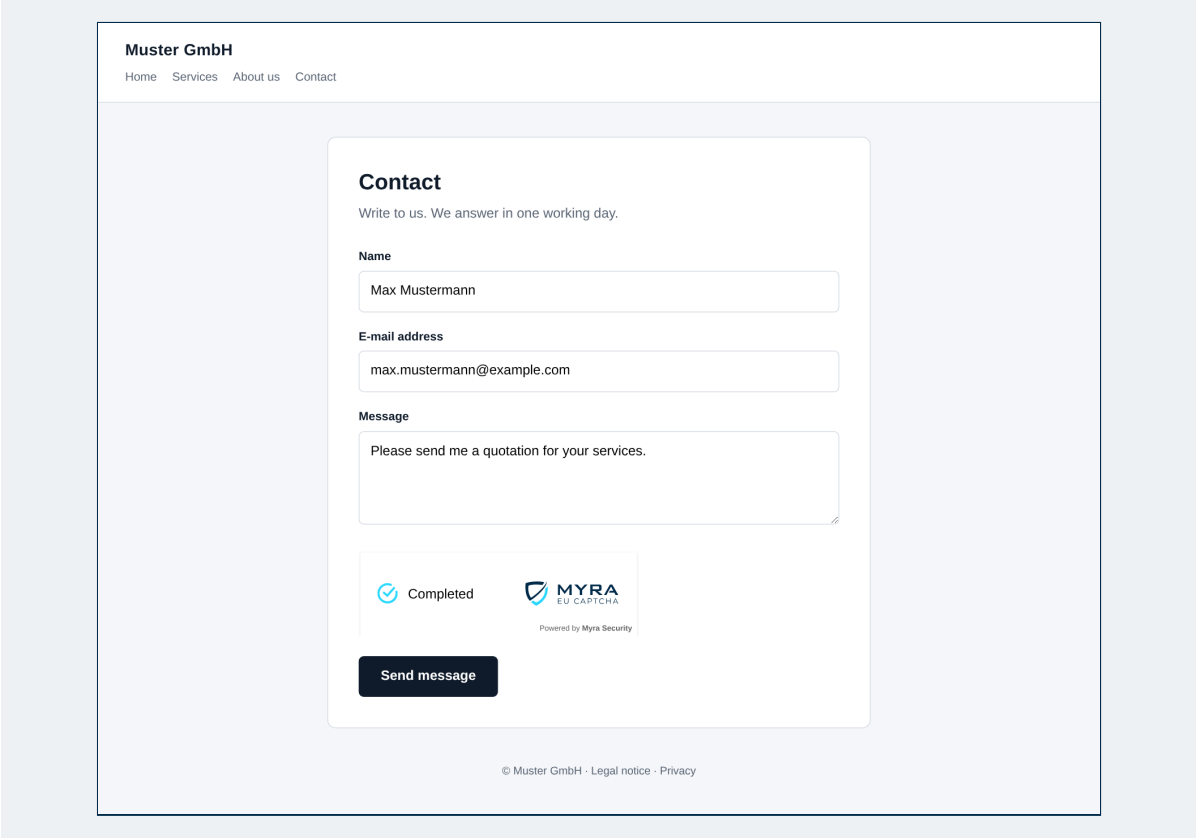


Figure 14: Protected form on the website of the customer

- ► Open the protected form.
- ► Send the form.
  - ↳ The widget solves the challenge in the background. The system processes the form.
- ► In the customer portal, open the **Challenges** view of the sitekey.
- ➔ The list contains an entry for the challenge that was solved.

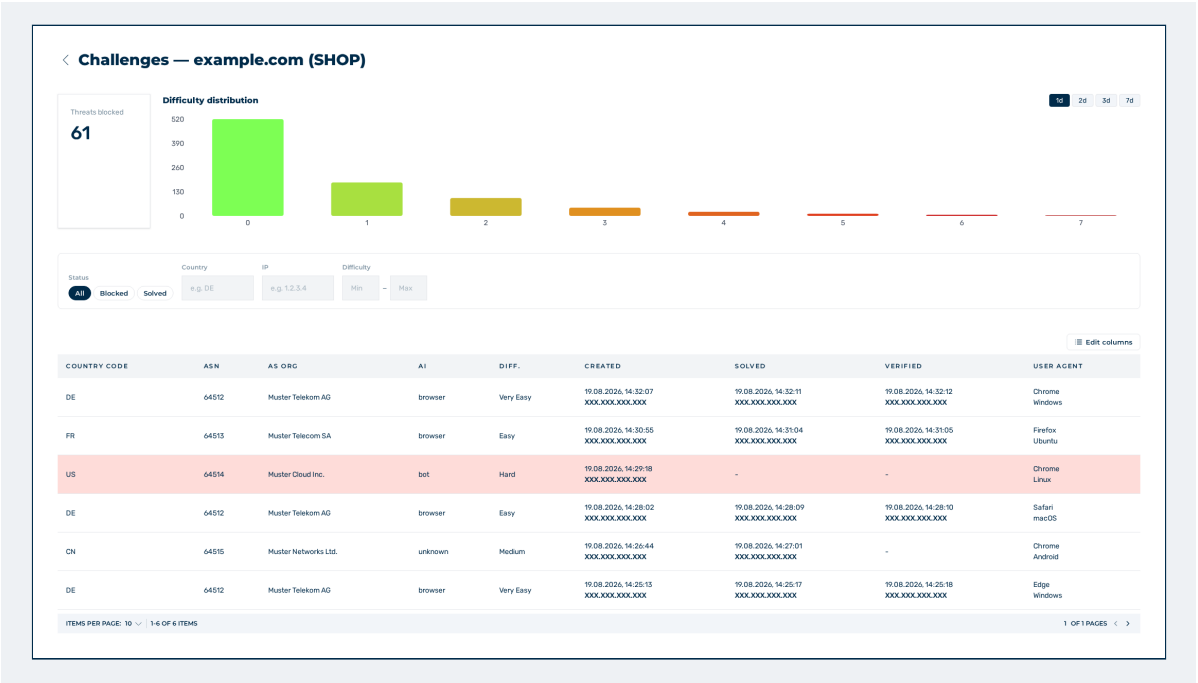


Figure 15: Challenges view

See **Challenges view**.

2.2.3.2 After the setup

The sitekey is ready for use. The following steps are useful, but not mandatory:

Step	Description	Described in
Configure the sitekey	Set the difficulty, the delay, the parallel challenges and the **EU CAPTCHA Protection** switch.	**Configuring a sitekey**
Grant access	Give other accounts read or write access to the sitekey.	**Grant access to a sitekey**
Monitor the traffic	Examine the challenges and their results for each sitekey.	**Challenges view**
Book a plan	Select a plan before the end of the trial period.	**Trial period**

While the **EU CAPTCHA Protection** switch is set to **Off**, the widget sets no challenge and the requests pass unverified. The integration test can then be successful although no protection is in place. Set the switch to **On** before the productive operation.

2.3 Trial period

Every new account starts with a free trial period of 30 days and full access to all functions. No payment details are required to start the trial period.

The trial period is followed by a grace period of seven days. After that, the account rests until a plan has been selected.

The following sections explain the following topics:

- **Commit to a plan during the trial period** describes how the **Captcha Starter** plan is reserved without an immediate debit.

2.3.1 Course of the trial period

The customer portal shows the state of the trial period in a banner at the top of every page. The banner changes with the remaining time:

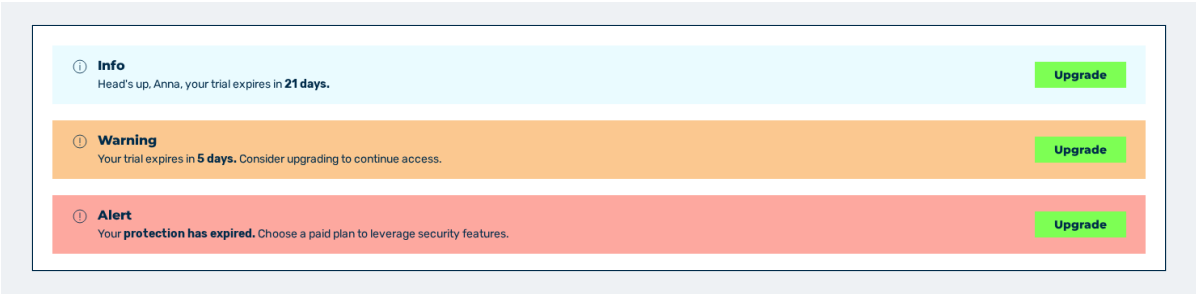


Figure 16: The three banners of the trial period: Info, Warning and Alert

The following states are possible:

State	Banner	Message
More than 7 days remaining	Blue **Info** banner	**Head's up, [first name], your trial expires in [N] days.**
7 days or fewer remaining	Yellow **Warning** banner	**Your trial expires in [N] days. Consider upgrading to continue access.**
Grace period	Red **Alert** banner	**Your protection has expired. Choose a paid plan to leverage security features.**
After the grace period	Lock over the customer portal	**TRIAL ENDED**

Every banner contains the **Upgrade** button. It leads to the **Plans** tab of the **Account** view. See **Plans tab**.

2.3.2 Grace period

The grace period starts with the end of the trial period and lasts seven days. During the grace period, the customer portal remains available.

2.3.3 After the grace period

Once the grace period has expired as well, the **TRIAL ENDED** lock covers the customer portal. The **Choose a plan** button leads to the **Plans** tab.

The state has the following consequences:

Area	Consequence
Site keys	The sitekeys are set to the **trial_expired** state. Embedded widgets remain in place but no longer block any requests. The protection is thus lifted.
Secrets	New secrets can no longer be created. The service answers with the `TRIAL_EXPIRED` error.
Account	The account remains in place and is not deleted.

To restore the protection, select a plan in the **Plans** tab. See **Book a plan** and **Trial period expired**.

During a running trial period, a plan can be booked in advance without an immediate payment. The plan then carries the **Committed** badge and starts after the end of the trial period. See **Commit to a plan during the trial period**.

2.3.4 Commit to a plan during the trial period

During a running trial period, the **Captcha Starter** plan can be committed to in advance without an immediate debit. The customer portal stores the payment method and debits it only at the end of the trial period. The plan becomes active automatically at the same moment.

All other plans as well as every purchase after the end of the trial period run through the ordinary payment process and are debited immediately. See **Book a plan**.

The following requirements must be met:

☑ Your account is in a running trial period. See **Trial period**.

☑ Neither a paid plan nor a commitment exists for your account.

Proceed as follows to commit to the **Captcha Starter** plan:

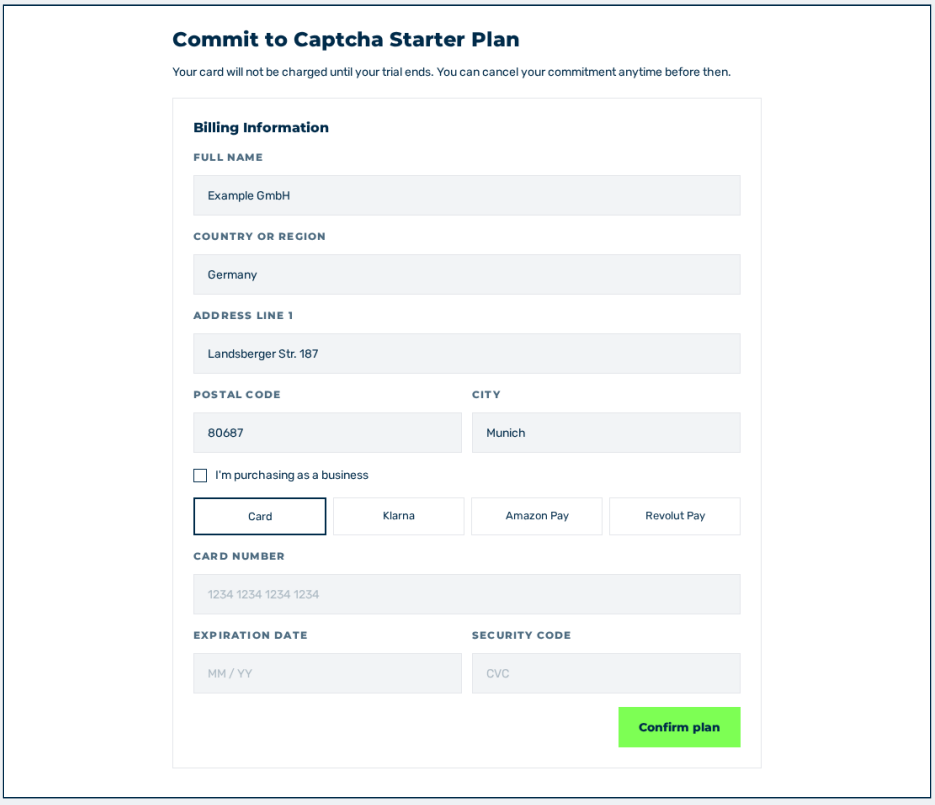


Figure 17: Commit to Captcha Starter Plan view with the Billing Information area

► Open the entry **Account** › **Profile** through the side bar and change to the **Plans** tab.

► Click the **Buy now** button in the **Starter** plan.

↳ The **Commit to Captcha Starter Plan** view is shown with the notice **Your card will not be charged until your trial ends. You can cancel your commitment anytime before then.**

- ▶ Fill in the billing address in the **Billing Information** area.
- ▶ Select a payment method and enter the corresponding data.
- ▶ Click the **Confirm plan** button.
  - ↳ During the processing, the button carries the label **Processing...**.
- ➔ The **You're all set!** view is shown and names the date of the first debit.

The following payment methods are available:

Payment method	Description
Card	Credit or debit card.
Klarna	Payment through Klarna.
Amazon Pay	Payment through Amazon Pay.
Revolut Pay	Payment through Revolut Pay.

As a company, select the **I'm purchasing as a business** check box and enter your VAT identification number in the **VAT Number (optional)** field. Enter the name of the company in the **Full name** field. This name appears on the invoice.

2.3.4.1 Withdraw a commitment

The commitment can be withdrawn at any time until the end of the trial period. Until then, no payment has taken place.

If a plan is committed to, the mark above the offer in the **Plans** tab gives the start date, for example **Captcha Starter Plan committed – activates on 13.06.2026**. The **Subscription History** tab carries the entry with the **Committed** mark and the **Cancel commitment** button.

How a commitment is withdrawn is described under Cancel a subscription.

After the withdrawal, the trial period continues until its original end date. The dialog names this date.

3. Views

3.1 General

3.1.1 Views

The customer portal of Myra EU CAPTCHA has a sidebar and a work area. The sidebar goes to the views, the work area shows the selected view.

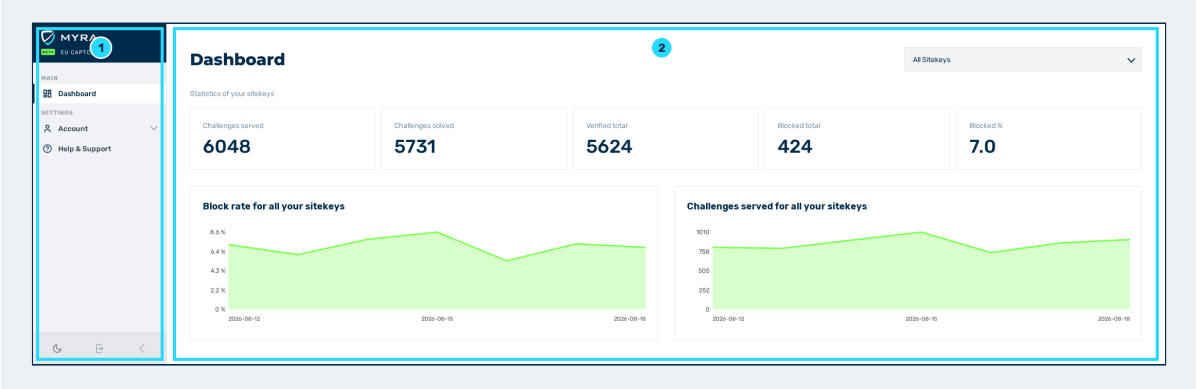


Figure 18: Structure of the customer portal

3.1.1.1 Sidebar

The sidebar has areas. These entries are available:

Area	Entry	Destination
MAIN	**Dashboard**	**Dashboard** view with key figures, graphs, and the list of the sitekeys
SETTINGS	**Account** › **Profile**	**Account** view with the tabs for profile, logins, plans, subscriptions, and deletion
–	**Help & Support**	**Help** view with the links to documentation and support

The **Account** entry can be opened. A click on the entry shows the subordinate entries.

At the top of the sidebar, the Myra logo is shown with the **BETA** mark. At the bottom, these buttons are available:

- **Switch to dark mode** and **Switch to light mode** change between the light and the dark appearance.
- **Logout** logs you out of the customer portal.
- **Collapse sidebar** and **Expand sidebar** make the sidebar smaller and larger. When it is smaller, the sidebar shows only the icons. The name of the entry is shown as a tooltip.

The **ADMINISTRATION** area with the **Admin Control** entry is only for the administration by Myra. This online help does not describe it.

3.1.1.2 Overview of the views

These views are available:

View	Contents
Dashboard view	Key figures and graphs about the challenges, and below them the list of the sitekeys
Sitekey List area	Table of all sitekeys with domain, status, and options; part of the **Dashboard** view
Sitekey view	Details, configuration, integration, integration test, permissions, and challenges of one sitekey
Account view	Profile, logins, plans, subscriptions, and deletion of the account
Organization view	Members of the organization and parameters for SAML and SSO
Help view	Links to documentation and support

3.2 Dashboard

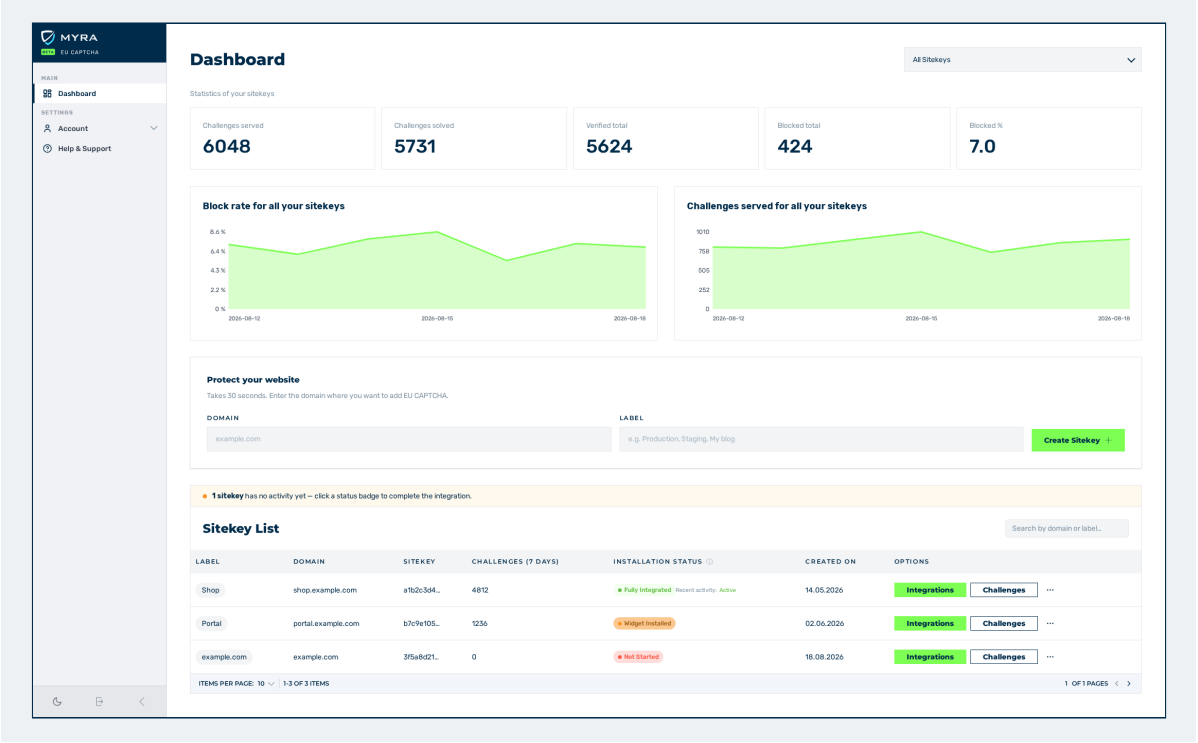


Figure 19: Dashboard view

The **Dashboard** view is the start view of the customer portal. It shows the key figures and the trend of the challenges. Below them, the view shows the list of the sitekeys.

> **Video:** The Dashboard view and its metrics — playable in the online help

3.2.1 Selection of the sitekeys

Above the row of the key figures, a drop-down list with the sitekeys is shown. The **All Sitekeys** entry collects all sitekeys. If you select one sitekey, the key figures and the graphs refer only to this sitekey.

A search field limits the list. A button resets the search.

3.2.2 Key figures

The **Statistics of your sitekeys** row contains these key figures:

Key figure	Meaning
Challenges served	Count of the supplied challenges
Challenges solved	Count of the solved challenges
Verified total	Count of the tokens that were confirmed on the server
Blocked total	Count of the blocked requests
Blocked %	Part of the blocked requests in all challenges

3.2.3 Graphs

Two trend graphs appear below the key figures:

Graph	Contents
Block rate for …	Trend of the block rate
Challenges served for …	Trend of the supplied challenges

The title names the label of the selected sitekey. If you selected the **All Sitekeys** entry, **all your sitekeys** is shown there.

While no data is available, the graph shows the **No data yet. Data appears within minutes of your first verified CAPTCHA challenge.** note.

3.2.4 Notes for the start

Depending on the state of the setup, the dashboard shows one of these notes:

Note	Meaning	Button
Protect your first website	No sitekey is available yet.	**Add your first domain**
Complete your integration	A sitekey is available, but no challenge was supplied yet.	**Go to integration guide**

After you create a sitekey, the **Sitekey created** message is also shown.

3.2.5 List of the sitekeys

The list of the sitekeys is shown below the graphs. See **Sitekey List area**.

3.2.6 Sitekey List area

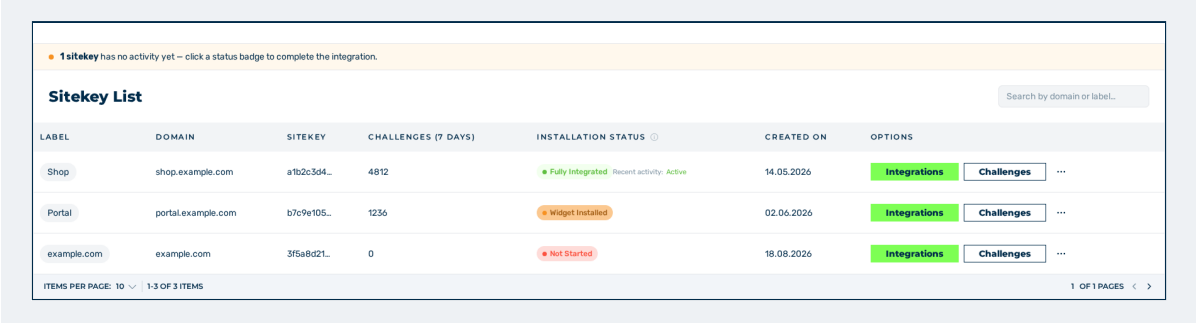


Figure 20: Sitekey List area

The **Sitekey List** area lists all sitekeys that you have access to. From the list, you open a sitekey, create a new one, and delete the available entries.

> **Video:** The list of the sitekeys and its columns — playable in the online help

3.2.6.1 Columns

The table contains the following columns:

Column	Contents
Label	Label of the sitekey. Without a label, the domain is shown.
Domain	Protected domain of the sitekey
Sitekey	First eight characters of the public key
Challenges (7 days)	Count of the challenges of the last seven days
Installation Status	State of the integration
Created on	Date of the creation
Options	Actions for the entry

A search field with the **Search by domain or label...** placeholder decreases the list. While no sitekey is available, the table shows the **No Sitekeys added** note.

Above the table, a note is shown as soon as one sitekey or more supplied no challenge yet: **N sitekeys have no activity yet — click a status badge to complete the integration.** In the note, `N` is the count of the applicable sitekeys.

Above the new sitekey, the **Click here for integration details** bubble is shown. The bubble is removed as soon as you click on the **Integrations** button.

For a sitekey with the protection switched off, the **Training mode** badge is also shown in the **Label** column.

For a sitekey that a different account shared, the **Read** or **Write** badge follows in the **Label** column.

3.2.6.2 Installation status

The **Installation Status** column has one of these values:

Value	Meaning
Not Started	No activity of the widget was detected yet.
Widget Installed	The widget loads, but the server-side verification is still missing.
Fully Integrated	The widget loads, and your server verifies the token.
Active	A server-side verification occurred in the last 24 hours.
Inactive	No server-side verification occurred in the last 24 hours.

A tooltip tells about the applicable value. The **Active** and **Inactive** values are shown after the **Recent activity:** label and occur only for the **Fully Integrated** value.

3.2.6.3 Actions

The **Options** column contains the following actions:

Action	Effect
Integrations	Opens the sitekey with the **Details** view.
Challenges	Opens the **Challenges** view of the sitekey.
CONFIG	Opens the **Configuration Settings** dialog.
PERMISSIONS	Opens the **Permissions** view of the sitekey.
DELETE	Opens the **Delete Sitekey?** dialog with the **Cancel** and **Delete** buttons.

The **CONFIG**, **PERMISSIONS**, and **DELETE** actions are in the menu behind the button with the three dots. The **PERMISSIONS** and **DELETE** actions occur only for the owner of the sitekey.

See **Create a sitekey** and **Delete a sitekey**.

3.3 Sitekey

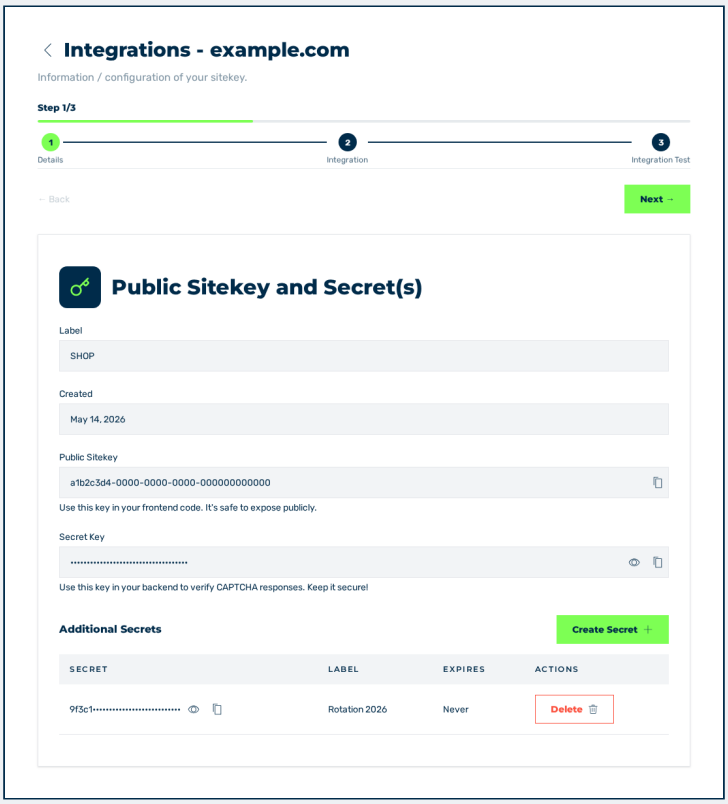


Figure 21: Sitekey view with the details of the sitekey

The **Sitekey** view collects all data and settings of one sitekey. You open the view from the list of the sitekeys. See **Sitekey List view**.

Above the content, the **Sitekey -** title is shown. In the three steps of the wizard, the title is **Integrations -** . Below the title, this description is shown: **Information / configuration of your sitekey.**

3.3.1 Views

The sitekey has these views:

View	Contents
Details	Public sitekey, secrets, and master data
Integration	Wizard with the instructions for the selected platform
Integration Test	Test of the widget and of the server-side verification
Configuration	Protection, difficulty, delay, and parallel challenges
Permissions	Access of other accounts to the sitekey
Challenges	Log of the supplied challenges

3.3.2 Wizard sequence

For the initial setup, a wizard goes through three of the views:

Step	View
1	**Details**
2	**Integration**
3	**Integration Test**

Above the steps, the wizard shows the **Step ⅓** to **Step 3/3** data and a progress bar. Below, the three steps are shown with their number and their label.

Below the steps and above the content of the view, the **Next** → button goes to the next step. The ← **Back** button goes to the previous step. In the first step, ← **Back** is locked. In the last step, **Next** → is not shown.

The **Configuration** and **Permissions** views are not part of the wizard and do not show the steps.

3.3.3 Access tier

Your access tier for the sitekey controls if you can change settings. With the **Read** tier, the input fields are locked. A tooltip names the **Read-only access** cause. See **Permissions view**.

3.3.4 Wizard

3.3.4.1 Details

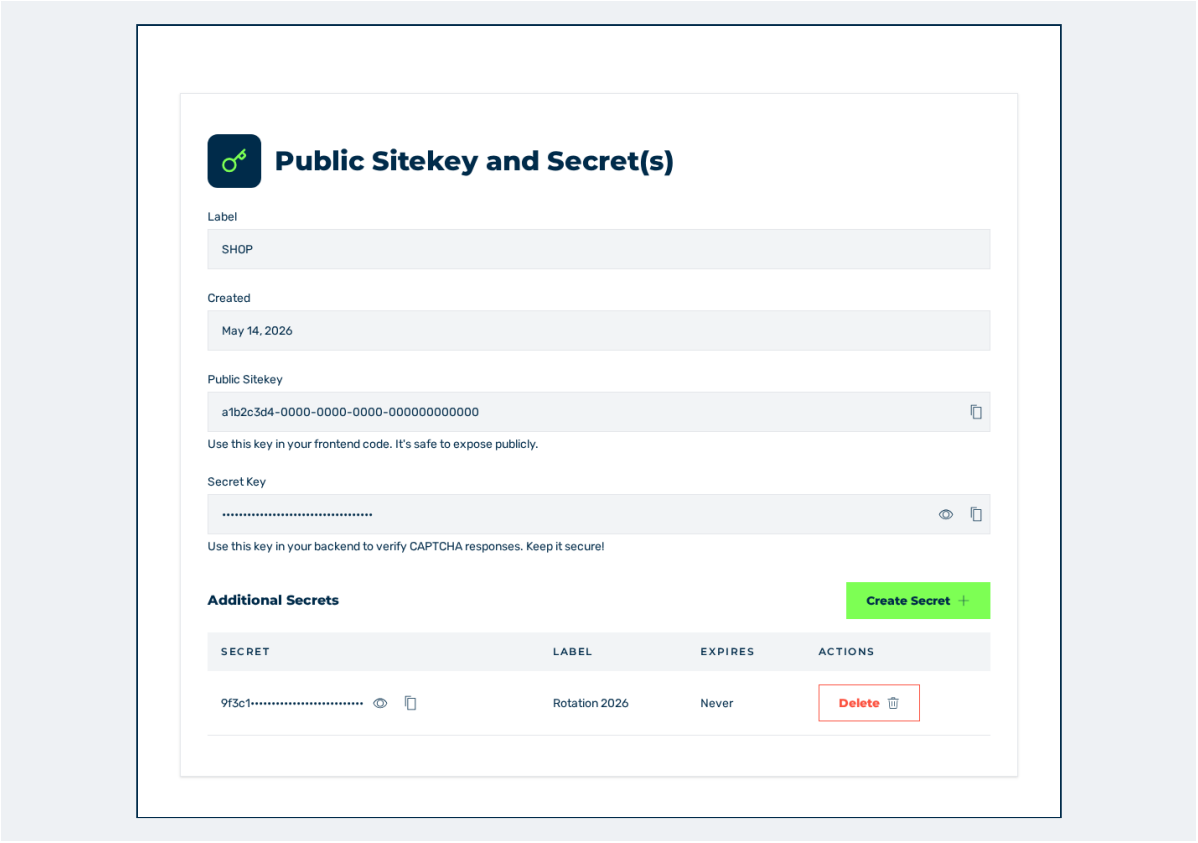


Figure 22: Details view

Below the **Public Sitekey and Secret(s)** title, the **Details** view shows the keys of the sitekey and its master data.

3.3.4.1.1 Fields

The view contains these fields:

Field	Contents
Label	Label of the sitekey
Created	Date of the creation
Public Sitekey	Public key for the widget element
Secret Key	Secret key for the server-side verification

The public sitekey is a UUID. It is used in the `data-sitekey` attribute of the widget element and is visible in the source code of your page.

The secret is shown hidden. It is only for your server.

> Never show the secret in the HTML or in JavaScript. With the secret, each token can be made valid.

3.3.4.1.2 Copy a key

Adjacent to each key, a button for the copy operation is available. The tooltip is **Copy key to clipboard.** After the copy operation, it changes to **Key copied to clipboard.**

Adjacent to the secret key, a related button shows the value. The tooltip is **Show secret**. When the value is shown, the tooltip is **Hide secret**.

3.3.4.1.3 The Additional Secrets area

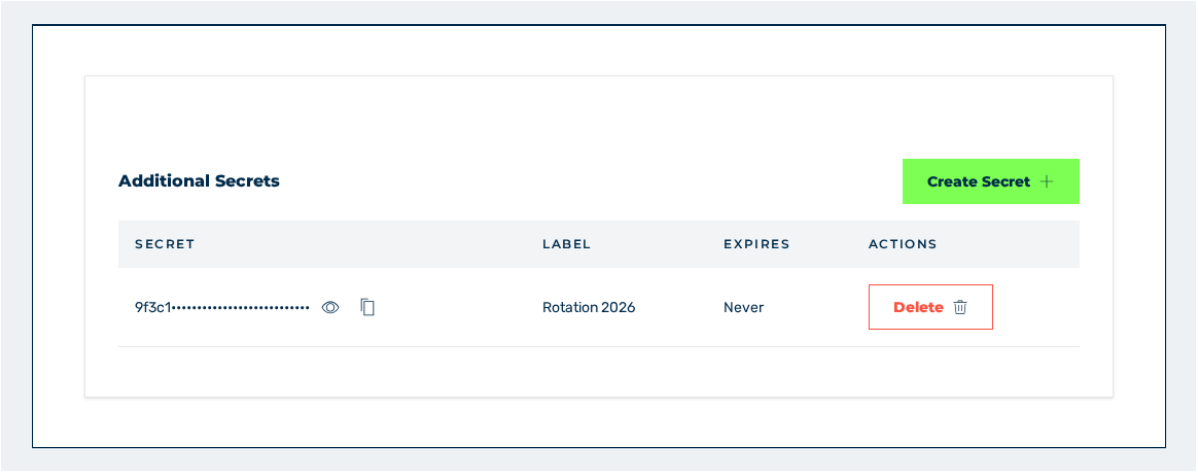


Figure 23: Additional Secrets area

The **Additional Secrets** area controls the other secrets of this sitekey. With the **Read** access tier, the view does not show the area.

Other secrets let you replace the secret during operation. You create a second secret, change your servers to it one by one, and then delete the old secret.

The table contains these columns:

Column	Contents
Secret	Secret, hidden except for the first five characters

Column	Contents
Label	Label of the secret. Without a label, a hyphen is shown.
Expires	Expiry date. Without an expiry date, the **Never** value is shown.
Actions	The **Delete** button

While no other secret is available, the table shows the **No additional secrets** note.

In the **Secret** column, a button shows and hides the value. The tooltip is **Show secret** and **Hide secret**. A related button copies the value. Its tooltip is **Copy to clipboard**. After the copy operation, it changes to **Copied!**

3.3.4.1.4 The Create Secret dialog

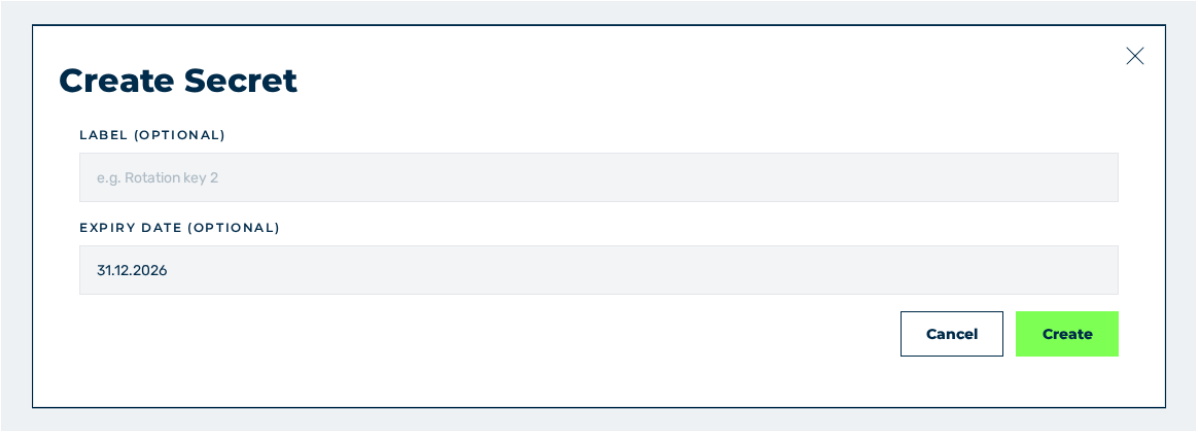


Figure 24: Create Secret dialog

The **Create Secret** button opens the **Create Secret** dialog. The dialog contains these options:

Option	Description
The **Label (optional)** field	Label of the secret. The placeholder is `e.g. Rotation key 2`.
The **Expiry date (optional)** field	Expiry date of the secret
The **Cancel** button	Cancels the operation
The **Create** button	Creates the secret

3.3.4.1.5 The Delete Secret dialog

Figure 25: Delete Secret dialog

The **Delete** button opens the **Delete Secret** dialog with this question: **Are you sure you want to delete the secret ...? This action cannot be undone.** The **Delete** button deletes the secret. The **Cancel** button cancels the operation.

After the delete operation, the system shows the **Secret deleted** message with the **The secret has been removed** text.

See Embedding the widget.

3.3.4.2 Integration

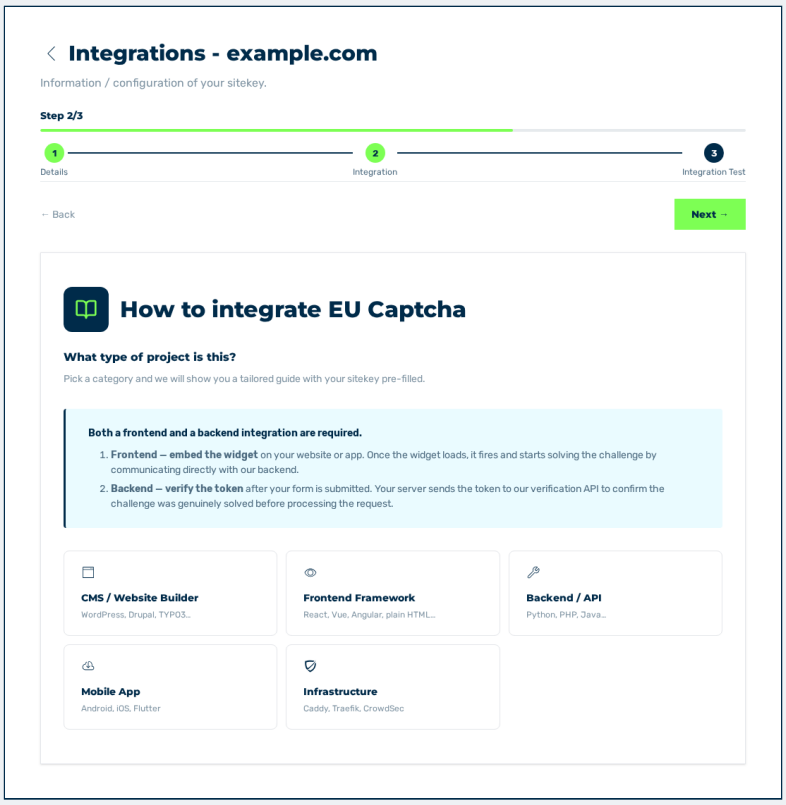


Figure 26: Integration view

Below the **How to integrate EU Captcha** title, the **Integration** view goes through the embedding. You select the platform of your website. The wizard shows the applicable guide with the sitekey of this entry.

> **Video:** The Integration view and the selection of the platform — playable in the online help

3.3.4.2.1 Categories

The wizard gives these categories:

Category	Worked examples
CMS / Website Builder	WordPress, Drupal, TYPO3
Frontend Framework	React, Vue, Angular, plain HTML
Backend / API	Python, PHP, Java
Mobile App	Android, iOS, Flutter
Infrastructure	Caddy, Traefik, CrowdSec

After you select a category, the wizard shows the technologies of this category. For platforms without a related library, an entry for the embedding with HTML and the REST API is available at the end of each list.

3.3.4.2.2 Guide

For the selected technology, the wizard shows the steps in areas that can be opened. The guide contains the public sitekey of the open entry. Thus you can use the examples without changes.

Related to the technology, the wizard also tells you that the other part of the integration is still missing:

- For frontend technologies, the note about the server-side verification.
- For backend technologies, the note about the embedding of the widget.

> The protection starts only when the two parts are set up: the widget in the page and the verification of the token on your server.

The full instructions of this online help are given in **Integration**.

3.3.4.3 Integration Test

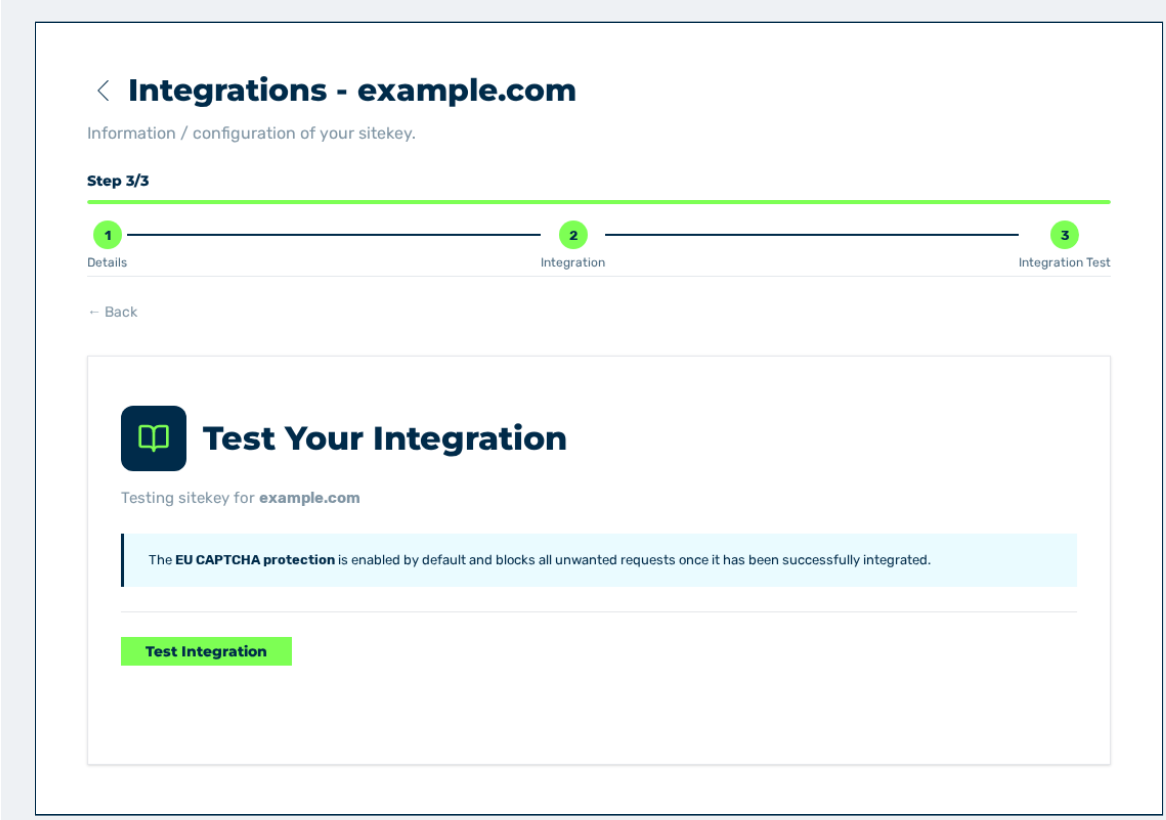


Figure 27: Integration Test view

Below the **Test Your Integration** title, the **Integration Test** view examines if the widget loads on your page, and if your server verifies the token.

3.3.4.3.1 Sequence

The **Test Integration** button starts the test. The result has one of these values:

Result	Meaning
Fully Integrated	The widget loads, and the server-side verification occurred.
Widget is working but your server didn't verify the token.	The widget loads, but the server-side verification is missing.
Widget not detected.	The widget was not detected.

The test examines the activity of the sitekey. Thus, open the protected page and send the form before the test.

See **Testing the integration**.

3.3.5 Configuration

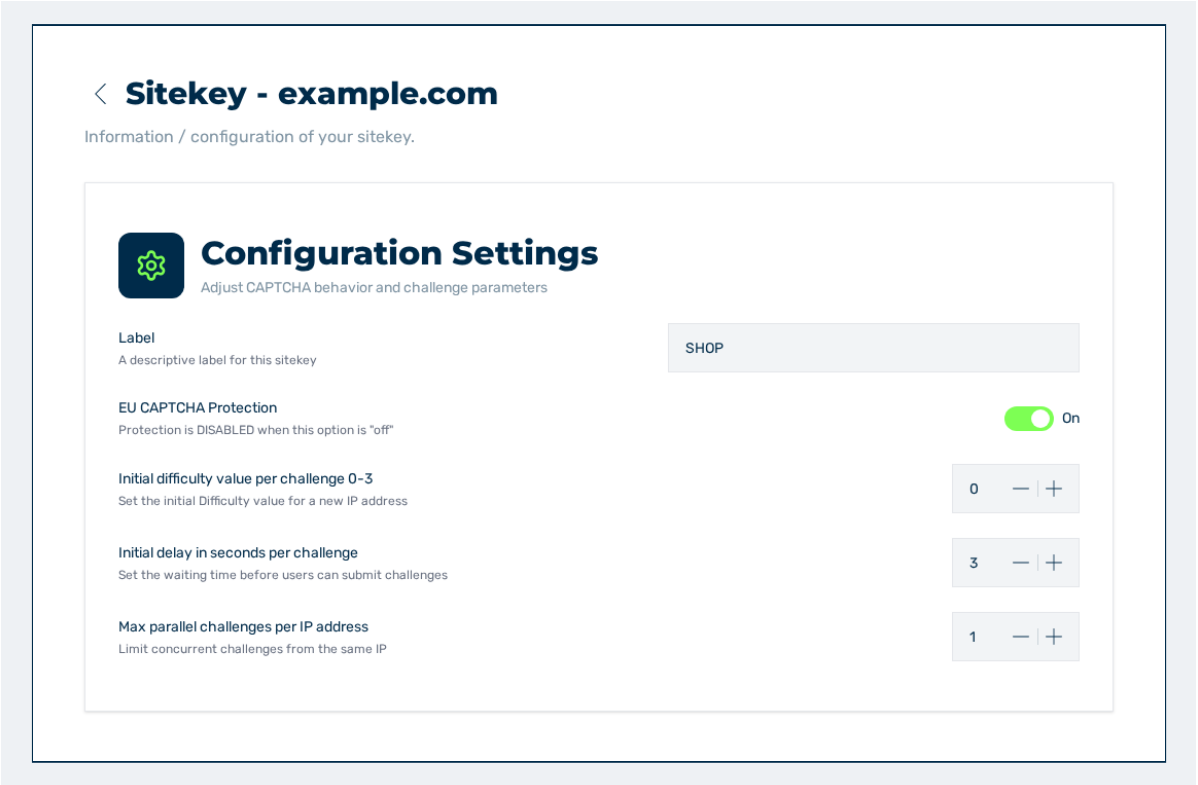


Figure 28: Configuration view

The **Configuration** view controls the behaviour of the protection for this sitekey. The title is **Configuration Settings**, the description is **Adjust CAPTCHA behavior and challenge parameters**.

> **Video:** The Configuration view and its settings — playable in the online help

3.3.5.1 Settings

The view contains these settings:

Setting	Description in the interface	Value range	Default
Label	A descriptive label for this sitekey	Text	empty
EU CAPTCHA Protection	Protection is DISABLED when this option is off	**On**, **Off**	**On**

Setting	Description in the interface	Value range	Default
Initial difficulty value per challenge 0-3	Set the initial Difficulty value for a new IP address	0 to 3	0
Initial delay in seconds per challenge	Set the waiting time before users can submit challenges	3 to 30	3
Max parallel challenges per IP address	Limit concurrent challenges from the same IP	1 to 10	1

If a value is out of the range, the field shows a note about the upper limit or the lower limit.

3.3.5.2 Locks

Two conditions lock the settings:

Condition	Tooltip	Effect
Read access tier	**Read-only access**	All settings are locked.
free plan	**Upgrade to unlock**	The three numerical values are locked. The **Label** and **EU CAPTCHA Protection** options stay changeable. The **Upgrade** button goes to the selection of the plans.

See Configure a sitekey and Plans tab.

3.3.6 Permissions

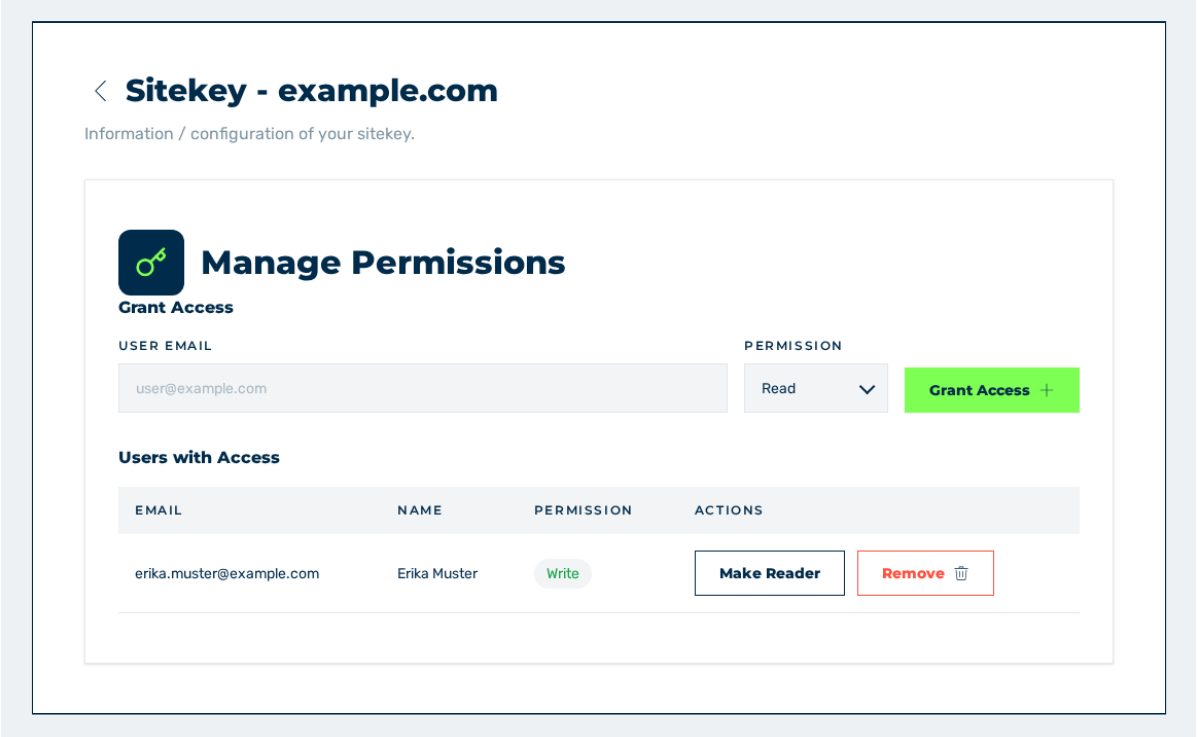


Figure 29: Permissions view

Below the **Manage Permissions** title, the **Permissions** view controls which other accounts get access to this sitekey. The access is applicable to one sitekey.

> **Video:** The Permissions view and the assignment of access rights — playable in the online help

3.3.6.1 Grant Access area

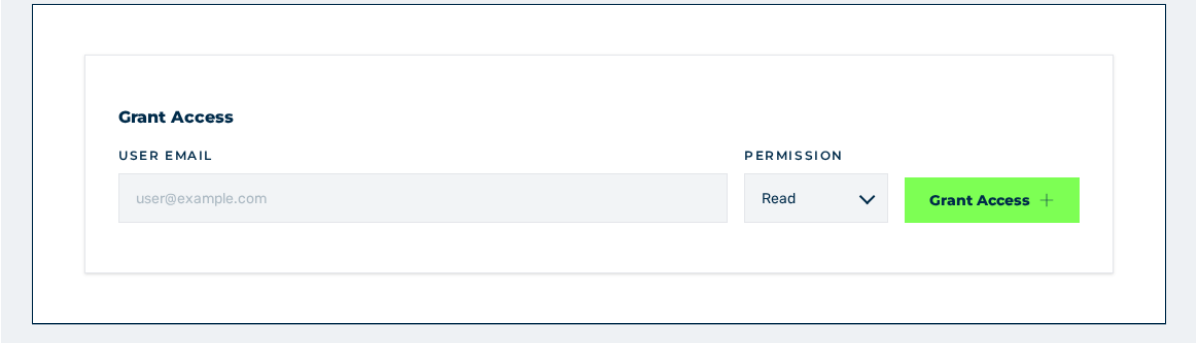


Figure 30: Grant Access area

The **Grant Access** area gives the access. It contains these elements:

Element	Description
User Email field	Account email address, placeholder `user@example.com`
Permission drop-down list	**Read** or **Write** access tier
Grant Access button	Gives the access

3.3.6.2 Users with Access area

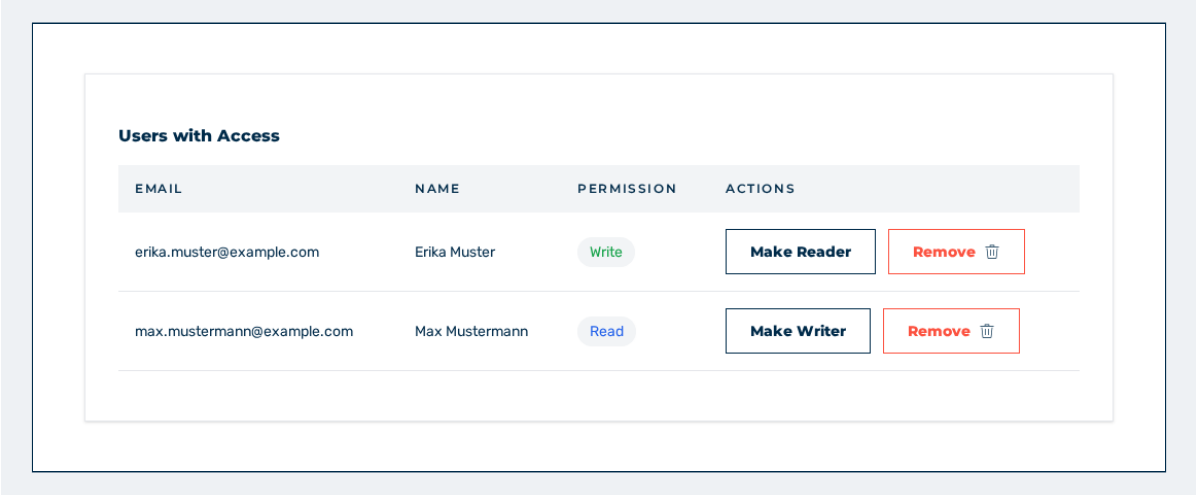


Figure 31: Users with Access area

The **Users with Access** area lists the accounts with permission. The table contains the following columns:

Column	Contents
Email	Account email address
Name	Name of the account
Permission	**Read** or **Write** access tier
Actions	The **Make Writer** and **Make Reader** buttons, and the **Remove** button

While no other account has permission, the table shows the **No users have been granted access** note.

3.3.6.3 Access tiers

These access tiers are available:

Tier	Rights
Read	The sitekey and its settings are visible. Changes are locked.
Write	The sitekey and its settings are visible and can be changed.

3.3.6.4 Dialogs

Each action opens a dialog for the confirmation:

Dialog	Question
Remove Permission	**Are you sure you want to remove access for …?**
Change to Write and **Change to Read**	**Change permission for … from … to …?**

The two dialogs contain the **Cancel** and **Confirm** buttons.

See Grant access to a sitekey.

3.3.7 Challenges

Figure 32: Challenges view

The **Challenges** view logs the challenges of this sitekey. From the log, you can see which requests were blocked and which were permitted. The heading is **Challenges – ()**.

> **Video:** The log of the challenges and its filters — playable in the online help

3.3.7.1 Columns

The table contains the following columns:

Column	Contents
Country Code	Country code of the source
ASN	Autonomous system of the IP address
As ORG	Operator of the autonomous system
AI	Classification of the request: **browser**, **bot**, or **unknown**
Diff.	Difficulty of the challenge
Created	Time and IP address of the challenge

Column	Contents
Solved	Time and IP address of the solution
Verified	Time and IP address of the server-side verification
User Agent	Browser and operating system of the request

A blocked challenge has a coloured row background. Without a value, the cell shows a hyphen.

While no challenge is available, the table shows the **No challenges yet** note.

The **Edit columns** button above the table opens a list with one check box for each column. Use it to hide and to show the applicable columns.

3.3.7.2 Difficulties

The **Diff.** column shows the difficulty as text:

Value	Display
0	**Very Easy**
1 and 2	**Easy**
3 and 4	**Medium**
5 and 6	**Hard**
7	**Very Hard**

3.3.7.3 Filter

Above the table, these filters are available:

Filter	Operation
Status	The **All**, **Blocked**, and **Solved** buttons
Country	Field for the country code, with the `e.g. DE` placeholder
IP	Field for the IP address, with the `e.g. 1.2.3.4` placeholder
Difficulty	Two fields, **Min** and **Max**, for the range 0 to 7

As soon as one filter is set, the **Clear filters** button is shown. It resets all the filters.

The filters apply to the loaded page of the table, not to all the records.

3.3.7.4 Distribution of the difficulty

Above the filters, the **Difficulty distribution** graph is shown. Its bars show how many challenges apply to the difficulties 0 to 7. The **1d**, **2d**, **3d**, and **7d** buttons select the time range.

To the left, the **Threats blocked** key figure names the count of the blocked requests in the selected time range.

See **How EU CAPTCHA works**.

3.4 Account

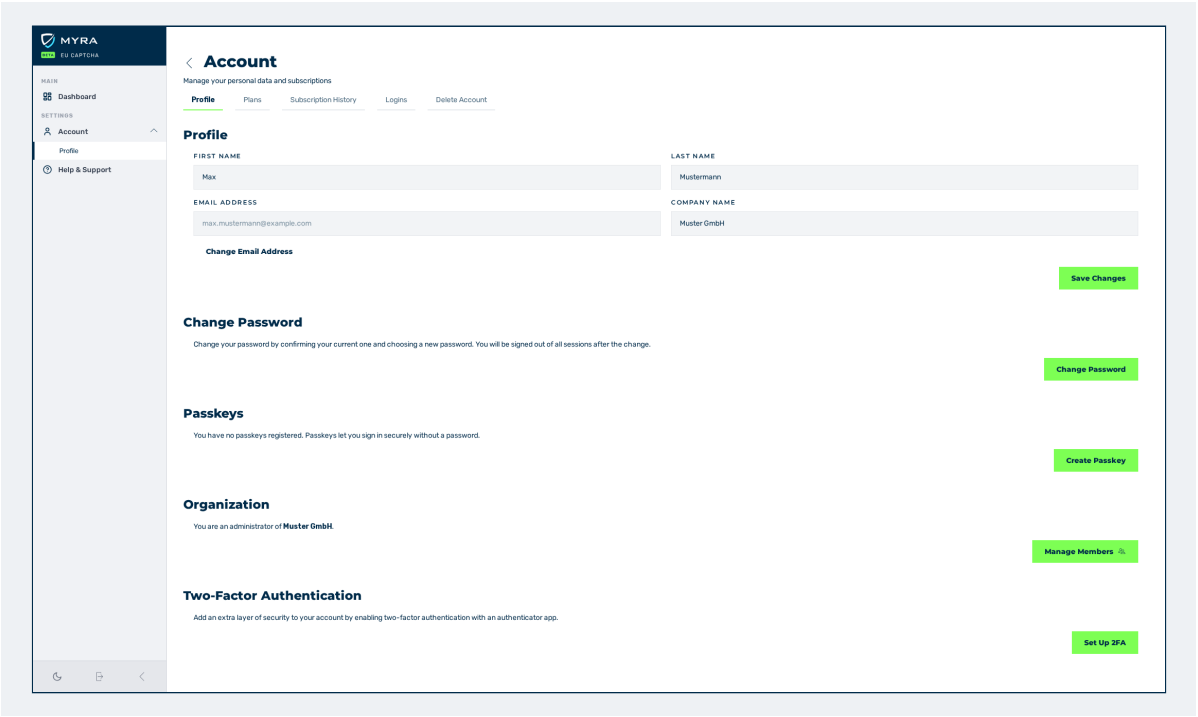


Figure 33: Account view

The **Account** view collects your personal data and your subscription. The description below the title is **Manage your personal data and subscriptions**. To open the view, use the **Account** › **Profile** entry of the sidebar.

3.4.1 Tabs

The view has these tabs:

Tab	Contents
Profile	Personal data, password, e-mail address, passkeys, and two-factor authentication
Plans	Available plans and their booking
Subscription History	History of the subscriptions and their status
Logins	Log of the logins to your account
Delete Account	Permanent deletion of the account

3.4.2 Profile tab

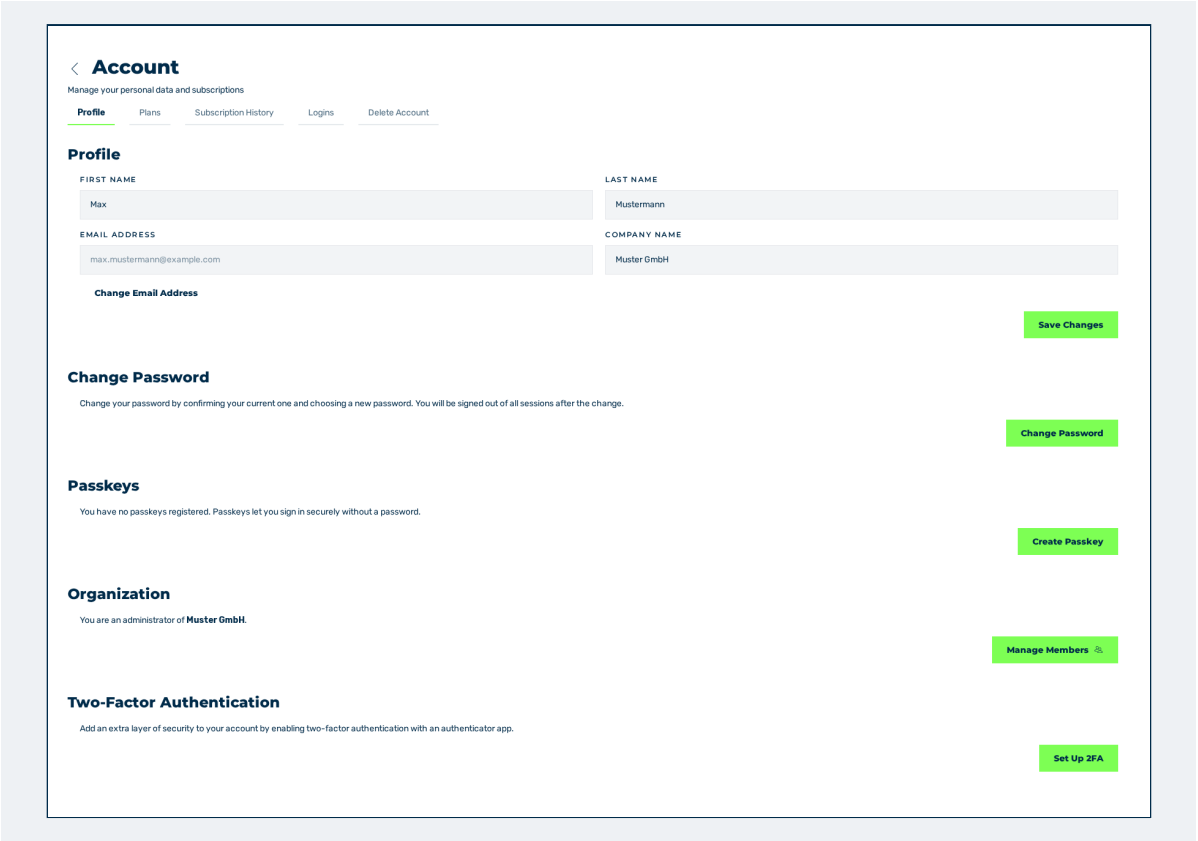


Figure 34: Profile tab

The **Profile** tab contains your personal data and the settings for the security of your account.

> **Video:** The Profile tab and its settings — playable in the online help

3.4.2.1 Profile area

These fields are available:

Field	Contents
First name	First name
Last name	Last name
Email address	E-mail address of the account. The field is disabled.
Company name	Name of the company

The **Save Changes** button applies the changes. The **Change Email Address** button opens the dialog with the same name.

3.4.2.2 Change Password area

The area describes the change of the password. The **Change Password** button opens the dialog with the same name. After the change, the system signs you out of all sessions.

3.4.2.3 Passkeys area

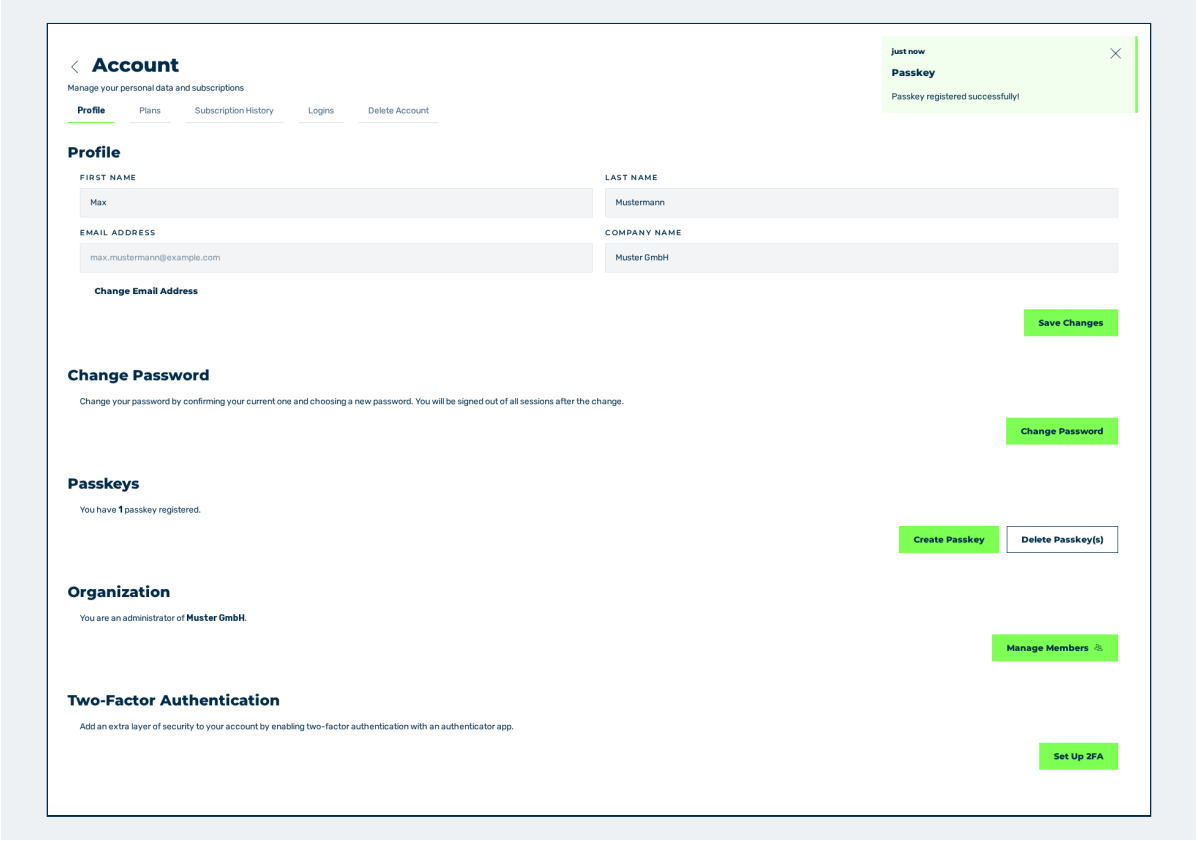


Figure 35: Profile tab with a passkey that was registered

The area gives the count of the passkeys that are registered. These buttons are available:

Button	Effect
Create Passkey	Registers a passkey for the login without a password.
Delete Passkey(s)	Deletes the passkeys that are registered. The button is shown only after the first passkey.

3.4.2.4 Organization area

The area is shown only when you are an Org Admin of an organisation. It gives the name of the organisation. The **Manage Members** button opens the management of the members.

3.4.2.5 Two-Factor Authentication area

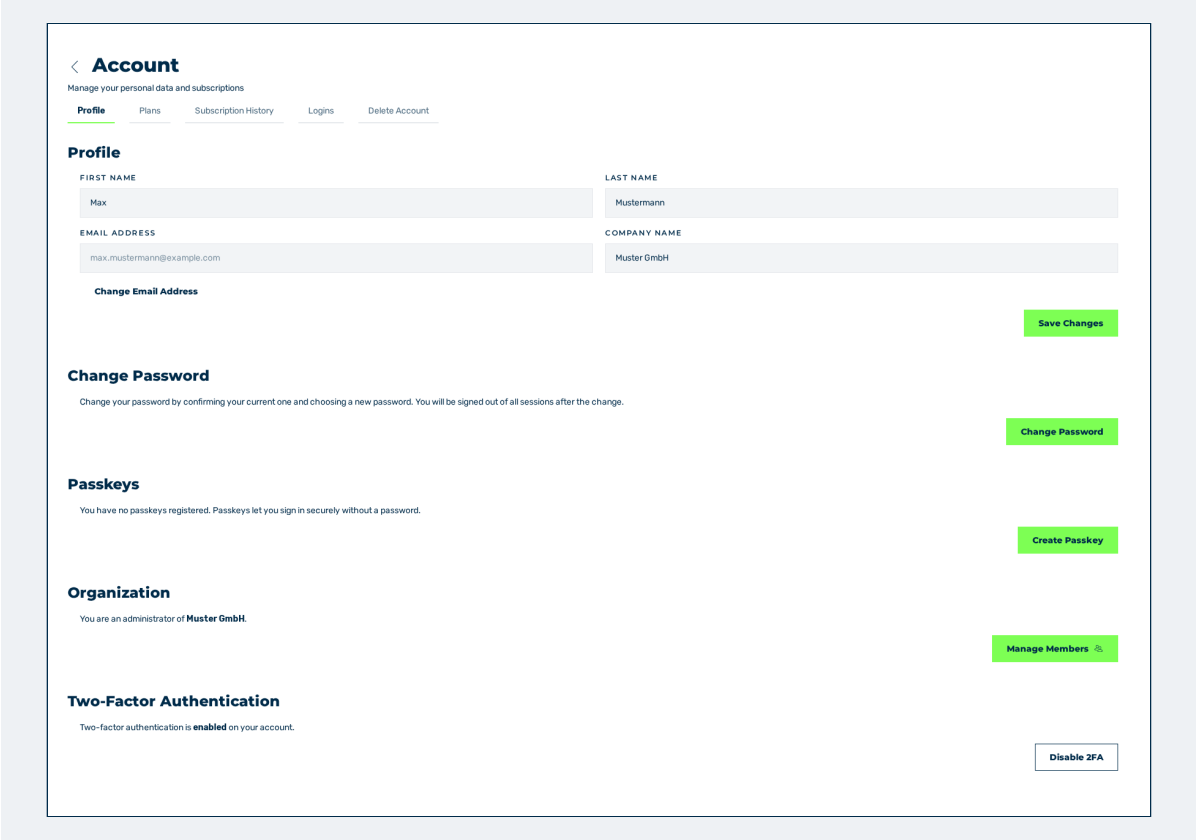


Figure 36: Profile tab with the two-factor authentication activated

The area gives the condition of the two-factor authentication. These buttons are available:

Button	Effect
Set Up 2FA	Starts the set-up of the two-factor authentication.
Disable 2FA	Opens the **Disable Two-Factor Authentication** dialog.

3.4.2.6 Messages

The portal confirms the operations with these messages:

Message	Cause
Password changed – Your password was changed. Please sign in again.	The password was changed.
Email change requested – Check your current inbox to confirm the change.	The change of the e-mail address was requested.
Passkey – Passkey registered successfully!	A passkey was registered.
Passkey – All passkeys have been deleted.	All passkeys were deleted.
2FA – Two-factor authentication has been disabled.	The two-factor authentication was deactivated.

See **Change the password**.

See **Change the e-mail address**.

See **Set up two-factor authentication**.

3.4.3 Plans tab

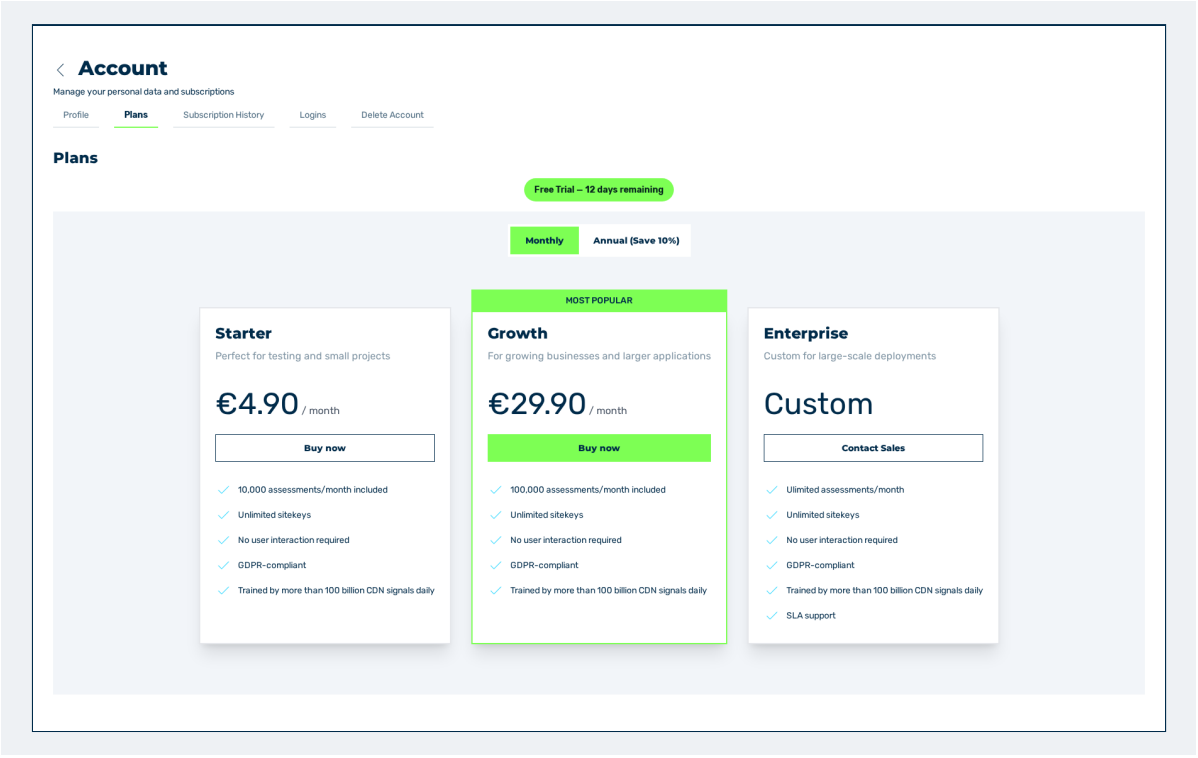


Figure 37: Plans tab

The **Plans** tab shows the available plans and starts their booking.

> **Video:** The available plans and how to book them — playable in the online help

3.4.3.1 Plan status

Above the offer, a coloured mark names the condition of your plan:

Mark	Meaning
Free Trial – 12 days remaining	The trial period is in operation. The mark gives the remaining days.
Free Trial Expired – 3 days to upgrade	The trial period ended. The mark gives the remaining days before the lock.
No active plan	The trial period ended, and there is no subscription.
Captcha Starter Plan – Active	

Mark	Meaning
	The subscription is in operation. The mark gives the plan that was booked.
Captcha Starter Plan committed – activates on 13.06.2026	You booked a plan during the trial period. The mark gives the start date.
Plan cancelled – access until 19.09.2026	You cancelled the subscription. The mark gives the end of the period you paid for.

3.4.3.2 Billing period

Two buttons select the billing period:

Button	Meaning
Monthly	monthly billing
Annual (Save 10%)	annual billing with a discount of 10 % on the monthly price

3.4.3.3 Plans

These plans are available:

Plan	Description	Price per month	Assessments per month	Button
Starter	Perfect for testing and small projects	€4.90	10,000	**Buy now**
Growth	For growing businesses and larger applications	€29.90	100,000	**Buy now**
Enterprise	Custom for large-scale deployments	on request	unlimited	**Contact Sales**

The **Growth** plan has the **MOST POPULAR** banner.

All plans contain these services:

- unlimited count of sitekeys
- verification without an operation of the visitor

- conformity with the GDPR
- assessment with more than 100 billion CDN signals each day

The **Enterprise** plan also contains the support with an SLA.

With annual billing, the monthly price decreases to €4.40 for the **Starter** plan and to €26.90 for the **Growth** plan.

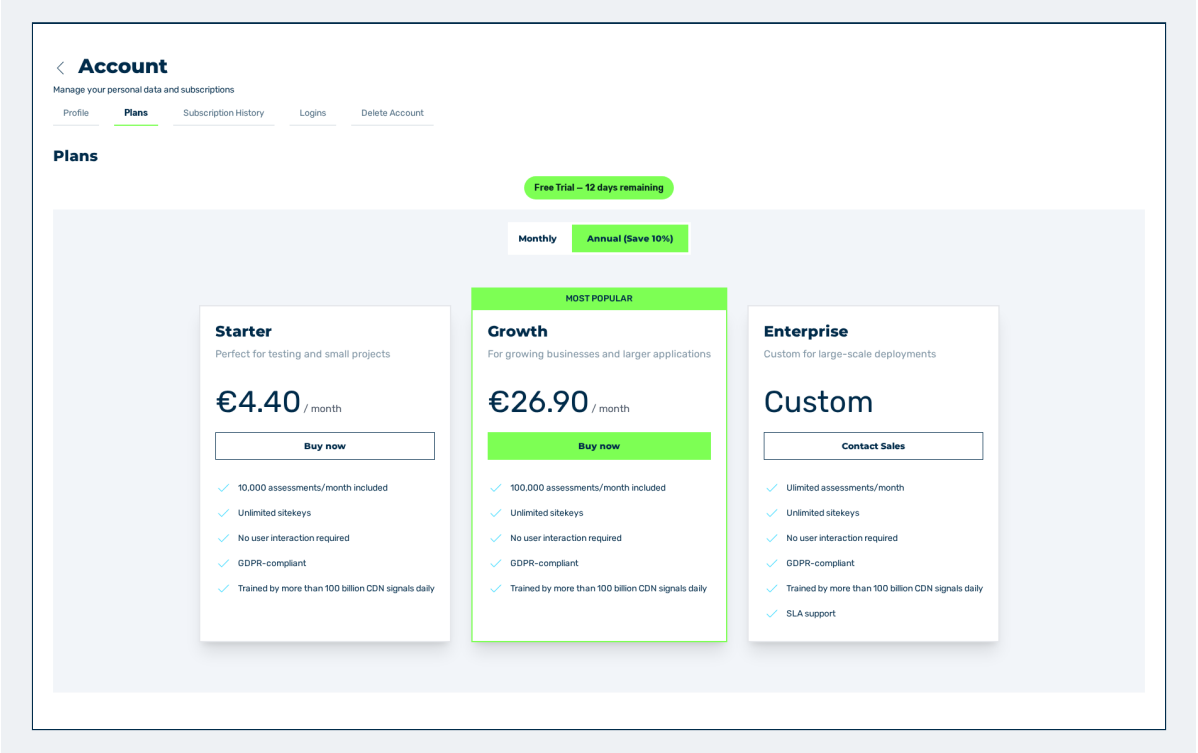


Figure 38: Plans tab with annual billing

While a subscription is in operation, the buttons of the **Starter** and **Growth** plans have the **Already subscribed** text and are disabled.

At this time, the monthly limit for assessments is not enforced. If the limit is exceeded, the widget continues to operate.

See **Book a plan**.

3.4.4 Subscription History tab

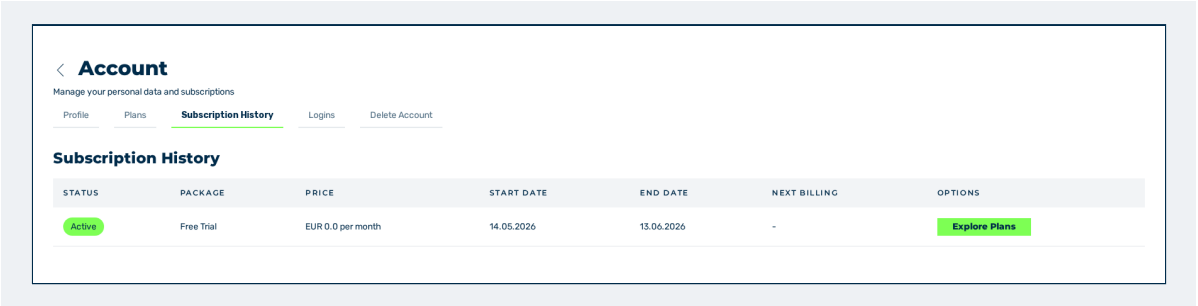


Figure 39: Subscription History tab

The **Subscription History** tab shows the history of your subscriptions. The trial period is its own row in the table, with the **Free Trial** plan.

> **Video:** The history of the subscriptions — playable in the online help

3.4.4.1 Columns

The table contains these columns:

Column	Contents
Status	Condition of the subscription as a coloured mark
Package	Name of the plan: **Free Trial**, **Captcha Starter**, or **Captcha Growth**
Price	Amount with the currency and the period, for example `EUR 4.90 per month`. An annual plan has the `per year` text.
Start date	Start of the subscription in the DD.MM.YYYY format
End date	End of the current period in the DD.MM.YYYY format
Next billing	Time of the next billing
Options	Buttons for the entry

3.4.4.2 Status

The **Status** column has these marks:

Mark	Meaning
Active	The subscription is in operation.
Committed	You booked a plan during the trial period. The plan starts after the end of the trial period.
Cancelled	You cancelled the subscription.
Expired	The subscription ended.

3.4.4.3 Next billing

The **Next billing** column shows these values:

Value	Meaning
`-`	Row of the trial period
`Activates 13.06.2026`	The plan that was booked starts on this date.
`Ends 19.09.2026`	The cancelled subscription continues until this date.
`Subscription cancelled`	The cancelled subscription ended.
`19.09.2026`	Date of the next billing

3.4.4.4 Options

The **Options** column contains these buttons, related to the condition of the entry:

Button	Row	Effect
Explore Plans	Row of the trial period	Opens the **Plans** tab.
Cancel commitment	**Committed** status	Opens the **Cancel commitment** dialog.

Button	Row	Effect
Manage	Subscription in operation	Opens the customer portal of the payment service provider. There you manage the payment data and the invoices.
Cancel	Subscription in operation with an open billing	Opens the **Cancel subscription** dialog.

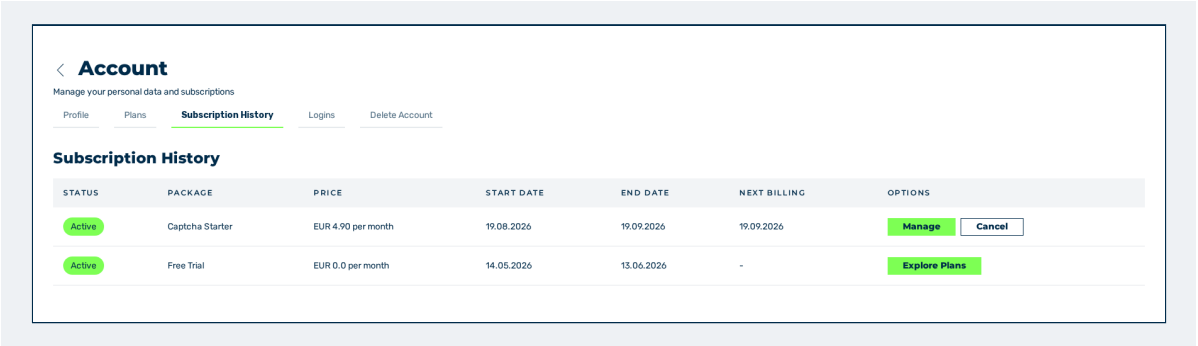


Figure 40: Subscription History tab with a plan that was booked

See **Book a plan**.

See **Cancel a subscription**.

3.4.5 Logins tab

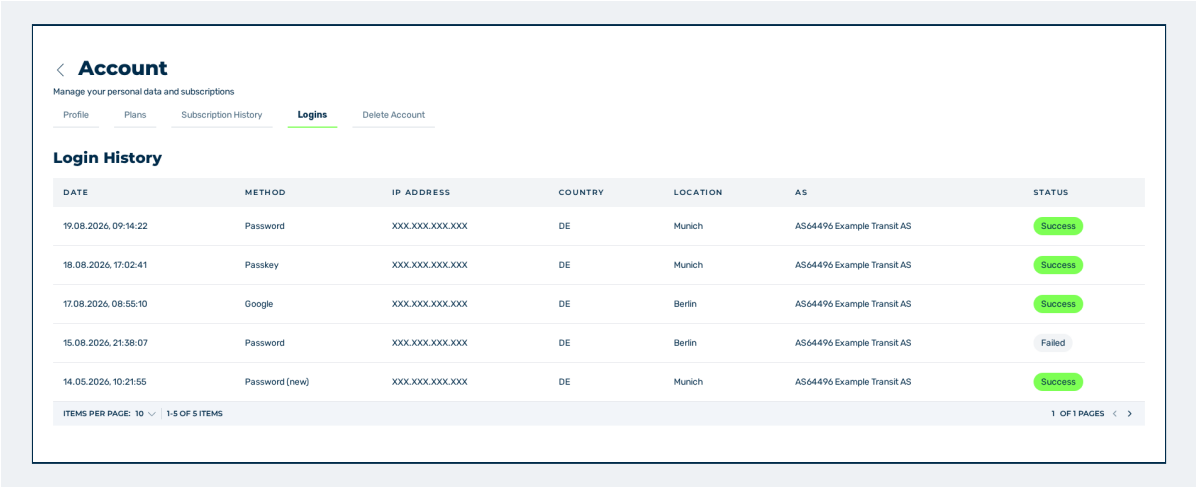


Figure 41: Logins tab

The **Logins** tab logs the logins to your account. From the log, you can see when and from where a login occurred. The table has the **Login History** heading.

> **Video:** The log of the logins — playable in the online help

3.4.5.1 Columns

The table contains the following columns:

Column	Contents
Date	Time of the login in the DD.MM.YYYY, hh:mm:ss format
Method	Login method that was used, for example **Password**, **Passkey**, or **Google**. The first login of an account has the **(new)** text.
IP Address	IP address that the login came from
Country	Country code of the IP address, for example **DE**
Location	Location of the IP address. If there is no location, the country is shown.

Column	Contents
AS	Autonomous system of the IP address with its number and its name, for example `AS64496 Example Transit AS`
Status	Result of the login as a coloured mark: **Success** or **Failed**

A column with no value contains a `-`.

Below the table, you go through the log and select the count of the entries for each page. At the start, the table shows 10 entries for each page.

While no entry is available, the table shows the **No login events** note.

If the log contains a login that you cannot identify, change your password immediately and activate the two-factor authentication. See **Change the password**.

3.4.6 Delete Account tab

Figure 42: Delete Account tab

The **Delete Account** tab deletes your account permanently. The **Delete Account Permanently** button starts the deletion.

> **Video:** The Delete Account tab and the deletion of the account — playable in the online help

3.4.6.1 Results of the deletion

The deletion has these results:

- Your account and your personal data are deleted permanently.
- All your sitekeys and the related CAPTCHA data are removed.
- You are logged out and cannot log in again.

> You cannot undo the deletion. Protected websites lose their protection as soon as the sitekeys are removed.

3.4.6.2 The Delete Account Permanently dialog

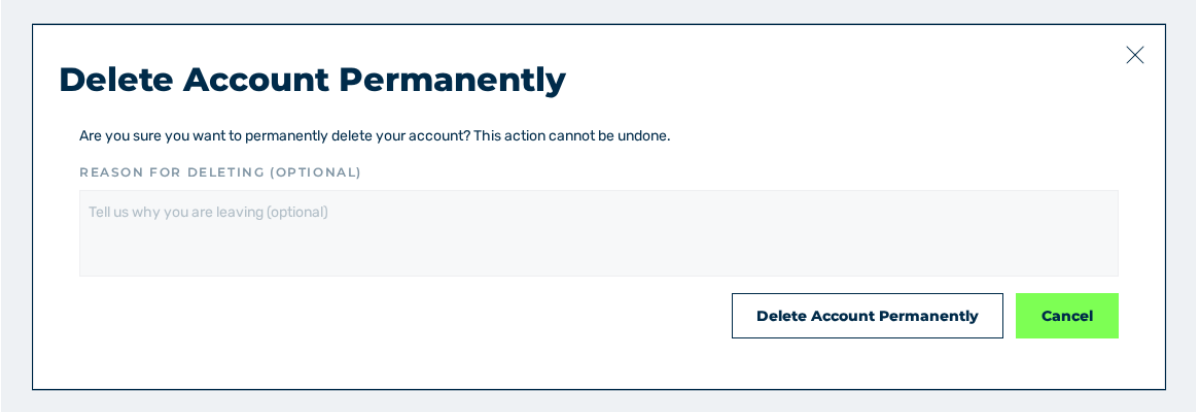


Figure 43: Delete Account Permanently dialog

The **Delete Account Permanently** button opens the dialog with the same name.

The **Reason for deleting (optional)** field with the **Tell us why you are leaving (optional)** placeholder accepts a reason. The reason is optional and holds a maximum of 2048 characters.

The dialog contains these buttons:

Button	Effect
Delete Account Permanently	Deletes the account and signs you out.
Cancel	Closes the dialog and does not delete.

3.4.6.3 Conditions

The deletion is possible only when none of these conditions is applicable:

Condition	Title of the dialog	Measure
A subscription is active.	**Cancel your subscription first**	Cancel the subscription. The **Go to Subscription History** button opens the related tab.
A payment is repeated again.	**A payment is still being retried**	Wait until the payment is complete.

The **Close** button closes the dialog.

See **Subscription History tab**.

See **Cancel a subscription**.

3.5 Organization

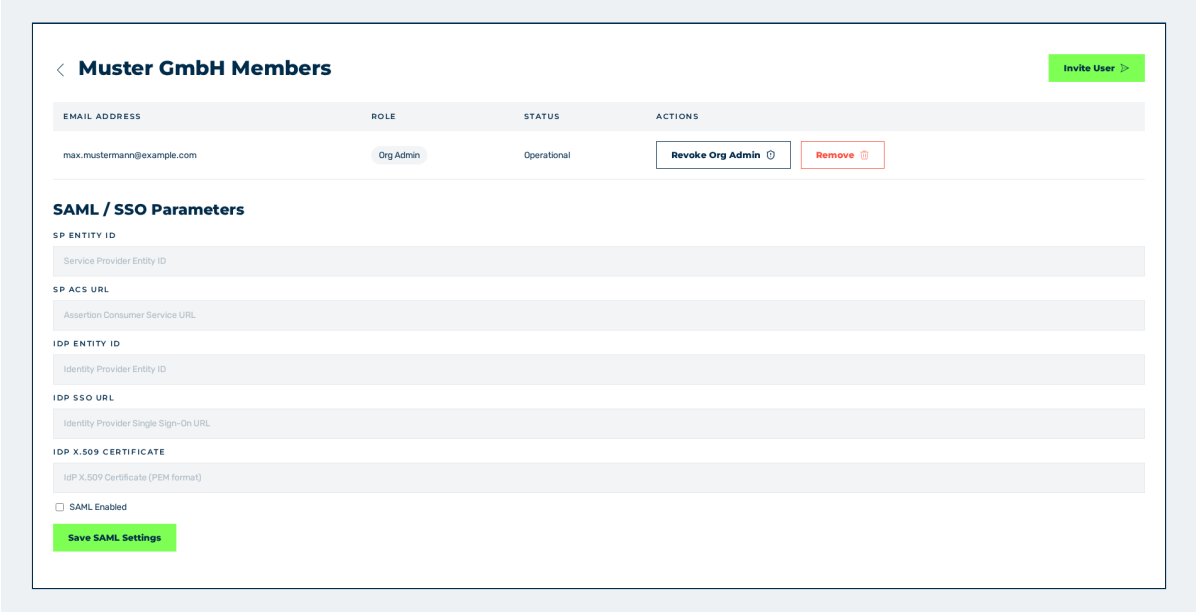


Figure 44: Organization view

The view manages the members of your organisation and the parameters for the login with SAML and SSO. The view is available to accounts with the **Org Admin** role. The title gives the name of your organisation, followed by **Members**.

To open the view, use the **Manage Members** button in the **Organization** area of the **Profile** tab.

3.5.1 Table of the members

The table contains these columns:

Column	Contents
Email Address	E-mail address of the member
Role	Role of the member as a mark
Status	Condition of the account
Actions	Buttons for the entry

While no member is available, the table shows the **No Members** note.

The **Invite User** button next to the title opens the **Invite User** dialog.

3.5.2 Roles

These roles are available:

Role	Rights
Member	Uses the product in the limits of the related sitekeys.
Org Admin	Also manages the members and the settings for SAML and SSO.

3.5.3 Status

The **Status** column has one of these values:

Value	Meaning
Operational	The account is confirmed and ready for use.
Pending Verification	The invitation was sent, but the account is not confirmed yet.

3.5.4 Actions

The **Actions** column contains these buttons:

Button	Effect
Make Org Admin	Opens the **Grant Org Admin** dialog. The button is shown for a member with the **Member** role.
Revoke Org Admin	Opens the **Revoke Org Admin** dialog. The button is shown for a member with the **Org Admin** role.
Remove	Opens the **Remove Member** dialog.

Each of the three dialogs gives the e-mail address of the member and contains the **Close** and **Confirm** buttons.

Figure 45: Grant Org Admin dialog

3.5.5 SAML / SSO Parameters area

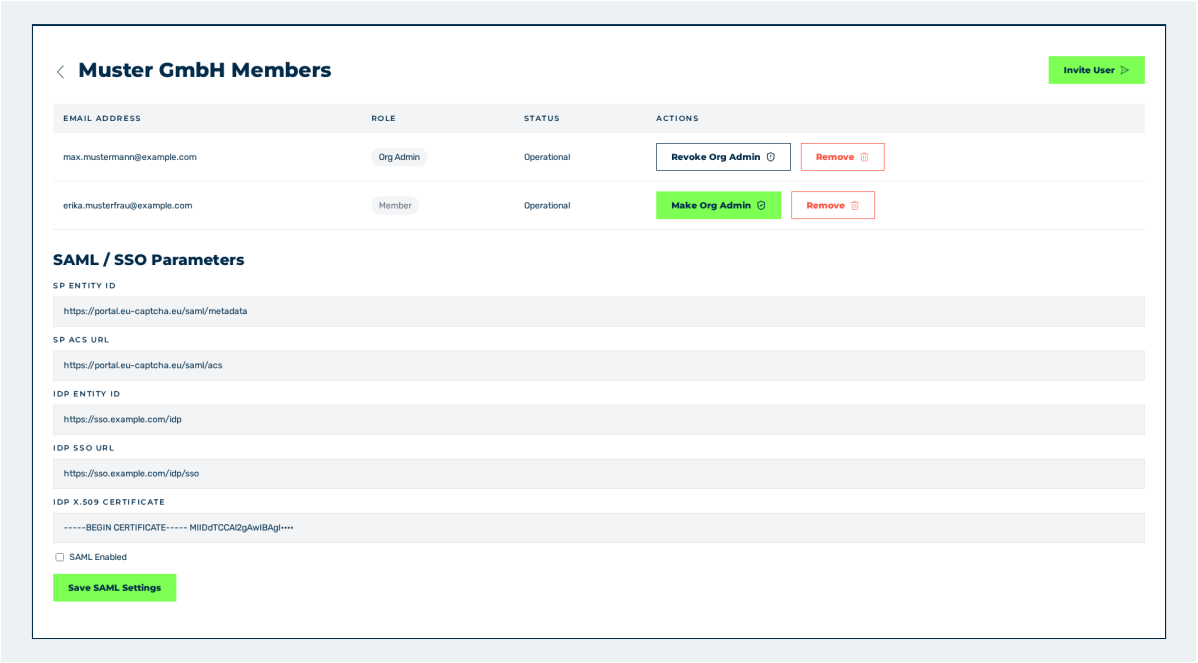


Figure 46: SAML / SSO Parameters area

The **SAML / SSO Parameters** area connects your organisation with an identity provider. It contains these fields:

Field	Contents
SP Entity ID	Identifier of the service provider, given by Myra EU CAPTCHA
SP ACS URL	Address of the service provider for the return of the login
IdP Entity ID	Identifier of your identity provider
IdP SSO URL	Login address of your identity provider
IdP X.509 Certificate	Certificate of your identity provider in the PEM format

The **SAML Enabled** check box activates the login with SAML. The **Save SAML Settings** button applies the data. After the save operation, the **SAML Settings** message with the **SAML settings saved successfully.** text is shown.

See Invite a member.

See Set up SAML.

3.6 Help & Support

3.6.1 Help

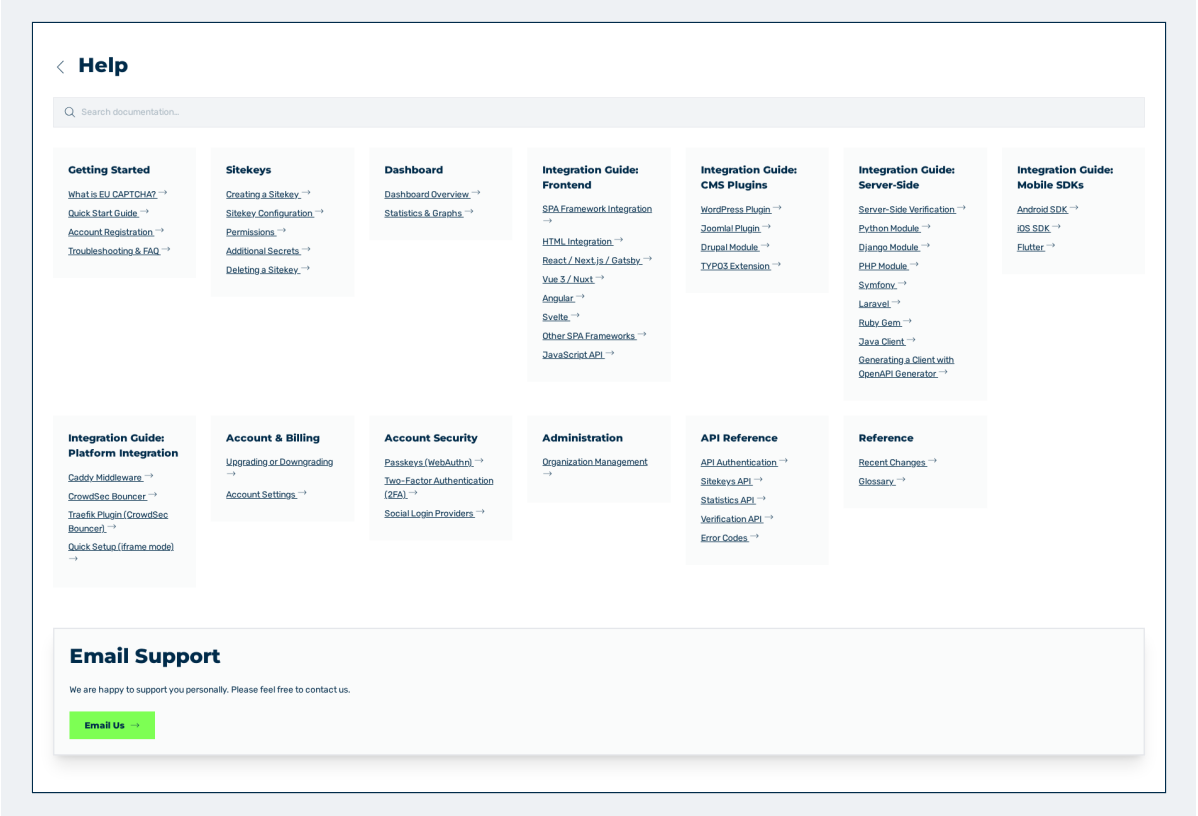


Figure 47: Help view

The **Help** view goes to the documentation and to the support. To open the view, use the **Help & Support** entry of the sidebar.

> **Video:** The Help view and the access to support — playable in the online help

3.6.1.1 Search

At the top, a search field with the **Search documentation...** placeholder is shown. The search examines the documentation and shows the hits below the field. To select a hit, use the arrow keys and the enter key.

The search field is also available in the header of the customer portal.

3.6.1.2 Topic cards

The view divides the documentation into cards. These cards are available:

Card	Contents
Getting Started	Introduction, quick start, registration, troubleshooting, and FAQ
Sitekeys	Creation and management of sitekeys
Dashboard	Key figures and analyses
Integration Guide: Frontend	Embedding in frontend frameworks
Integration Guide: CMS Plugins	Extensions for CMS platforms
Integration Guide: Server-Side	Server-side verification
Integration Guide: Mobile SDKs	Embedding in mobile apps
Integration Guide: Platform Integration	Embedding in proxies and gateways
Account & Billing	Account and billing
Account Security	Security of the account
Administration	Management of the organization
API Reference	Description of the endpoints
Reference	Reference work

3.6.1.3 Support

At the bottom, the **Email Support** card is shown. The card contains the **We are happy to support you personally. Please feel free to contact us.** text and the **Email Us** button. The button opens an e-mail to `eucaptcha.support@myrasecurity.com`.

The support is available in each plan.

See Plans tab.

4. Configuration

4.1 Sitekey

4.1.1 Creating a sitekey

For each domain that you want to protect, create a related sitekey. The sitekey connects the domain with Myra EU CAPTCHA and supplies the keys for the widget and for the server-side verification.

> **Video:** Creation of an additional sitekey — playable in the online help

The following requirements must be met:

☑ You are logged in to the customer portal.

☑ You know the domain that you want to protect.

Proceed as follows to create a subsequent sitekey:

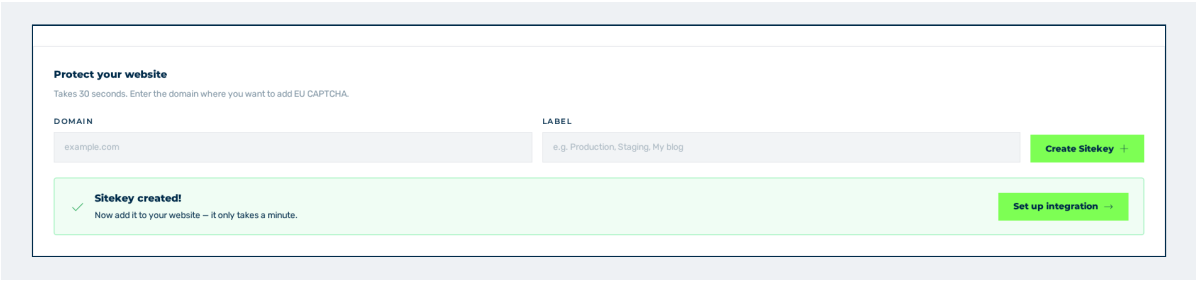


Figure 48: Protect your website area

- Open the **Dashboard** view and go to the **Protect your website** area. The description is **Takes 30 seconds. Enter the domain where you want to add EU CAPTCHA.**
- In the **Domain** field, enter the domain to protect, for example `example.com`.
- If required, enter a label in the **Label** field, for example `Production`, `Staging`, or `My blog`.
- Click on the **Create Sitekey** button.

↳ The system creates the sitekey and shows the **Sitekey created!** message. The text is **Now add it to your website – it only takes a minute.** The system shows the new entry in the **Sitekey List** list.

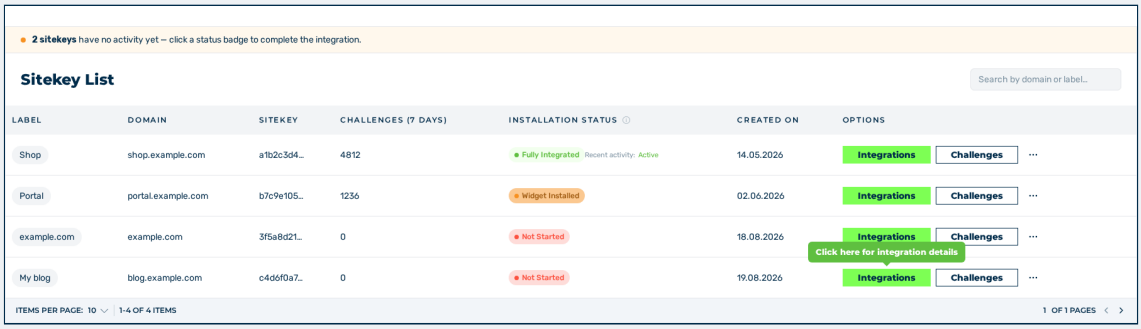


Figure 49: Sitekey List area with the new entry

► Click on the **Set up integration** button.

➔ The **Details** view of the new sitekey is shown.

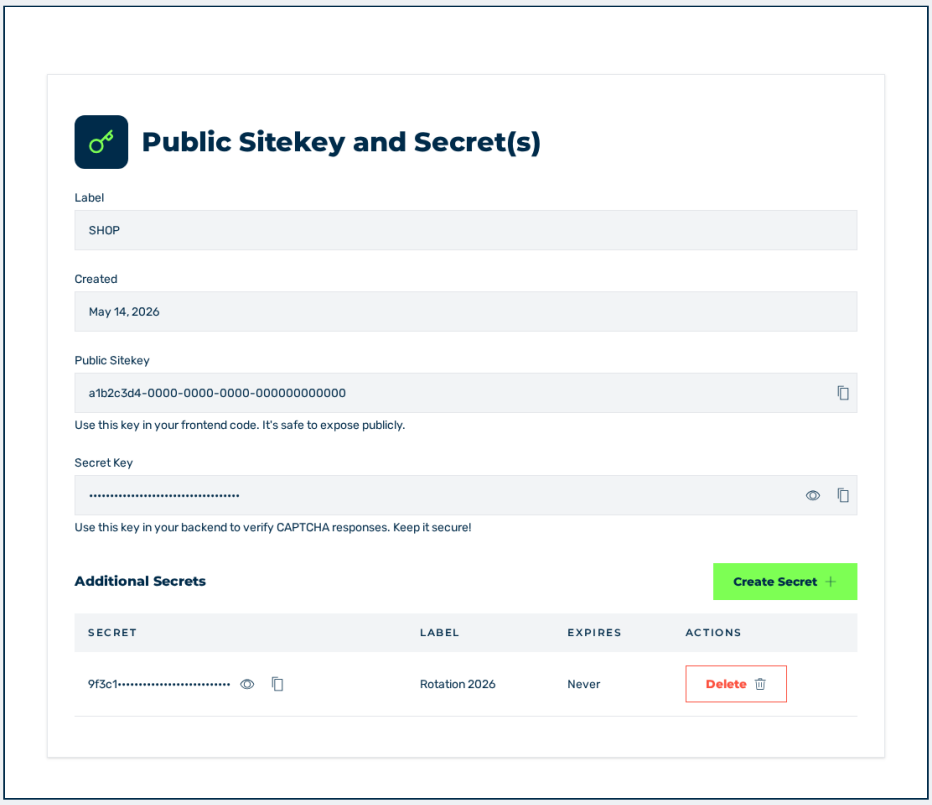


Figure 50: Details view

4.1.1.1 Verification of the input

The **Domain** field refuses incorrect input:

Message	Cause
Please enter a domain.	The field is empty.
Please enter a valid domain name (e.g. example.com).	The input is not a valid domain name.

An initial `http://` or `https://` and an attached path are removed from the input. Uppercase and lowercase have no effect: the system keeps the domain in lowercase.

At this time, the system does not prevent more than one sitekey for the same domain. Examine the list and delete the double entries. See **Two sitekeys for the same domain**.

See **Configuring a sitekey**.

4.1.2 Configuring a sitekey

The configuration controls how strictly Myra EU CAPTCHA examines the visitors of a domain. You switch the protection on and off. You also set the initial difficulty, the delay, and the count of parallel challenges.

> **Video:** Change of the configuration of a sitekey — playable in the online help

The following requirements must be met:

☑ You have the **Write** access tier for the sitekey.

☑ A plan with costs is active for your account. With the **free** plan, the three number values are locked. The **Label** field and the **EU CAPTCHA Protection** switch stay available.

The view has no button to save. The system applies each change immediately and shows **Success** with the text **Sitekey config updated successfully**.

Proceed as follows to configure a sitekey:

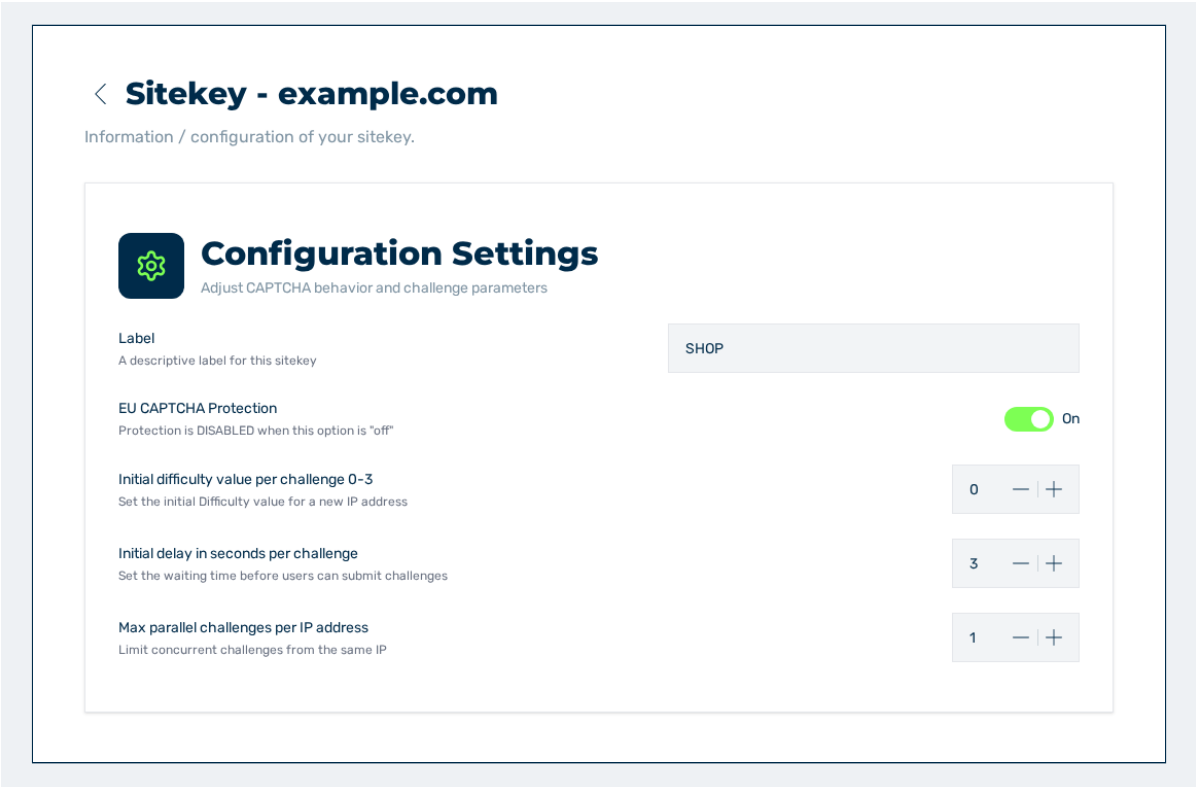


Figure 51: Configuration view

► Open the sitekey and go to the **Configuration** view.

- If required, enter a label in the **Label** field and push the Enter key.
- Set the **EU CAPTCHA Protection** switch to **On**.
- In the **Initial difficulty value per challenge 0-3** field, enter a value between 0 and 3.
- In the **Initial delay in seconds per challenge** field, enter a value between 3 and 30.
- In the **Max parallel challenges per IP address** field, enter a value between 1 and 10.

➔ The system applies each change immediately and uses the values for the subsequent challenges.

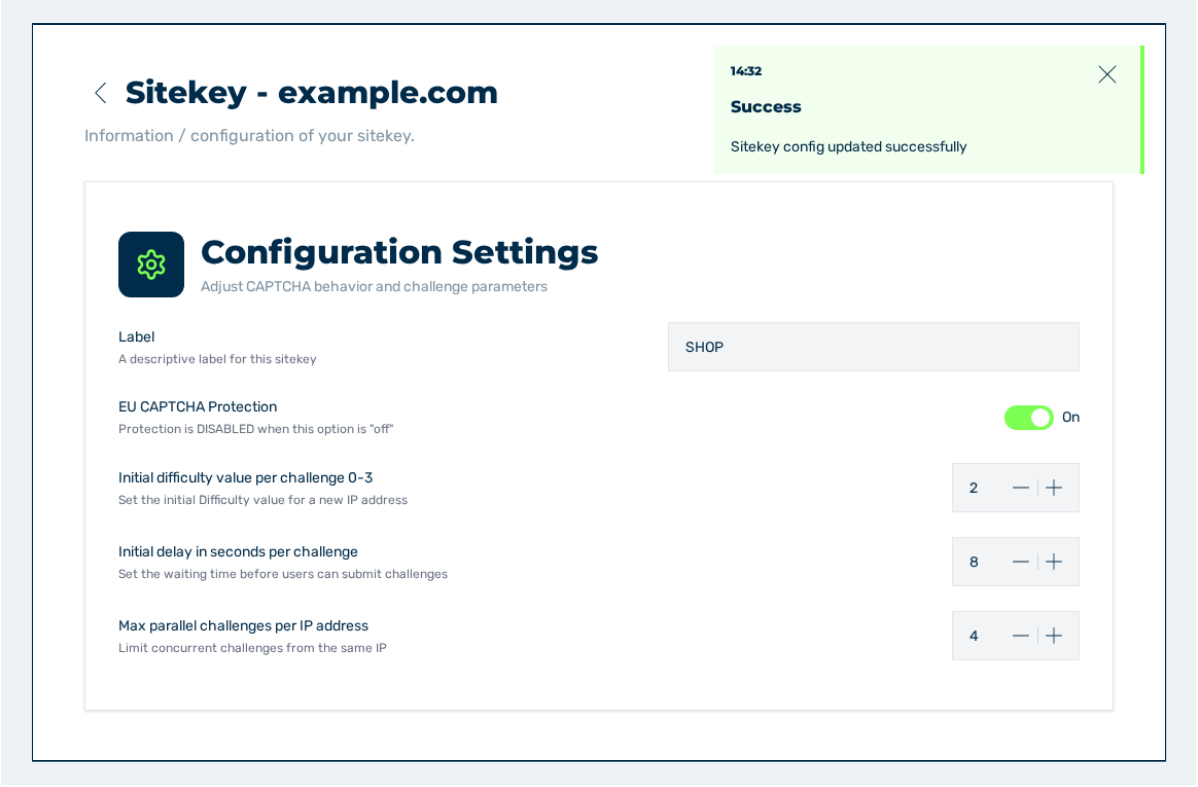


Figure 52: Configuration view with the changed values

4.1.2.1 Effect of the settings

The settings have this effect:

Setting	Effect	Default
EU CAPTCHA Protection	With **Off**, the widget does not set a challenge, and the requests continue without a verification.	**On**

Setting	Effect	Default
Initial difficulty value per challenge 0-3	Sets the difficulty of the first challenge of an unknown IP address. A higher value makes the verification longer for all visitors.	0
Initial delay in seconds per challenge	Sets the delay before the transmission is possible.	3
Max parallel challenges per IP address	Limits the concurrent challenges of one IP address.	1

The risk detection sets the higher difficulties up to level 7 by itself. See **How EU CAPTCHA works**.

With **Off**, there is no protection. The widget stays embedded. The requests are not examined.

4.1.2.2 Locked settings

If the settings are locked, a tooltip names the cause:

Tooltip	Cause	Measure
Read-only access	Your access tier is **Read**.	Ask for the **Write** tier. See **Change the access to a sitekey**.
Upgrade to unlock	No plan with costs is active for your account.	Book a plan. See **Book a plan**.

4.1.3 Managing additional secrets

In addition to the original secret, you create further secrets for a sitekey. With them, you change the key during operation without an interruption of the verification: you create a second secret, change your servers to it one after the other, and delete the old one afterwards.

4.1.3.1 Creating a secret

The following requirements must be met:

☑ You have the **Write** access tier for the sitekey.

Proceed as follows to create an additional secret:

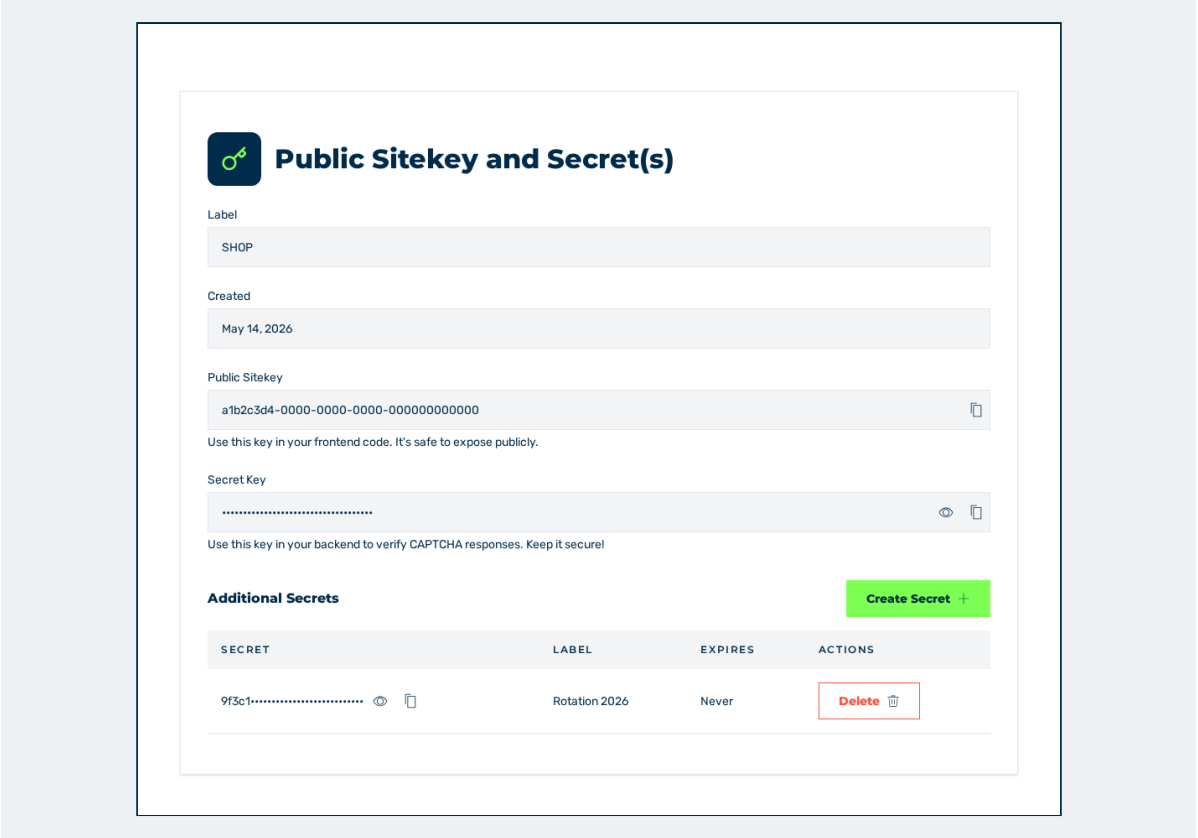


Figure 53: Details view

- ► Open the sitekey and change to the **Details** view.
- ► Click on the **Create Secret** button in the **Additional Secrets** area.
  - ↳ The **Create Secret** dialog opens.

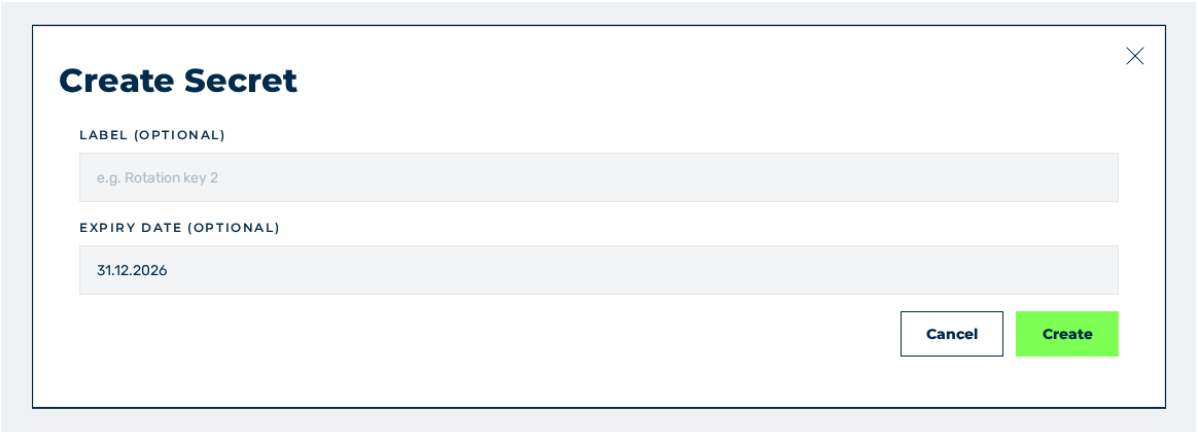


Figure 54: Create Secret dialog

- If required, enter a label in the **Label (optional)** field, for example `Rotation key 2`.
- If required, select an expiry date in the **Expiry date (optional)** field.
- Click on the **Create** button.

➔ The system creates the secret and shows it in the **Additional Secrets** area.

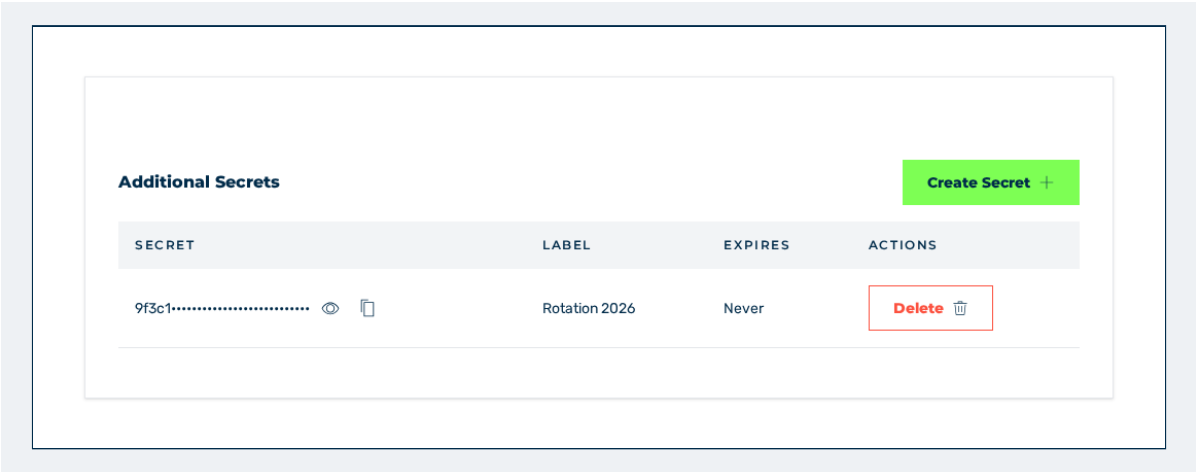


Figure 55: Additional Secrets area with the new secret

Without an expiry date, the secret does not expire. In this case, the **Expires** column shows the **Never** value.

To stop the operation, click on the **Cancel** button in the dialog.

Treat every additional secret like the original secret. It belongs on your server only and never in HTML or JavaScript.

4.1.3.2 Deleting a secret

The following requirements must be met:

☑ You have the **Write** access tier for the sitekey.

☑ None of your servers uses the secret any more.

Proceed as follows to delete an additional secret:

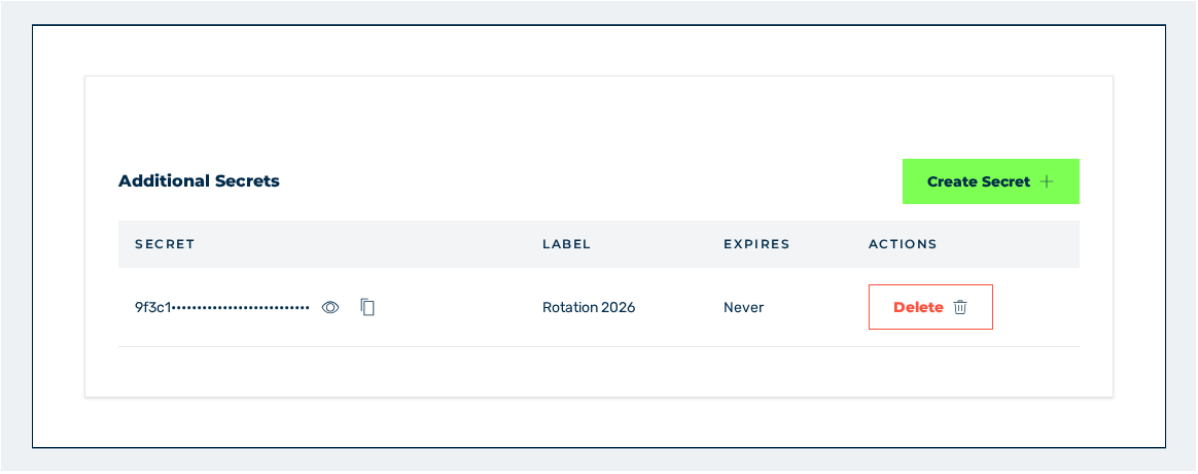


Figure 56: Additional Secrets area

► Click on the **Delete** button in the row of the secret in the **Additional Secrets** area.

↳ The **Delete Secret** dialog opens.

Figure 57: Delete Secret dialog

► Click on the **Delete** button.

➔ The system deletes the secret and shows the **Secret deleted** message with the **The secret has been removed** text.

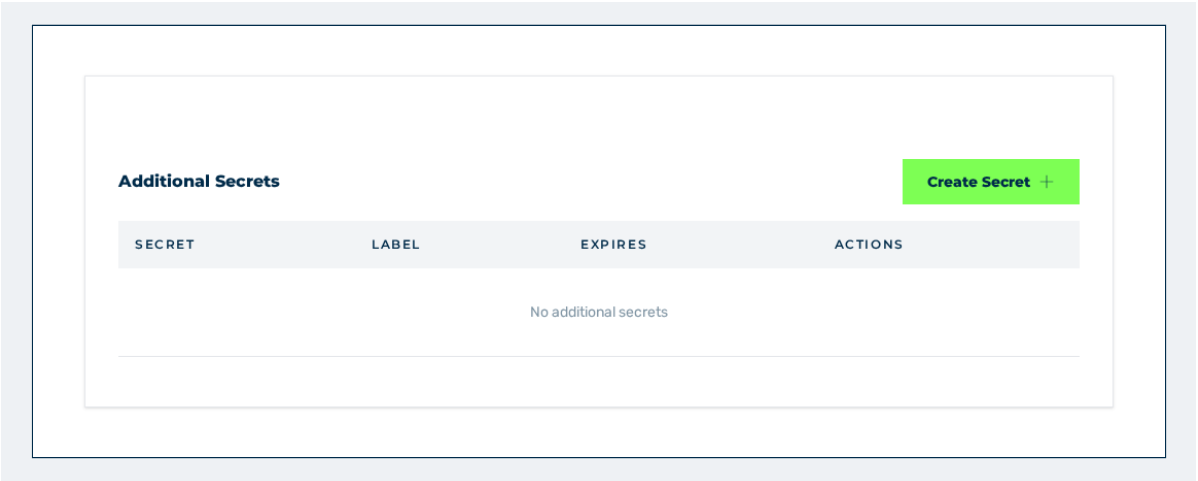


Figure 58: Additional Secrets area without additional secrets

To stop the operation, click on the **Cancel** button in the dialog.

A deleted secret cannot be restored. Requests that are still verified with this secret fail immediately.

See **Details**.

4.1.4 Deleting a sitekey

A sitekey that was deleted does not protect its domain any more. Delete a sitekey only after you removed the widget from the related website.

The following requirements must be met:

☑ You have the **Write** access tier for the sitekey.

☑ The widget is removed from the protected website, or the website is not in operation any more.

Proceed as follows to delete a sitekey:

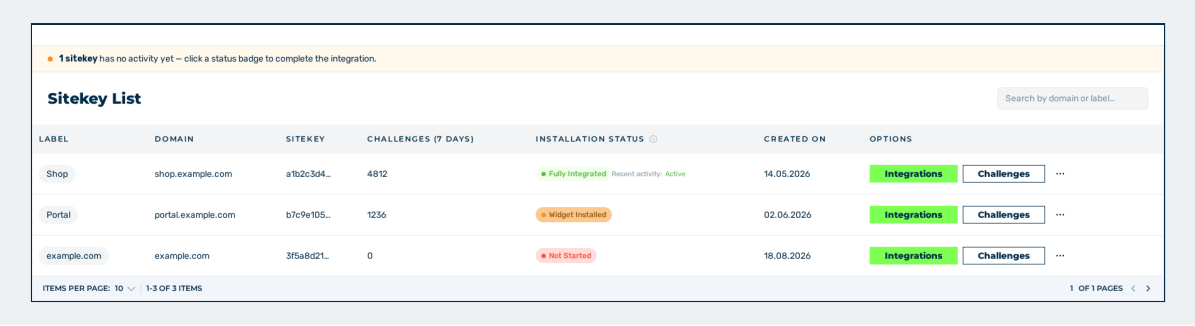


Figure 59: Sitekey List area

► Open the **Dashboard** view and go to the **Sitekey List** area.

► In the row of the sitekey, click on the action for the deletion in the **Options** column.

↳ The **Delete Sitekey?** dialog is shown.

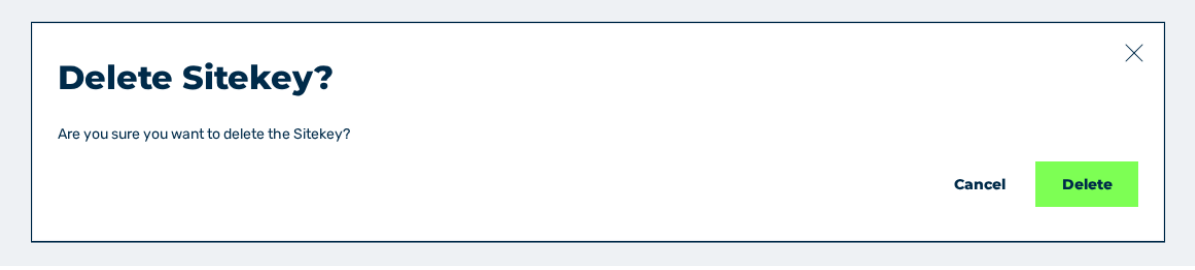


Figure 60: Delete Sitekey? dialog

► Click on the **Delete** button.

➔ The system deletes the sitekey. The entry is removed from the list.

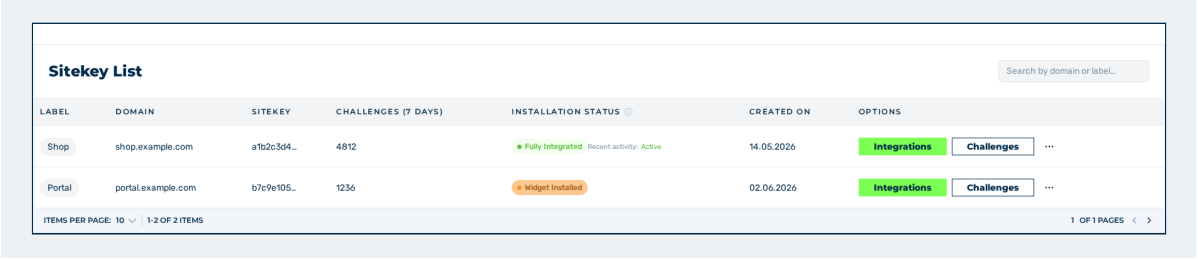


Figure 61: Sitekey List area without the deleted entry

After the deletion, the widget refuses the sitekey. Forms that still use the key do not get a valid token.

To stop the operation, click on the **Cancel** button in the dialog.

4.2 Permissions

4.2.1 Grant access to a sitekey

You give access to a sitekey to single accounts. You do not have to give your own credentials to them. The access is applicable to one sitekey.

The following requirements must be met:

☑ You have the **Write** access tier for the sitekey.

☑ The account that must get the access is registered in the customer portal.

☑ You know the e-mail address of this account.

Proceed as follows to grant access to a sitekey:

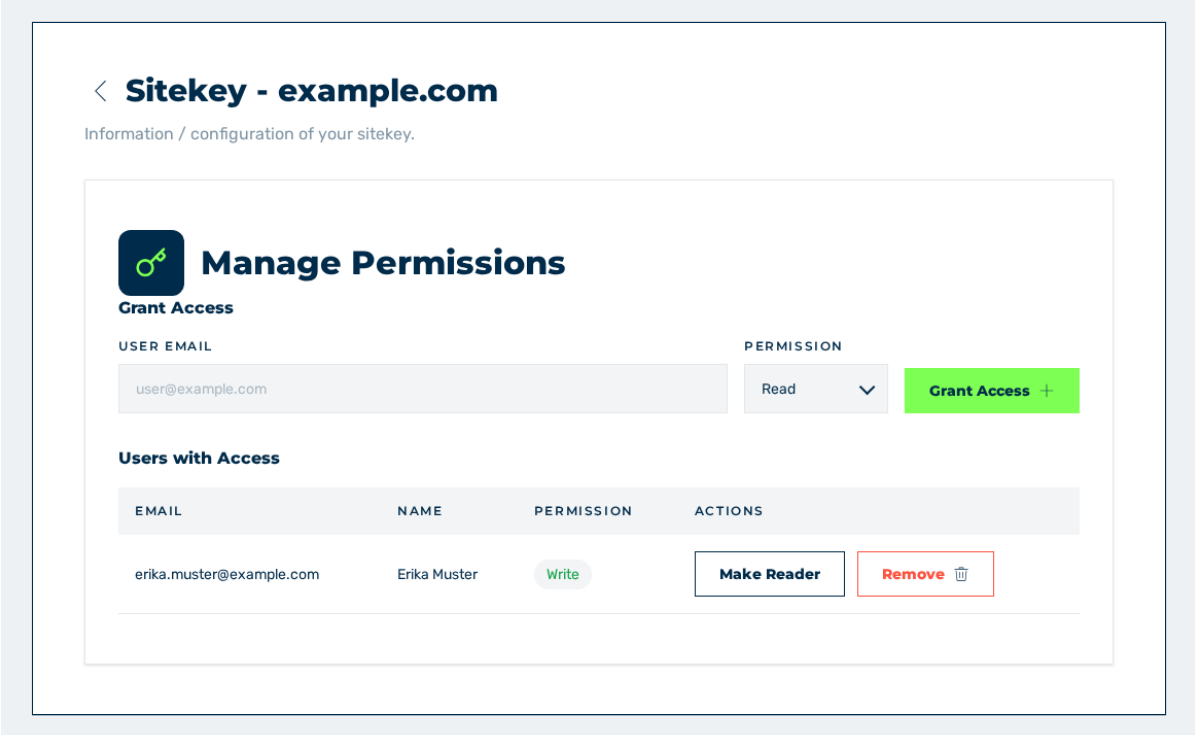


Figure 62: Permissions view

► Open the sitekey and go to the **Permissions** view.

► In the **Grant Access** area, enter the e-mail address of the account in the **User Email** field.

► In the **Permission** drop-down list, select the **Read** or **Write** access tier.

- ► Click on the **Grant Access** button.
- ➔ The system gives the access and shows the **Permission granted** message. The entry is shown in the **Users with Access** area.

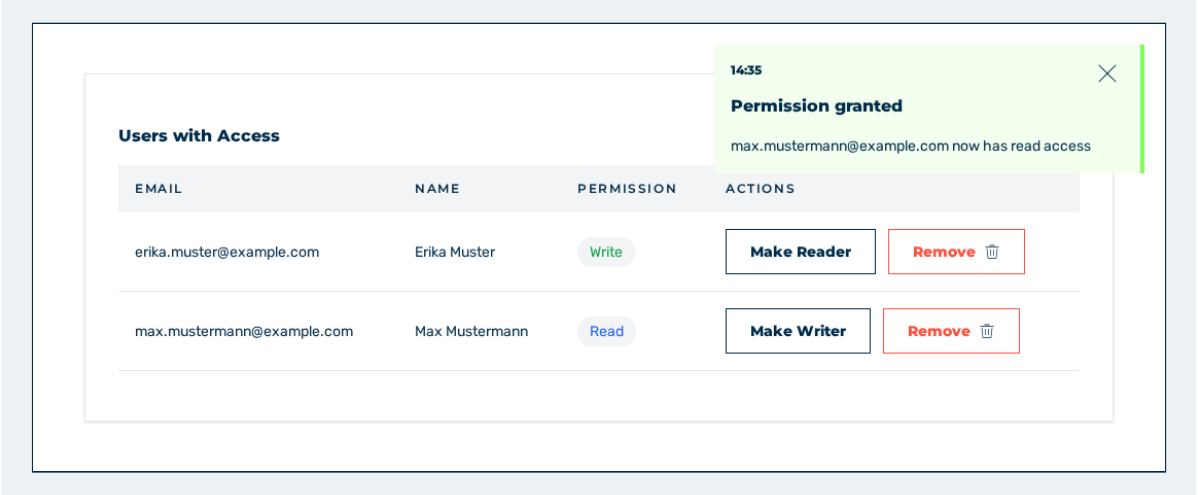


Figure 63: Users with Access area with the new entry

These access tiers are available:

Tier	Rights
Read	The sitekey and its settings are visible. Changes are locked.
Write	The sitekey and its settings are visible and can be changed.

See **Change the access to a sitekey**.

4.2.2 Change the access to a sitekey

You change the access tier of an account with permission between **Read** and **Write**.

The following requirements must be met:

☑ You have the **Write** access tier for the sitekey.

☑ The account already has permission. See **Grant access to a sitekey**.

Proceed as follows to change the access tier:

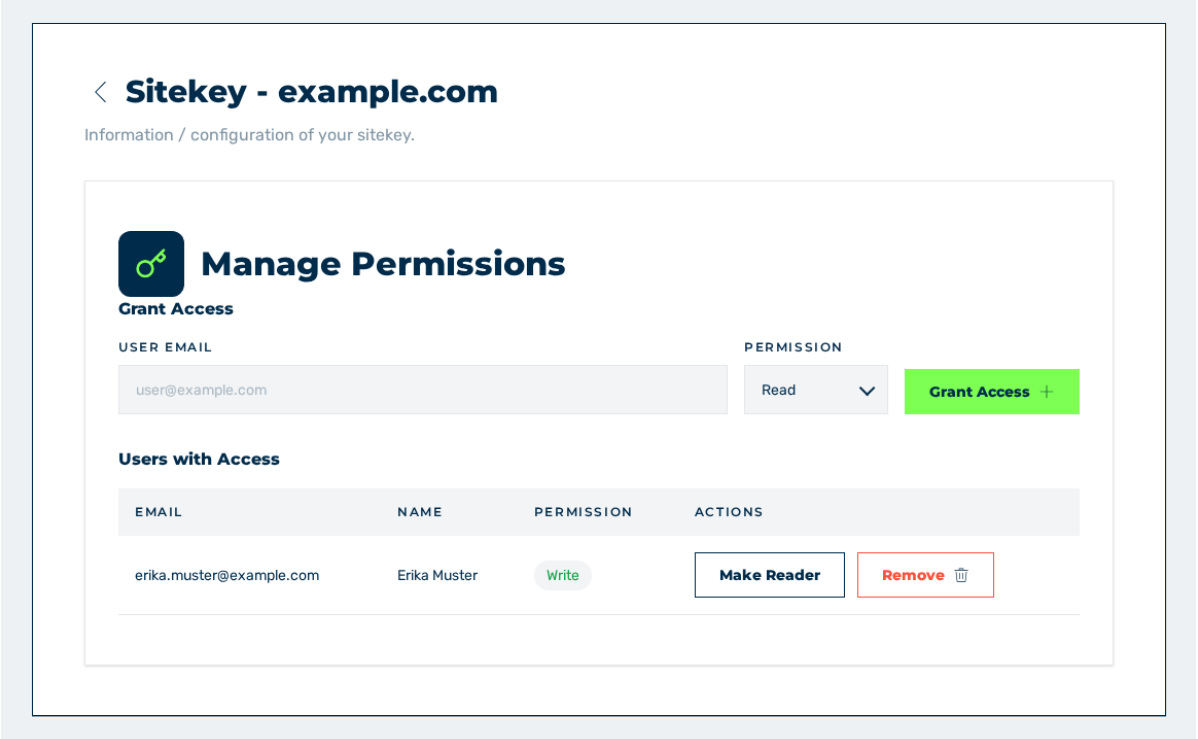


Figure 64: Users with Access area

- ▶ Open the sitekey and go to the **Permissions** view.
- ▶ In the **Users with Access** area, click on the tier action in the row of the account.

  ↳ The **Change to Write** or **Change to Read** dialog is shown. The question names the account and the two tiers.

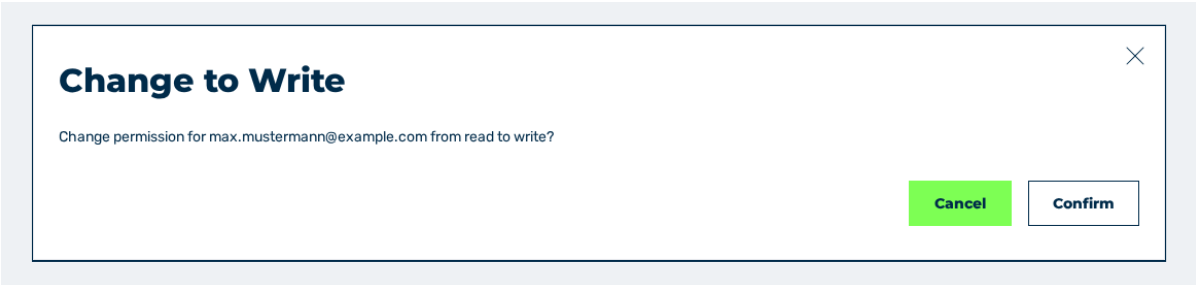


Figure 65: Change to Write dialog

- ► Click on the **Confirm** button.
- ➔ The system changes the tier and shows the **Permission updated** message. The **Permission** column shows the new tier.

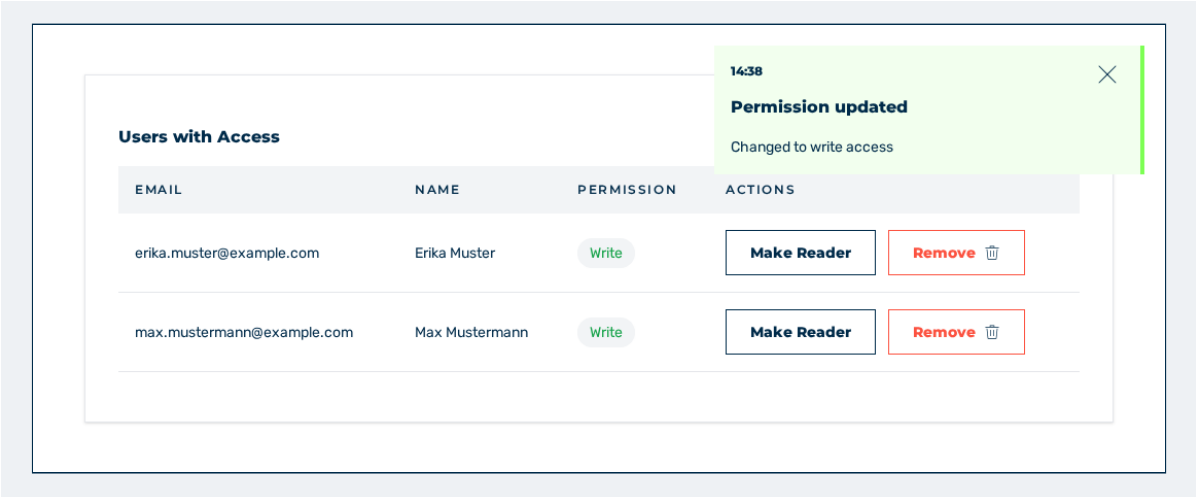


Figure 66: Users with Access area with the changed tier

To stop the operation, click on the **Cancel** button in the dialog.

4.2.3 Revoke the access to a sitekey

You revoke the access of an account to a sitekey. The sitekey itself continues to exist.

The following requirements must be met:

☑ You have the **Write** access tier for the sitekey.

☑ The account has permission.

Proceed as follows to revoke the access:

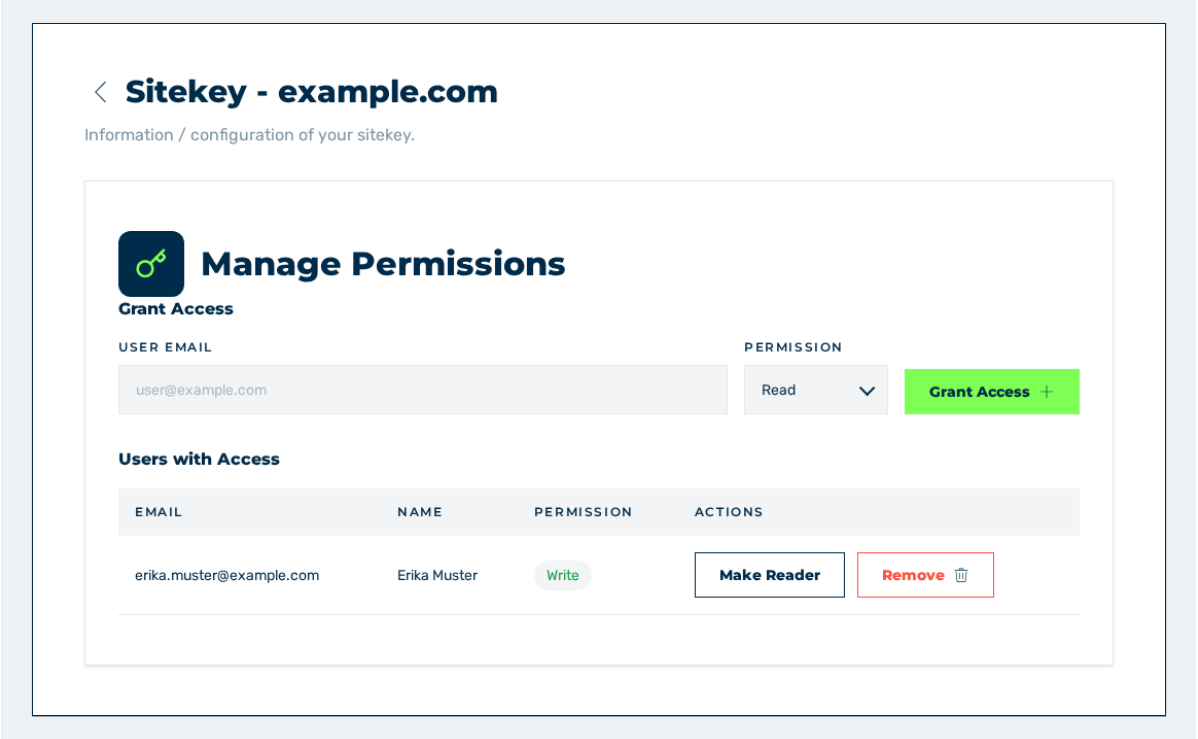


Figure 67: Users with Access area

► Open the sitekey and go to the **Permissions** view.

► In the **Users with Access** area, click on the **Remove** action in the row of the account.

↳ The **Remove Permission** dialog is shown. The question names the e-mail address of the account.

Figure 68: Remove Permission dialog

- ► Click on the **Confirm** button.
- ➔ The system revokes the access and shows the **Permission revoked** message. The entry is removed from the list.

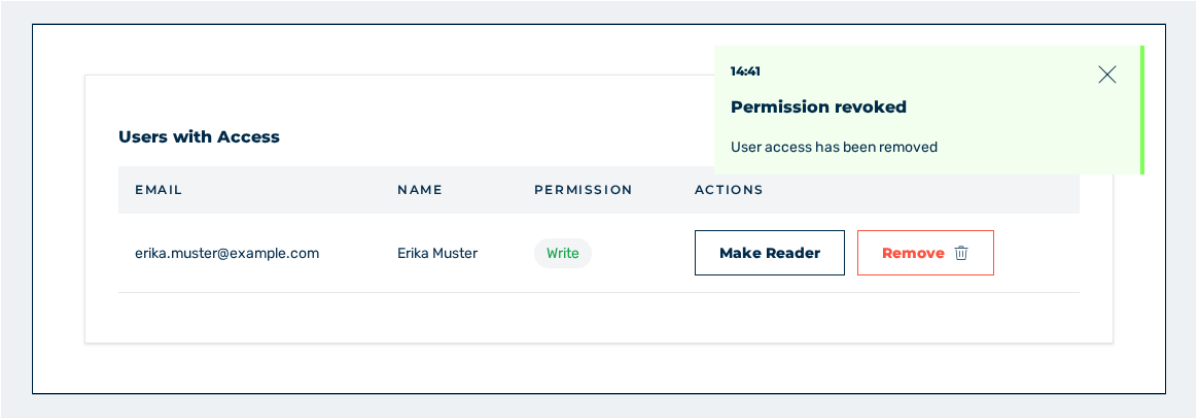


Figure 69: Users with Access area without the entry

To stop the operation, click on the **Cancel** button in the dialog.

4.3 Account and security

4.3.1 Changing the personal data

The first name, the last name, and the company name are in the **Profile** area of the tab with the same name. You do not change the e-mail address there directly: its field is locked.

The following requirements must be met:

☑ You are logged in to the customer portal.

Proceed as follows to change the personal data:

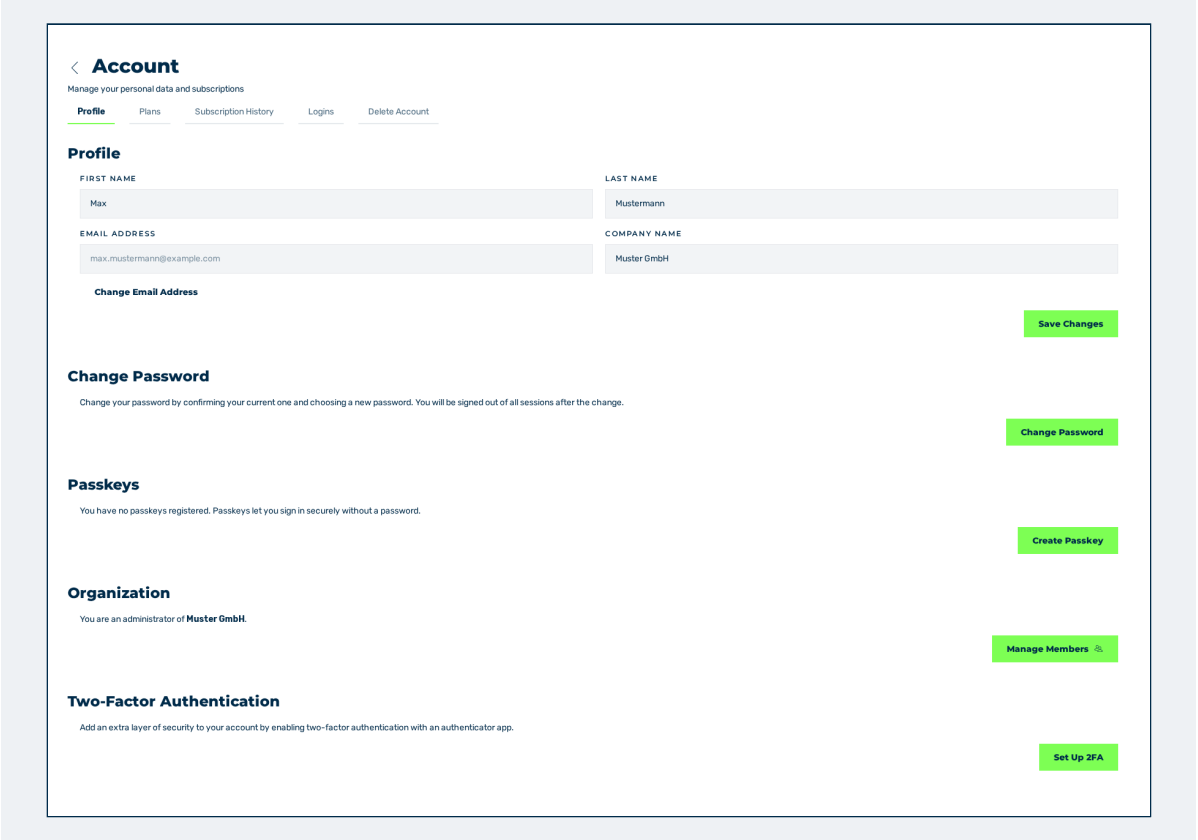


Figure 70: Profile tab

- ► In the sidebar, open the **Account** › **Profile** entry.
- ► Enter your first name in the **First name** field.
- ► Enter your last name in the **Last name** field.

- ► If required, enter the name of your company in the **Company name** field.
- ► Click on the **Save Changes** button.
- ➔ The system saves the data.

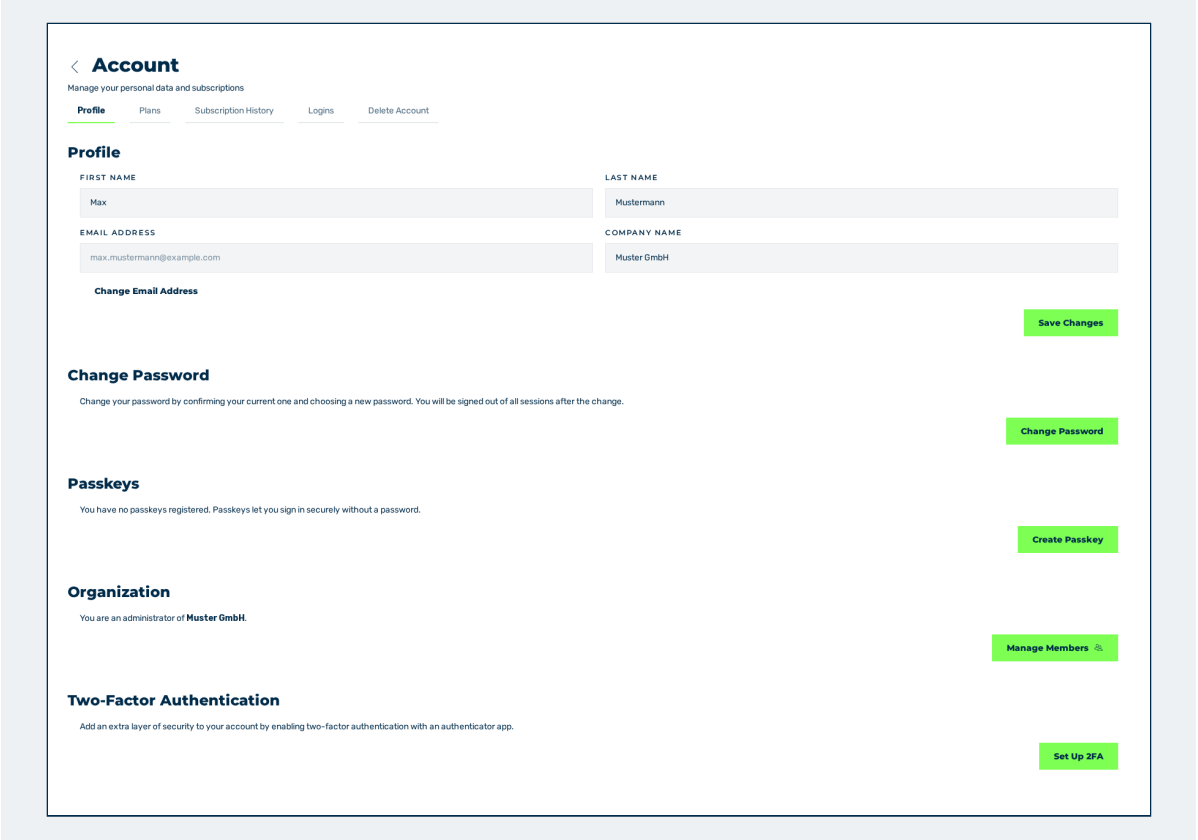


Figure 71: Profile tab with the saved data

The **Email address** field gives the e-mail address of your account. It is locked and can only be changed with the **Change Email Address** button. See Changing the e-mail address.

See Profile tab.

4.3.2 Changing the e-mail address

You change the e-mail address of your account in the **Profile** tab. The change becomes effective only after you confirm it with the previous address.

The following requirements must be met:

☑ You are logged in to the customer portal.

☑ You have access to the mailbox of the previous e-mail address.

Proceed as follows to change the e-mail address:

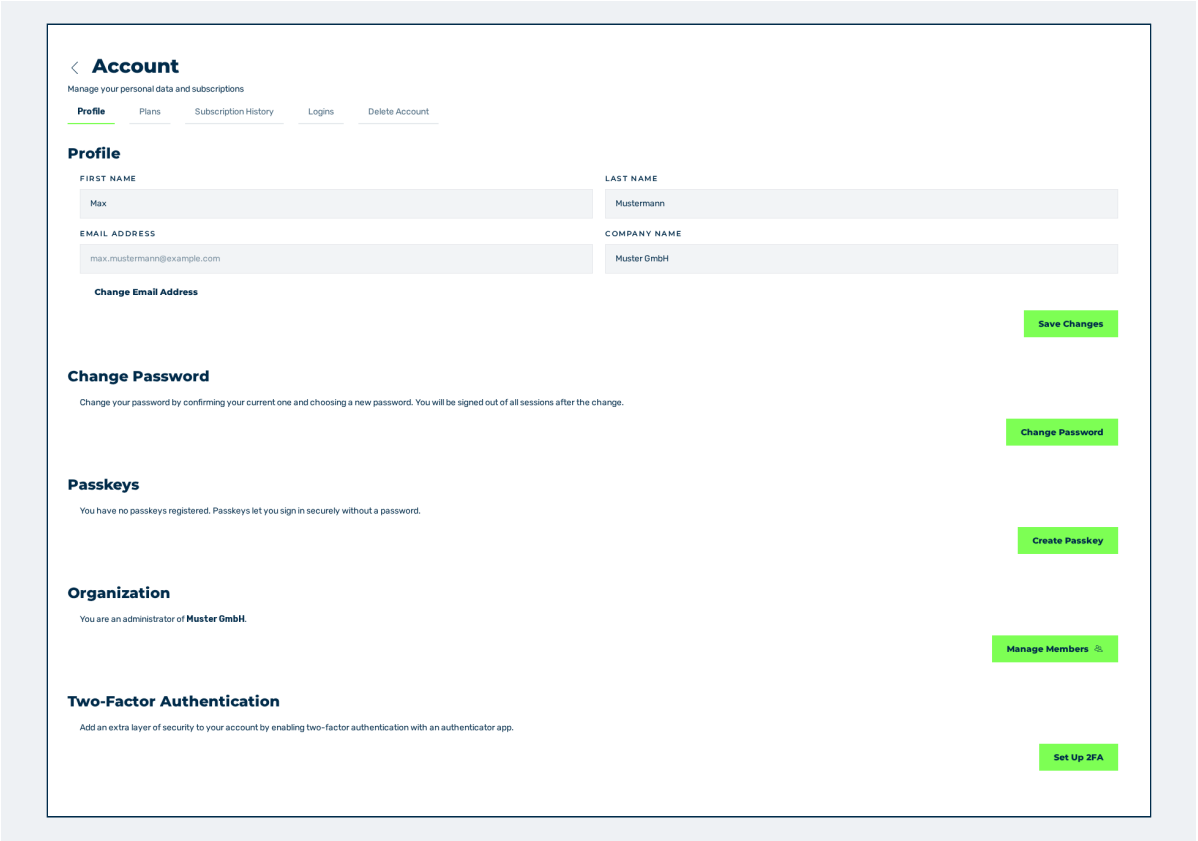


Figure 72: Profile tab

► In the sidebar, open the **Account** › **Profile** entry.

► Click on the **Change Email Address** button.

↳ The **Change Email Address** dialog is shown. First, the dialog asks for your current password.

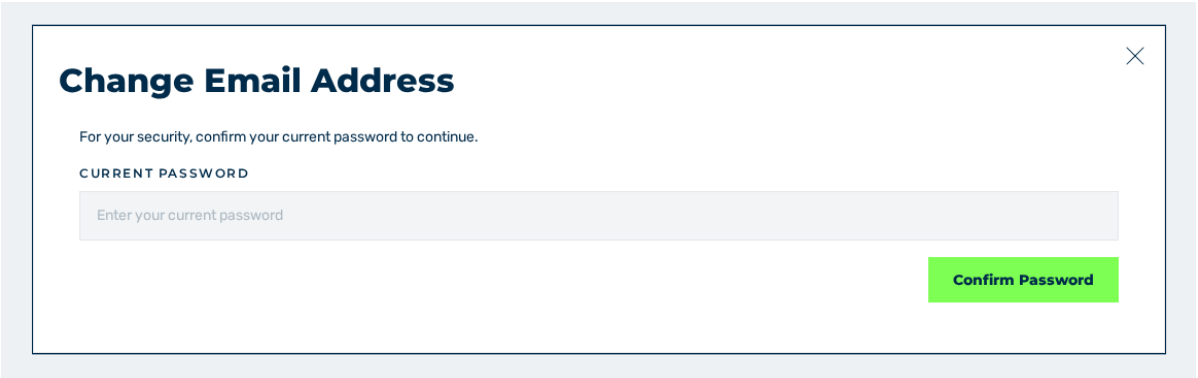


Figure 73: Change Email Address dialog

- In the **Current password** field, enter your current password.
- Click on the **Confirm Password** button.

  ↳ The dialog shows the field for the new e-mail address. The text gives the mailbox that receives the confirmation.

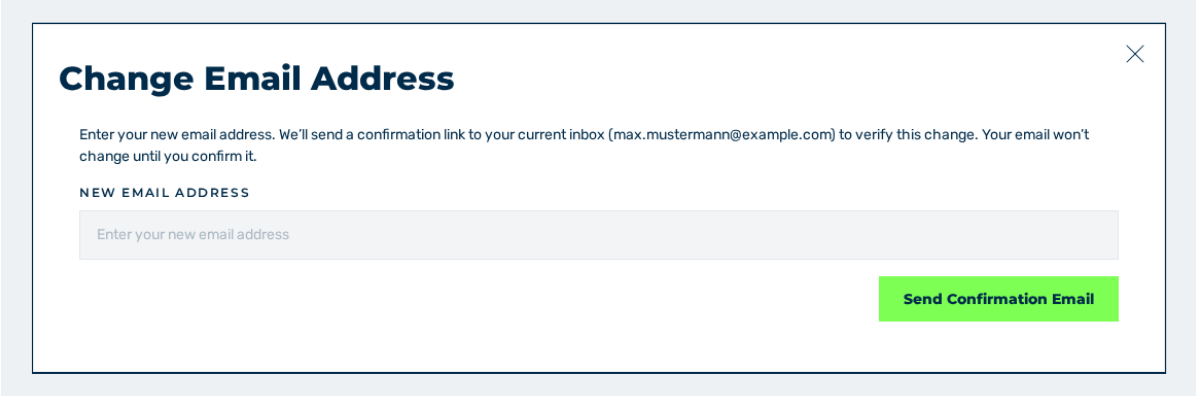


Figure 74: Change Email Address dialog with the New email address field

- In the **New email address** field, enter the new e-mail address.
- Click on the **Send Confirmation Email** button.

→ The system requests the change and shows the **Email change requested** message with the **Check your current inbox to confirm the change.** text.

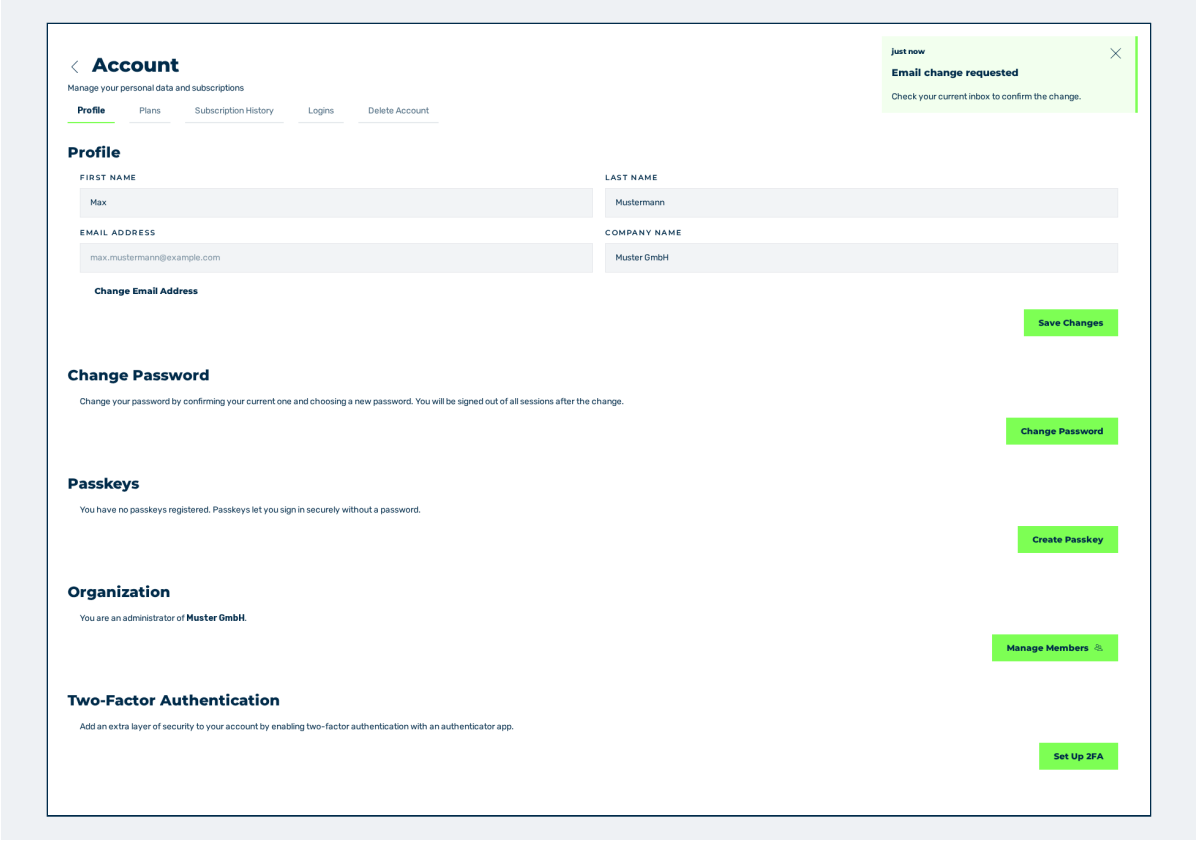


Figure 75: Profile tab with the Email change requested message

► Open the mailbox of the previous address and confirm the change.

➔ The new e-mail address is effective.

Until you confirm the change, the previous e-mail address stays applicable.

4.3.3 Change password

You change the password of your account in the **Profile** tab. After the change, the portal logs you out.

The following requirements must be met:

☑ You are logged in to the customer portal.

☑ You know your current password.

Proceed as follows to change the password:

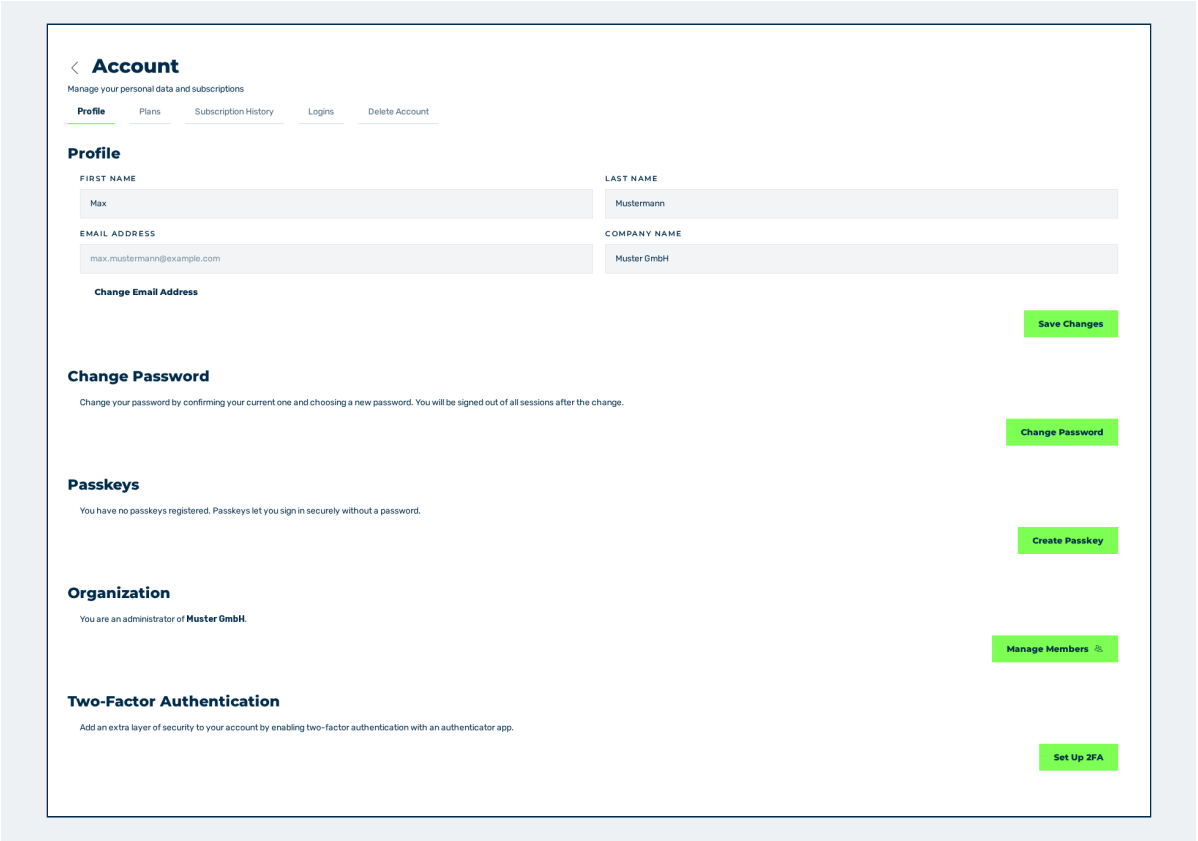


Figure 76: Profile tab

► In the sidebar, open the **Account** › **Profile** entry.

► Click on the **Change Password** button.

↳ The **Change Password** dialog is shown.

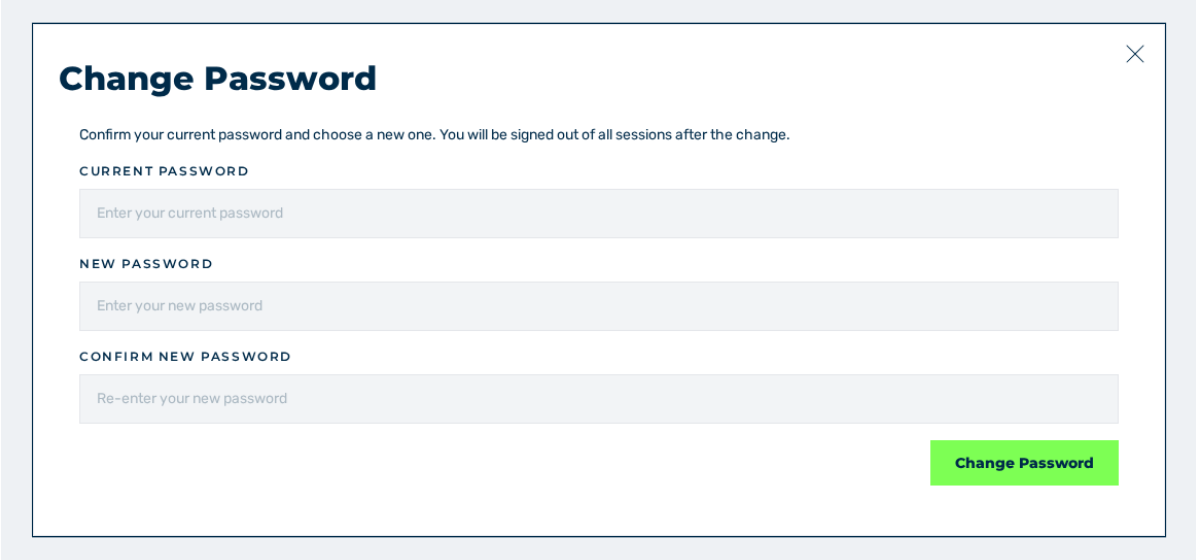


Figure 77: Change Password dialog

- In the **Current password** field, enter your current password.
- In the **New password** field, enter the new password.
- In the **Confirm new password** field, enter the new password again.
- Click on the **Change Password** button.

→ The system changes the password and shows the **Password changed** message with the **Your password was changed. Please sign in again.** text. The login page is shown.

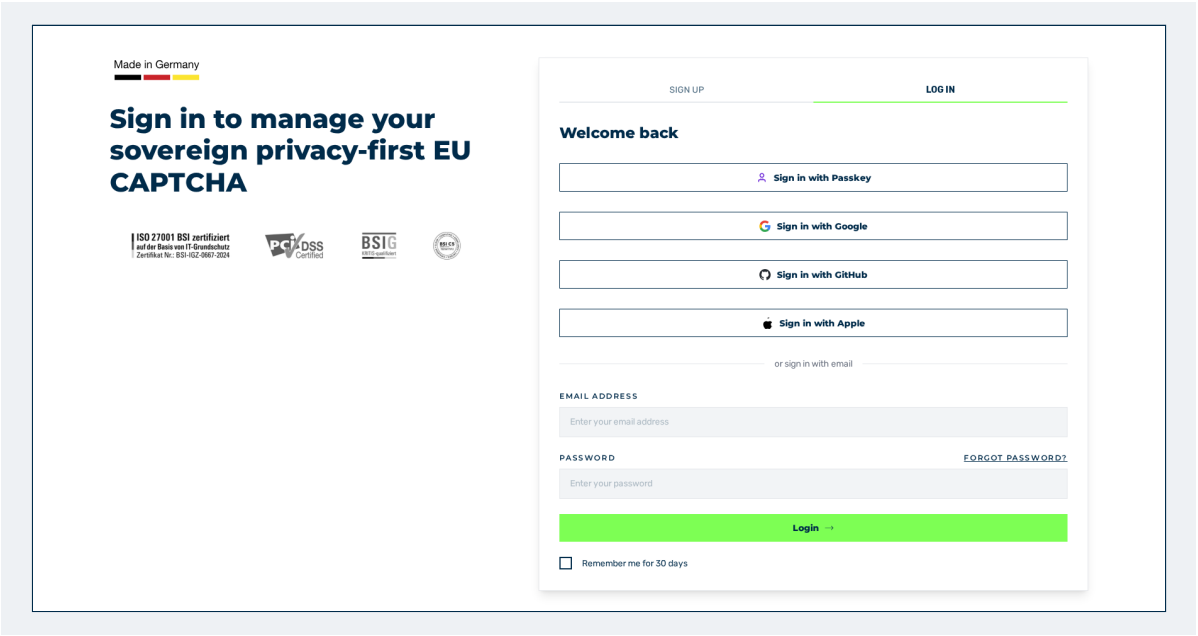


Figure 78: Welcome back view

► Log in with the new password.

See **Log in**.

4.3.4 Reset password

If the password is lost, you request an e-mail through the login page and set a new password with the link in it. The account stays as it is.

The following requirements must be met:

☑ You have access to the mailbox of your e-mail address.

Proceed as follows to reset the password:

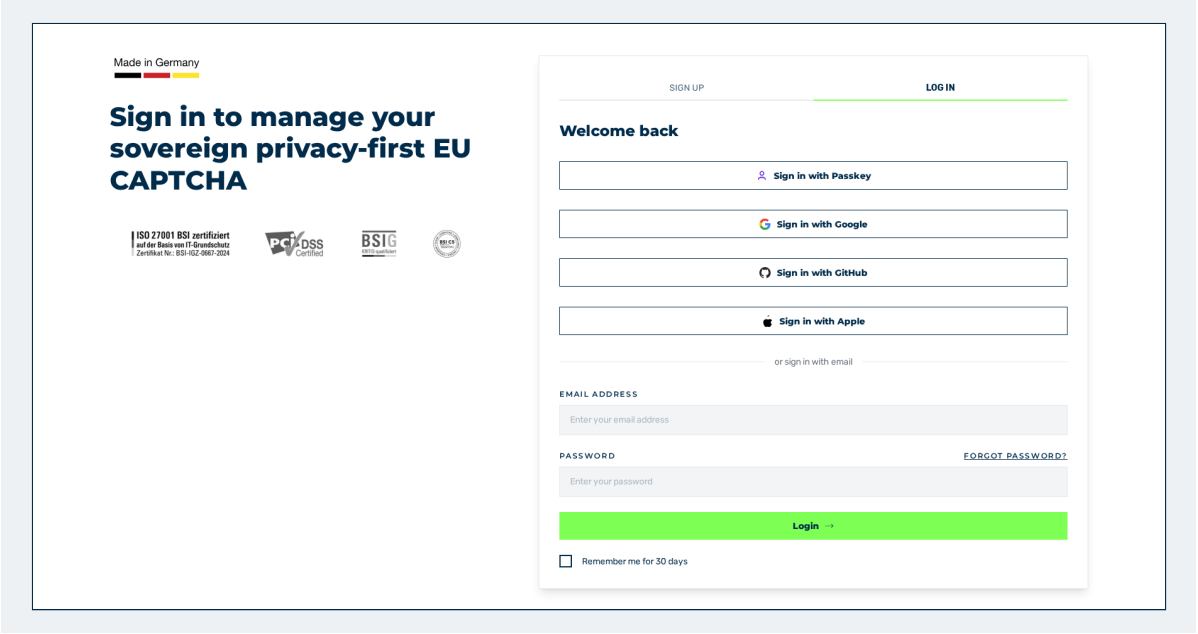


Figure 79: Welcome back view

- ► Open the login page of the customer portal.
- ► Click on the **Forgot Password?** link.
  - ↳ The **Reset Password** view opens.

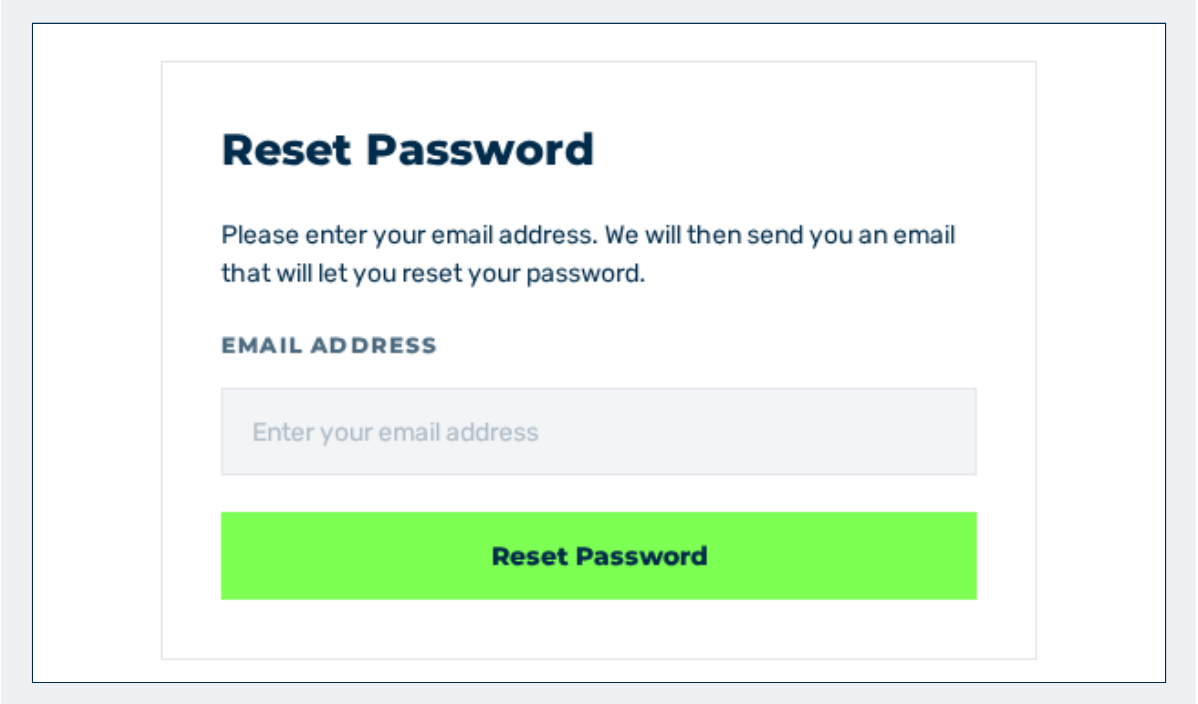


Figure 80: Reset Password view

- In the **Email address** field, enter your e-mail address.
- Click the **Reset Password** button.
  - ↳ The system sends an e-mail with a link to the given address.
- Follow the link in the e-mail.
  - ↳ The **Reset Password** view opens with the fields for the new password.

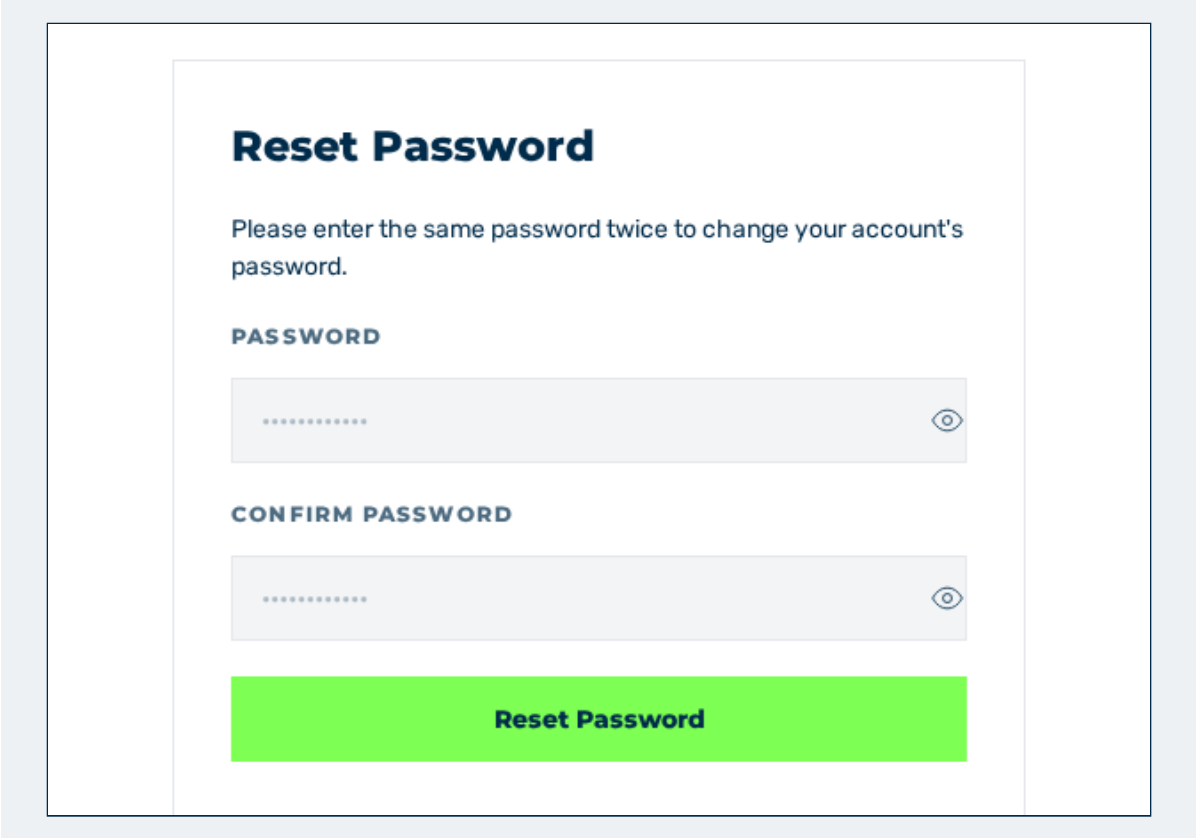


Figure 81: Reset Password view with the Password and Confirm Password fields

- ▶ Enter the new password in the **Password** field.
- ▶ Enter the same password in the **Confirm Password** field.
- ▶ Click the **Reset Password** button.
- ➔ The system sets the password and shows the login page. Log in with the new password.

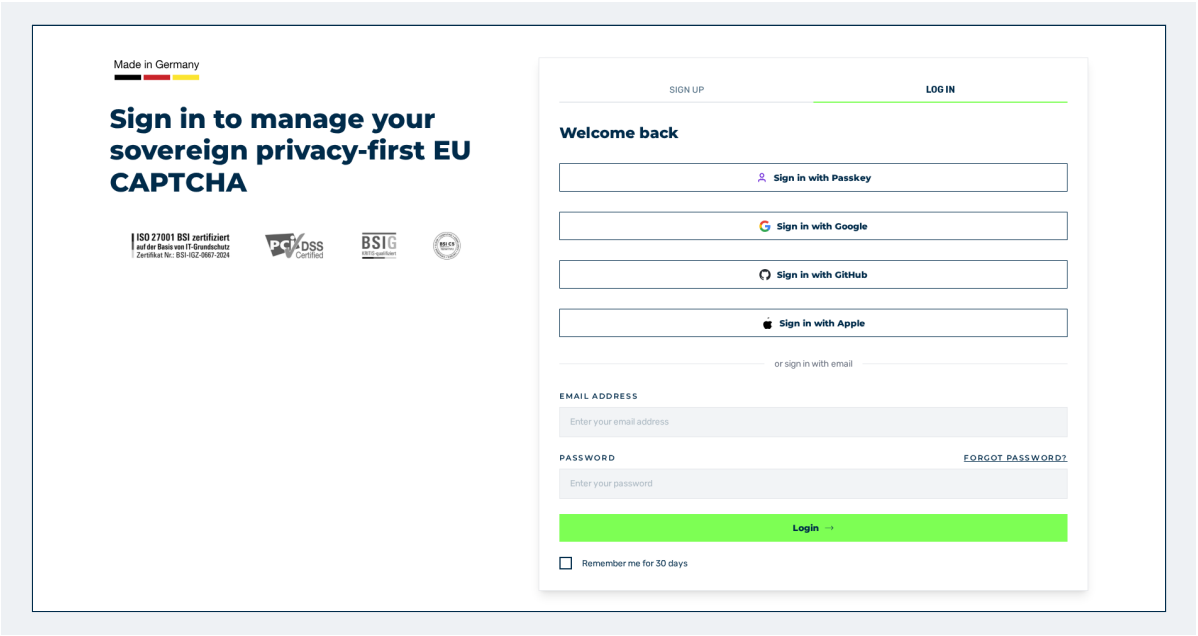


Figure 82: Welcome back view

4.3.4.1 Fields

These fields are available:

Field	Contents
Email address	E-mail address of the account, placeholder `Enter your email address`
Password	New password. A button shows and hides the entry.
Confirm Password	Repetition of the new password

After the dispatch, the system gives the note to also examine the spam folder. The **Back to Login** button goes back to the login.

4.3.4.2 Messages

During the reset, the following messages can occur:

Message	Cause	Remedy
We couldn't find an account for that email address.	No account exists for the email address.	Examine the address or create an account.

Message	Cause	Remedy
Passwords do not match, please enter the same password twice.	The two inputs do not match.	Enter the same password in both fields.
Your password reset link has expired. Please request a new one.	The link has expired.	Request a new e-mail.
This password reset link is invalid. Please request a new one.	The link is invalid.	Request a new e-mail.
This password was used recently. Please choose a different one.	The password has already been used for this account.	Select a different password.

For the new password, the same requirements apply as at the registration. See **Password policy**.

See **Log in**.

4.3.5 Password policy

The same policy applies to every password — at the registration, at the reset, and at the change in the **Profile** tab. The system examines every new password against the policy and rejects it with a message as soon as a rule is broken.

4.3.5.1 Rules

A password must fulfil the following rules:

Policy	Requirement
Length	At least 12 characters
Capital letters	At least one capital letter
Small letters	At least one small letter
Digits	At least one digit
Special characters	At least one special character

The following characters count as special characters:

```
#()^,.<>/\|{}[]-=_+$@!%*?&
```

The minimum length is adjustable on the server through the `PASSWORD_MIN_LENGTH` environment variable. Without a value for the variable, the value 12 applies.

4.3.5.2 Check during the entry

As soon as you fill in a password field, the list of the rules is below the field. Rules that are fulfilled are marked:

- **At least [number] characters**
- **At least one uppercase character**
- **At least one lowercase character**
- **At least one number**

- **At least one special character ([characters])**

A button at the right edge of the field shows and hides the password. The tooltip changes between **Show password** and **Hide password**.

4.3.5.3 Confirmation and reuse

In addition to the rules, the following applies:

- The confirmation must agree with the new password.
- A password that was used recently is rejected. At the reset and at the change, the system compares the new password with the passwords that were used last for the account. At the registration, the check does not apply, because no earlier password exists yet.

4.3.5.4 Messages

The following messages can occur:

Message	Cause
Your password does not meet the requirements.	The password breaks a rule for the length or the composition.
The passwords do not match.	The new password and the confirmation do not match.
Passwords do not match.	The confirmation differs from the new password. The message is directly below the confirmation field, before the form is sent.
This password was used recently. Please choose a different one.	The password has already been used for this account.
The current password is incorrect.	During the change, the previous password was entered incorrectly.
Too many attempts. Please try again later.	Too many attempts in quick succession.
Password login is not enabled for this account.	The account logs in through a provider.
	The link for the reset has expired.

Message	Cause
Your password reset link has expired. Please request a new one.	
This password reset link is invalid. Please request a new one.	The link for the reset is invalid.
We couldn't find an account for that email address.	No account exists for the email address.

A password alone does not give the account sufficient protection. Also set up a passkey or the two-factor authentication.

See **Change password**.

See **Reset password**.

4.3.6 Create a passkey

A passkey makes the login without a password possible. The device confirms the login, for example with a fingerprint, face recognition, or the device PIN.

The following requirements must be met:

☑ You are logged in to the customer portal.

☑ Your device and your browser support passkeys.

Proceed as follows to create a passkey:

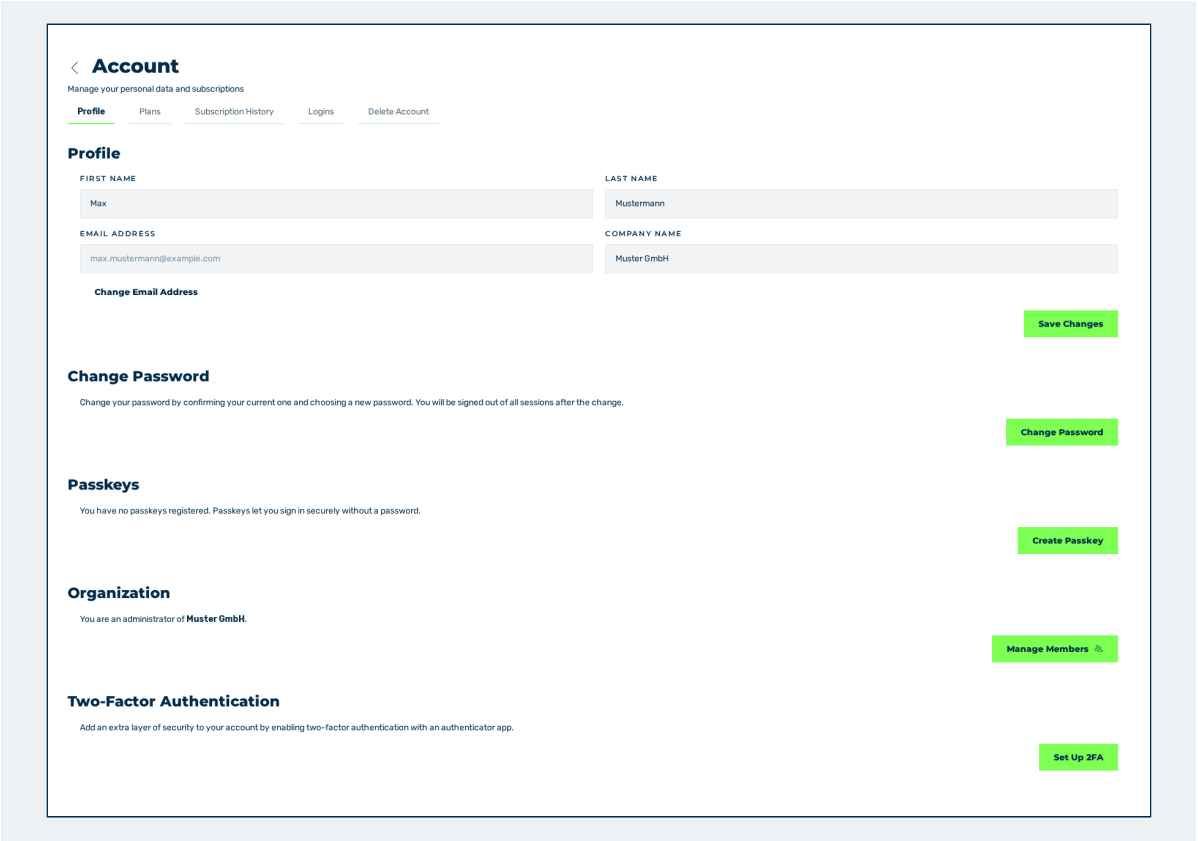


Figure 83: Profile tab

► In the sidebar, open the **Account** › **Profile** entry.

► Click on the **Create Passkey** button.

↳ The browser asks for the confirmation by the device.

► Confirm the request on your device.

➔ The system creates the passkey and shows the **Passkey registered successfully!** message.

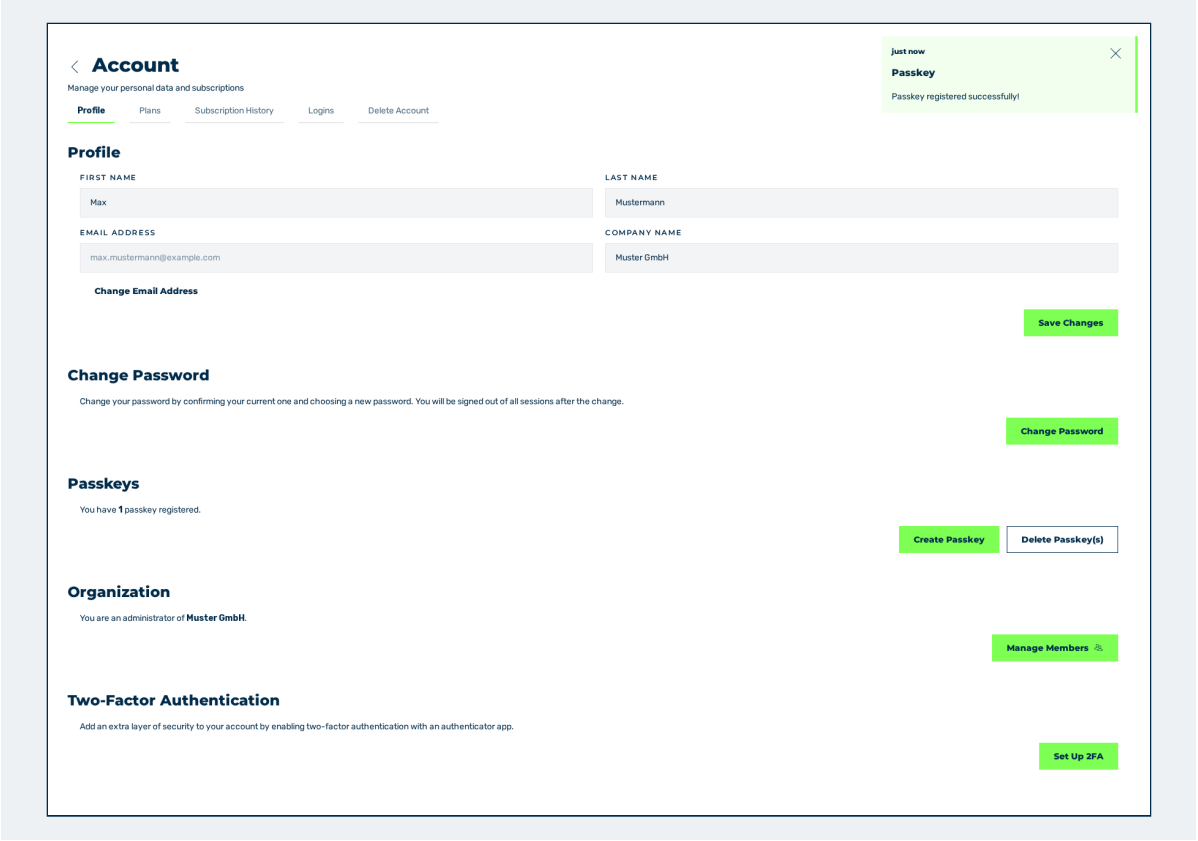


Figure 84: Profile tab with the passkey that was created

To remove all passkeys, click on the **Delete Passkey(s)** button. See **Deleting passkeys**.

At this time, the login with a passkey has a malfunction. Until the correction, use the login with the e-mail address and the password. See **The login with a passkey fails**.

4.3.7 Deleting passkeys

The **Delete Passkey(s)** button removes all passkeys of the account at once. Single passkeys cannot be deleted.

The following requirements must be met:

☑ You are logged in to the customer portal.

☑ At least one passkey is created for your account.

Proceed as follows to delete all passkeys:

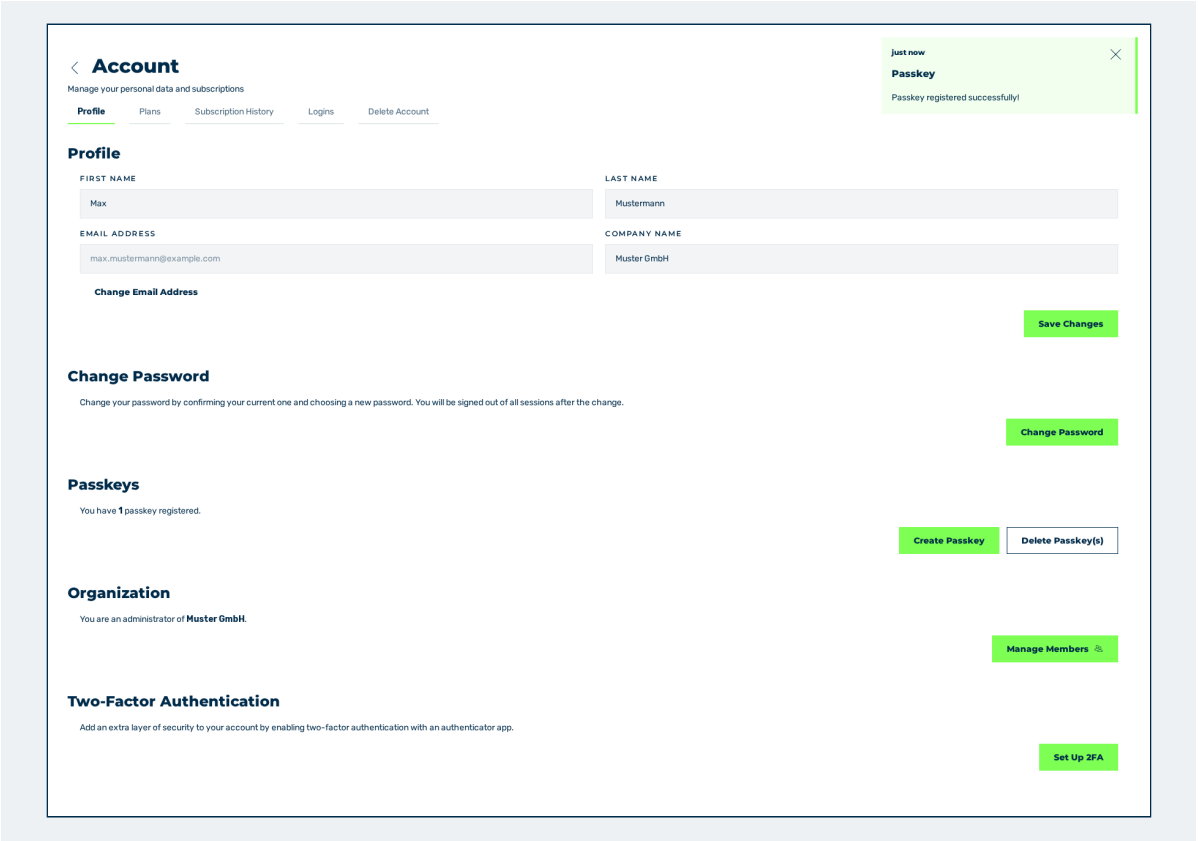


Figure 85: Profile tab with the passkey that was created

► In the sidebar, open the **Account** › **Profile** entry.

► Click on the **Delete Passkey(s)** button in the **Passkeys** area.

➔ The system deletes all passkeys and shows the **All passkeys have been deleted.** message.

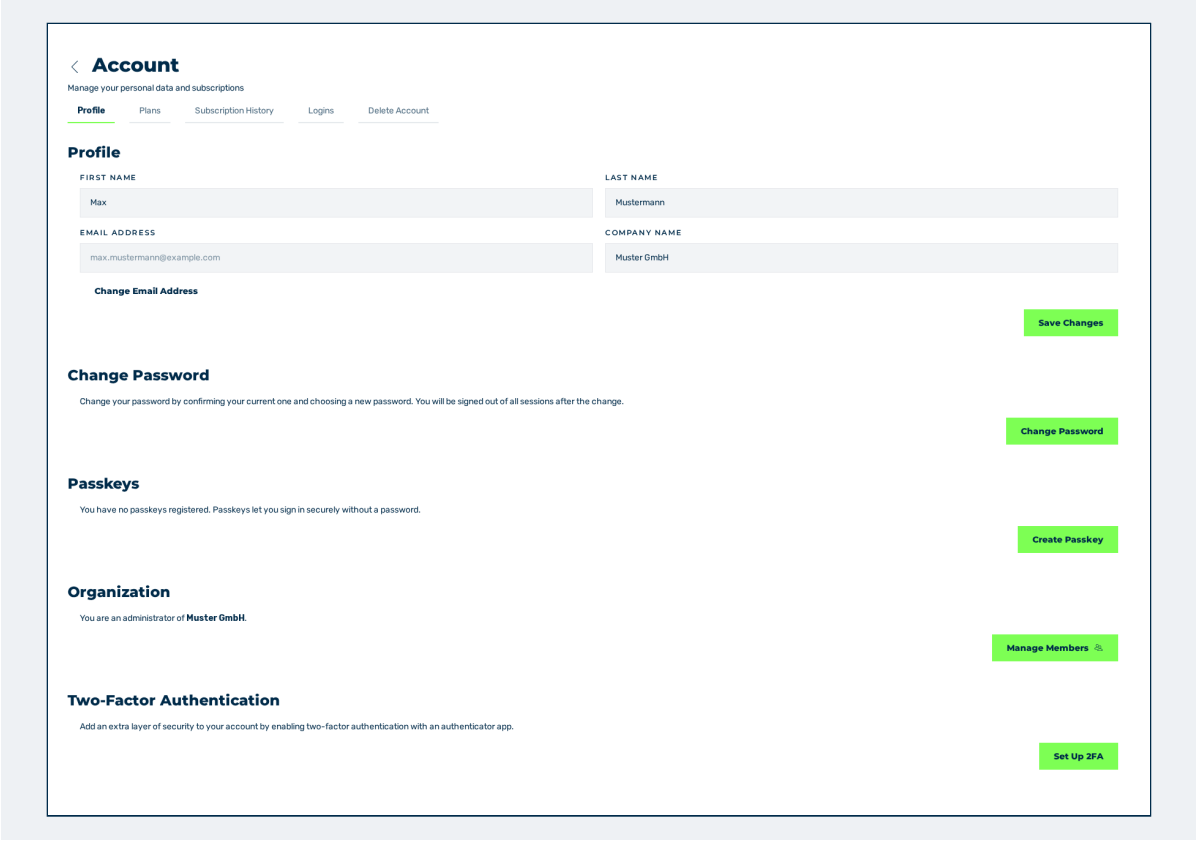


Figure 86: Profile tab without passkeys

The **Delete Passkey(s)** button appears only with the first passkey that is created.

After the deletion, the login with a passkey is not possible any more. Log in with the e-mail address and the password or through a provider.

See **Create a passkey**.

4.3.8 Set up two-factor authentication

For each login, the two-factor authentication also asks for a six-digit code from an authenticator app.

The following requirements must be met:

☑ You are logged in to the customer portal.

☑ An authenticator app is installed on your mobile device.

Proceed as follows to set up the two-factor authentication:

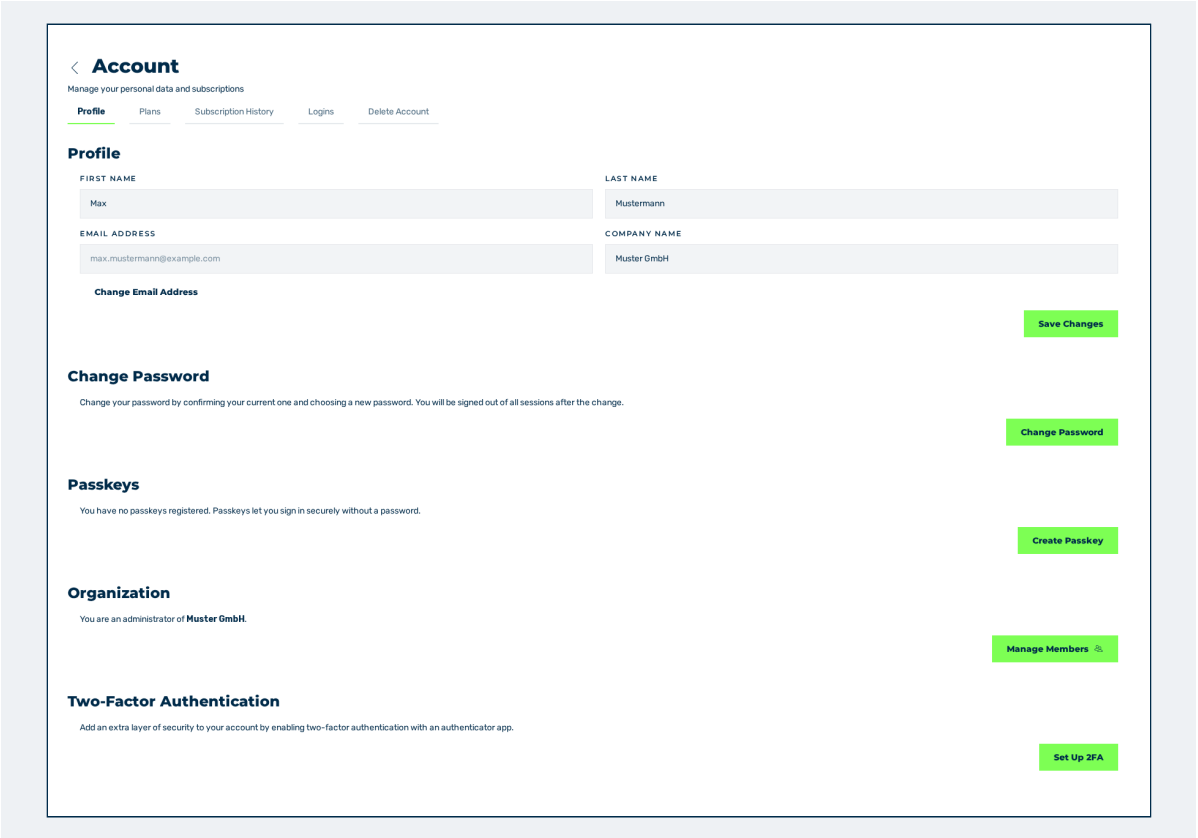


Figure 87: Profile tab

► In the sidebar, open the **Account** › **Profile** entry.

► Click on the **Set Up 2FA** button.

↳ The **Set Up Two-Factor Authentication** view is shown.

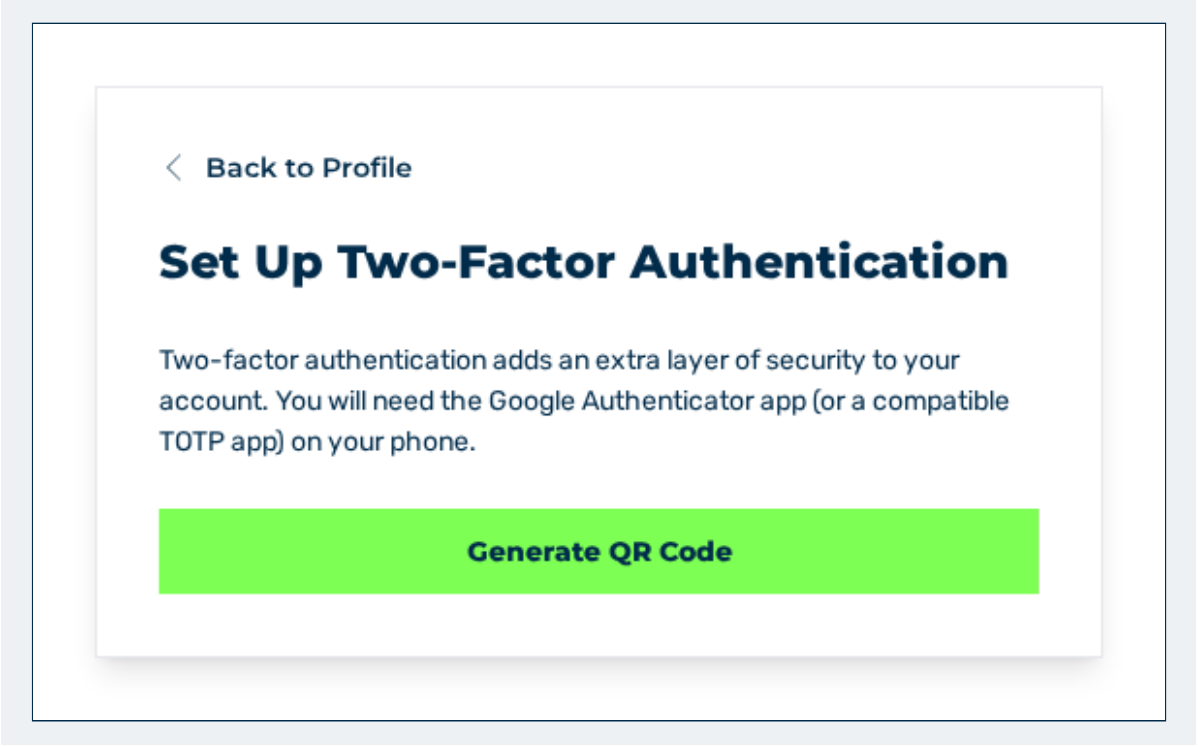


Figure 88: Set Up Two-Factor Authentication view

- Click on the **Generate QR Code** button.
  - ↳ The system shows the QR code. Below it, the secret is shown as text below **Or enter this secret manually:**, for example XXXX XXXX XXXX XXXX .

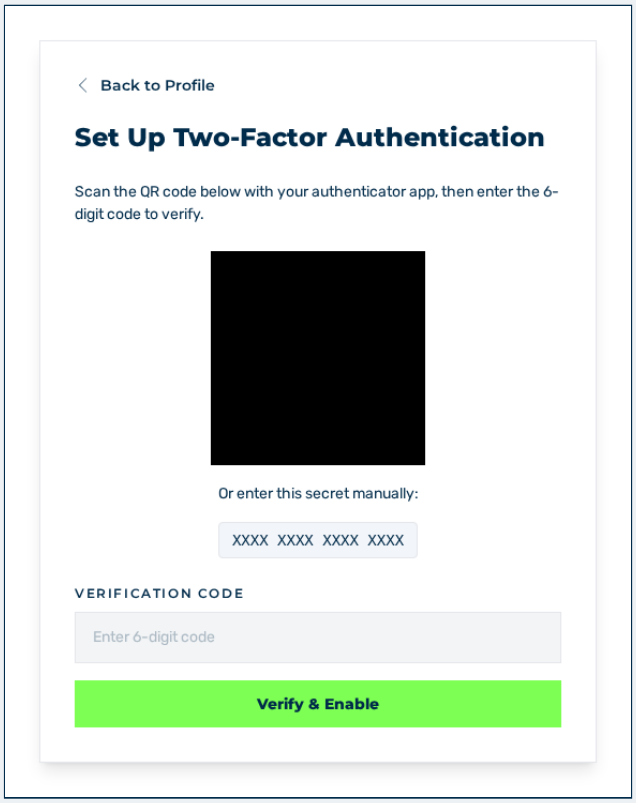


Figure 89: Set Up Two-Factor Authentication view with the masked QR code

- ► Scan the QR code with your authenticator app. As an alternative, enter the secret in the app.
- ► In the **Verification code** field, enter the six-digit code from the app.
- ► Click on the **Verify & Enable** button.
- ➔ The system activates the two-factor authentication and shows the message **Two-factor authentication has been enabled successfully.**

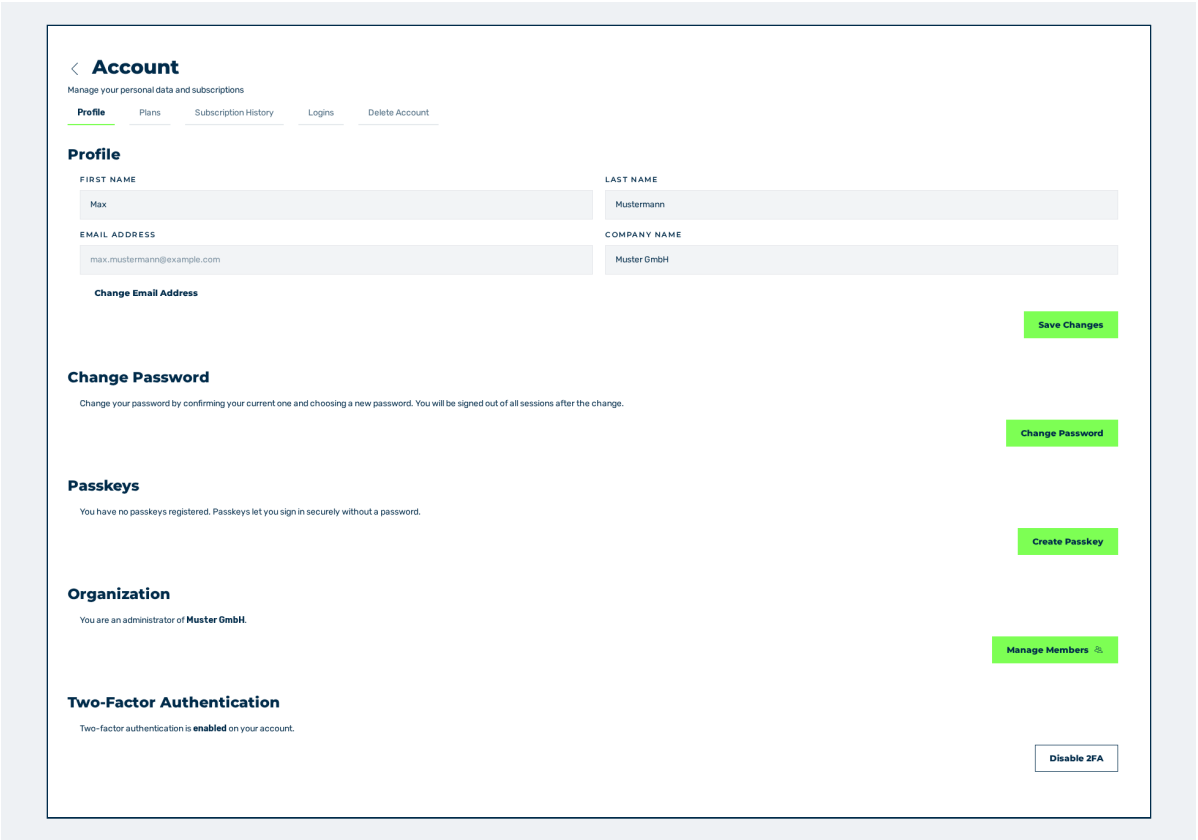


Figure 90: Profile tab with the two-factor authentication activated

► Click on the **Back to Profile** button.

Keep the secret in a safe location. Without the authenticator app and without the secret, the login is not possible.

See **Deactivating the two-factor authentication**.

4.3.9 Deactivating the two-factor authentication

After the deactivation, the e-mail address and the password are sufficient for the login.

The following requirements must be met:

☑ You are logged in to the customer portal.

☑ The two-factor authentication is active for your account.

Proceed as follows to deactivate the two-factor authentication:

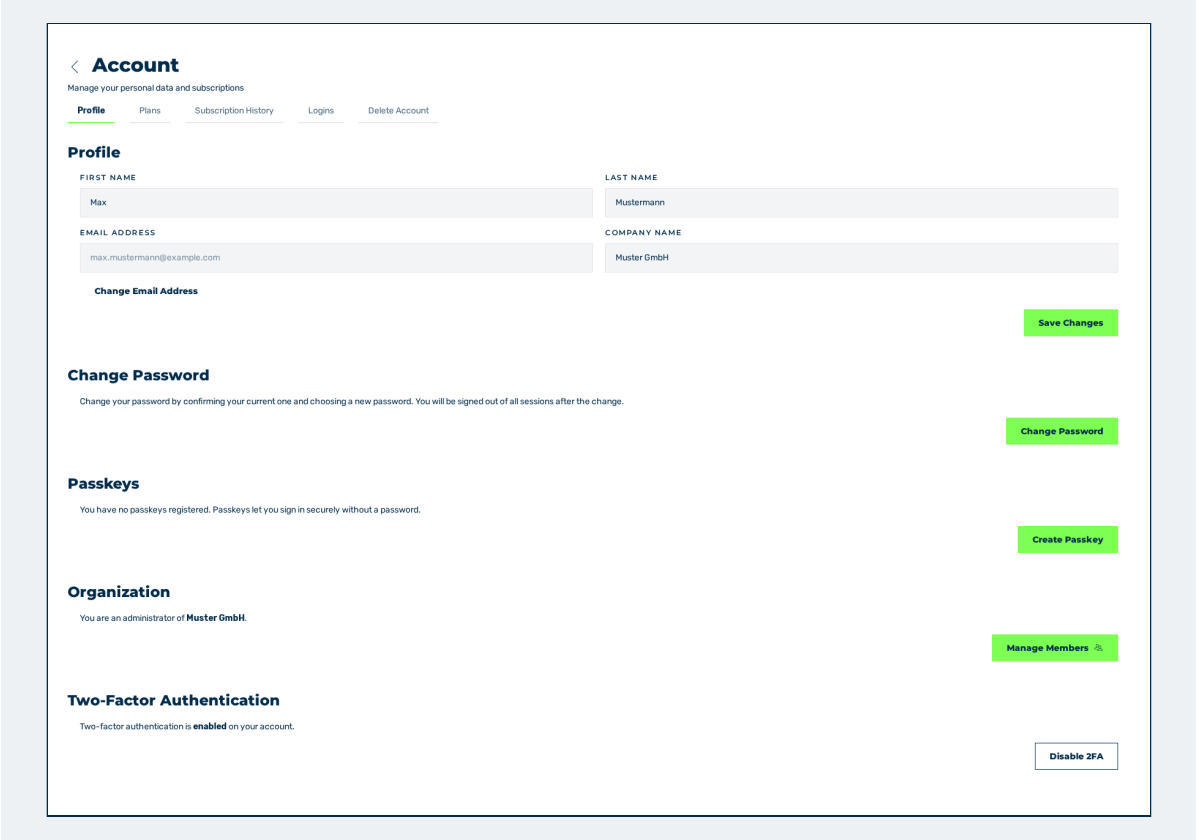


Figure 91: Profile tab

► In the sidebar, open the **Account** › **Profile** entry.

► Click on the **Disable 2FA** button.

↳ The **Disable Two-Factor Authentication** dialog is shown.

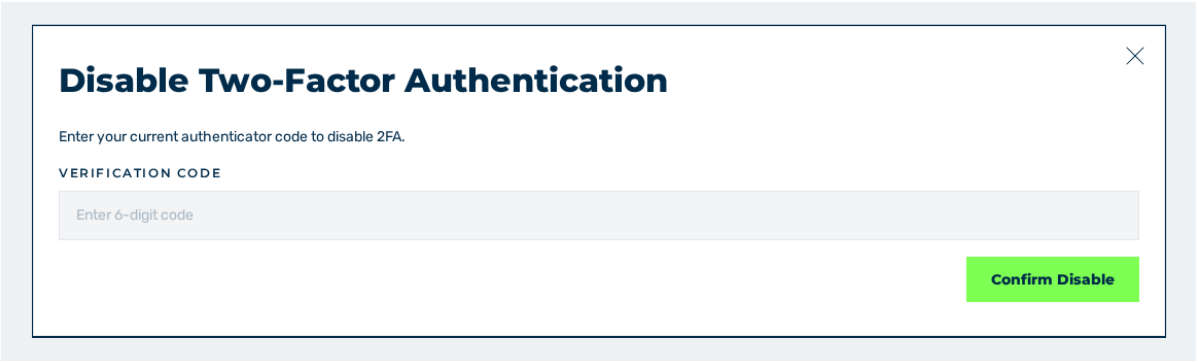


Figure 92: Disable Two-Factor Authentication dialog

- In the **Verification code** field, enter the six-digit code from your authenticator app.
- Click on the **Confirm Disable** button.

➔ The system deactivates the two-factor authentication and shows the **Two-factor authentication has been disabled.** message.

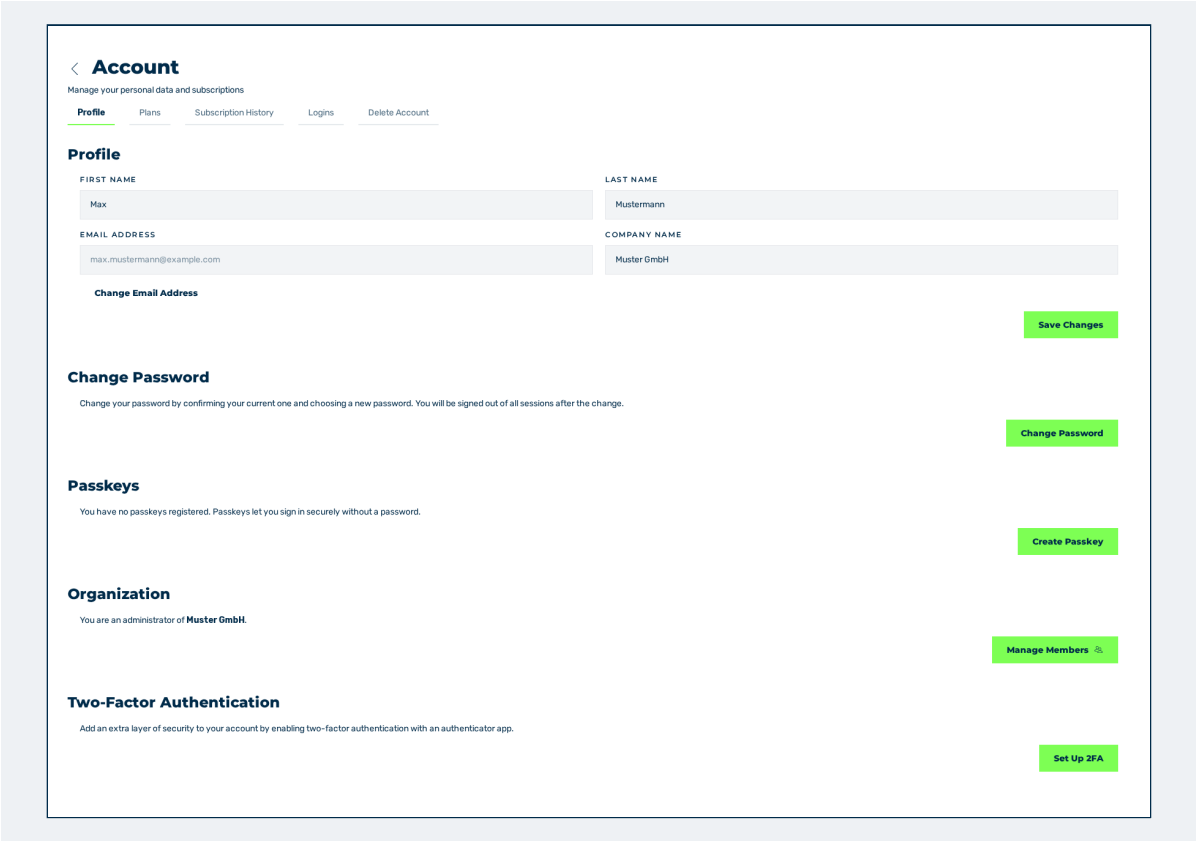


Figure 93: Profile tab without the two-factor authentication

Without the two-factor authentication, only the password protects your account. Examine the log of the logins regularly. See **Logins tab**.

4.3.10 Delete account

The **Delete Account** tab deletes your account permanently. With the account, all sitekeys go, and with them the protection of the related websites.

The following requirements must be met:

☑ You are logged in to the customer portal.

☑ No subscription is active for your account any more. See **Cancel a subscription**.

☑ No payment is repeated for your account any more.

Proceed as follows to delete the account:

Figure 94: Delete Account tab

► In the sidebar, open the **Account** › **Profile** entry.

► Change to the **Delete Account** tab.

► Click on the **Delete Account Permanently** button.

↳ The **Delete Account Permanently** dialog opens.

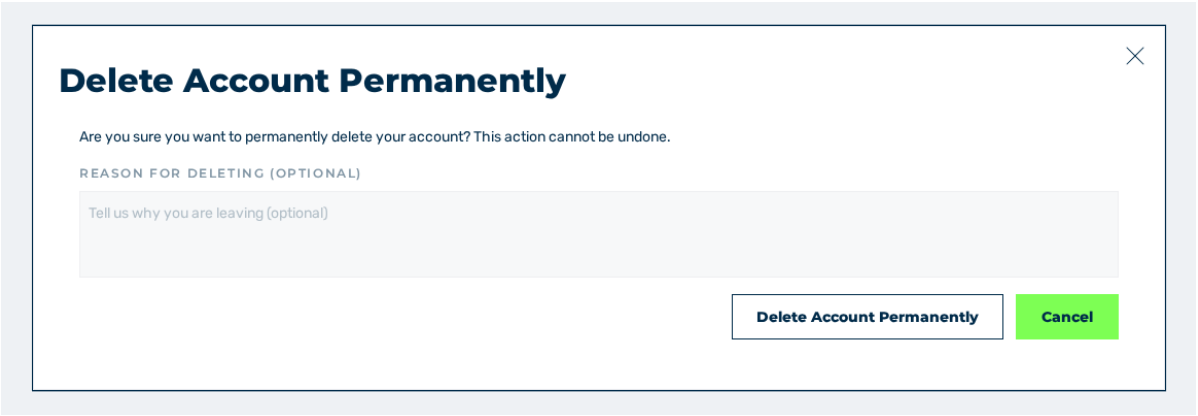


Figure 95: Delete Account Permanently dialog

- ▶ If required, enter a reason in the **Reason for deleting (optional)** field.
- ▶ Click on the **Delete Account Permanently** button.
- ➔ The system deletes the account and logs you out.

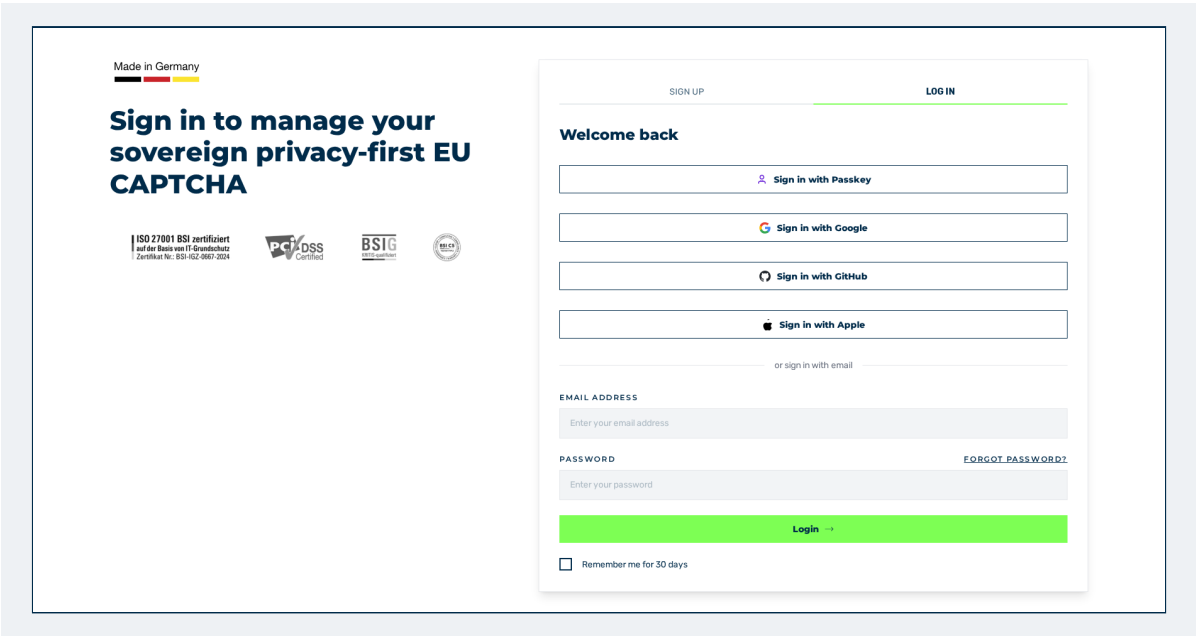


Figure 96: Welcome back view after the deletion

The reason is voluntary and holds 2048 characters at most. To stop the operation, click on the **Cancel** button in the dialog.

The deletion cannot be undone. Your account and your personal data are deleted permanently, all sitekeys and the related CAPTCHA data are removed, and a new login is not possible.

4.3.10.1 Blocked deletion

As long as one of the following conditions applies, the system rejects the deletion:

Condition	Title of the dialog	Measure
A subscription is active.	**Cancel your subscription first**	Cancel the subscription. The **Go to Subscription History** button opens the related tab.
A payment is repeated again.	**A payment is still being retried**	Wait until the payment is complete.

The **Close** button closes the dialog.

See **Delete Account tab**.

4.3.11 Cookie and tracking settings

The customer portal evaluates the use of the interface with analysis tools. At the first call, the **We use tracking tools to improve your experience** dialog asks for the consent. The consent is voluntary and can be revoked at any time.

The settings apply to the customer portal only. The protection itself works without cookies: the widget sets no cookies on the protected website and stores nothing in the browser of the visitors. See **How EU CAPTCHA works**.

4.3.11.1 Categories

The consent is divided into the following categories:

Category	Content	Default
Necessary	Basic functions, security and operability of the website. Cannot be deselected.	**Enabled**
User Analytics	Evaluation of the use of website and product through Matomo and Userpilot.	**Disabled**
Session Replay	Recording of pseudonymised session data such as mouse movements, clicks and scrolling behaviour through Userpilot.	**Disabled**

Session Replay requires **User Analytics**. If you switch on **Session Replay**, **User Analytics** is switched on as well; if you switch off **User Analytics**, **Session Replay** is switched off as well.

4.3.11.2 Giving the consent

Proceed as follows to give the consent in the first dialog:

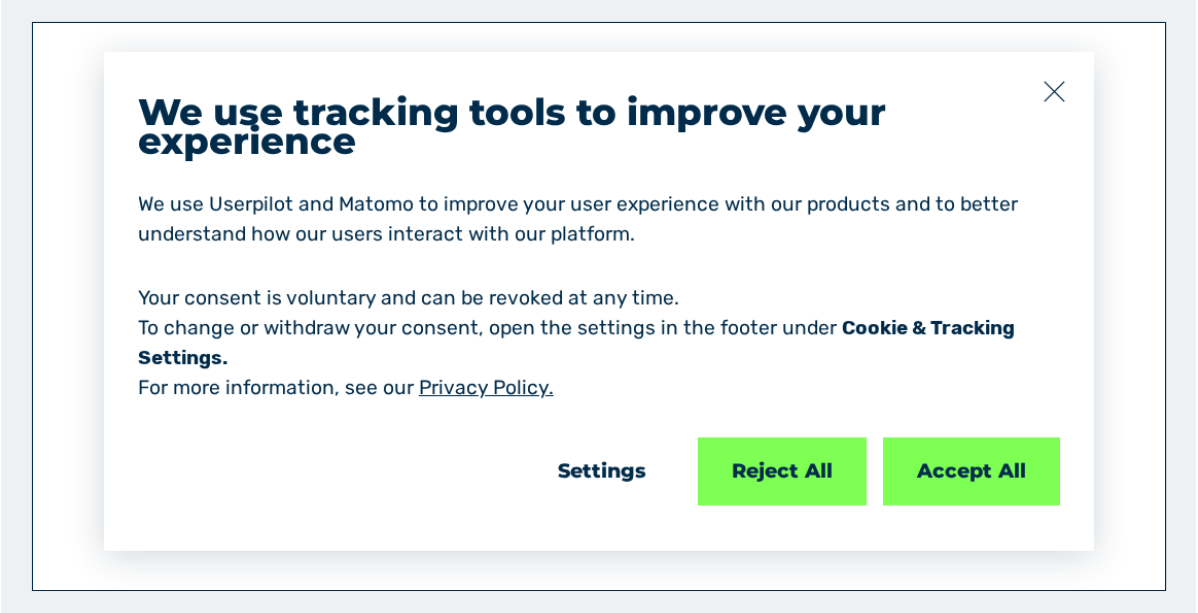


Figure 97: We use tracking tools to improve your experience dialog

► Click on one of the following buttons:

Button	Effect
Accept All	Switches on all categories.
Reject All	Switches off **User Analytics** and **Session Replay**.
Settings	Shows the single categories for selection.

➔ The system saves the selection and closes the dialog.

4.3.11.3 Selecting single categories

Proceed as follows to select single categories:

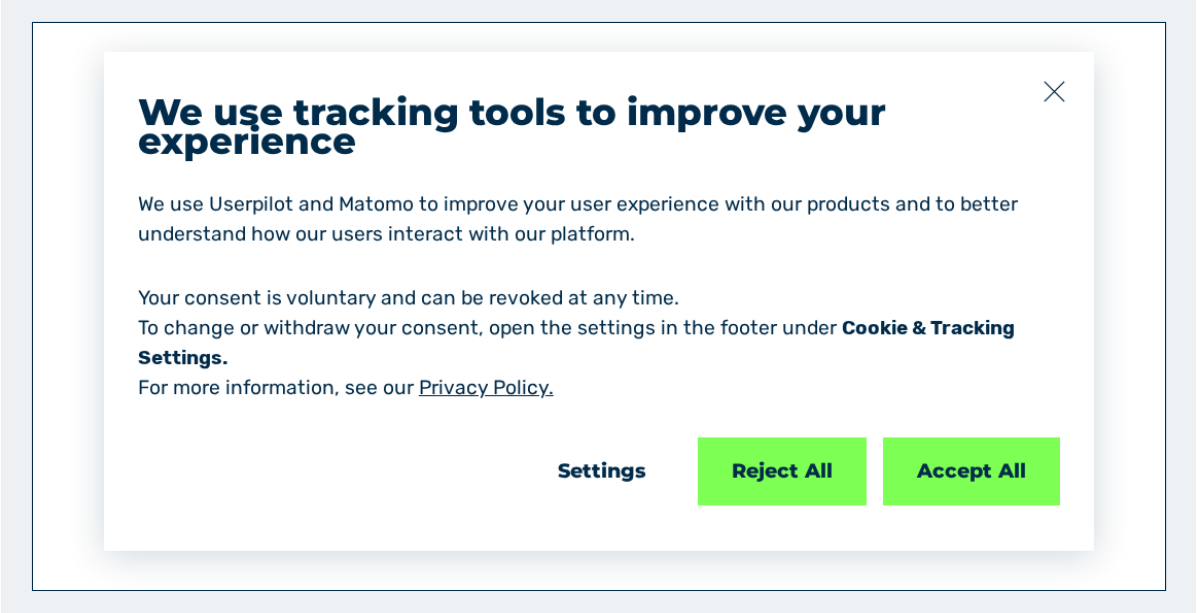


Figure 98: Dialog with the categories

- ► Click on the **Settings** button in the dialog.
  - ↳ The dialog shows the three categories, each with one switch.

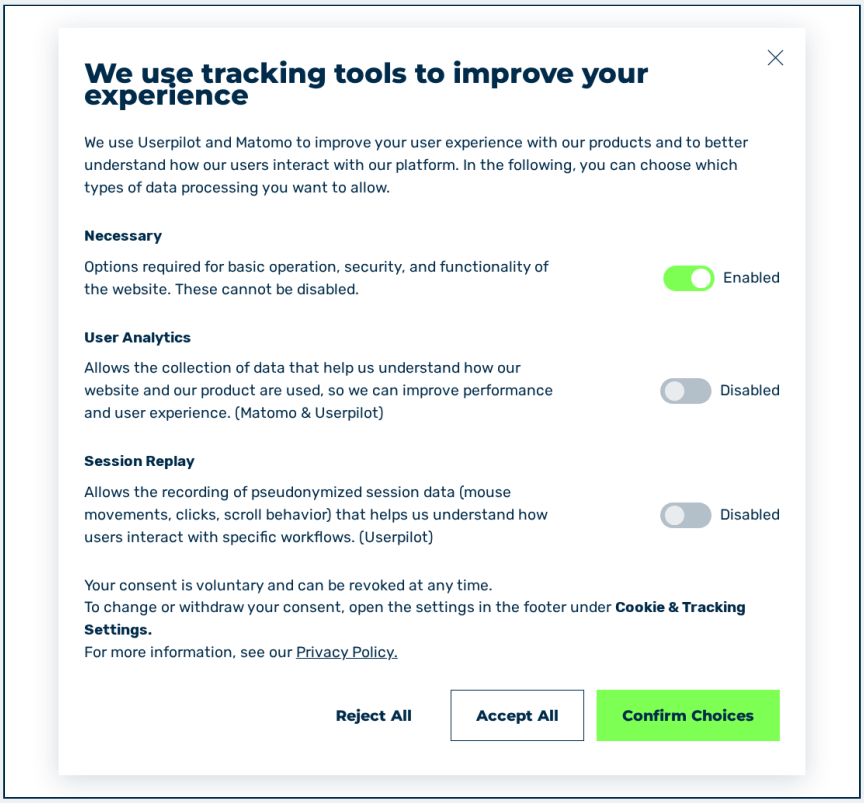


Figure 99: Dialog with the switches of the three categories

- ► Set the required categories to **Enabled**.
- ► Click on the **Confirm Choices** button.
- ➔ The system uses only the selected categories and closes the dialog.

4.3.11.4 Changing or revoking the consent

Proceed as follows to change the consent subsequently:

- ► In the footer, click on the **Cookie & Tracking Settings** entry.
  - ↳ The dialog with the categories opens.

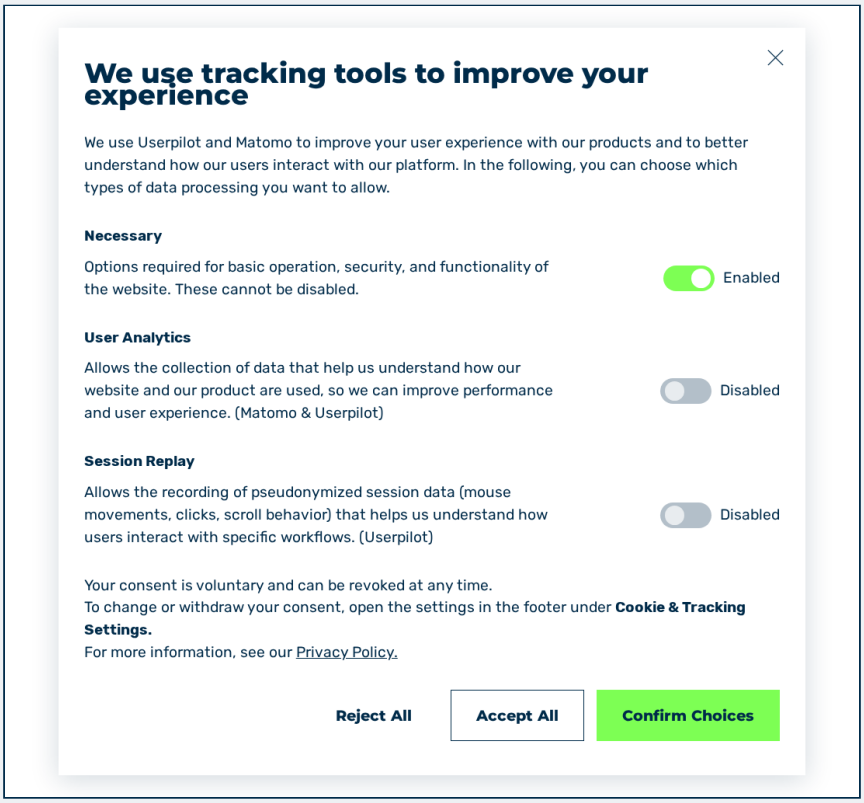


Figure 100: Dialog with the switches of the three categories

- ► Change the switches of the categories.
- ► Click on the **Confirm Choices** button.
- ➔ The system applies the changed selection immediately.

The **privacy policy** of Myra Security GmbH gives details on the processing.

4.4 Organization

4.4.1 Inviting a member

You invite other persons into your organization. The account that was invited gets an e-mail and confirms its admission with it.

The following requirements must be met:

☑ Your account has the **Org Admin** role.

☑ You know the e-mail address of the person to invite.

Proceed as follows to invite a member:

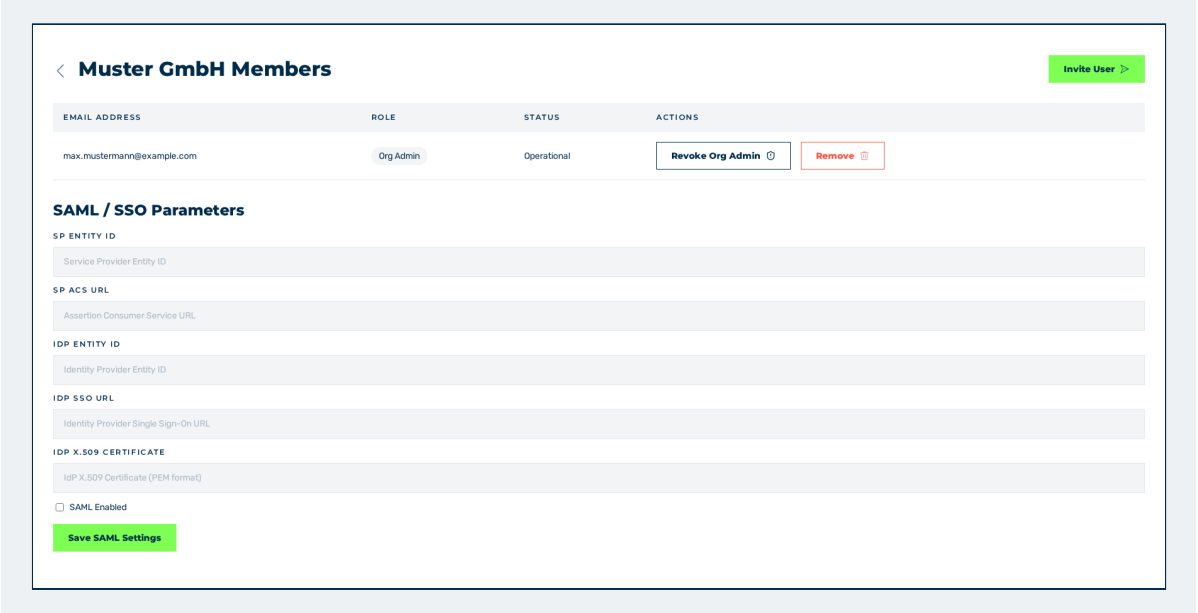


Figure 101: Organization view

► In the sidebar, open the **Account** › **Profile** entry and click on the **Manage Members** button in the **Organization** area.

► Click on the **Invite User** button.

↳ The **Invite User** dialog is shown.

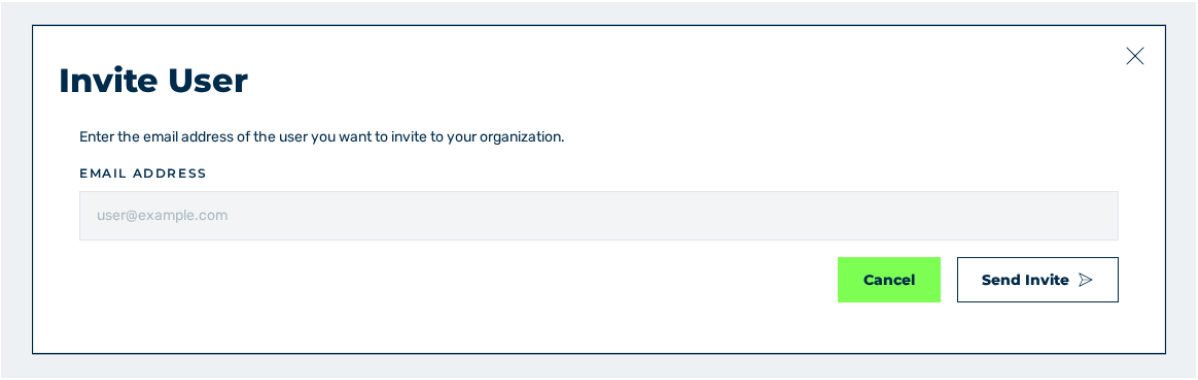


Figure 102: Invite User dialog

- ▶ In the **Email address** field, enter the e-mail address.
- ▶ Click on the **Send Invite** button.
- ➔ The system sends the invitation and shows the **Invitation sent.** message. The entry is shown in the table with the **Pending Verification** status.

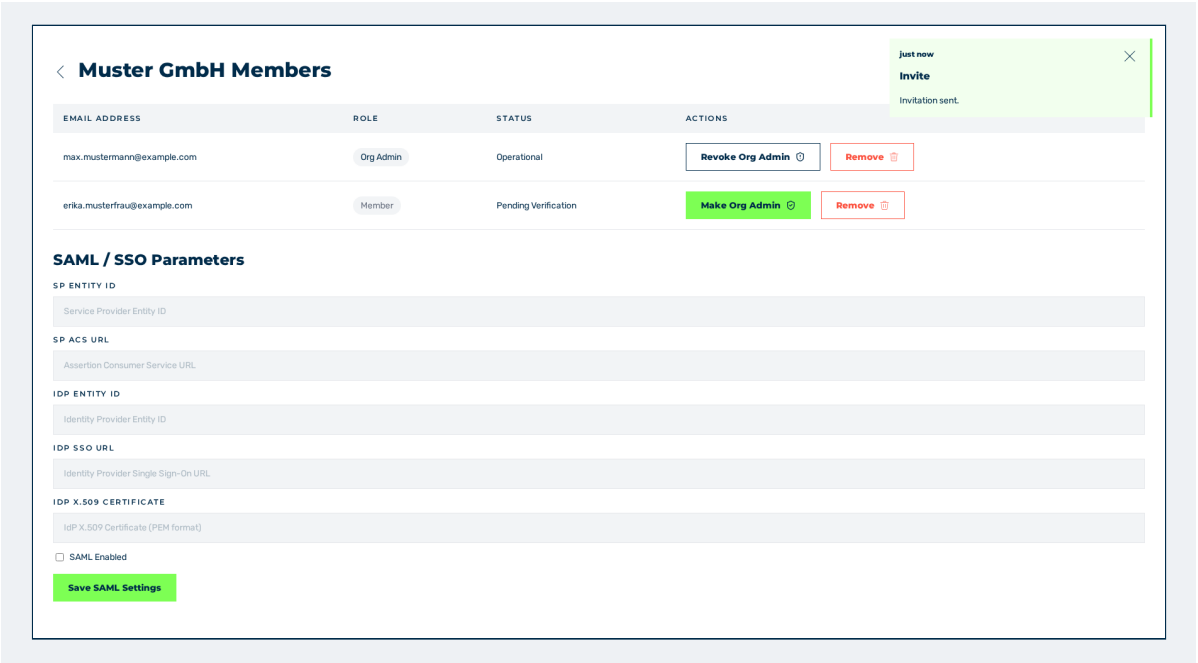


Figure 103: Organization view with the new entry

As soon as the person confirms the invitation, the status changes to **Operational**.

See **Organization view**.

4.4.2 Assign and revoke the Org Admin role

The **Org Admin** role permits a member to manage other members and to change the settings for SAML and SSO.

The following requirements must be met:

☑ Your account has the **Org Admin** role.

☑ The member is already part of the organization.

4.4.2.1 Assign the role

Proceed as follows to assign the role:

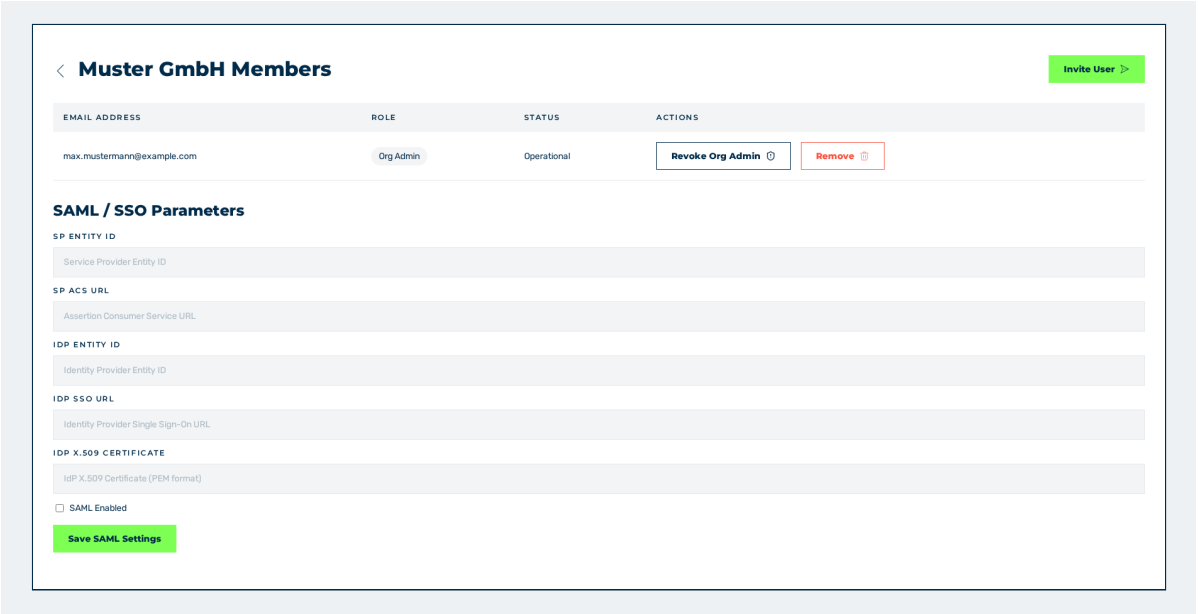


Figure 104: Organization view

► In the sidebar, open the **Account** › **Profile** entry and click on the **Manage Members** button in the **Organization** area.

► In the row of the member, click on the **Make Org Admin** button in the **Actions** column.

↳ The **Grant Org Admin** dialog is shown.

Figure 105: Grant Org Admin dialog

- ► Click on the **Confirm** button.
- ➔ The system assigns the role and closes the dialog. The **Role** column shows the **Org Admin** value.

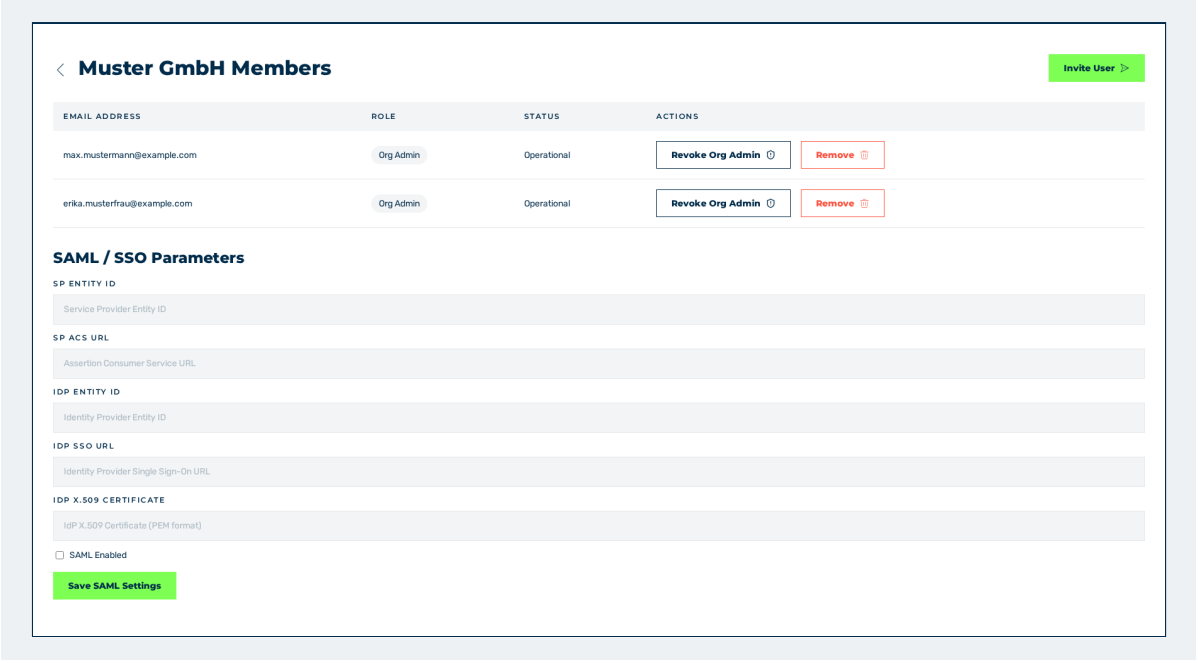


Figure 106: Organization view with the role that was assigned

To stop the procedure, click on the **Close** button.

4.4.2.2 Revoke the role

Proceed as follows to revoke the role:

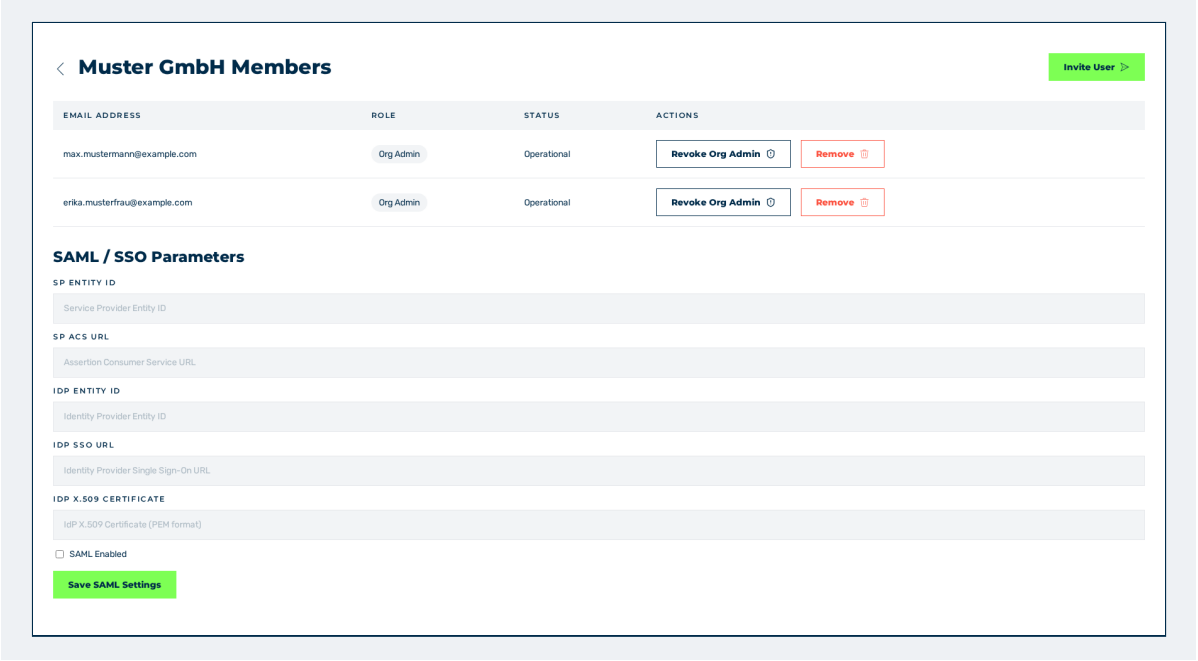


Figure 107: Organization view with the role that was assigned

► In the row of the member, click on the **Revoke Org Admin** button in the **Actions** column.

↳ The **Revoke Org Admin** dialog is shown.

Figure 108: Revoke Org Admin dialog

► Click on the **Confirm** button.

➔ The system revokes the role and closes the dialog. The **Role** column shows the **Member** value.

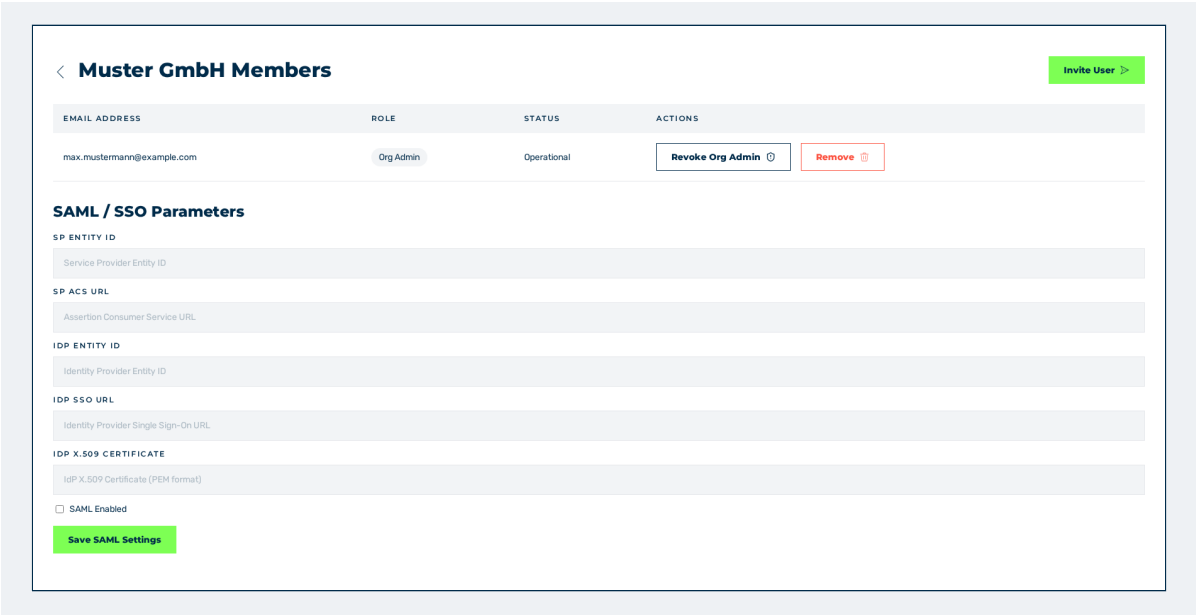


Figure 109: Organization view without the role

See **Removing a member**.

See **Organization view**.

4.4.3 Removing a member

You remove a member from your organization. Thus the account loses the access to the sitekeys of the organization.

The following requirements must be met:

☑ Your account has the **Org Admin** role.

☑ The member is part of the organization.

Proceed as follows to remove a member:

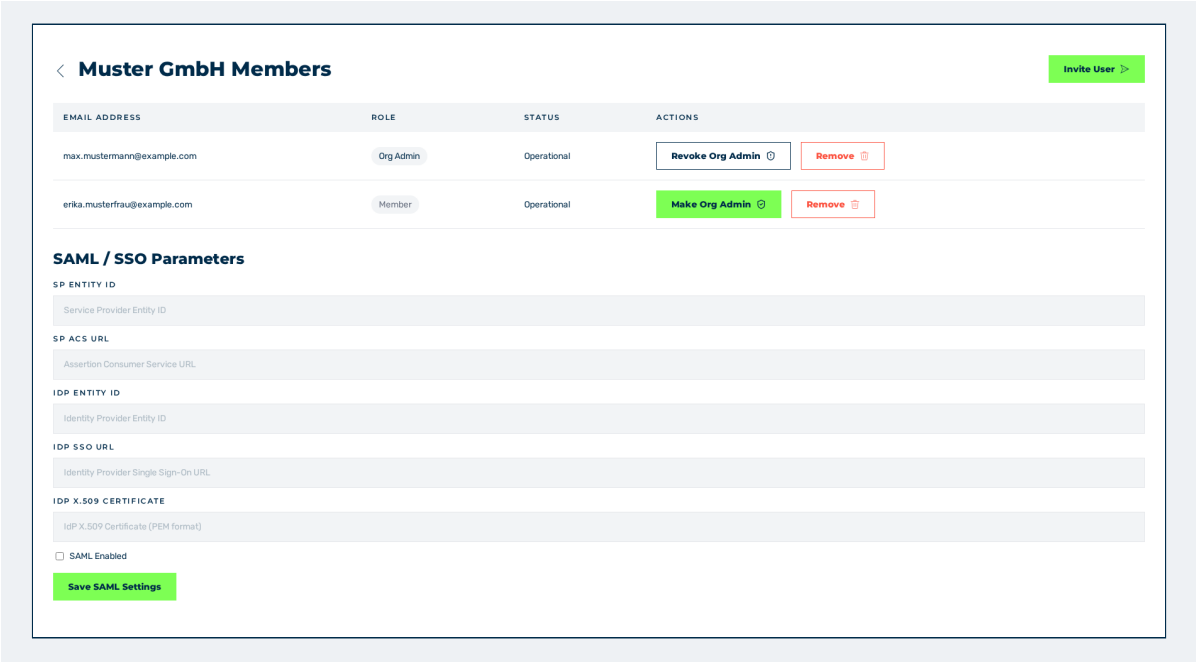


Figure 110: Organization view with two members

► In the sidebar, open the **Account** › **Profile** entry and click on the **Manage Members** button in the **Organization** area.

► In the row of the member, click on the **Remove** button in the **Actions** column.

↳ The **Remove Member** dialog is shown. The dialog gives the e-mail address of the member.

Figure 111: Remove Member dialog

► Click on the **Confirm** button.

➔ The system removes the member and closes the dialog. The entry is removed from the table.

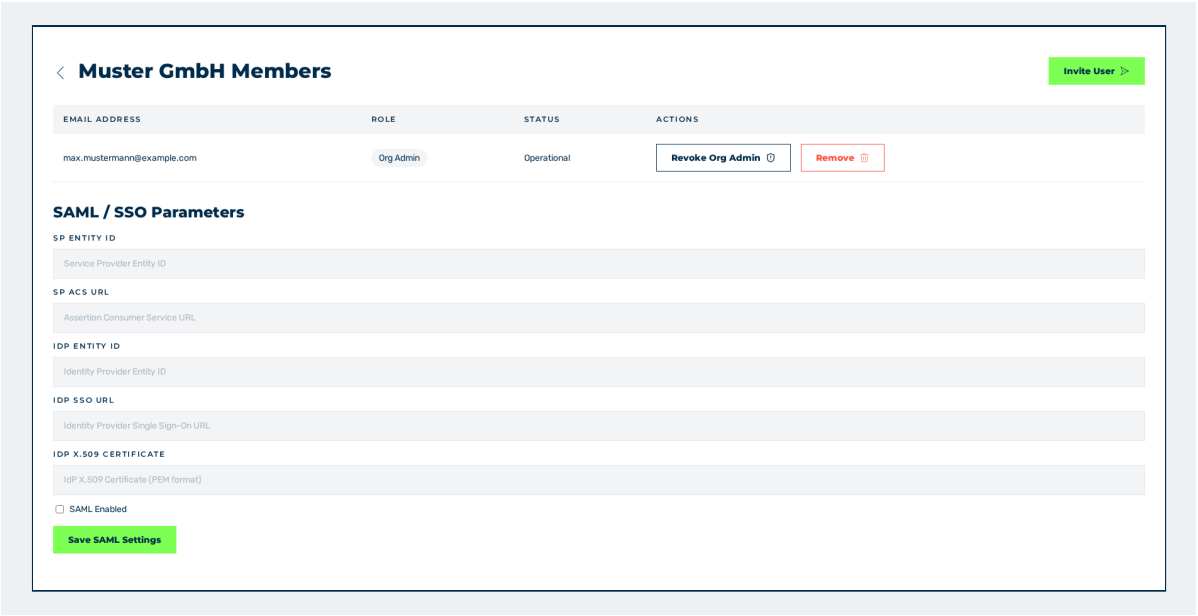


Figure 112: Organization view without the member that was removed

To stop the procedure, click on the **Close** button.

See **Assign and revoke the Org Admin role**.

See **Organization view**.

4.4.4 Set up SAML

With SAML, the members of your organization log in to the customer portal with the credentials of your own identity provider.

The following requirements must be met:

☑ Your account has the **Org Admin** role.

☑ You manage an identity provider with support for SAML 2.0.

☑ You know the identifier, the login address, and the certificate of your identity provider.

Proceed as follows to set up SAML:

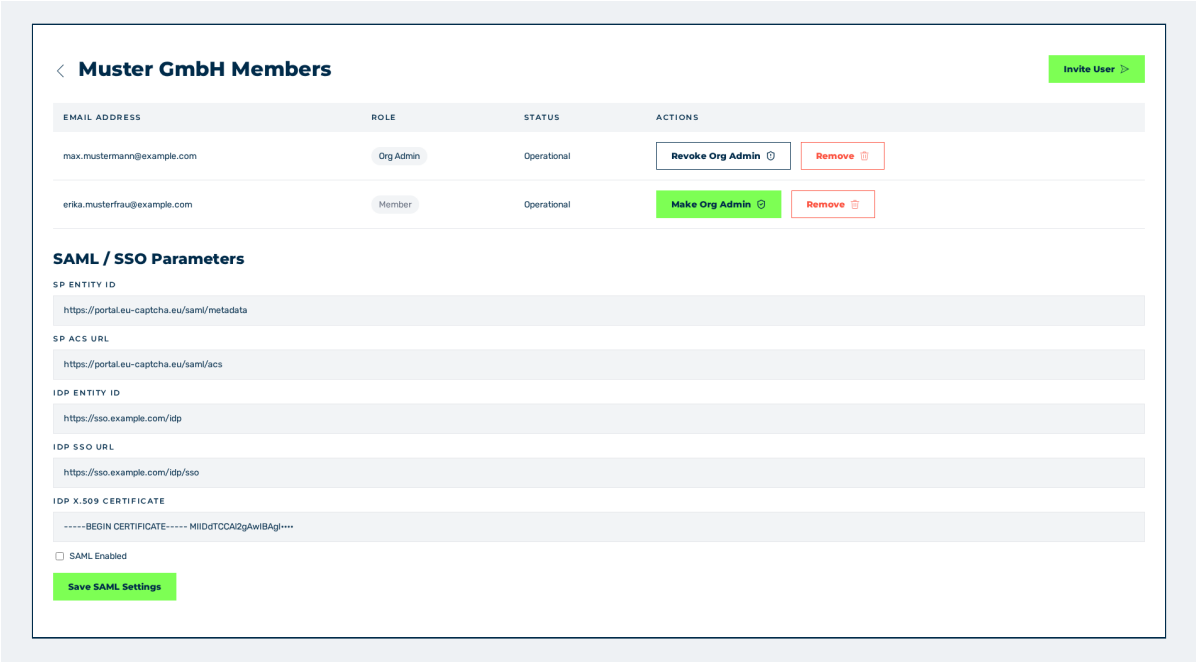


Figure 113: SAML / SSO Parameters area

- In the sidebar, open the **Account** › **Profile** entry and click on the **Manage Members** button in the **Organization** area.
- Go to the **SAML / SSO Parameters** area.
- Apply the values of the **SP Entity ID** and **SP ACS URL** fields in the configuration of your identity provider.
- In the **IdP Entity ID** field, enter the identifier of your identity provider.
- In the **IdP SSO URL** field, enter the login address of your identity provider.
- In the **IdP X.509 Certificate** field, add the certificate in the PEM format.

- ► Select the **SAML Enabled** check box.
- ► Click on the **Save SAML Settings** button.
- ➔ The system saves the data and shows the **SAML Settings** message with the **SAML settings saved successfully.** text.

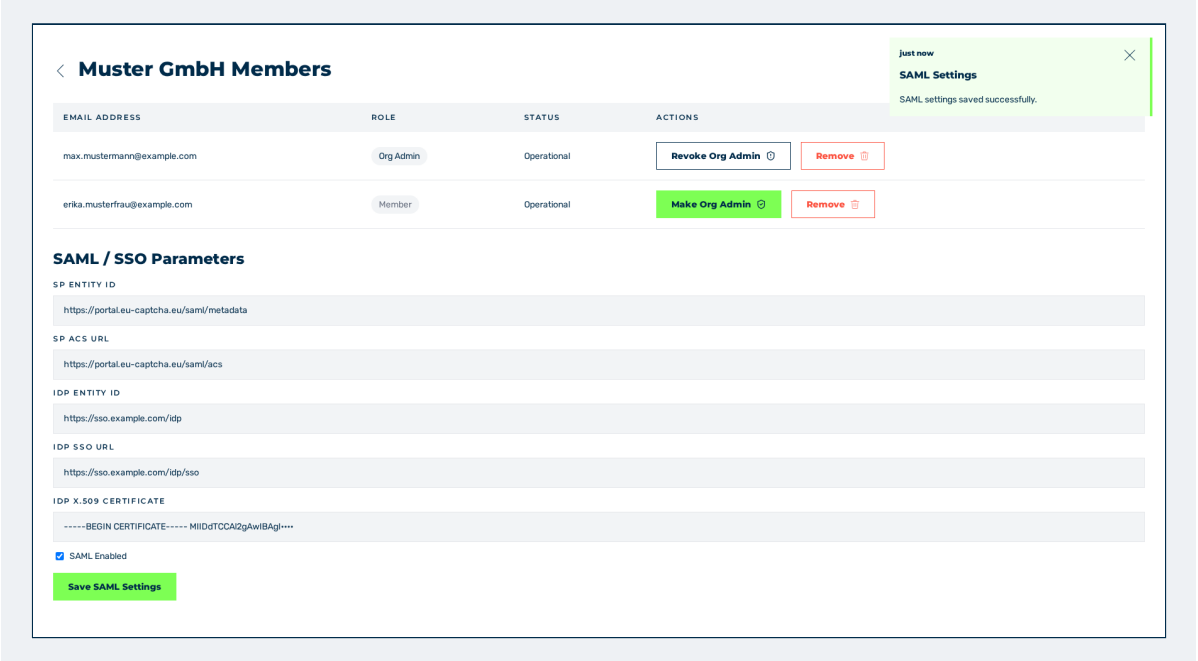


Figure 114: SAML / SSO Parameters area after the save operation

- ► Examine the login with your identity provider with a test account.

4.5 Subscription

4.5.1 Book a plan

With a plan with costs, the settings of the sitekeys and the personal support are available. The billing occurs with a payment service provider.

The following requirements must be met:

☑ You are logged in to the customer portal.

☑ No subscription is active for your account.

☑ The billing data and a means of payment are available to you.

Proceed as follows to book a plan:

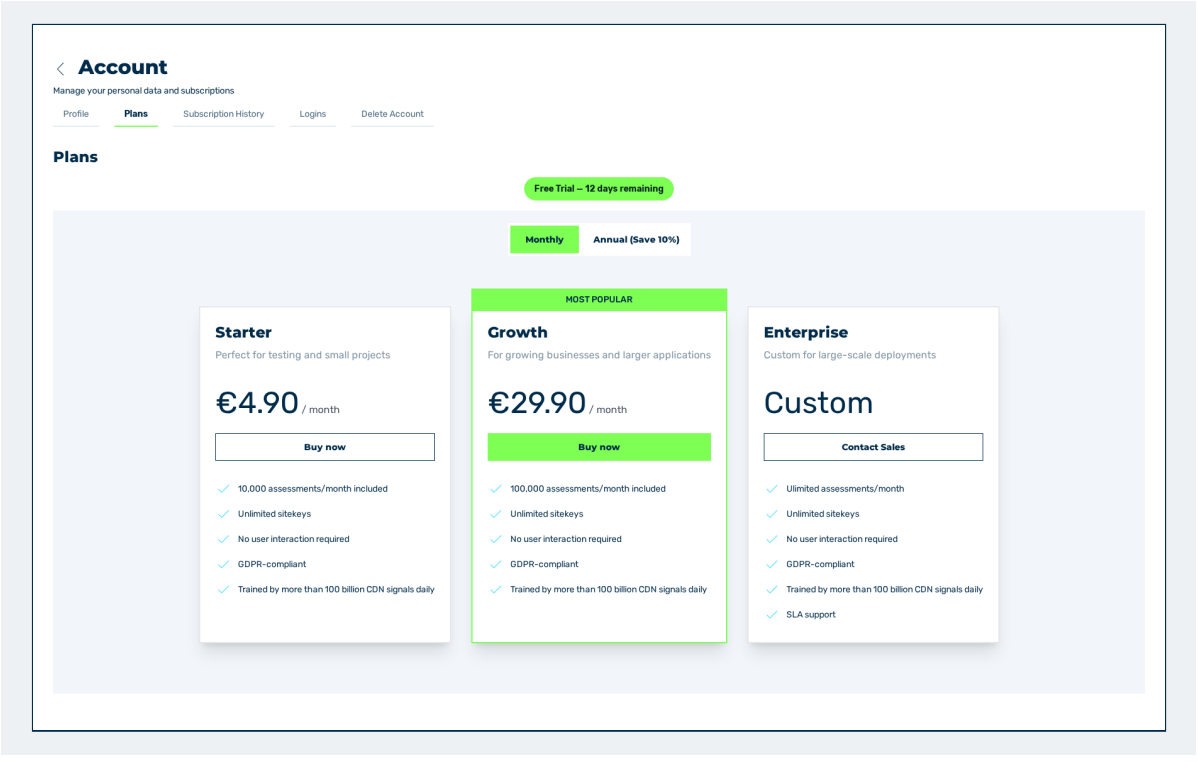


Figure 115: Plans tab

► In the sidebar, open the **Account** › **Profile** entry and go to the **Plans** tab.

► Click on the **Monthly** or **Annual (Save 10%)** button.

↳ The prices of the plans change to the selected billing period.

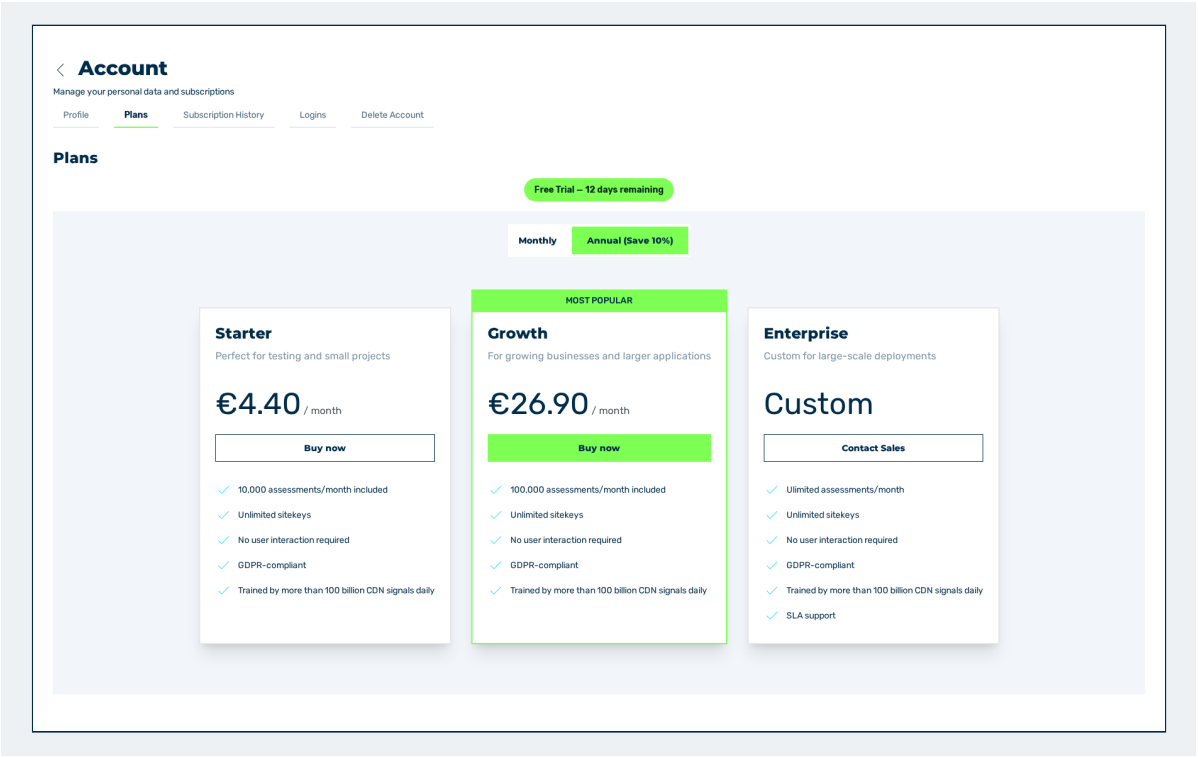


Figure 116: Plans tab with the annual billing

- ► In the applicable plan, click on the **Buy now** button.
  - ↳ The **Complete Your Purchase** view is shown.

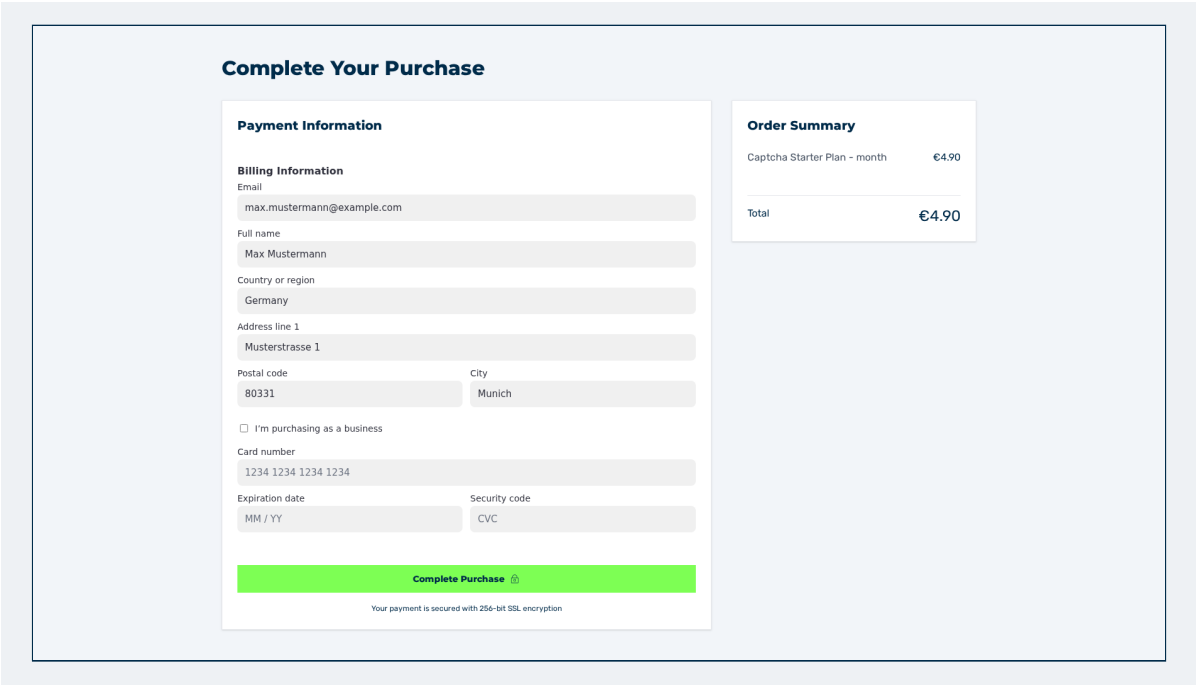


Figure 117: Complete Your Purchase view

- ► In the **Billing Information** area, enter the billing data.
- ► In the **Payment Information** area, enter the data of your means of payment.
- ► Click on the **Complete Purchase** button.

➔ The system activates the plan. The **Subscription History** tab contains an entry with the status of the subscription.

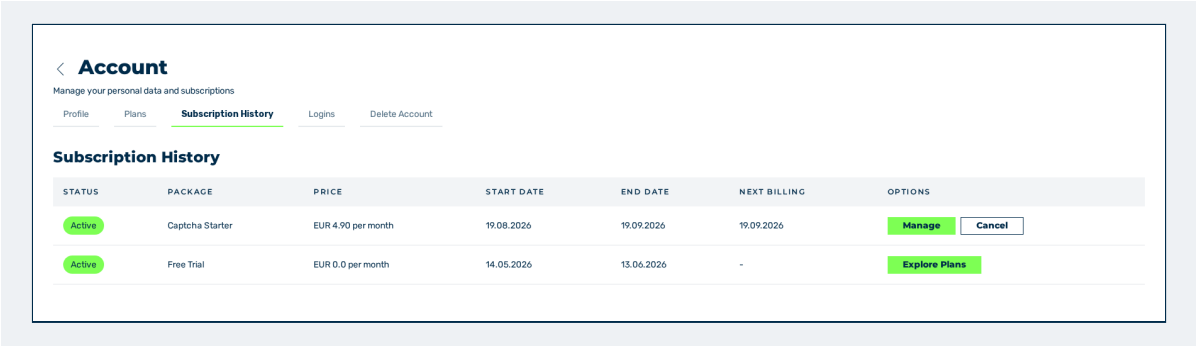


Figure 118: Subscription History tab with the new subscription

4.5.1.1 Booking as a business

As a business, select the **I'm purchasing as a business** check box. The view then shows a hint and the **VAT Number (optional)** field.

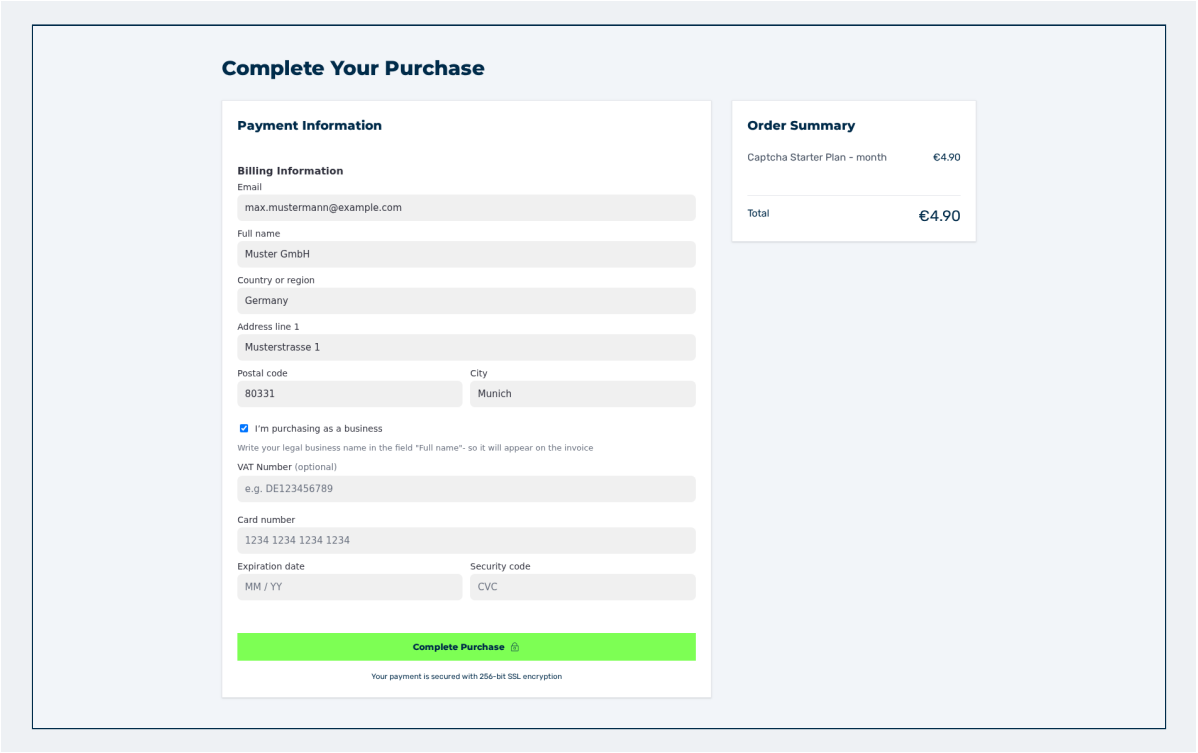


Figure 119: Complete Your Purchase view with the data of a business

- ► Select the **I'm purchasing as a business** check box.
  - ↳ A hint about the business name in the **Full name** field and the **VAT Number (optional)** field are shown.
- ► In the **Full name** field, enter the name of your business.
- ► In the **VAT Number (optional)** field, enter your VAT identification number, for example `DE123456789` .
- ➔ The invoice has the name of the business. With a VAT identification number from the EU, the system bills with the reverse charge procedure.

4.5.1.2 Enterprise plan

For the **Enterprise** plan, the **Contact Sales** button goes to the contact form of Myra. The price and the range of services are agreed for each case.

4.5.1.3 Manage the subscription

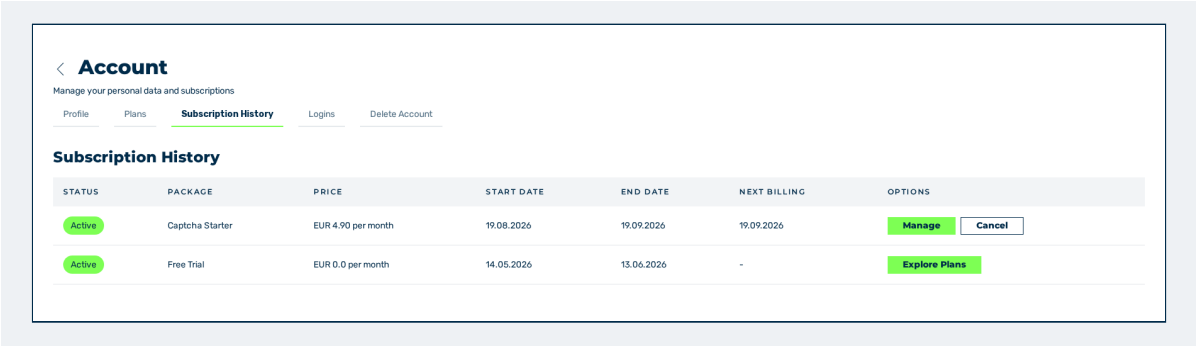


Figure 120: Subscription History tab with the Manage button

To manage the payment data and the invoices, use the **Manage** button in the **Subscription History** tab. The button opens the customer portal of the payment service provider.

See **Subscription History tab**.

See **Cancel a subscription**.

4.5.2 Cancel a subscription

You cancel a subscription in the **Subscription History** tab. The subscription stays active until the end of the period you paid for. After that, there is no more billing.

The following requirements must be met:

☑ You are logged in to the customer portal.

☑ A subscription is active for your account.

Proceed as follows to cancel a subscription:

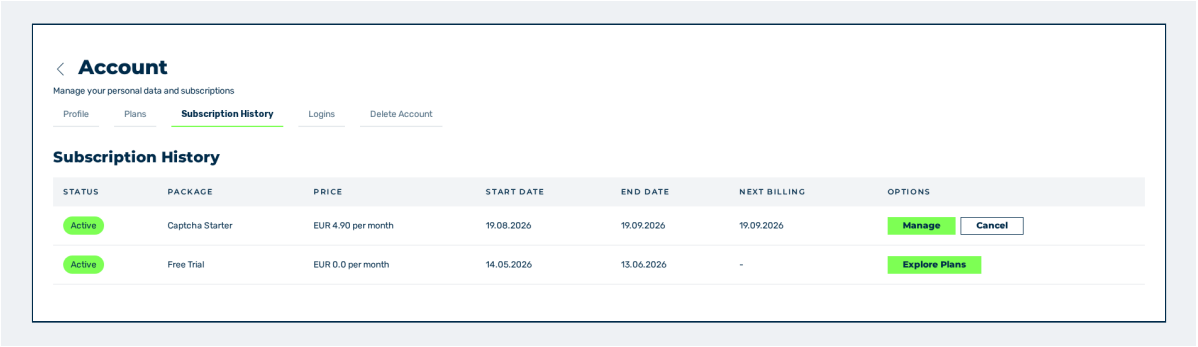


Figure 121: Subscription History tab with a plan that was booked

► In the sidebar, open the **Account** › **Profile** entry and go to the **Subscription History** tab.

► In the row of the subscription, click on the **Cancel** button.

↳ The **Cancel subscription** dialog is shown. The dialog gives the plan and the end of the period you paid for.

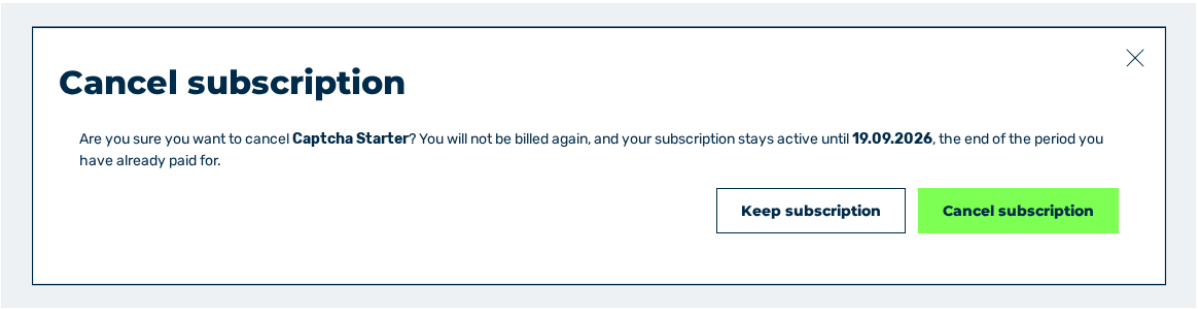


Figure 122: Cancel subscription dialog

► Click on the **Cancel subscription** button.

➔ The system cancels the subscription. The **Status** column shows the **Cancelled** mark. The **Next billing** column shows the `Ends` value with the end of the period you paid for.

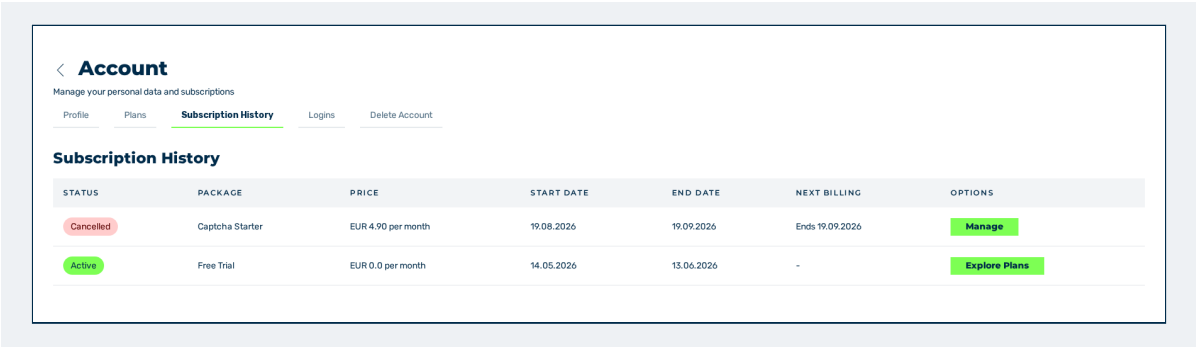


Figure 123: Subscription History tab with the subscription that was cancelled

4.5.2.1 Cancel a booking from the trial period

A plan that you booked during the trial period has the **Committed** mark and starts only after the end of the trial period. To cancel this booking, use the **Cancel commitment** button.

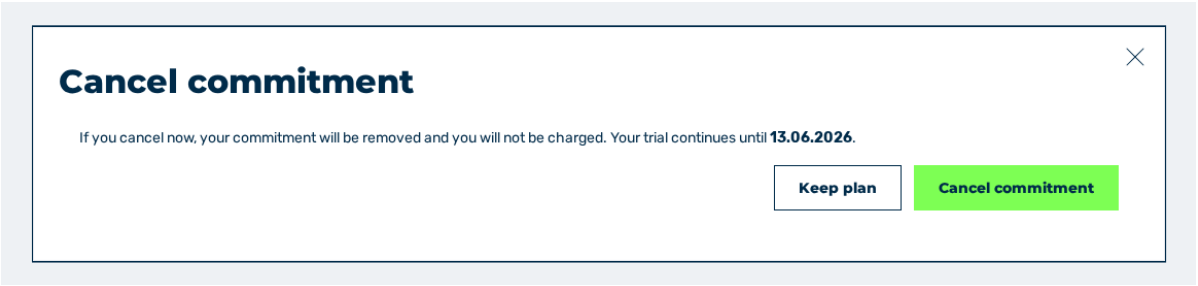


Figure 124: Cancel commitment dialog

- ▶ In the row of the subscription, click on the **Cancel commitment** button.
  - ↳ The **Cancel commitment** dialog is shown.
- ▶ Click on the **Cancel commitment** button.
- ➔ The system removes the booking. There is no billing. The trial period continues until its end.

> The **Keep subscription** and **Keep plan** buttons close the dialog and do not cancel.

See **Subscription History tab**.

See **Delete the account**.

5. Integration

5.1 General

5.1.1 Integration

The integration of the Myra EU CAPTCHA always has two parts: the widget in your page and the verification of the token on your server. Only both parts together give protection.

For the usual platforms, there are libraries, extensions, and examples that do these two parts. The subsequent chapters give the installation, the configuration, and the verification for each platform.

The guides of this chapter need a sitekey that is already created. The way from a new account to the examined integration is given under **Initial setup** with the steps **Create the first sitekey**, **Embedding the widget**, and **Testing the integration**.

5.1.1.1 Structure of an integration

Each integration does these three steps:

Step	Location	Content
1	Frontend	Load the `verify.js` script from the CDN.
2	Frontend	Add the widget element with the public sitekey.
3	Backend	Verify the token with the `/verify` endpoint before the processing.

See **Embedding the widget**.

The two frontend steps occur in the browser of the visitors. If your website sends a Content Security Policy, then it must permit the addresses of the service. This requirement applies to each platform of this chapter. See **Content Security Policy**.

5.1.1.2 Available instructions

These instructions are available:

Category	Platforms
CMS	**WordPress**, **Joomla**, **Drupal**, **TYPO3**
Frontend	**HTML and JavaScript**, **React**, **Vue**, **Angular**, **Svelte**
Backend	**PHP**, **Symfony and Laravel**, **Python**, **Django**, **Ruby**, **Java**, **Client from the specification**
Mobile apps	**Android**, **iOS**, **Flutter**
Infrastructure	**Caddy**, **CrowdSec**, **Traefik**

5.1.1.3 Full examples

The instructions for each platform give one side of the integration. The examples put both sides together in one continuous case: you protect the contact form of an application that already operates. You change only a small number of files.

These examples are available:

Example	Frontend	Backend
React and PHP	React with Vite	PHP
Vue and Python	Vue with Vite	Python with Flask
Angular and Java	Angular	Java with Spring Boot
Svelte and Ruby	Svelte with Vite	Ruby with Sinatra
HTML and Django	HTML without a framework	Django

5.1.1.4 Data from the customer portal

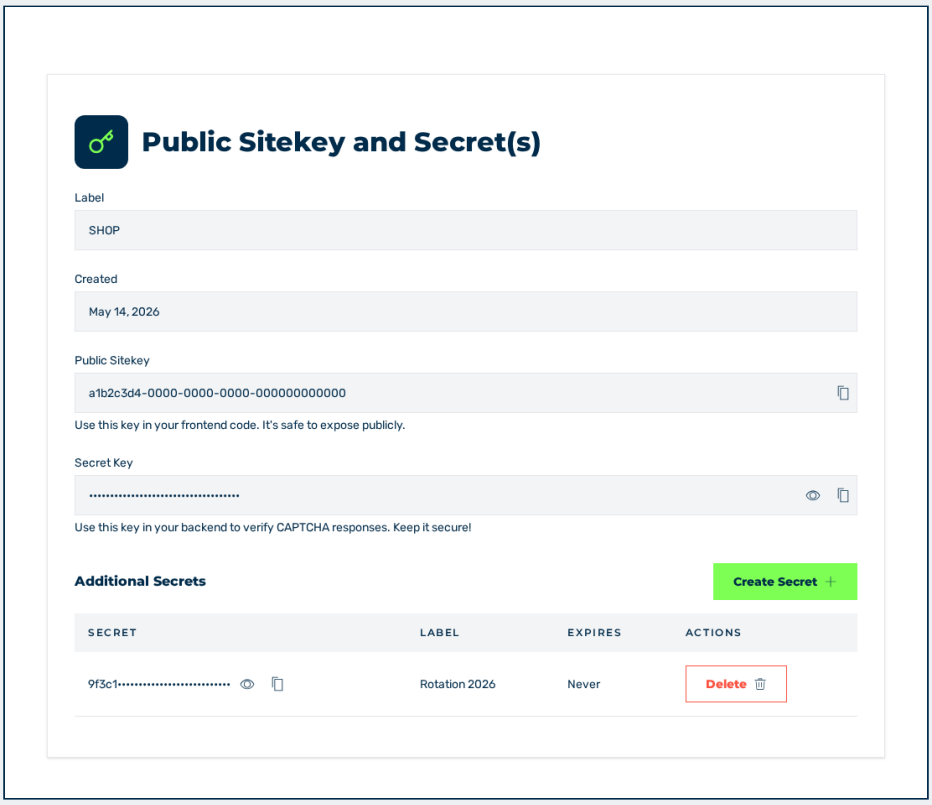


Figure 125: Details view

All instructions need this data from the **Details** view of the sitekey:

Data	Use
Public Sitekey	In the widget element and in the configuration of the frontend. The value is public.
Secret	Only in the verification on the server. The value stays on the server.

See The Details view.

5.2 CMS

5.2.1 WordPress

The **EU-Captcha** plugin protects the forms of WordPress and of the usual form plugins. It does the widget and the server-side verification.

> **Video:** Setup of the plugin in WordPress — playable in the online help

5.2.1.1 Requirements

The following requirements must be met:

Requirement	Value
WordPress	from version 5.8, tested up to version 6.9
PHP	from version 7.4
Access	Administration area of WordPress

5.2.1.2 Protected forms

The plugin protects these forms:

- login of WordPress
- registration of WordPress
- comments of WordPress
- Contact Form 7
- Ninja Forms
- WooCommerce Checkout
- WPForms

5.2.1.3 Set up the plugin

Proceed as follows to set up the plugin:

► Upload the `eu-captcha` folder into the `/wp-content/plugins/` directory of your installation.

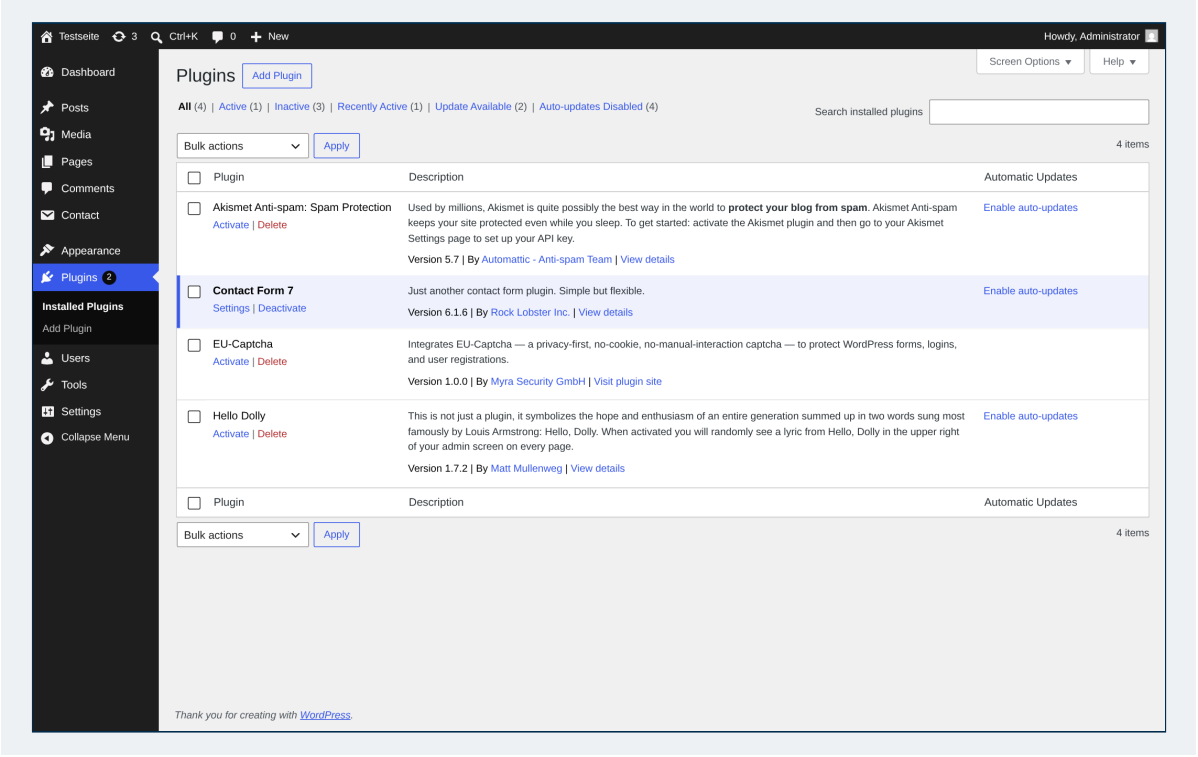


Figure 126: List of the plugins with the EU-Captcha plugin

► Activate the plugin in the **Plugins** menu.

↳ The **EU-Captcha** settings page is available.

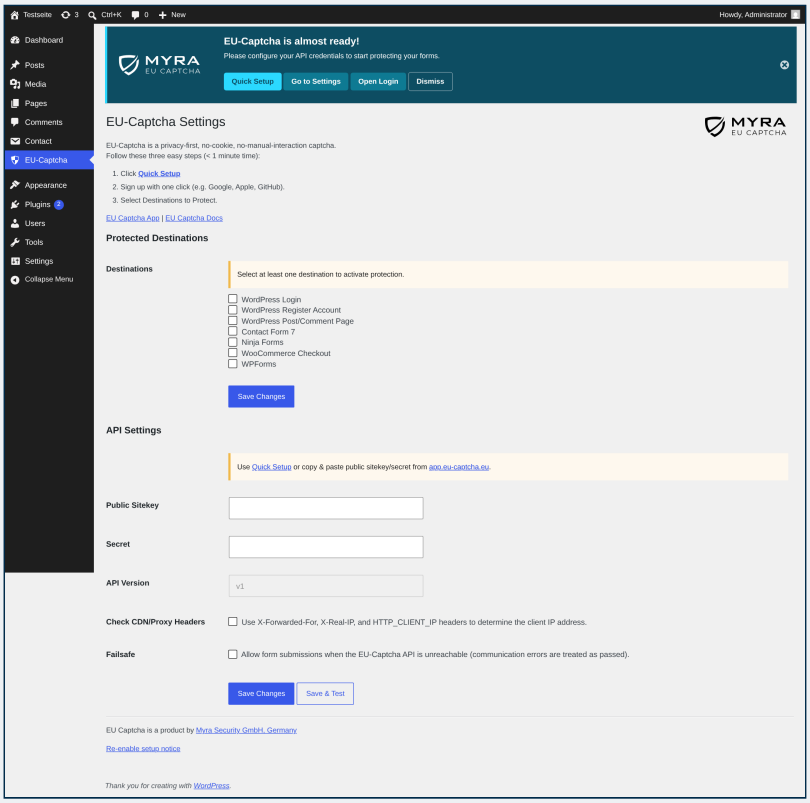


Figure 127: EU-Captcha settings page

- ► Open the **EU-Captcha** settings page.
- ► Click on the **Quick Setup** button. As an alternative, enter the public sitekey and the secret manually.
- ► Below **Protected Destinations**, select the forms to protect.
- ► Click on the **Save Changes** button.
- ➔ The plugin embeds the widget in the selected forms and verifies the tokens on the server.

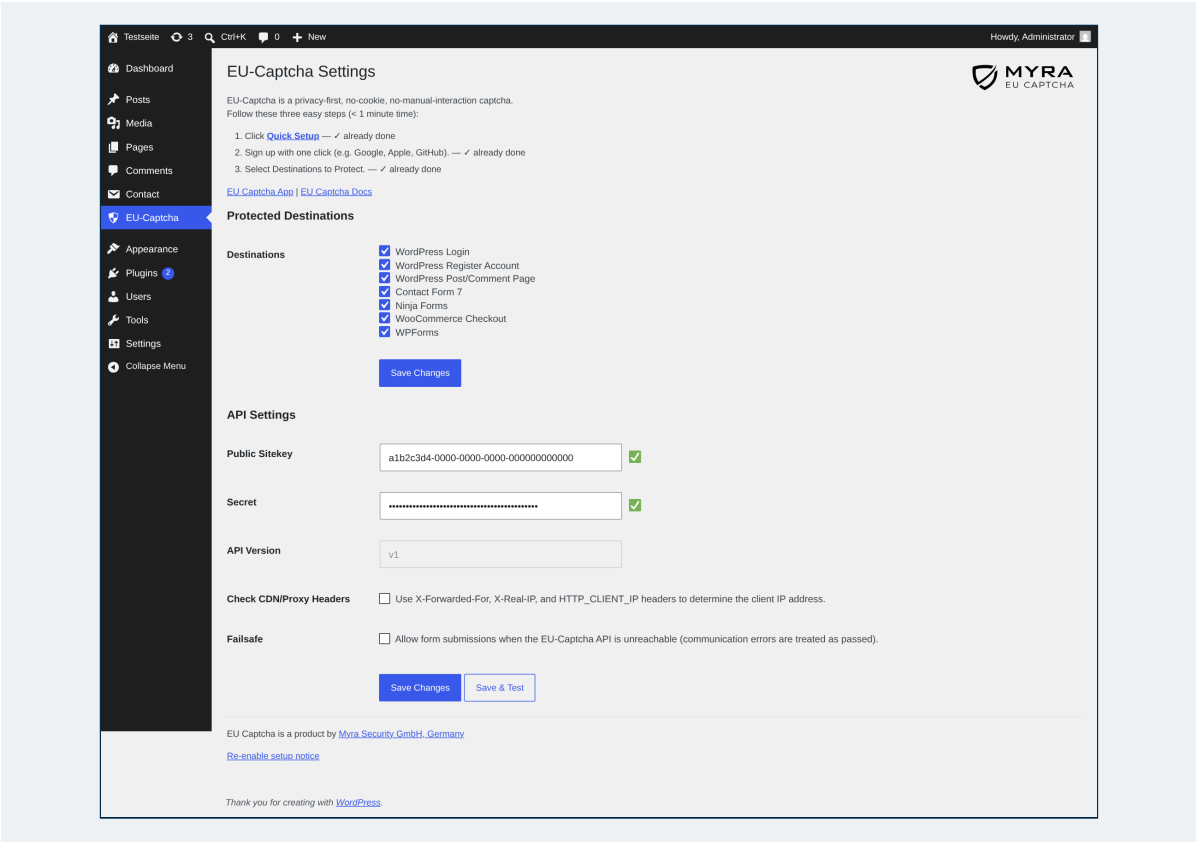


Figure 128: EU-Captcha settings page after saving

5.2.1.4 Behaviour during a malfunction

By default, the plugin refuses a transmission if the API is not reachable. Thus, the protection cannot be bypassed during a malfunction.

The **Failsafe** option in the settings below **API Settings** changes this behaviour: transmission errors such as timeouts or name resolution errors then count as a verification that passed. The plugin continues to refuse invalid responses.

With the **Failsafe** option activated, your form stays available during a malfunction. The form is unprotected during this time.

5.2.1.5 Protect your own forms

The plugin supplies the `eu_captcha_verify()` function. With it, other plugins and themes verify a token with the credentials that are already stored.

Add the widget element to your form:

```
<div class="eu-captcha" data-sitekey="a1b2c3d4-0000-0000-0000-000000000000"></
div>
```

Verify the token during the transmission:

```
if ( function_exists( 'eu_captcha_verify' ) ) {
    $token  = sanitize_text_field( $_POST['eu-captcha-response'] );
    $result = eu_captcha_verify( $token );
    if ( ! $result ) {
        wp_die( 'Captcha verification failed.' );
    }
}
```

The function returns `true` for a valid verification, and `false` for an invalid token, missing credentials, or an error of the API. Always examine with `function_exists()` first, so that your source code also operates when the plugin is deactivated.

The plugin embeds the `verify.js` script on all pages of the frontend and on the login page itself.

5.2.2 Joomla

The plugin for Joomla connects to the standard CAPTCHA interface of Joomla. Thus, each form with a CAPTCHA field is protected without your own source code.

5.2.2.1 Requirements

The following requirements must be met:

Requirement	Value
Joomla	version 4.x, 5.x, or 6.x
PHP	from version 8.1
Access	Administration area of Joomla

5.2.2.2 Install the plugin

Proceed as follows to install the plugin:

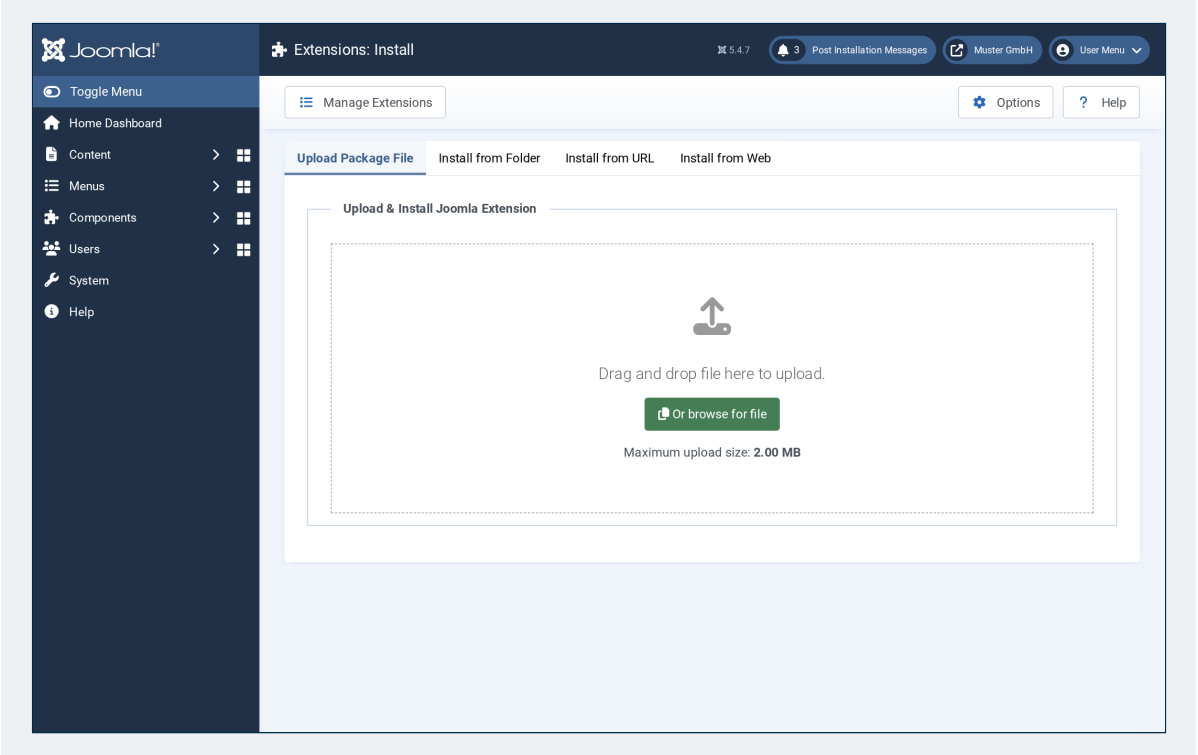


Figure 129: Extensions: Install view

► Download the `pkg_eucaptcha.zip` package.

- In the administration area, open the **Extensions** → **Install** entry.
- In the **Upload Package File** tab, upload the `pkg_eucaptcha.zip` file.
  - ↳ Joomla installs the two plugins that are included and shows **Installation of the package was successful**.

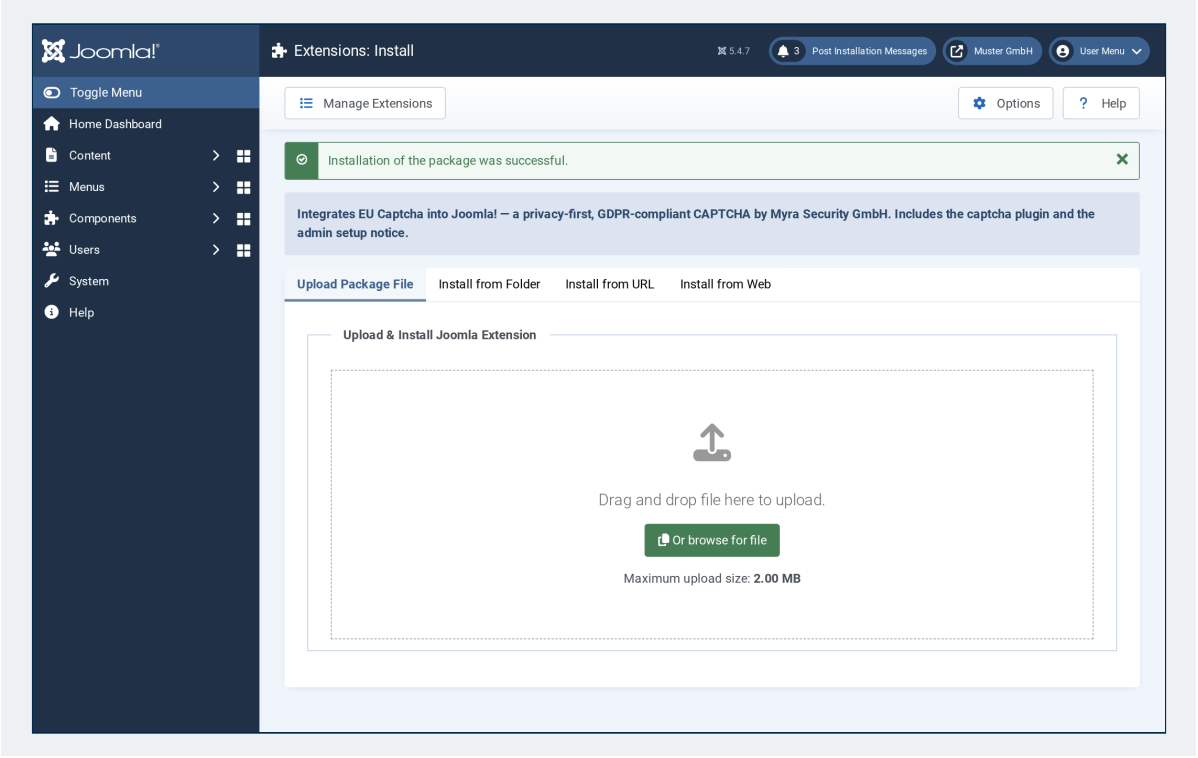


Figure 130: Message after the installation

- Open the **Extensions** → **Plugins** entry and search for `eu captcha`.
  - ↳ The list shows the **EU Captcha** and **EU Captcha - Admin Notice** plugins. The two plugins are disabled.

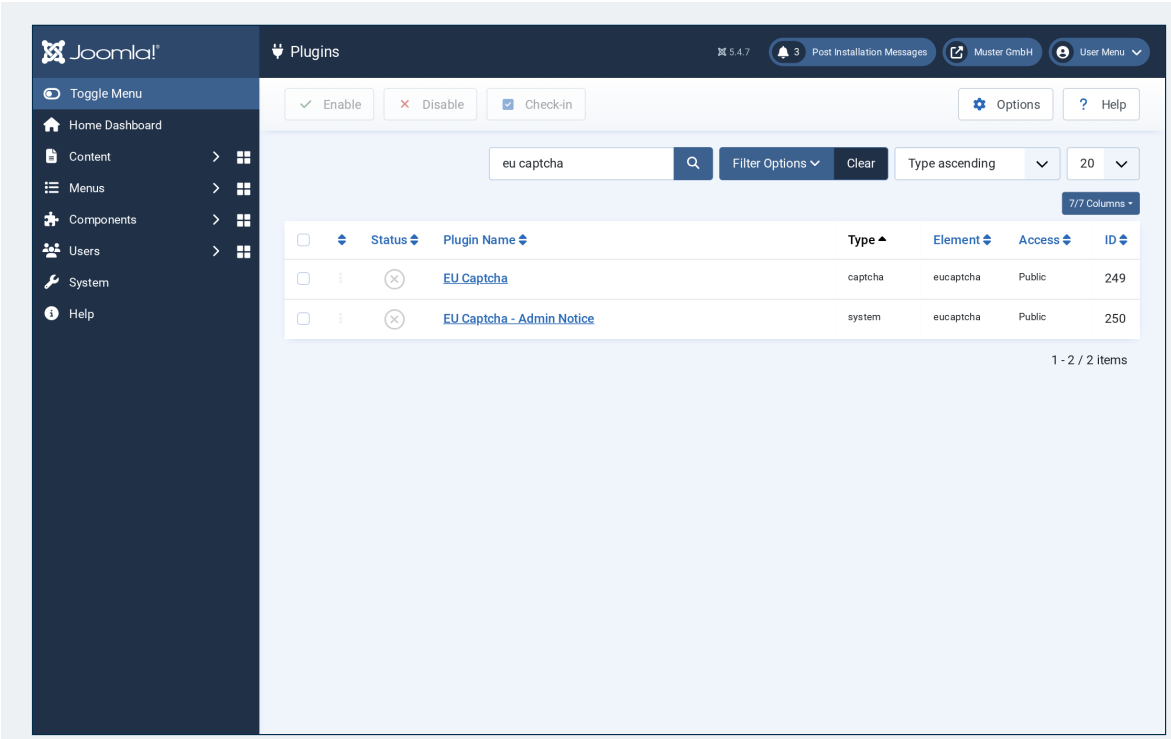


Figure 131: List of the plugins with the two extensions

- ► Select the two plugins and click on the **Enable** button.
- ➔ The two plugins are enabled.

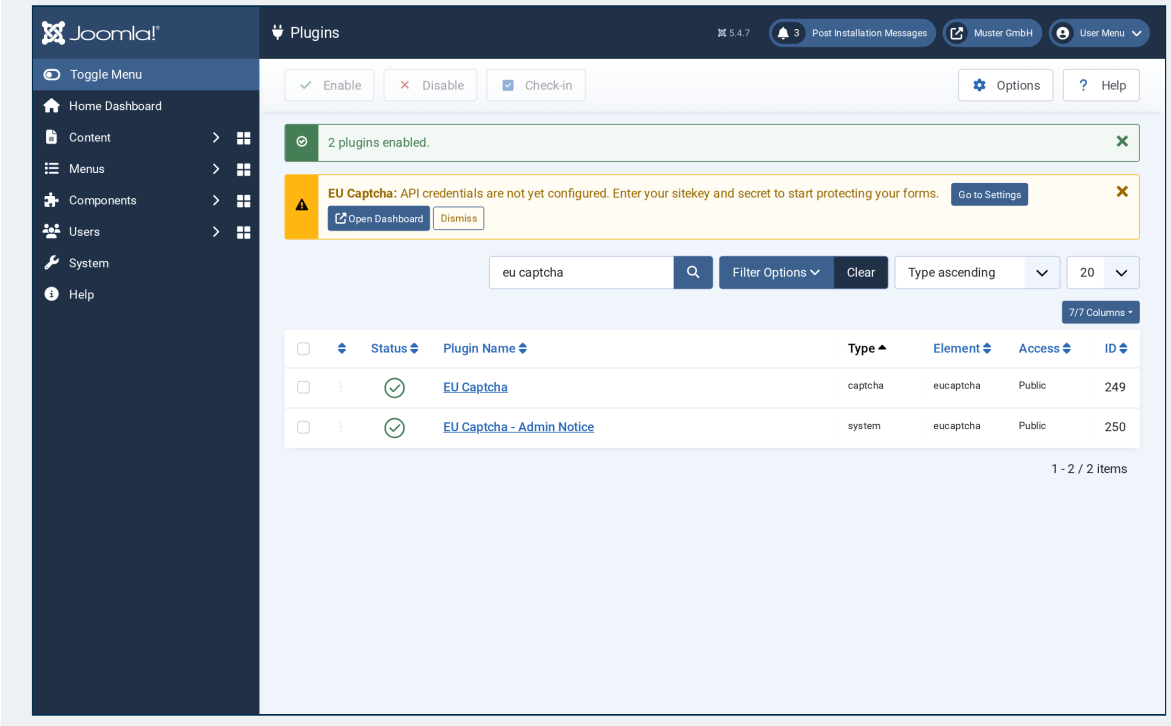


Figure 132: List of the plugins after they are enabled

5.2.2.3 Store the credentials

Until the configuration is complete, Joomla shows a note on each page of the administration area.

Proceed as follows to store the credentials:

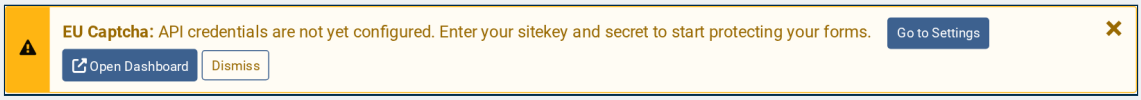


Figure 133: Note in the administration area

- ► In the note, click on the **Go to Settings** button. As an alternative, open **Extensions** → **Plugins** and then the **EU Captcha** plugin.
  - ↳ The settings of the plugin are shown.

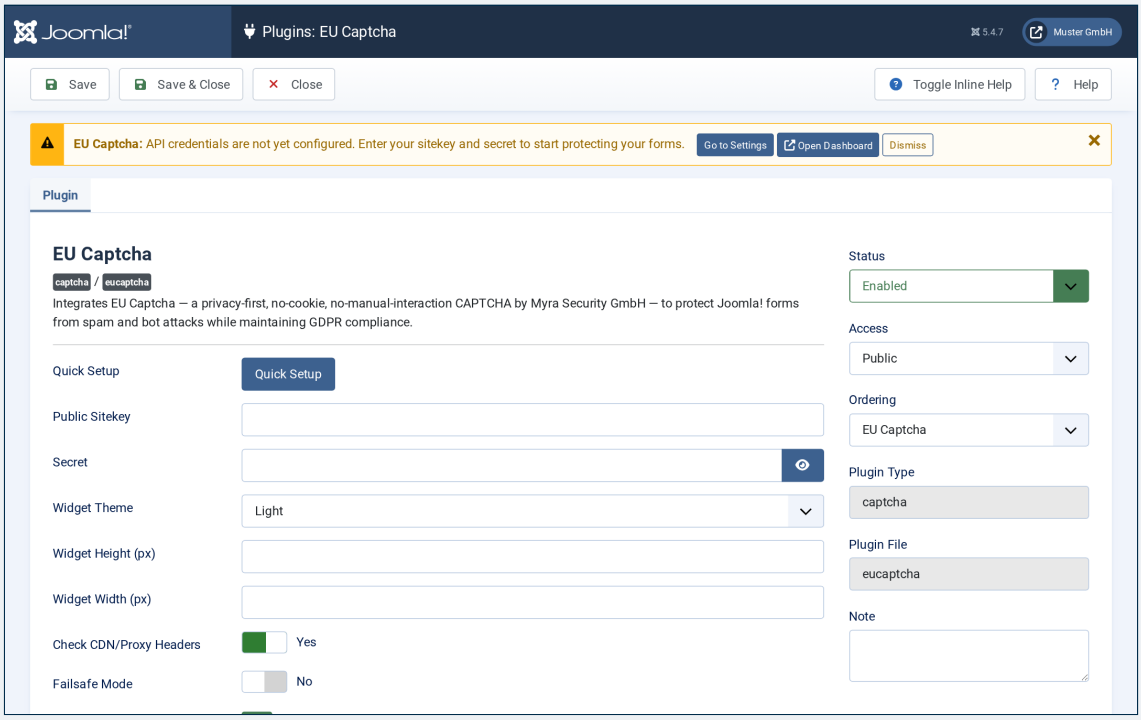


Figure 134: Settings of the EU Captcha plugin

- ► Click on the **Quick Setup** button.
  - ↳ A dialog with the registration portal of Myra EU CAPTCHA is shown.
- ► Complete the registration.
- ➔ The plugin applies the public sitekey and the secret.

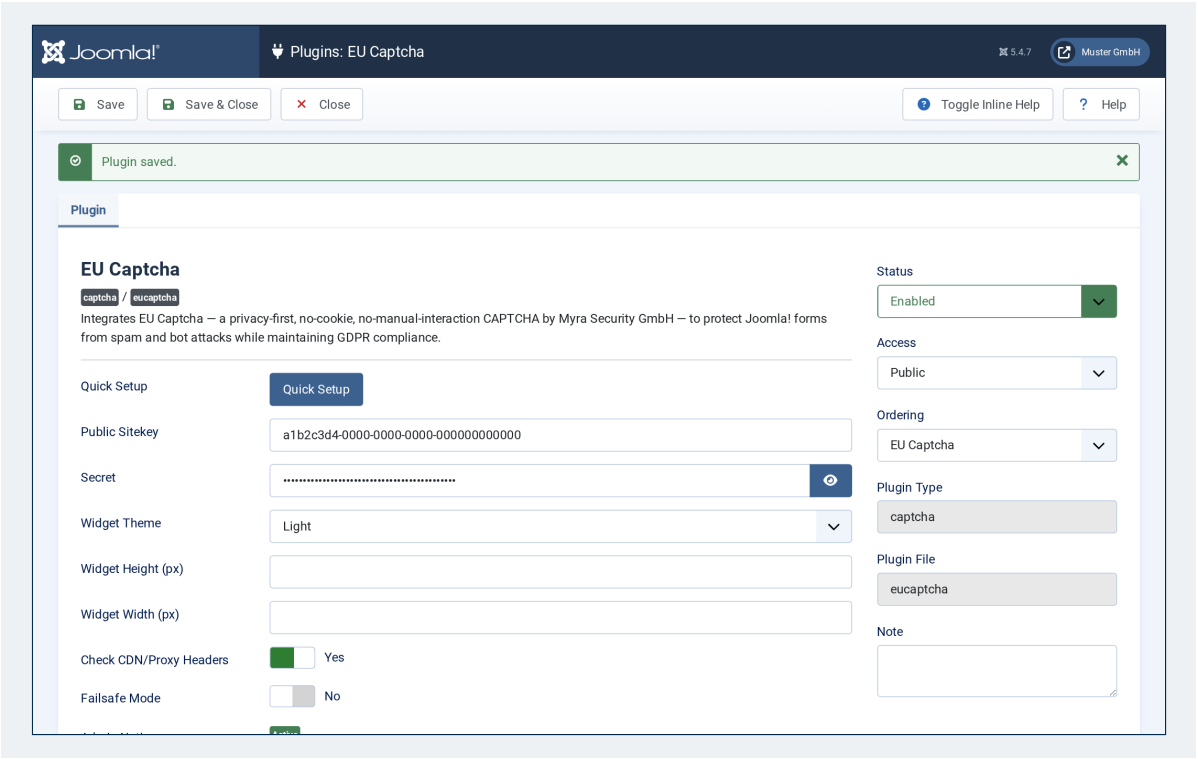


Figure 135: Settings of the plugin after the save operation

As an alternative, enter the available credentials manually:

Field	Contents
Public Sitekey	Public key in the UUID format. The value is permitted in the frontend.
Secret	Secret key in the Base64 format. The value stays in the database of Joomla.

5.2.2.4 Options of the API

These options are available:

Option	Default	Effect
Check CDN/ Proxy	**Yes**	Reads the true IP address of the visitor from the `HTTP_CLIENT_IP`, `HTTP_X_FORWARDED_FOR`, or `HTTP_X_REAL_IP` headers, before `REMOTE_ADDR` is used. Activate the option if your website is behind a CDN or a load balancer.

Option	Default	Effect
Headers		
Failsafe Mode	**No**	With **No**, a network error blocks the transmission. With **Yes**, a transmission error counts as a verification that passed. In the two cases, the plugin refuses invalid tokens.

5.2.2.5 Appearance of the widget

These options control the appearance:

Option	Default	Effect
Widget Theme	**Light**	Light or dark appearance, values **Light** and **Dark**.
Widget Height (px)	empty	Fixed height in pixels, minimum 48.
Widget Width (px)	empty	Fixed width in pixels, minimum 1.

5.2.2.6 Activate the CAPTCHA

Proceed as follows to activate the CAPTCHA:

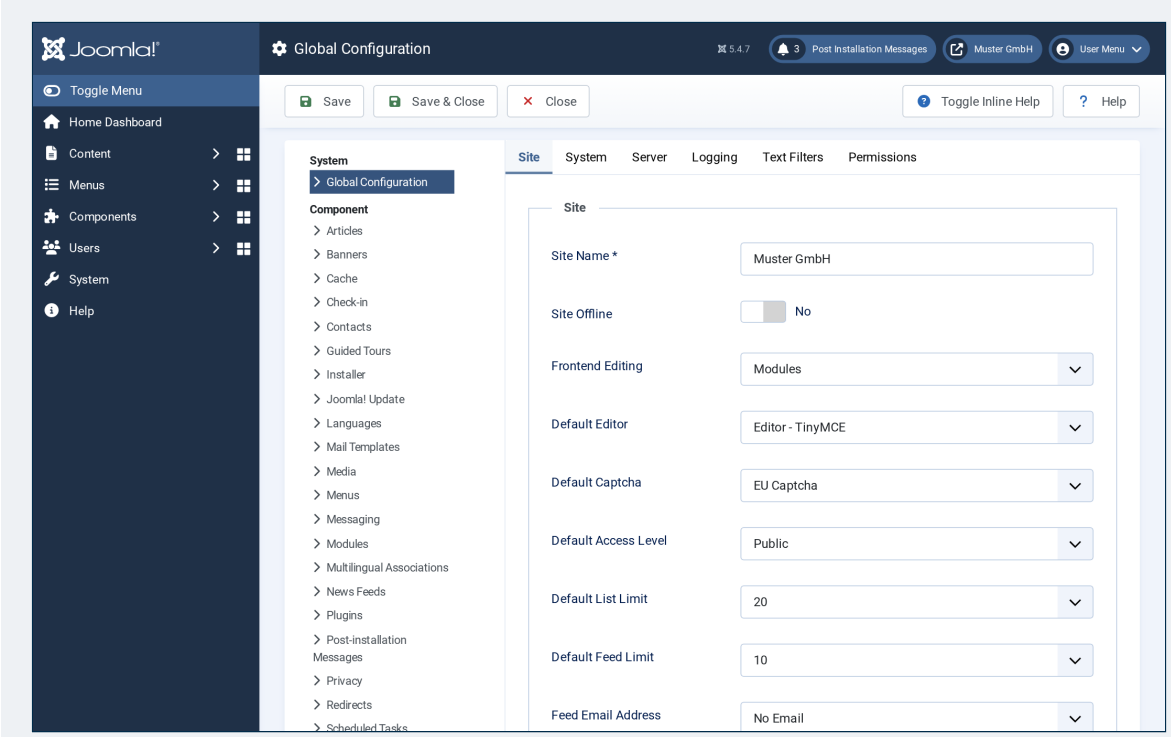


Figure 136: Global Configuration view with the Default Captcha option

- ► Open **System** → **Global Configuration** and go to the **Site** tab.
- ► Set the **Default Captcha** option to the **EU Captcha** value.
- ► Click on the **Save** button.
- ➔ All forms with a CAPTCHA field are protected.

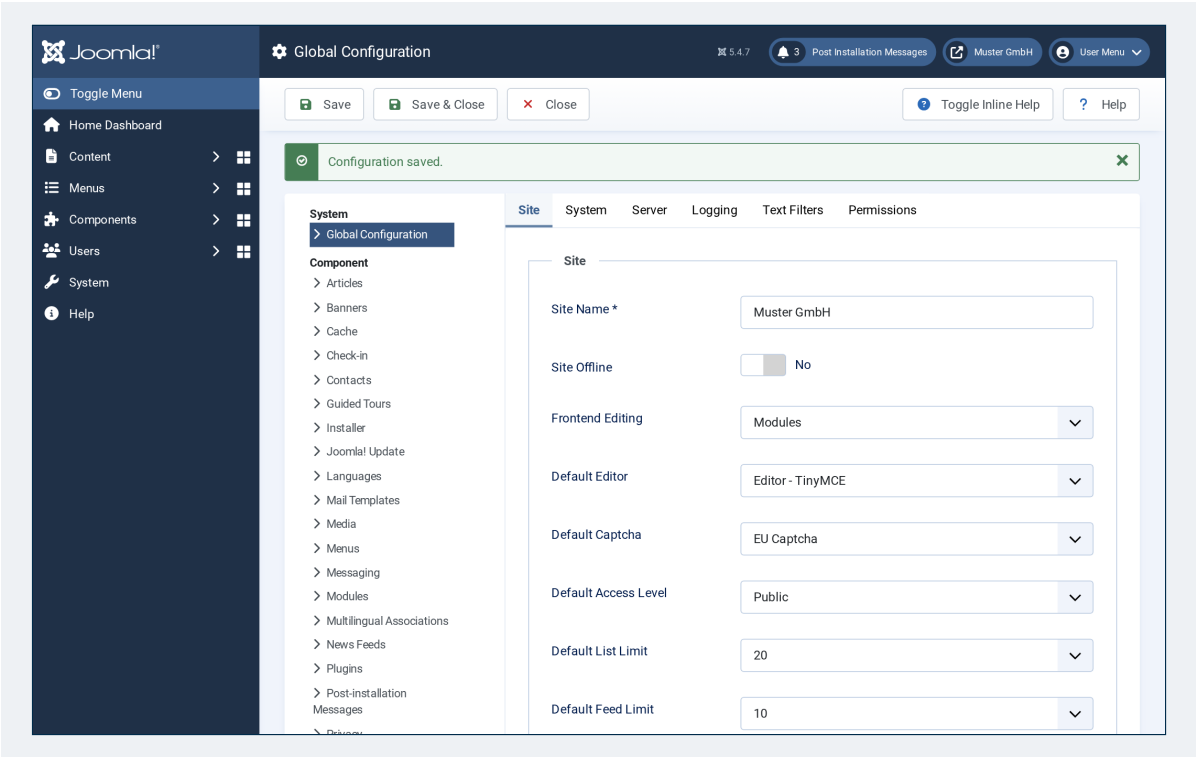


Figure 137: Global Configuration view after the save operation

For the contact form that is included, also activate the CAPTCHA field below **Components** → **Contact** → **Options** in the **Form** tab.

5.2.2.7 Protected forms

Each form with a standard CAPTCHA field is protected, for example:

- registration of users
- reset of the password
- `com_contact` contact component
- components of other suppliers with a standard CAPTCHA field

5.2.3 Drupal

The module for Drupal protects the forms of the core and the forms of the Webform module. The widget is loaded from the CDN. Your own JavaScript source code is not necessary.

5.2.3.1 Requirements

The following requirements must be met:

Requirement	Value
Drupal	version 10 or 11
Permission	`administer eu captcha`
Access	Command line with Drush

5.2.3.2 Install the module

Proceed as follows to install the module:

- Put the module in the `web/modules/custom/eu_captcha/` directory.
- Activate the module on the command line:

```
drush en eu_captcha
```

- In the administration area, open the **Administration** → **Configuration** → **System** → **EU-Captcha** entry.

→ The `/admin/config/system/eu-captcha` settings page is shown.

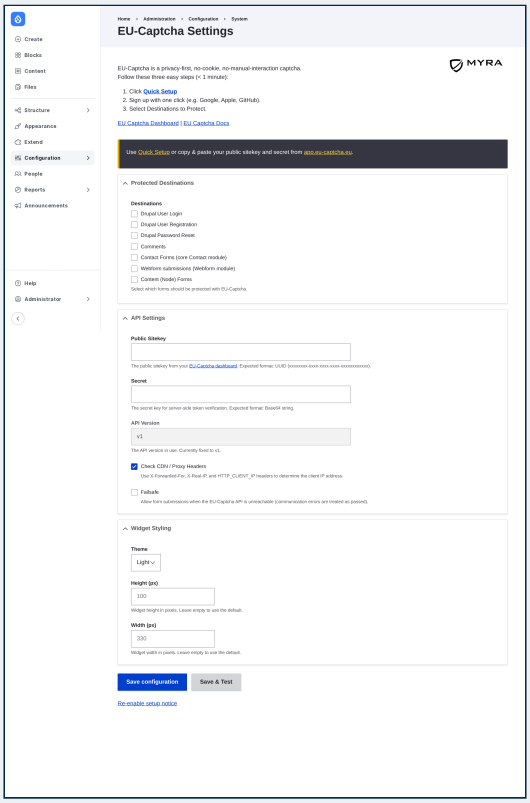


Figure 138: EU-Captcha Settings page

5.2.3.3 Store the credentials

Proceed as follows to store the credentials:

- ► On the settings page, click on the **Quick Setup** button.
  - ↳ A dialog with the login of Myra EU CAPTCHA is shown.
- ► Log in with Google, Apple, or GitHub.
- ➔ The module applies the public sitekey and the secret.

As an alternative, enter the available credentials manually:

Field	Contents
Public Sitekey	Public key in the UUID format. The widget in the frontend uses this value.
Secret	Secret key in the Base64 format for the server-side verification. After the save operation, Drupal does not show the value again.

The module examines the format of the two values immediately during the input.

5.2.3.4 Options of the API

These options are available:

Option	Effect
Check CDN / Proxy Headers	Reads the true IP address of the visitor from the `X-Forwarded-For`, `X-Real-IP`, or `HTTP_CLIENT_IP` headers. Activate the option if Drupal is behind a load balancer or a CDN.
Failsafe	With **On**, the module permits a transmission if the API of Myra EU CAPTCHA is not reachable. With **Off**, the module blocks the transmission when an API error occurs.

5.2.3.5 Select the forms

Proceed as follows to select the protected forms:

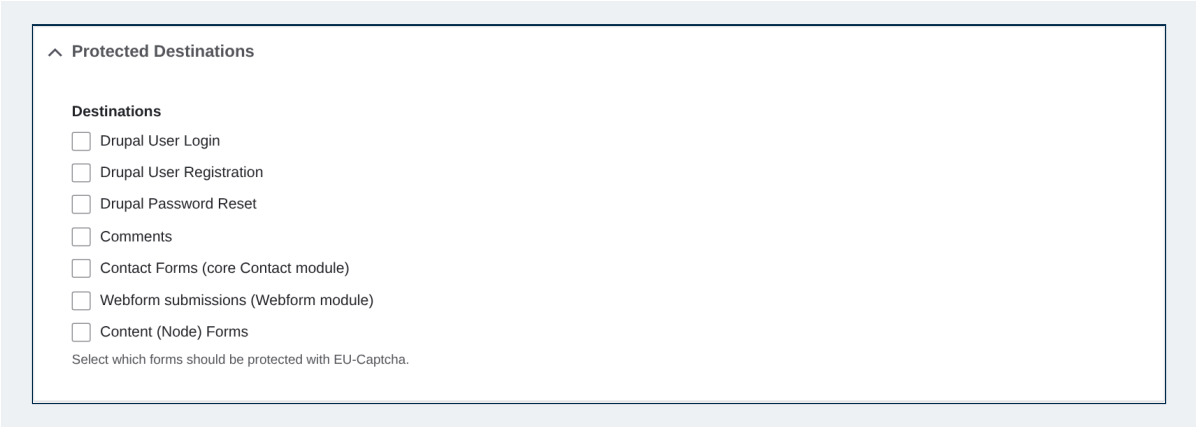


Figure 139: Protected Destinations area

- ▶ In the **Protected Destinations** area, select the applicable destinations.
- ▶ Click on the **Save configuration** button.
- ➔ The widget is shown in the selected forms.

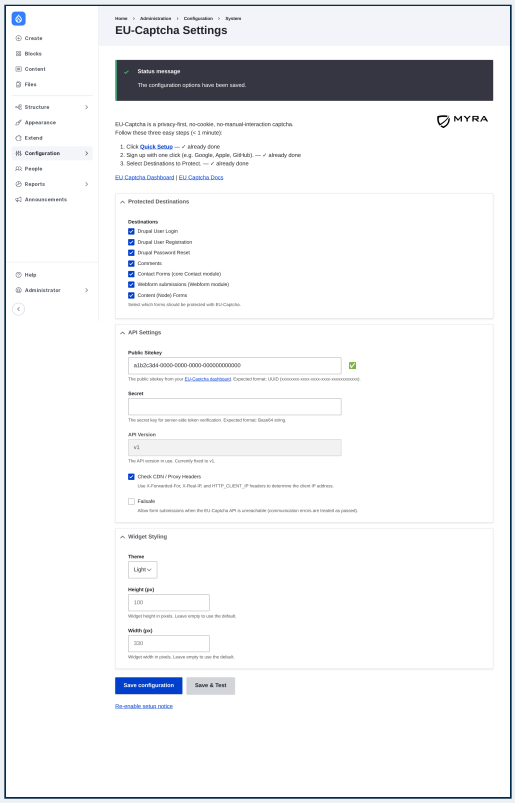


Figure 140: Settings page after the save operation

These destinations are available:

Destination	Label	Range
`user_login`	**Drupal User Login**	Login form of Drupal
`user_register`	**Drupal User Registration**	Registration form of Drupal
`user_pass`	**Drupal Password Reset**	Form for the password reset
`comment`	**Comments**	All comment forms (`comment_{bundle}_form`)
`contact`	**Contact Forms (core Contact module)**	Forms of the Contact core module (`contact_message_{id}_form`)
`webform`		Forms of the Webform module (`webform_submission_{id}_add_form`)

Destination	Label	Range
	Webform submissions (Webform module)	
`node`	**Content (Node) Forms**	Forms for content (`node_{type}_form`)

5.2.3.6 Permission

Permission	Effect
`administer eu captcha`	Access to the settings page, to **Quick Setup**, and to the notice bar in the administration area. Give the permission only to roles that you trust.

5.2.3.7 Notice bar

While no sitekey is saved or no destination is selected, Drupal shows a notice bar in the administration area. Each user can hide the bar with the **Dismiss** button. The bar is removed permanently as soon as a sitekey and a minimum of one destination are saved.

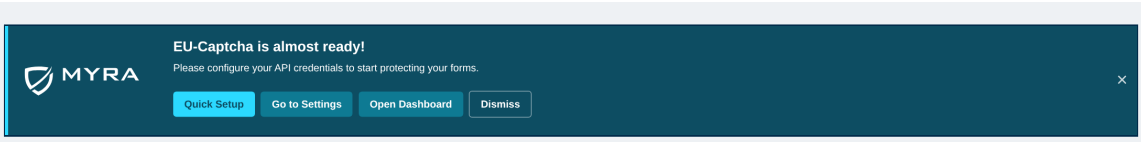


Figure 141: Notice bar in the administration area

To show the notice bar again, click on the **Re-enable setup notice** link at the end of the settings page.

5.2.3.8 Sequence of the verification

1. The module loads `verify.js` from the CDN and adds a `<div data-sitekey="…">` element to each protected form.
2. During the transmission, the widget adds the hidden `eu-captcha-response` field.

3. The server sends the token, the sitekey, the secret, and the IP address of the visitor to `https://api.eu-captcha.eu/v1/verify`. The `{"success": true}` response permits the transmission. Each other response makes a validation error in the form.

   See **Widget not detected**.

5.2.4 TYPO3

The extension for TYPO3 protects the login in the backend, the login in the frontend with `EXT:felogin`, and the forms of the `EXT:form` form framework.

5.2.4.1 Requirements

The following requirements must be met:

Requirement	Value
TYPO3	version 14
Extensions	`felogin` for the login in the frontend, `form` for the form framework
Permission	Administration rights in TYPO3

5.2.4.2 Install the extension

Proceed as follows to install the extension:

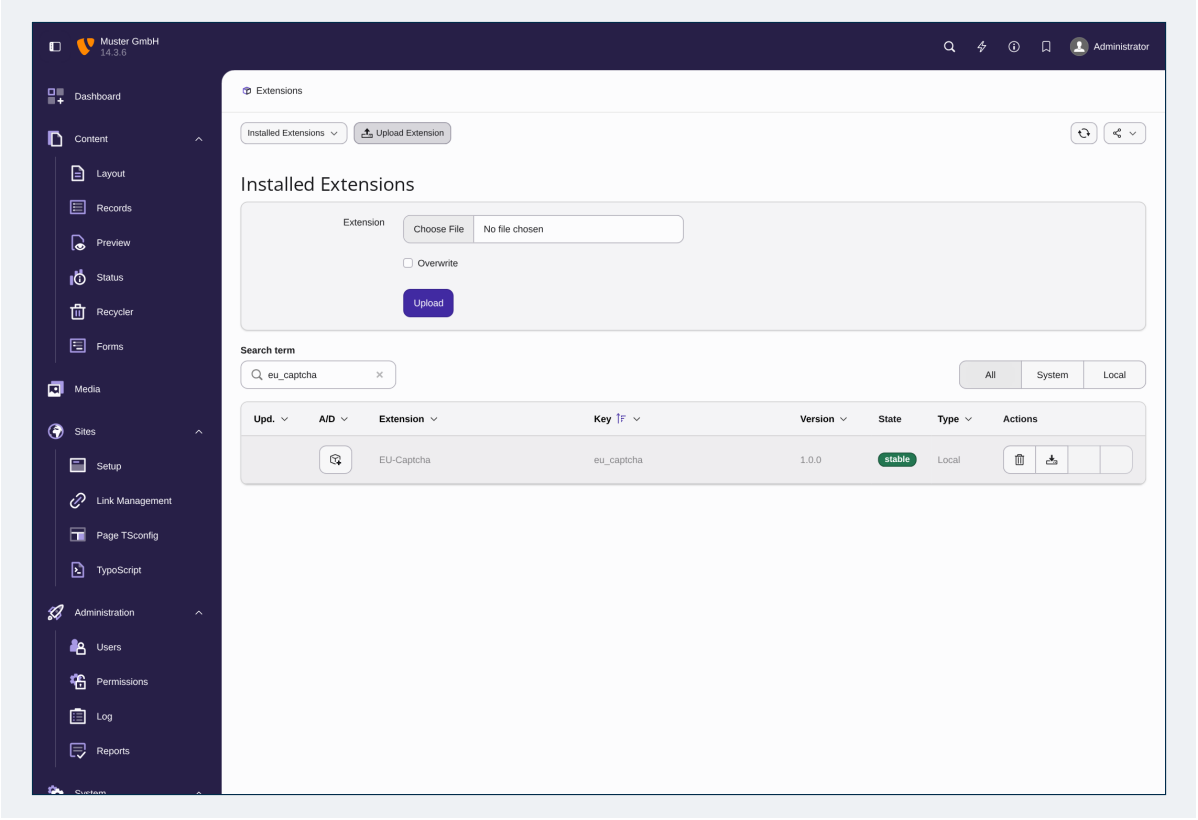


Figure 142: Extensions module with the open form for the upload

- In the **Integration** view, click on the **Download zip** button.
- In TYPO3, open the **System** → **Extensions** module.
- Click on the **Upload Extension** button.
- Select the archive and click on the **Upload** button.

  ↳ TYPO3 shows **Extension Upload** and puts the extension in the **Installed Extensions** list. The extension is not enabled yet.

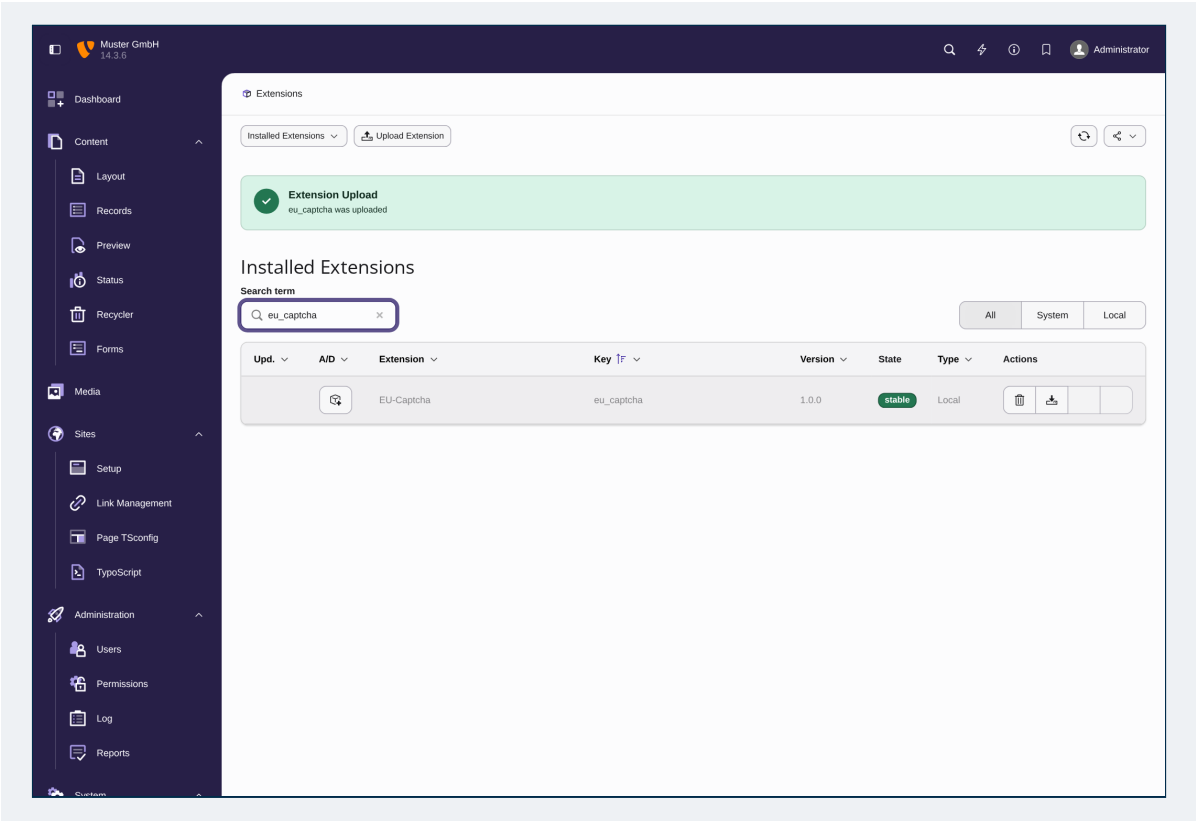


Figure 143: Installed Extensions list after the upload

- ► In the **A/D** column, click on the icon adjacent to the **EU-Captcha** extension.
- ► Confirm the prompt.
  - ↳ TYPO3 reloads the backend.
- ➔ The extension is enabled.

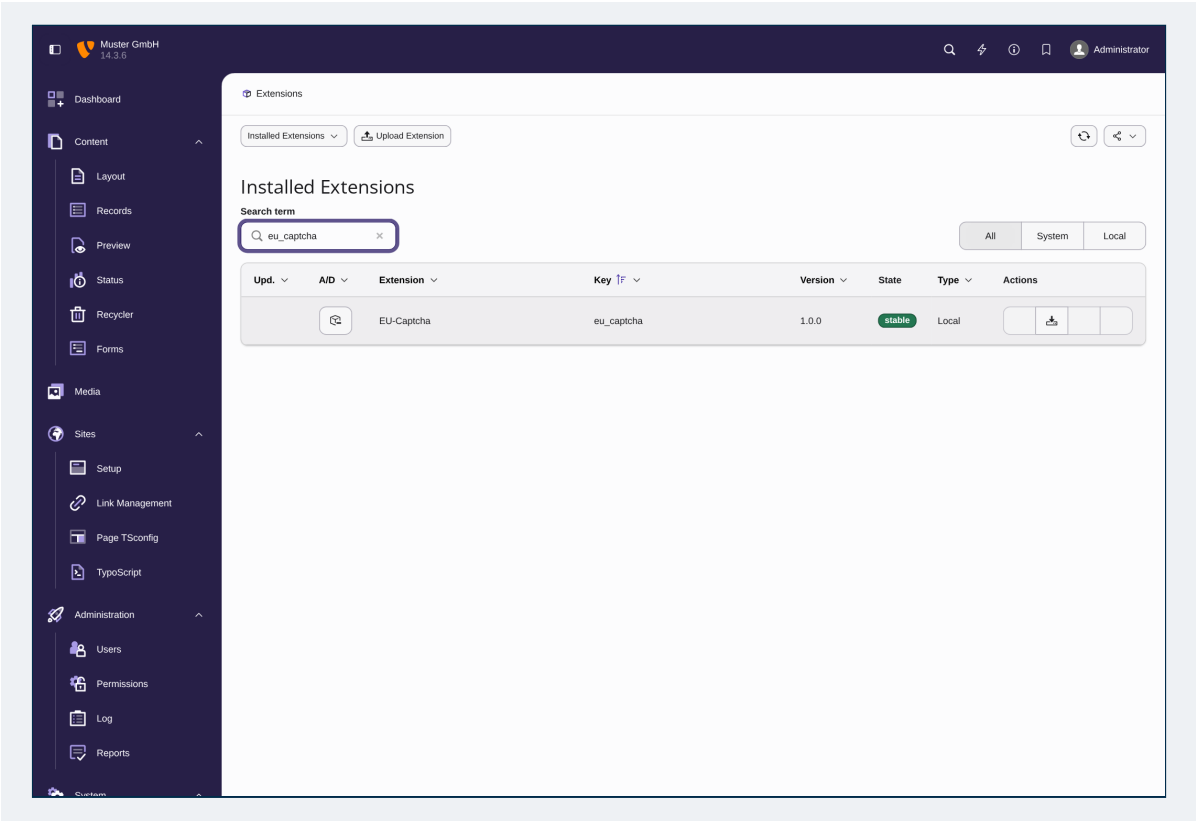


Figure 144: Installed Extensions list with the enabled extension

5.2.4.3 Embed the TypoScript

The TypoScript loads the script of the widget on each page in the frontend and adds the widget to the login form of `felogin`. Add this line to the TypoScript file of your website, for example `typo3conf/sites/main/setup.typoscript`:

```
@import 'EXT:eu_captcha/Configuration/TypoScript/setup.typoscript'
```

From TYPO3 14, the TypoScript of a website comes from a site set. The **Sites → TypoScript** module only shows the definitions and gives the note that they are changed in the file system.

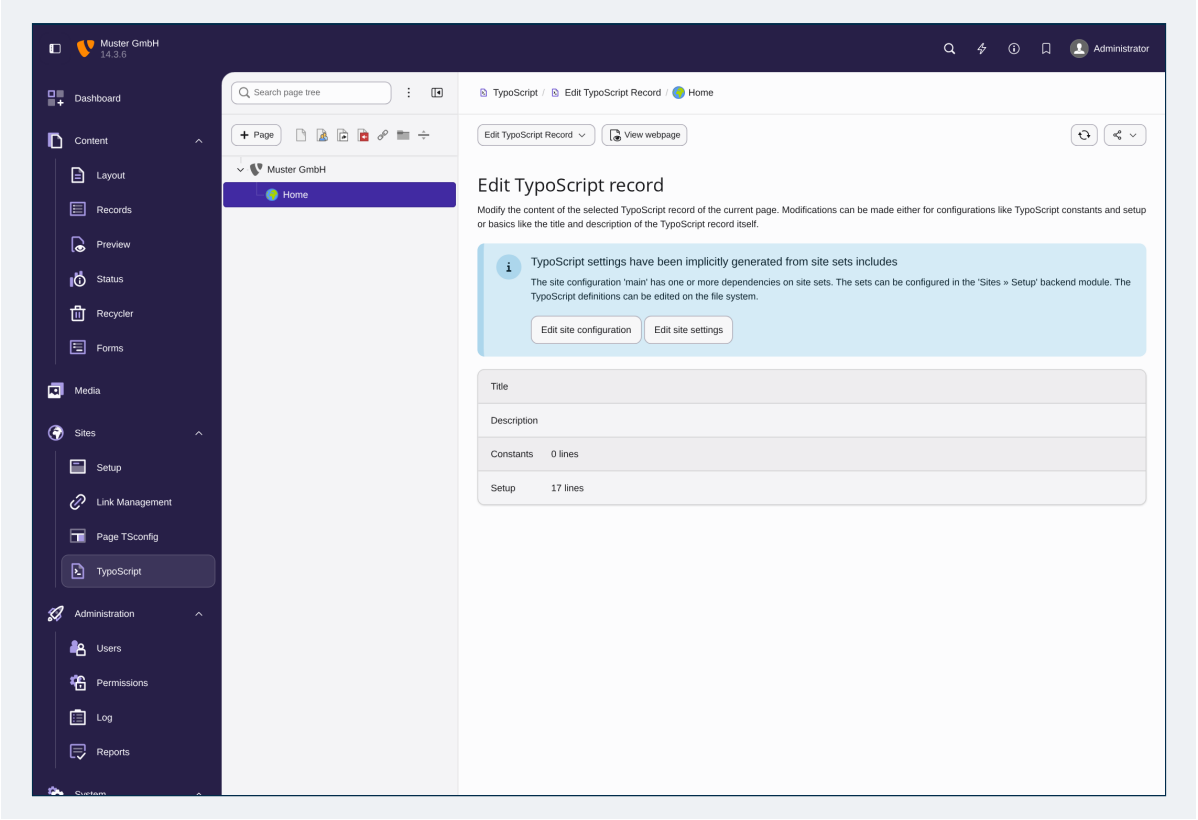


Figure 145: TypoScript module with the note about the site set

5.2.4.4 Store the credentials

Proceed as follows to store the credentials:

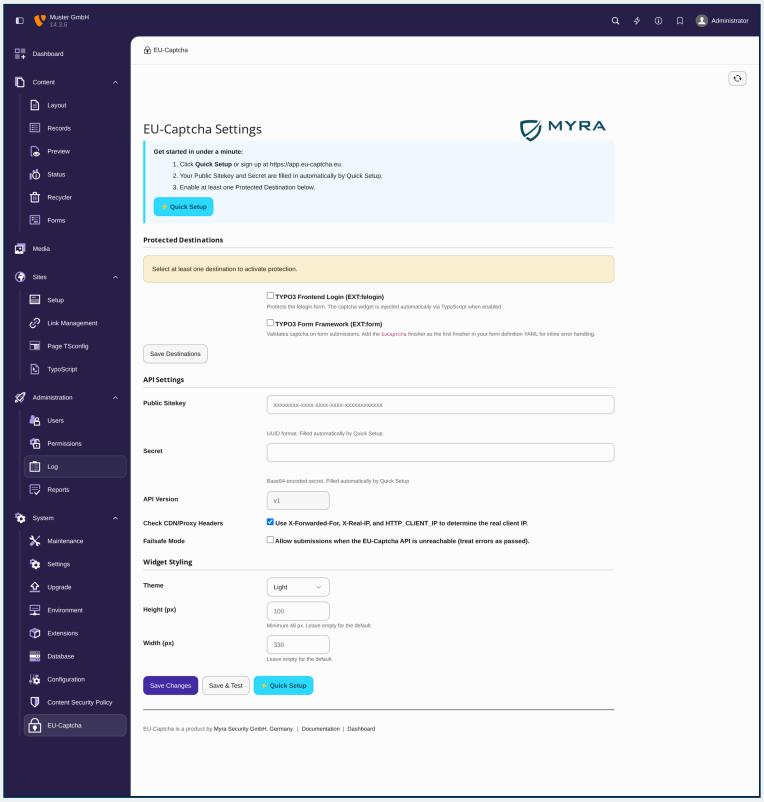


Figure 146: EU-Captcha module without credentials

- ▶ In TYPO3, open the **System** → **EU-Captcha** module.
- ▶ Click on the **Quick Setup** button and log in. As an alternative, enter the public sitekey in the **Public Sitekey** field and the secret in the **Secret** field manually. The two values are shown in the **Details** view of the sitekey.
- ▶ Click on the **Save Changes** button.
- ➔ The extension saves the credentials.

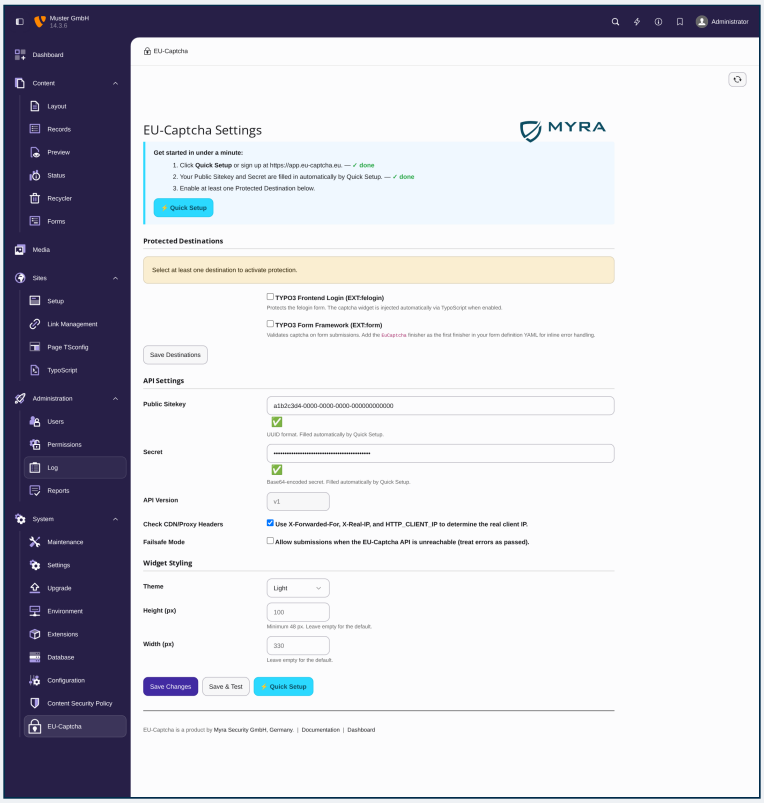


Figure 147: EU-Captcha module with the saved credentials

The **Save & Test** button saves the credentials and additionally examines them against the API.

The login in the backend of TYPO3 is protected as soon as the sitekey and the secret are saved. You cannot switch this verification off.

5.2.4.5 Select the forms

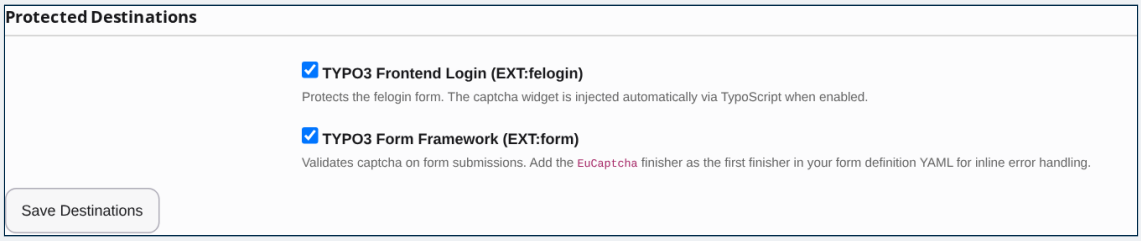


Figure 148: Protected Destinations area with the two selected destinations

In the **Protected Destinations** area, you select the protected forms. While no destination is selected, TYPO3 shows the `Select at least one destination to activate protection.` note.

Option	Range
TYPO3 Frontend Login (EXT:felogin)	Protects the login form of `felogin`. The widget is added automatically with the TypoScript.
TYPO3 Form Framework (EXT:form)	Verifies the transmission of forms from `EXT:form`. In the YAML definition of the form, enter the `EuCaptcha` finisher as the first finisher, so that errors are shown immediately in the form.

Then click on the **Save Destinations** button.

5.2.4.6 Options of the API

These options are available:

Option	Effect
Check CDN/ Proxy Headers	Reads the true IP address of the visitor from the headers of the system in front. Activate the option if TYPO3 is behind a CDN or a load balancer.
Failsafe Mode	Permits a transmission if the API of Myra EU CAPTCHA is not reachable.

5.2.4.7 Appearance of the widget

These options control the appearance:

Option	Default	Effect
Theme	**Light**	Light or dark appearance, values **Light** and **Dark**.
Height (px)	empty	Fixed height in pixels, minimum 48. With an empty field, the default value is applicable.

Option	Default	Effect
Width (px)	empty	Fixed width in pixels, minimum 1. With an empty field, the default value is applicable.

Click on the **Save Changes** button to save the appearance.

5.2.4.8 Content Security Policy

The extension supplies its own Content Security Policy. This policy permits the `script-src` directive for the `https://cdn.eu-captcha.eu` address in the backend and in the frontend.

The supplied policy contains the `frame-src` and `connect-src` directives only for the backend. If you use a Content Security Policy in the frontend, then add these two directives. Without this addition, the script loads, but the hidden iframe stays empty.

See Content Security Policy.

5.2.4.9 Test the integration

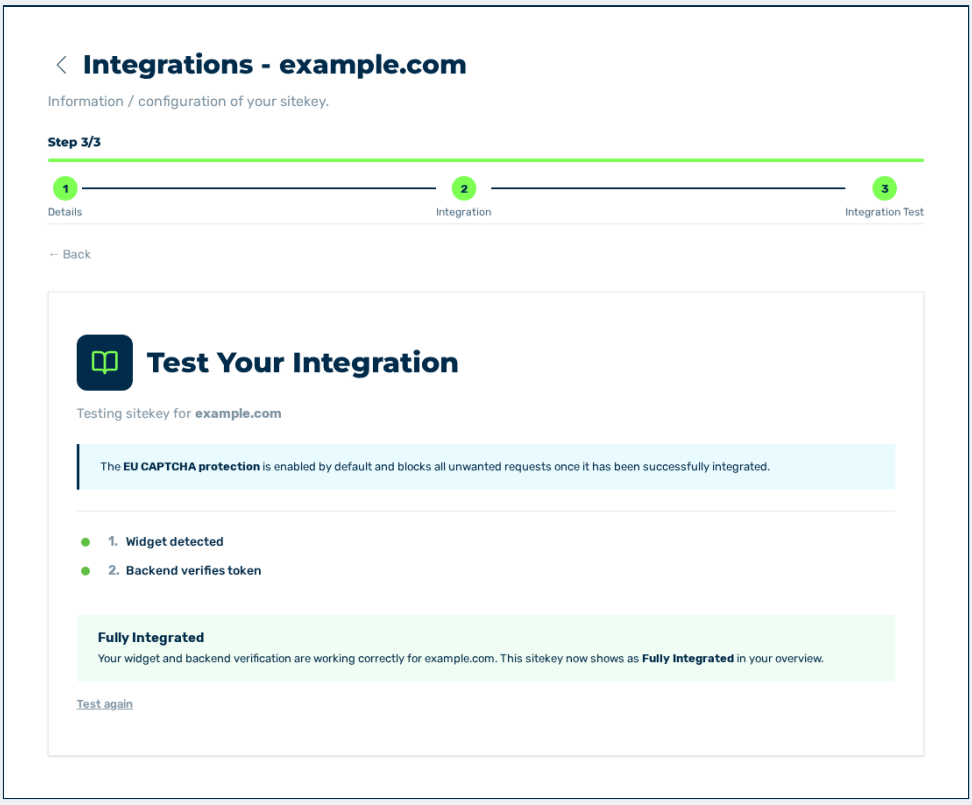


Figure 149: Integration Test view with the Fully Integrated result

After the setup, go to the **Integration Test** view of the sitekey and examine the embedding.

See **Testing the integration**.

5.3 Frontend

5.3.1 HTML and JavaScript

Without a framework, you embed the widget with two lines of HTML: a script link in the head area of the page and an element in the form. Use this method also for each content management system that permits your own HTML.

5.3.1.1 Requirements

The following requirements must be met:

Requirement	Value
Sitekey	Public sitekey from the **Details** view
Access	Write rights for the HTML template of the page
Server side	Endpoint that verifies the token

5.3.1.2 Embed the script

Add this line to the `<head>` area of your page:

```
<script src="https://cdn.eu-captcha.eu/verify.js" async defer></script>
```

If your website sends a Content Security Policy, then it must permit the `https://cdn.eu-captcha.eu` and `https://api.eu-captcha.eu` addresses. See **Content Security Policy**.

5.3.1.3 Add the widget

Add this element to each form to protect:

```
<div class="eu-captcha" data-sitekey="a1b2c3d4-0000-0000-0000-000000000000"></
div>
```

Replace the value of `data-sitekey` with your public sitekey.

Thus, the full form is as follows:

```
<form method="POST" action="/your-endpoint">
  <!-- your fields -->

  <div class="eu-captcha" data-
sitekey="a1b2c3d4-0000-0000-0000-000000000000"></div>

  <button type="submit">Submit</button>
</form>
```

During the transmission, the widget adds the hidden `eu-captcha-response` field with the token.

5.3.1.4 Attributes of the element

The `verify.js` script uses these attributes:

Attribute	Default	Effect
`data-sitekey`	—	Public sitekey. The value is necessary.
`data-theme`	`"light"`	Appearance, values `"light"` and `"dark"`.
`data-width`	`330`	Width of the widget in pixels.
`data-height`	`100`	Height of the widget in pixels.
`data-widgetid`	automatic	Your own identifier of the widget.
`data-autostart`	`"true"`	The `"false"` value delays the start of the challenge.
`data-callback`	—	Name of a global function that is called after the challenge passed.
`data-expired-callback`	—	Name of a global function that is called when the token expires.
`data-error-callback`	—	Name of a global function that is called when an error occurs.

5.3.1.5 Embed the package with npm

For projects with a bundler, the same widget is available as an npm package. The package supplies TypeScript types and a version control. The widget continues to load `verify.js` from the CDN.

Proceed as follows to embed the package:

- ▸ Install the package:

```
npm i @myrasec/eu-captcha-vanilla
```

- ▸ Make a target element in the form:

```
<form id="contact-form">
  <!-- your fields -->
  <div id="captcha"></div>
  <button type="submit">Submit</button>
</form>
```

- ▸ Render the widget into the target element:

```
import { renderEuCaptcha, isEuCaptchaDone } from "@myrasec/eu-captcha-
vanilla";

const captchaSitekey = "EUCAPTCHA_SITE_KEY";

renderEuCaptcha("#captcha", {
  sitekey: captchaSitekey,
  onComplete: (token: string) => console.log("token:", token),
}).catch((err) => console.error("EU CAPTCHA failed to render", err));
```

- ➔ The widget starts the verification in the background.

As the first argument, `renderEuCaptcha` accepts a CSS selector or an `HTMLElement`.

5.3.1.6 Options

These options are available:

Option	Type	Default	Effect
`sitekey`	`string`	—	Public sitekey. The value is necessary.
`theme`	`string`	`"light"`	Appearance, values `"light"` and `"dark"`.
`width`	`number`	`330`	Width of the widget in pixels.
`height`	`number`	`100`	Height of the widget in pixels.
`widgetId`	`string`	—	Your own identifier of the widget. Without a value, the package makes an identifier. For `euCaptcha.execute()`, the value is necessary.
`autostart`	`boolean`	`true`	Starts the challenge automatically. The `false` value delays the start until the `euCaptcha.execute(widgetId)` call.
`onComplete`	`(token: string) => void`	—	Is called with the token as soon as the challenge passed.
`onExpired`	`() => void`	—	Is called when the token expires. A token expires 60 minutes after the solution.
`onError`	`() => void`	—	Is called when a network error or a server error occurs.

5.3.1.7 Delay the start of the challenge

Set `autostart` to `false` and give an identifier, to start the challenge yourself:

```
renderEuCaptcha("#captcha", {
  sitekey: captchaSitekey,
```

```
  widgetId: "my-captcha",
  autostart: false,
});

document.getElementById("verify-btn")!.addEventListener("click", () => {
  (window as any).euCaptcha.execute("my-captcha");
});
```

5.3.1.8 Get the condition

Before the transmission, examine if the challenge passed:

```
import { isEuCaptchaDone } from "@myrasec/eu-captcha-vanilla";

function handleSubmit(e: SubmitEvent): void {
    e.preventDefault();

    if (!isEuCaptchaDone()) {
        // challenge not yet complete
        return;
    }
    // proceed with form submission
}
```

Without an argument, `isEuCaptchaDone()` is reliable only with exactly one widget without its own identifier. With more than one widget, give an identifier to each widget and get `isEuCaptchaDone(widgetId)`.

As an alternative, wait for the `euCaptchaCompleted` message at the window. Examine the origin of the message, because each script and each extension in the browser can send messages:

```
const CAPTCHA_ORIGIN = "https://cdn.eu-captcha.eu";

function listenForCaptchaDone(msg: MessageEvent): void {
    if (msg.origin !== CAPTCHA_ORIGIN) return;
    const data = (msg.data ?? {}) as { type?: string };
    if (data.type === "euCaptchaCompleted") {
        // enable submit button, update state, etc.
    }
}

window.addEventListener("message", listenForCaptchaDone, false);
```

The verification in the browser controls only the operation. Always also verify the token on the server.

5.3.1.9 Remove the widget

`renderEuCaptcha` returns a reference. The `handle.destroy()` call removes the message listener, clears the target element, and resets the condition. Thus, `isEuCaptchaDone()` gives `false` again.

```
const handle = await renderEuCaptcha("#captcha", { sitekey: captchaSitekey });
// later
handle.destroy();
```

Call `destroy()` for each removal. Each render with a callback function adds a message listener to the window, and only `destroy()` removes it again. Without this call, one listener and one detached DOM subtree stay in the memory for each render.

5.3.1.10 Verify the token

Verify the token on your server. See **Verify a client token**.

5.3.1.11 Full example

See **HTML and Django**.

5.3.2 React

The `@myrasec/eu-captcha` package supplies the `EuCaptcha` component. The component renders the widget in the form and gives the token to your source code.

5.3.2.1 Requirements

The following requirements must be met:

Requirement	Value
React	From version 18
Sitekey	Public sitekey from the **Details** view
Server side	Endpoint that verifies the token

5.3.2.2 Install the package

Install the package:

```
npm i @myrasec/eu-captcha
```

5.3.2.3 Embed the component

Embed the component in the form to protect:

```
import { EuCaptcha, isEuCaptchaDone } from "@myrasec/eu-captcha";

const captchaSitekey = "EUCAPTCHA_SITE_KEY";

export function ContactForm() {
  function handleSubmit(e: React.FormEvent<HTMLFormElement>): void {
    e.preventDefault();
    if (!isEuCaptchaDone()) {
      // challenge not yet complete
      return;
    }
    // proceed with form submission
  }

  return (
    <form onSubmit={handleSubmit}>
      {/* your fields */}
```

```
        <EuCaptcha sitekey={captchaSitekey} />
        <button type="submit">Submit</button>
      </form>
    );
  }
```

You can test the embedding with an invented sitekey. For an unknown sitekey, the CAPTCHA operates with the default values, and all traffic goes through.

5.3.2.4 Properties

These properties are available:

Property	Type	Default	Effect
`sitekey`	`string`	—	Public sitekey. The value is necessary.
`theme`	`string`	`"light"`	Appearance, values `"light"` and `"dark"`.
`width`	`number`	`330`	Width of the widget in pixels.
`height`	`number`	`100`	Height of the widget in pixels.
`widgetId`	`string`	—	Your own identifier of the widget. Without a value, the package makes an identifier. For `euCaptcha.execute()`, the value is necessary.
`autostart`	`boolean`	`true`	Starts the challenge automatically. The `false` value delays the start until the `euCaptcha.execute(widgetId)` call.
`onComplete`	`(token: string) => void`	—	Is called with the token as soon as the challenge passed.
		—	

Property	Type	Default	Effect
`onExpired`	`() => void`		Is called when the token expires. A token expires 60 minutes after the solution.
`onError`	`() => void`	—	Is called when a network error or a server error occurs.

Examples of the appearance:

```
<EuCaptcha sitekey={captchaSitekey} theme="dark" />
<EuCaptcha sitekey={captchaSitekey} width={280} />
<EuCaptcha sitekey={captchaSitekey} height={60} />
```

5.3.2.5 Use the callbacks

```
<EuCaptcha
  sitekey={captchaSitekey}
  onComplete={(token: string) => console.log("token:", token)}
  onExpired={() => console.log("token expired")}
  onError={() => console.log("challenge error")}
/>
```

5.3.2.6 Delay the start of the challenge

Set `autostart` to `false` and give an identifier:

```
<EuCaptcha
  sitekey={captchaSitekey}
  widgetId="my-captcha"
  autostart={false}
/>

<button onClick={() => (window as any).euCaptcha.execute("my-captcha")}
>Verify</button>
```

5.3.2.7 Next.js

The component uses interfaces of the browser. Thus, it operates only in the browser.

In the **App Router**, put `'use client'` at the start of the file:

```
'use client';

import { EuCaptcha, isEuCaptchaDone } from "@myrasec/eu-captcha";
```

In the **Pages Router**, load the component with `next/dynamic` and without rendering on the server:

```
import dynamic from "next/dynamic";
import { isEuCaptchaDone } from "@myrasec/eu-captcha";

const EuCaptcha = dynamic(
  () => import("@myrasec/eu-captcha").then((m) => m.EuCaptcha),
  { ssr: false }
);
```

The same procedure applies to Gatsby, because Gatsby also uses the `@myrasec/eu-captcha` package.

5.3.2.8 Use the package without React

You can also use the package without React. React stays a dependency of the package.

Proceed as follows to use the package without React:

- ► Load the parts asynchronously:

```
import { loadEuCaptcha, isEuCaptchaDone } from "@myrasec/eu-captcha";

loadEuCaptcha();
```

- ► Add the element to the form:

```
<div class="eu-captcha" data-sitekey="EUCAPTCHA_SITE_KEY"></div>
```

- ➔ The widget is rendered in the form.

The permitted attributes are given in **HTML and JavaScript**.

5.3.2.9 Get the condition

Before the transmission, examine if the challenge passed:

```
import { isEuCaptchaDone } from "@myrasec/eu-captcha";

function handleSubmit(e: React.FormEvent<HTMLFormElement>): void {
    e.preventDefault();

    if (!isEuCaptchaDone()) {
        // challenge not yet complete
        return;
    }
    // proceed with form submission
}
```

As an alternative, wait for the `euCaptchaDone` message at the window:

```
function listenForCaptchaDone(msg: MessageEvent): void {
    if (msg.data.type === "euCaptchaDone") {
        // enable submit button, update state, etc.
    }
}

window.addEventListener("message", listenForCaptchaDone, false);
```

The verification in the browser controls only the operation. Always also verify the token on the server. See **Verify a client token**.

5.3.2.10 Full example

See **React and PHP**.

5.3.3 Vue

The `@myrasec/eu-captcha-vue` package supplies the `EuCaptcha` component for Vue 3 and Nuxt.

5.3.3.1 Requirements

The following requirements must be met:

Requirement	Value
Vue	Version 3, also as the basis of Nuxt
Sitekey	Public sitekey from the **Details** view
Server side	Endpoint that verifies the token

5.3.3.2 Install the package

Install the package:

```
npm i @myrasec/eu-captcha-vue
```

5.3.3.3 Embed the component

Embed the component in the form to protect:

```
<script setup lang="ts">
import { EuCaptcha, isEuCaptchaDone } from "@myrasec/eu-captcha-vue";

const captchaSitekey = "EUCAPTCHA_SITE_KEY";

function handleSubmit(): void {
  if (!isEuCaptchaDone()) {
    // challenge not yet complete
    return;
  }
  // proceed with form submission
}
</script>

<template>
  <form @submit.prevent="handleSubmit">
    <!-- your fields -->
```

```
    <EuCaptcha :sitekey="captchaSitekey" />
    <button type="submit">Submit</button>
  </form>
</template>
```

> You can test the embedding with an invented sitekey. For an unknown sitekey, the CAPTCHA operates with the default values, and all traffic goes through.

5.3.3.4 Properties

These properties are available:

Property	Type	Default	Effect
`sitekey`	`string`	—	Public sitekey. The value is necessary.
`theme`	`string`	`"light"`	Appearance, values `"light"` and `"dark"`.
`width`	`number`	`330`	Width of the widget in pixels.
`height`	`number`	`100`	Height of the widget in pixels.
`widgetId`	`string`	—	Your own identifier of the widget. Without a value, the package makes an identifier. For `euCaptcha.execute()`, the value is necessary.
`autostart`	`boolean`	`true`	Starts the challenge when the component is mounted. The `false` value delays the start until the `euCaptcha.execute(widgetId)` call.
`onComplete`	`(token: string) => void`	—	Is called with the token as soon as the challenge passed.
		—	

Property	Type	Default	Effect
`onExpired`	`() => void`		Is called when the token expires. A token expires 60 minutes after the solution.
`onError`	`() => void`	—	Is called when a network error or a server error occurs.

Examples of the appearance:

```
<EuCaptcha :sitekey="captchaSitekey" theme="dark" />
<EuCaptcha :sitekey="captchaSitekey" :width="280" />
<EuCaptcha :sitekey="captchaSitekey" :height="60" />
```

5.3.3.5 Use the callbacks

```
<EuCaptcha
  :sitekey="captchaSitekey"
  :on-complete="(token) => console.log('token:', token)"
  :on-expired="() => console.log('token expired')"
  :on-error="() => console.log('challenge error')"
/>
```

5.3.3.6 Delay the start of the challenge

Set `autostart` to `false` and give an identifier:

```
<EuCaptcha
  :sitekey="captchaSitekey"
  widget-id="my-captcha"
  :autostart="false"
/>

<button @click="euCaptcha.execute('my-captcha')">Verify</button>
```

5.3.3.7 Get the condition

Before the transmission, examine if the challenge passed:

```
import { isEuCaptchaDone } from "@myrasec/eu-captcha-vue";

function handleSubmit(): void {
    if (!isEuCaptchaDone()) {
        // challenge not yet complete
        return;
    }
    // proceed with form submission
}
```

As an alternative, wait for the `euCaptchaDone` message at the window:

```
import { onMounted, onUnmounted } from "vue";

function listenForCaptchaDone(msg: MessageEvent): void {
    if (msg.data.type === "euCaptchaDone") {
        // enable submit button, update reactive state, etc.
    }
}

onMounted(() => window.addEventListener("message", listenForCaptchaDone,
false));
onUnmounted(() => window.removeEventListener("message", listenForCaptchaDone,
false));
```

The verification in the browser controls only the operation. Always also verify the token on the server. See **Verify a client token**.

5.3.3.8 Full example

See **Vue and Python**.

5.3.4 Angular

For Angular, there is a different package for each major version. All packages supply the `eu-captcha` component with the same inputs and outputs.

5.3.4.1 Requirements

The following requirements must be met:

Requirement	Value
Angular	Version 19, 20, or 21
Sitekey	Public sitekey from the **Details** view
Server side	Endpoint that verifies the token

5.3.4.2 Select the package

Select the package for the major version of your project:

Package	Angular
`@myrasec/eu-captcha-angular19`	Version 19
`@myrasec/eu-captcha-angular20`	Version 20
`@myrasec/eu-captcha-angular21`	Version 21

Install the selected package. The example shows Angular 21:

```
npm i @myrasec/eu-captcha-angular21
```

5.3.4.3 Embed the component

In a standalone component, embed `EuCaptchaComponent`:

```
import { Component } from '@angular/core';
import { EuCaptchaComponent, isEuCaptchaDone } from '@myrasec/eu-captcha-
angular21';

@Component({
```

```
  standalone: true,
  imports: [EuCaptchaComponent],
  template: `
    <form (submit)="handleSubmit($event)">
      <!-- your fields -->
      <eu-captcha sitekey="EUCAPTCHA_SITE_KEY" />
      <button type="submit">Submit</button>
    </form>
  `,
})
export class MyFormComponent {
  handleSubmit(event: Event): void {
    event.preventDefault();
    if (!isEuCaptchaDone()) {
      // challenge not yet complete
      return;
    }
    // proceed with form submission
  }
}
```

In an application with modules, embed `EuCaptchaModule` instead:

```
import { NgModule } from '@angular/core';
import { EuCaptchaModule } from '@myrasec/eu-captcha-angular21';

@NgModule({
  imports: [EuCaptchaModule],
})
export class AppModule {}
```

```
<!-- in your template -->
<eu-captcha sitekey="EUCAPTCHA_SITE_KEY" />
```

You can test the embedding with an invented sitekey. For an unknown sitekey, the CAPTCHA operates with the default values, and all traffic goes through.

5.3.4.4 Inputs

These inputs are available:

Input	Type	Default	Effect
`sitekey`	`string`	—	Public sitekey. The value is necessary.
`theme`	`string`	`"light"`	Appearance, values `"light"` and `"dark"`.
`width`	`number`	`330`	Width of the widget in pixels.
`height`	`number`	`100`	Height of the widget in pixels.
`widgetId`	`string`	—	Your own identifier of the widget. Without a value, the package makes an identifier. For `euCaptcha.execute()`, the value is necessary.
`autostart`	`boolean`	`true`	Starts the challenge when the component is mounted. The `false` value delays the start until the `euCaptcha.execute(widgetId)` call.

5.3.4.5 Outputs

These outputs are available:

Output	Value	Effect
`completed`	`string`	Is triggered with the token as soon as the challenge passed.
`expired`	—	Is triggered when the token expires. A token expires 60 minutes after the solution.
`error`	—	Is triggered when a network error or a server error occurs.

Example of the evaluation:

```
<eu-captcha
  sitekey="EUCAPTCHA_SITE_KEY"
  (completed)="onToken($event)"
  (expired)="onExpired()"
  (error)="onError()"
/>
```

```
onToken(token: string): void {
  console.log('token:', token);
}

onExpired(): void {
  console.log('token expired');
}

onError(): void {
  console.log('challenge error');
}
```

Examples of the appearance:

```
<eu-captcha sitekey="EUCAPTCHA_SITE_KEY" theme="dark" />
<eu-captcha sitekey="EUCAPTCHA_SITE_KEY" [width]="280" />
<eu-captcha sitekey="EUCAPTCHA_SITE_KEY" [height]="60" />
```

5.3.4.6 Delay the start of the challenge

Set `autostart` to `false` and give an identifier:

```
<eu-captcha
  sitekey="EUCAPTCHA_SITE_KEY"
  widgetId="my-captcha"
  [autostart]="false"
/>

<button (click)="euCaptcha.execute('my-captcha')">Verify</button>
```

5.3.4.7 Get the condition

Before the transmission, examine if the challenge passed:

```
import { isEuCaptchaDone } from '@myrasec/eu-captcha-angular21';

function handleSubmit(): void {
    if (!isEuCaptchaDone()) {
        // challenge not yet complete
        return;
    }
    // proceed with form submission
}
```

As an alternative, wait for the `euCaptchaDone` message at the window:

```
import { Component, OnInit, OnDestroy } from '@angular/core';

@Component({ /* ... */ })
export class MyComponent implements OnInit, OnDestroy {
  private listener = (msg: MessageEvent) => {
    if (msg.data.type === 'euCaptchaDone') {
      // enable submit button, update state, etc.
    }
  };

  ngOnInit(): void {
    window.addEventListener('message', this.listener, false);
  }

  ngOnDestroy(): void {
    window.removeEventListener('message', this.listener, false);
  }
}
```

The verification in the browser controls only the operation. Always also verify the token on the server. See **Verify a client token**.

5.3.4.8 Full example

See **Angular and Java**.

5.3.5 Svelte

The `@myrasec/eu-captcha-svelte` package supplies the `EuCaptcha` component for Svelte 5.

5.3.5.1 Requirements

The following requirements must be met:

Requirement	Value
Svelte	Version 5, also as the basis of SvelteKit
Sitekey	Public sitekey from the **Details** view
Server side	Endpoint that verifies the token

5.3.5.2 Install the package

Install the package:

```
npm i @myrasec/eu-captcha-svelte
```

5.3.5.3 Embed the component

Embed the component in the form to protect:

```
<script lang="ts">
  import { EuCaptcha, isEuCaptchaDone } from "@myrasec/eu-captcha-svelte";

  const captchaSitekey = "EUCAPTCHA_SITE_KEY";

  function handleSubmit(e: SubmitEvent): void {
    e.preventDefault();
    if (!isEuCaptchaDone()) {
      // challenge not yet complete
      return;
    }
    // proceed with form submission
  }
</script>

<form onsubmit={handleSubmit}>
  <!-- your fields -->
```

```
  <EuCaptcha sitekey={captchaSitekey} />
  <button type="submit">Submit</button>
</form>
```

You can test the embedding with an invented sitekey. For an unknown sitekey, the CAPTCHA operates with the default values, and all traffic goes through.

5.3.5.4 Properties

These properties are available:

Property	Type	Default	Effect
`sitekey`	`string`	—	Public sitekey. The value is necessary.
`theme`	`string`	`"light"`	Appearance, values `"light"` and `"dark"`.
`width`	`number`	`330`	Width of the widget in pixels.
`height`	`number`	`100`	Height of the widget in pixels.
`widgetId`	`string`	—	Your own identifier of the widget. Without a value, the package makes an identifier. For `euCaptcha.execute()`, the value is necessary.
`autostart`	`boolean`	`true`	Starts the challenge when the component is mounted. The `false` value delays the start until the `euCaptcha.execute(widgetId)` call.
`onComplete`	`(token: string) => void`	—	Is called with the token as soon as the challenge passed.
`onExpired`	`() => void`	—	Is called when the token expires. A token expires 60 minutes after the solution.

Property	Type	Default	Effect
`onError`	`() => void`	—	Is called when a network error or a server error occurs.

Examples of the appearance:

```
<EuCaptcha sitekey={captchaSitekey} theme="dark" />
<EuCaptcha sitekey={captchaSitekey} width={280} />
<EuCaptcha sitekey={captchaSitekey} height={60} />
```

5.3.5.5 Use the callbacks

```
<EuCaptcha
  sitekey={captchaSitekey}
  onComplete={(token) => console.log("token:", token)}
  onExpired={() => console.log("token expired")}
  onError={() => console.log("challenge error")}
/>
```

5.3.5.6 Delay the start of the challenge

Set `autostart` to `false` and give an identifier:

```
<EuCaptcha
  sitekey={captchaSitekey}
  widgetId="my-captcha"
  autostart={false}
/>

<button onclick={() => (window as any).euCaptcha.execute("my-captcha")}
>Verify</button>
```

5.3.5.7 Get the condition

Before the transmission, examine if the challenge passed:

```
import { isEuCaptchaDone } from "@myrasec/eu-captcha-svelte";

function handleSubmit(e: SubmitEvent): void {
    e.preventDefault();
    if (!isEuCaptchaDone()) {
        // challenge not yet complete
        return;
    }
    // proceed with form submission
}
```

As an alternative, wait for the `euCaptchaDone` message at the window:

```
import { onMount, onDestroy } from "svelte";

function listenForCaptchaDone(msg: MessageEvent): void {
    if (msg.data.type === "euCaptchaDone") {
        // enable submit button, update reactive state, etc.
    }
}

onMount(() => window.addEventListener("message", listenForCaptchaDone,
false));
onDestroy(() => window.removeEventListener("message", listenForCaptchaDone,
false));
```

5.3.5.8 SvelteKit

The component uses interfaces of the browser and does its full setup in `onMount`. Thus, you can also embed it in pages that are rendered on the server. No more settings are necessary. A test for `browser` or a dynamic import is not necessary.

The verification in the browser controls only the operation. Always also verify the token on the server. See **Verify a client token**.

5.3.5.9 Full example

See **Svelte and Ruby**.

5.4 Backend

5.4.1 PHP

For PHP, there are two packages. Both verify the token on the server and give the same result object.

5.4.1.1 Select the package

Select the package for the PHP version of your application:

Package	PHP	Technology
`myra-security-gmbh/eu-captcha`	From version 8.0	Named arguments, type declarations, HTTP with Guzzle (`guzzlehttp/guzzle` ^6.0 or ^7.0)
`myra-security-gmbh/eu-captcha-old`	From version 5.0	No other dependencies, HTTP with `file_get_contents()` and a stream context

5.4.1.2 Requirements

The following requirements must be met:

Requirement	Value
PHP	From version 8.0 for `eu-captcha` , from version 5.0 for `eu-captcha-old`
Composer	Access to the command line of the server
Credentials	Public sitekey and secret from the **Details** view

5.4.1.3 Install the package

Install the package for PHP 8:

```
composer require myra-security-gmbh/eu-captcha
```

For PHP 5 to PHP 7, install this package instead:

```
composer require myra-security-gmbh/eu-captcha-old
```

5.4.1.4 Embed the widget

Add the script link to each page that has a form to protect:

```
<script src="https://cdn.eu-captcha.eu/verify.js" async defer></script>
```

Add the widget to the form:

```
<div class="eu-captcha" data-sitekey="EUCAPTCHA_SITE_KEY"></div>
```

In a single-page application with React, Vue, or Angular, embed the widget with the applicable npm package. Then the PHP package does only the verification on the server.

5.4.1.5 Verify the token

Verify the transmitted token on the server:

```
<?php

use Myrasec\EuCaptcha;

$captcha = new EuCaptcha(
    sitekey: EUCAPTCHA_SITE_KEY,
    secret:  EUCAPTCHA_SECRET_KEY,
);
```

```
$result = $captcha->validate();

if (!$result->success()) {
    // Reject the form submission
}
```

With `eu-captcha-old`, give the values as an associative array instead:

```
<?php

use Myrasec\EuCaptcha;

$captcha = new EuCaptcha([
    'sitekey' => EUCAPTCHA_SITE_KEY,
    'secret'  => EUCAPTCHA_SECRET_KEY,
]);

$result = $captcha->validate();
```

`validate()` reads the token from `$_POST['eu-captcha-response']`. If `$_POST` is empty, the method reads the body of the request as JSON. The method gets the IP address of the visitor from the headers of the request.

5.4.1.6 Options

These options are available:

Option	Type	Default	Effect
`sitekey`	string	—	Public sitekey. The value is necessary.
`secret`	string	—	Secret key. The value is necessary and must not occur in the browser.

Option	Type	Default	Effect
`failDefault`	`bool`	`true`	Return value for the network condition and the token condition when the API is not available. `true` permits the transmission, `false` rejects it.
`checkCdnHeaders`	`bool`	`true`	Gets the IP address of the visitor from the `HTTP_CLIENT_IP`, `HTTP_X_FORWARDED_FOR`, and `HTTP_X_REAL_IP` headers before `REMOTE_ADDR` is used. Set the value to `false` if the server is not behind an upstream system, or if you give the IP address yourself.

Only `eu-captcha` for PHP 8 has these additional options:

Option	Type	Default	Effect
`verifyUrl`	string	Address of the production environment	Overwrites the address of the `/verify` endpoint. Use the option for tests.
`credentialsUrl`	string	Address of the production environment	Overwrites the address of the `/verify-credentials` endpoint.
`client`	`?Client`	`null`	Your own instance of Guzzle for different settings or for tests.

5.4.1.7 The result object

`validate()` gives an `EuCaptchaResult` object with three methods:

Method	Gives `true` when
`success()`	the API was available and the token is valid.
`successNetwork()`	the call of the API completed without a network error or a transmission error.
`successToken()`	the API reported the transmitted token as valid.

The separate query tells a failed challenge from a malfunction of the API:

```php
<?php

$result = $captcha->validate();

if (!$result->successNetwork()) {
    // Could not reach the API — consider logging or alerting
}

if (!$result->successToken()) {
    // Token was rejected — the submission is likely automated
}
```

5.4.1.8 Give the token and the IP address yourself

For different field names, give the token and the IP address yourself:

```php
<?php

$token    = $_POST['my-captcha-field'] ?? '';
$clientIp = $_SERVER['REMOTE_ADDR'];

$result = $captcha->validate($token, $clientIp);
```

5.4.1.9 Verify the credentials

With `verifyCredentials()`, you verify the sitekey and the secret without a token from the browser, for example at the start of the application:

```php
<?php

$captcha = new EuCaptcha(sitekey: EUCAPTCHA_SITE_KEY, secret:
EUCAPTCHA_SECRET_KEY);

if (!$captcha->verifyCredentials()) {
    // Credentials are invalid or the API is unreachable — log and alert
}
```

For a network error or an API error, the method gives `false` and causes no exception. Thus, the call is also safe during the initialization.

See **Verify the sitekey and the secret**.

5.4.1.10 Symfony and Laravel

For the two frameworks, see **Symfony and Laravel**.

5.4.1.11 Full example

See **React and PHP**.

5.4.2 Symfony and Laravel

Symfony and Laravel use the same `myra-security-gmbh/eu-captcha` package as each other application with PHP 8. The two frameworks have their own function to get the IP address of the visitor, which you give to `validate()` .

5.4.2.1 Requirements

The following requirements must be met:

Requirement	Value
PHP	From version 8.0
Package	`myra-security-gmbh/eu-captcha`
Credentials	Public sitekey and secret from the **Details** view

The installation of the package is given in **PHP**.

5.4.2.2 Symfony

In your services and controllers, use the `EuCaptchaInterface` type. Then Symfony connects the service automatically. Your source code does not depend on the applicable class.

Enter the service in `config/services.yaml` and point the interface to it:

```
services:
    Myrasec\EuCaptcha:
        arguments:
            $sitekey: '%env(EUCAPTCHA_SITE_KEY)%'
            $secret:  '%env(EUCAPTCHA_SECRET_KEY)%'
    Myrasec\EuCaptchaInterface: '@Myrasec\EuCaptcha'
```

Give the interface to the controller:

```
<?php

namespace App\Controller;

use Myrasec\EuCaptchaInterface;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
```

```
use Symfony\Component\Routing\Attribute\Route;

class ContactController
{
    public function __construct(private EuCaptchaInterface $captcha) {}

    #[Route('/contact', methods: ['POST'])]
    public function submit(Request $request): Response
    {
        $result = $this->captcha->validate(
            $request->request->getString('eu-captcha-response'),
            $request->getClientIp() ?? '',
        );

        if (!$result->success()) {
            return new Response('CAPTCHA verification failed',
Response::HTTP_BAD_REQUEST);
        }

        // process the form...

        return new Response('OK');
    }
}
```

`$request->getClientIp()` obeys the setting of the trusted upstream systems of Symfony. Thus, behind a CDN or a load balancer, the true IP address of the visitor goes to the API. As the third argument, `validate()` also accepts the identifier of the browser.

5.4.2.3 Laravel

Put the credentials in the `.env` file and publish them with `config/services.php`.

The `.env` file:

```
EUCAPTCHA_SITE_KEY=YOUR_SITEKEY
EUCAPTCHA_SECRET_KEY=YOUR_SECRET
```

The `config/services.php` file:

```
'eucaptcha' => [
    'sitekey' => env('EUCAPTCHA_SITE_KEY'),
    'secret'  => env('EUCAPTCHA_SECRET_KEY'),
],
```

Verify the token in the controller:

```php
<?php

namespace App\Http\Controllers;

use Myrasec\EuCaptcha;
use Illuminate\Http\Request;
use Illuminate\Http\RedirectResponse;

class ContactController extends Controller
{
    public function submit(Request $request): RedirectResponse
    {
        $captcha = new EuCaptcha(
            sitekey: config('services.eucaptcha.sitekey'),
            secret:  config('services.eucaptcha.secret'),
        );

        $result = $captcha->validate(
            $request->input('eu-captcha-response'),
            $request->ip(),
        );

        if (!$result->success()) {
            return back()->withErrors(['captcha' => 'CAPTCHA verification
failed.']);
        }

        // process the form...

        return redirect()->route('contact.success');
    }
}
```

`$request->ip()` obeys the setting of the trusted upstream systems of Laravel.

5.4.2.4 Verification in a form request

For more than one controller, put the verification in its own `FormRequest`:

```php
<?php

namespace App\Http\Requests;

use Myrasec\EuCaptcha;
use Illuminate\Foundation\Http\FormRequest;
use Illuminate\Contracts\Validation\Validator;

class ContactRequest extends FormRequest
```

```
{
    public function rules(): array
    {
        return [
            'name'    => ['required', 'string', 'max:255'],
            'email'   => ['required', 'email'],
            'message' => ['required', 'string'],
        ];
    }

    protected function withValidator(Validator $validator): void
    {
        $validator->after(function (Validator $validator) {
            $captcha = new EuCaptcha(
                sitekey: config('services.eucaptcha.sitekey'),
                secret:  config('services.eucaptcha.secret'),
            );

            $result = $captcha->validate(
                $this->input('eu-captcha-response'),
                $this->ip(),
            );

            if (!$result->success()) {
                $validator->errors()->add('captcha', 'CAPTCHA verification
failed.');
            }
        });
    }
}
```

Then give `ContactRequest` to the method of the controller instead of `Request`. Laravel then verifies the request before it calls the method:

```
public function submit(ContactRequest $request): RedirectResponse
{
    // validation and CAPTCHA check already passed
    // process the form...

    return redirect()->route('contact.success');
}
```

5.4.3 Python

The `myra-eucaptcha` package verifies the token on the server. It supplies a synchronous interface and an asynchronous interface.

5.4.3.1 Requirements

The following requirements must be met:

Requirement	Value
Package manager	`pip` or `uv`
Credentials	Public sitekey and secret from the **Details** view

5.4.3.2 Install the package

Install the package from PyPI:

```
pip install myra-eucaptcha
```

With the `uv` package manager, the command is as follows:

```
uv add myra-eucaptcha
```

5.4.3.3 Embed the widget

Add the widget to the form to protect:

```
<div class="eu-captcha" data-sitekey="EUCAPTCHA_SITE_KEY"></div>
```

Add the script link to the page:

```
<script src="https://cdn.eu-captcha.eu/verify.js" async defer></script>
```

The script renders the widget, does the challenge, and adds the token to the form.

Thus, a full page is as follows:

```
<html>
    <head>
        <title>captcha integration sample</title>
    </head>
    <body>

    <form action="/login" method="post">
        <table>
            <tr>
                <td>user</td>
                <td>
                    <input type=text name="username" />
                </td>
            </tr>
            <tr>
                <td>pass</td>
                <td>
                    <input type=password name="password" />
                </td>
            </tr>
            <tr>
                <td colspan="2">
                    <div class="eu-captcha" data-
sitekey="EUCAPTCHA_SITE_KEY"></div>
                </td>
            </tr>
            <tr><td colspan="2"><input type="submit" name="submit" /></td></
tr>
        </table>
    </form>

    <script src="https://cdn.eu-captcha.eu/verify.js" async defer></script>
    </body>
</html>
```

5.4.3.4 Verify the token

This example shows the verification on the server with FastAPI:

```
from myra_eucaptcha import MyraEuCaptchaClient, MyraEuCaptchaClientConfig

@app.post("/login", response_class=HTMLResponse)
async def login(
    request:Request,
    username: str = Form(...),
    password: str = Form(...),
    eu_captcha_response: str = Form(..., alias="eu-captcha-response"),
):
```

```
    # configure the captcha settings and behaviour
    captcha_config = MyraEuCaptchaClientConfig(
        sitekey="EUCAPTCHA_SITE_KEY",
        secret="EUCAPTCHA_SECRET_KEY",
    )

    client = MyraEuCaptchaClient(config=captcha_config)

    # this is depending on your webserver-setup and whether we're behind
    # a reverse-proxy (nginx, apache, traefik, or something similar.)
    remote_addr = request.headers.get('x-real-ip','')
    validationresult = await client.avalidate(
                token=eu_captcha_response,
                remote_addr=remote_addr)

    if validationresult.success:
        # yay - all good.
        # <your-code-here>
        pass
    else:
        # validation failed, check validationresult.errors
        # <your-code-here>
        pass
```

The asynchronous verification uses `avalidate()`, the synchronous verification uses `validate()`.

The source of the IP address of the visitor depends on your web server. If the application is behind an upstream system such as nginx, Apache, or Traefik, read the address from the applicable header.

5.4.3.5 Behaviour for errors

By default, the package is fault-tolerant: for a timeout, an API error, or a network error, the verification gives the `True` result. No exception occurs. The errors that occurred are given in `MyraEuCaptchaResult.errors`.

You change this behaviour with `MyraEuCaptchaClientConfig`:

```
    captcha_config = MyraEuCaptchaClientConfig(
        sitekey="EUCAPTCHA_SITE_KEY",
        secret="EUCAPTCHA_SECRET_KEY",
        connect_timeout: int = 3
        read_timeout: int = 10
        write_timeout: int = 10
        pool_timeout: int = 3
        default_result_on_error:bool = True ## this is the validation result,
that is returned on exceptions
```

```
        suppress_exceptions:bool = True     ## if you want exceptions to be
raised, set this to False
    )
```

Setting	Default	Effect
`connect_timeout`	`3`	Time limit for the connection setup in seconds.
`read_timeout`	`10`	Time limit to read the response in seconds.
`write_timeout`	`10`	Time limit to send the request in seconds.
`pool_timeout`	`3`	Time limit for an available connection from the connection pool in seconds.
`default_result_on_error`	`True`	Result of the verification when an exception occurs.
`suppress_exceptions`	`True`	Set to `False`, the package causes the exception instead of suppressing it.

For Django, see **Django**.

5.4.3.6 Full example

See **Vue and Python**.

5.4.4 Django

The `myra-eucaptcha-django` package supplies a form field, a widget, and a management function for the configurations in the administration area of Django.

5.4.4.1 Requirements

The following requirements must be met:

Requirement	Value
Package manager	`pip` or `uv`
Access	Administration area of Django
Credentials	Public sitekey and secret from the **Details** view

5.4.4.2 Install the package

Proceed as follows to set up the package:

- Install the package:

```
pip install myra-eucaptcha-django
```

With the `uv` package manager, the command is as follows:

```
uv add myra-eucaptcha-django
```

- Enter the application in `settings.py` :

```
# settings.py
INSTALLED_APPS = [
    ...
    'myra_eucaptcha_django',
]
```

- Do the migrations:

```
python manage.py migrate
```

→ The administration area shows the **Captcha Configurations** entry.

5.4.4.3 Make a configuration

Proceed as follows to make a configuration:

- In the administration area, open the **Captcha Configurations** entry.
- Make a configuration with your sitekey and your secret.
- Set the configuration to **Default**, so that forms use it without a name.

→ The configuration is available in the forms.

5.4.4.4 Put the field in a form

Add the `CaptchaField` field and `CaptchaFormMixin` to the form:

```
from django import forms
from myra_eucaptcha_django import CaptchaField, CaptchaFormMixin

class ContactForm(CaptchaFormMixin, forms.Form):
    name = forms.CharField()
    email = forms.EmailField()
    message = forms.CharField(widget=forms.Textarea)

    # Uses the default configuration
    captcha = CaptchaField()
```

In the view, give the request to the field, so that the field can verify the IP address of the visitor:

```
from django.shortcuts import render, redirect
from .forms import ContactForm

def contact_view(request):
    if request.method == "POST":
        # Pass request= to enable IP validation
        form = ContactForm(request.POST, request=request)
        if form.is_valid():
            # Process the form
```

```
            return redirect("success")
    else:
        form = ContactForm()

    return render(request, "contact.html", {"form": form})
```

Render the form in the template:

```
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Send</button>
</form>
```

The widget supplies the necessary JavaScript itself.

5.4.4.5 Use more than one configuration

In the administration area, you make more than one configuration with unique names, for example `default`, `contact-form`, and `registration`. In the form, you give the name of the applicable configuration:

```
class ContactForm(CaptchaFormMixin, forms.Form):
    name = forms.CharField()
    captcha = CaptchaField("contact-form")

class RegistrationForm(CaptchaFormMixin, forms.Form):
    username = forms.CharField()
    captcha = CaptchaField("registration")

class CommentForm(CaptchaFormMixin, forms.Form):
    comment = forms.CharField()
    captcha = CaptchaField()  # Uses default configuration
```

5.4.4.6 Fields of a configuration

These fields are available:

Field	Default	Effect
`name`	necessary	Unique name of the configuration.
`description`	—	Free note about the use.

Field	Default	Effect
`sitekey`	necessary	Public sitekey for the widget.
`secret`	necessary	Secret key for the verification on the server.
`is_default`	`False`	Uses the configuration when the form gives no name.
`is_active`	`True`	Releases the configuration for use.

These fields change the addresses of the services:

Field	Default	Effect
`verify_url`	`https://api.eu-captcha.eu/v1/verify/`	Address of the endpoint for the verification.
`widget_url`	`https://cdn.eu-captcha.eu/verify.js`	Address of the script for the widget.

These fields set the time limits in seconds:

Field	Default	Effect
`connect_timeout`	`3`	Time limit for the connection setup.
`read_timeout`	`10`	Time limit to read the response.
`write_timeout`	`10`	Time limit to send the request.
`pool_timeout`	`3`	Time limit for an available connection from the connection pool.

These fields control the behaviour for errors:

Field	Default	Effect
`default_result_on_error`	`True`	For a network error, gives a positive result and permits the transmission.
	`True`	Suppresses exceptions and gives the result instead.

Field	Default	Effect
`suppress_exceptions`		

5.4.4.7 Configuration with the settings

Instead of the database, you use the `settings.py` file:

```
# settings.py
EUCAPTCHA_SITEKEY = "EUCAPTCHA_SITE_KEY"
EUCAPTCHA_SECRET = "EUCAPTCHA_SECRET_KEY"

# Optional
EUCAPTCHA_VERIFY_URL = "https://api.eu-captcha.eu/v1/verify/"
EUCAPTCHA_WIDGET_URL = "https://cdn.eu-captcha.eu/verify.js"
EUCAPTCHA_CONNECT_TIMEOUT = 3
EUCAPTCHA_READ_TIMEOUT = 10
EUCAPTCHA_WRITE_TIMEOUT = 10
EUCAPTCHA_POOL_TIMEOUT = 3
EUCAPTCHA_DEFAULT_RESULT_ON_ERROR = True
EUCAPTCHA_SUPPRESS_EXCEPTIONS = True
```

Configurations from the database have precedence over the settings.

5.4.4.8 Interface

Element	Use
`CaptchaField(config_name=None, **kwargs)`	Form field that renders the widget and verifies the response. Without `config_name`, the default configuration applies.
`CaptchaFormMixin`	Gives the request to the fields, so that the IP address can be verified.
`CaptchaWidget`	Renders the challenge. Usually, you do not use this element directly.

Element	Use
`validate_captcha(...)`	Verifies a token immediately. For an error, the function causes `django.core.exceptions.ValidationError`.

```
from myra_eucaptcha_django import validate_captcha

validate_captcha(
    token="captcha-response-token",
    remote_addr="client-ip",  # Optional
    config_name="my-config",  # Optional
)
```

5.4.4.9 Full example

See **HTML and Django**.

5.4.5 Ruby

The `eu_captcha` gem verifies the token on the server. It has no other dependencies and uses `net/http` from the standard library of Ruby.

5.4.5.1 Requirements

The following requirements must be met:

Requirement	Value
Ruby	From version 2.7
Credentials	Public sitekey and secret from the **Details** view

5.4.5.2 Install the gem

Install the gem:

```
gem install eu_captcha
```

As an alternative, add this line to the `Gemfile` file:

```
gem 'eu_captcha'
```

5.4.5.3 Embed the widget

Add the script link to each page that has a form to protect:

```
<script src="https://cdn.eu-captcha.eu/verify.js" async defer></script>
```

Add the widget to the form:

```
<div class="eu-captcha" data-sitekey="EUCAPTCHA_SITE_KEY"></div>
```

In a single-page application with React, Vue, or Angular, embed the widget with the applicable npm package. Then the gem does only the verification on the server.

5.4.5.4 Verify the token

Verify the transmitted token on the server:

```
require 'eu_captcha'

captcha = EuCaptcha::Client.new(
  sitekey: ENV['EUCAPTCHA_SITE_KEY'],
  secret:  ENV['EUCAPTCHA_SECRET_KEY']
)

result = captcha.validate(
  token:       params['eu-captcha-response'],
  remote_addr: request.ip
)

render_error unless result.success?
```

5.4.5.5 Options

You give these options to the constructor as keyword arguments:

Option	Type	Default	Effect
`sitekey`	String	—	Public sitekey. The value is necessary.
`secret`	String	—	Secret key. The value is necessary and must not occur in the browser.
`fail_default`	Boolean	`true`	Return value for the network condition and the token condition when the API is not available. `true` permits the transmission, `false` rejects it.
`check_cdn_headers`	Boolean	`true`	Gets the IP address of the visitor from the `HTTP_CLIENT_IP`, `HTTP_X_FORWARDED_FOR`, and `HTTP_X_REAL_IP` headers of the Rack environment before `REMOTE_ADDR` is used.

Option	Type	Default	Effect
`verify_url`	String	Address of the production environment	Overwrites the address of the `/verify` endpoint. Use the option for tests.
`credentials_url`	String	Address of the production environment	Overwrites the address of the `/verify-credentials` endpoint.

5.4.5.6 The result object

`validate` gives an `EuCaptcha::Result` object with these methods:

Method	Return value
`success?`	`true` when the API was available and the token is valid.
`success_network?`	`true` when the call of the API completed without a network error or a transmission error.
`success_token?`	`true` when the API reported the transmitted token as valid.
`train`	`true`, `false`, or `nil`.

The separate query tells a failed challenge from a malfunction of the API:

```
result = captcha.validate(token: params['eu-captcha-response'], remote_addr:
request.ip)

unless result.success_network?
  # Could not reach the API – consider logging or alerting
end

unless result.success_token?
  # Token was rejected – the submission is likely automated
end
```

> `train` gives `true` when the API did not do the applicable verification and forced a positive result. This occurs when the sitekey does not exist, when the secret does not agree with it, or when the protection of the sitekey is off. `success?` already reads this flag and gives `false` in this condition. For a network error, `train` gives the `nil` value.

5.4.5.7 Verify the credentials

With `verify_credentials`, you verify the sitekey and the secret without a token from the browser, for example at the start of the application:

```
captcha = EuCaptcha::Client.new(
  sitekey: ENV['EUCAPTCHA_SITE_KEY'],
  secret:  ENV['EUCAPTCHA_SECRET_KEY']
)

unless captcha.verify_credentials
  # Credentials are invalid or the API is unreachable – log and alert
end
```

For a network error or an API error, the method gives `false` and causes no exception.

See **Verify the sitekey and the secret**.

5.4.5.8 Ruby on Rails

Put the credentials in `config/credentials.yml.enc` and make the client in an initializer file.

The `config/initializers/eu_captcha.rb` file:

```
EU_CAPTCHA = EuCaptcha::Client.new(
  sitekey: Rails.application.credentials.dig(:eucaptcha, :sitekey),
  secret:  Rails.application.credentials.dig(:eucaptcha, :secret)
)
```

Verify the token in the controller:

```
class ContactController < ApplicationController
  def create
    result = EU_CAPTCHA.validate(
```

```
      token:       params['eu-captcha-response'],
      remote_addr: request.ip,
      user_agent:  request.user_agent
    )

    unless result.success?
      render json: { error: 'CAPTCHA verification failed' },
status: :unprocessable_entity
      return
    end

    # process the form...
  end
end
```

`request.ip` obeys the setting of the trusted upstream systems of Rails.

5.4.5.9 Sinatra and Rack

Give the Rack environment to the gem, so that the gem gets the IP address and the identifier of the browser itself:

```
require 'sinatra'
require 'eu_captcha'

CAPTCHA = EuCaptcha::Client.new(
  sitekey: ENV['EUCAPTCHA_SITE_KEY'],
  secret:  ENV['EUCAPTCHA_SECRET_KEY']
)

post '/contact' do
  result = CAPTCHA.validate(
    token:    params['eu-captcha-response'],
    rack_env: env
  )

  halt 422, 'CAPTCHA verification failed' unless result.success?

  # process the form...
end
```

When a Rack environment is given, `validate` obeys the `check_cdn_headers` setting.

5.4.5.10 Full example

See **Svelte and Ruby**.

5.4.6 Java

For Java, there are three clients. All three are made from the same OpenAPI description and call the same endpoint. They differ in the Spring technology that they use.

5.4.6.1 Select the client

Application	Client
Spring Boot 3 with Jakarta EE	`java-webflux-boot3`
Spring Boot 2 or Spring 5, reactive	`java-webflux-boot2`
Spring MVC on the servlet stack, Spring 6	`java-resttemplate`

Client	Requirements	Interface
`java-resttemplate`	Java 17, Spring Framework 6	Synchronous: `VerifyResponse verifyClientToken(VerifyRequest)`. Repeats requests for the 5xx and 429 status codes with an increasing interval.
`java-webflux-boot2`	Java 8, Spring Boot 2 with Spring WebFlux	Reactive: `Mono<VerifyResponse> verifyClientToken(VerifyRequest)`
`java-webflux-boot3`	Java 17, Spring Boot 3 with Jakarta EE 10	Reactive: `Mono<VerifyResponse> verifyClientToken(VerifyRequest)`. Uses annotations from `jakarta.*` throughout.

5.4.6.2 Embed the client

Embed the selected client with Maven. The example shows `java-resttemplate`:

```
<dependency>
  <groupId>com.myrasec</groupId>
  <artifactId>eu-captcha-java-resttemplate</artifactId>
  <version>1.0.0</version>
```

```
  <scope>compile</scope>
</dependency>
```

With Gradle, the entry is as follows:

```
implementation "com.myrasec:eu-captcha-java-resttemplate:1.0.0"
```

You make the client from the sources with this command:

```
mvn clean install
```

5.4.6.3 Verify the token

This example shows the synchronous verification with `java-resttemplate` :

```
import com.myrasec.client.ApiClient;
import com.myrasec.client.api.EuCaptchaApi;
import com.myrasec.client.model.VerifyRequest;
import com.myrasec.client.model.VerifyResponse;
import org.springframework.web.client.RestClientException;

ApiClient client = new ApiClient();
// client.setMaxAttemptsForRetry(3); // optional: retry up to 3 times on 5xx /
429

EuCaptchaApi api = new EuCaptchaApi(client);

VerifyRequest request = new VerifyRequest()
        .sitekey("YOUR_SITEKEY")
        .secret("YOUR_SECRET")
        .clientIp(clientIp)              // real end-user IP – see below
        .clientToken(euCaptchaToken)     // value of the "eu-captcha-response"
POST field
        .clientUserAgent(userAgent);     // value of the User-Agent request
header

try {
    VerifyResponse response = api.verifyClientToken(request);

    if (response.isTrainingMode()) {
        // Training mode: real validation was not performed – always allow.
        // Occurs when the sitekey does not exist, the secret is wrong,
        // or the sitekey is configured with train=true.
        allowAccess();
    } else if (Boolean.TRUE.equals(response.getSuccess())) {
```

```
            allowAccess();
        } else {
            denyAccess();
        }
    } catch (RestClientException e) {
        // Network error or unexpected HTTP status.
        // Decide your fallback policy: allow or deny.
        log.error("EU CAPTCHA verification failed: {}", e.getMessage());
    }
```

5.4.6.4 Reactive verification

The two clients for WebFlux give a `Mono<VerifyResponse>`. Put it in your processing chain instead of a `.block()` call:

```
import com.myrasec.client.ApiClient;
import com.myrasec.client.api.EuCaptchaApi;
import com.myrasec.client.model.VerifyRequest;
import com.myrasec.client.model.VerifyResponse;
import
org.springframework.web.reactive.function.client.WebClientResponseException;
import reactor.core.publisher.Mono;

ApiClient client = new ApiClient(); // thread-safe; share a single instance

EuCaptchaApi api = new EuCaptchaApi(client);

VerifyRequest request = new VerifyRequest()
        .sitekey("YOUR_SITEKEY")
        .secret("YOUR_SECRET")
        .clientIp(clientIp)
        .clientToken(euCaptchaToken)
        .clientUserAgent(userAgent);

Mono<VerifyResponse> result = api.verifyClientToken(request)
        .map(response -> {
            if (response.isTrainingMode()) {
                allowAccess();
            } else if (Boolean.TRUE.equals(response.getSuccess())) {
                allowAccess();
            } else {
                denyAccess();
            }
            return response;
        })
        .onErrorResume(WebClientResponseException.class, e -> {
            log.error("EU CAPTCHA verification failed: {}", e.getMessage());
            return Mono.empty();
        });
```

5.4.6.5 Get the IP address of the visitor

Always give the true IP address of the visitor, not the address of your upstream system.

In an application on the servlet stack:

```
// Without a proxy
String clientIp = httpServletRequest.getRemoteAddr();

// Behind a reverse proxy or CDN (use the header your provider documents)
String forwarded = httpServletRequest.getHeader("X-Forwarded-For");
if (forwarded != null && !forwarded.isBlank()) {
    // X-Forwarded-For is a comma-separated list; the leftmost entry is the
client IP
    clientIp = forwarded.split(",")[0].trim();
}
```

In an application with WebFlux:

```
// In a Spring WebFlux handler (ServerWebExchange)
String clientIp =
exchange.getRequest().getRemoteAddress().getAddress().getHostAddress();

// Behind a reverse proxy or CDN (use the header your provider documents)
String forwarded = exchange.getRequest().getHeaders().getFirst("X-Forwarded-
For");
if (forwarded != null && !forwarded.isBlank()) {
    clientIp = forwarded.split(",")[0].trim();
}
```

The `X-Forwarded-For` header contains a list with commas between the entries. The first entry is the IP address of the visitor.

5.4.6.6 Fields of the request

Field	Type	Necessary	Content
`sitekey`	String	yes	Public sitekey from the **Details** view.
`secret`	String	yes	Secret key for the sitekey.

Field	Type	Necessary	Content
`clientIp`	String	yes	IPv4 or IPv6 address of the visitor.
`clientToken`	String	yes	Token from `verify.js`. The value can be empty.
`clientUserAgent`	String	yes	Value of the `User-Agent` header from the request of the visitor.

5.4.6.7 Fields of the response

Field	Type	Content
`success`	Boolean	`true` when the challenge passed.
`train`	Boolean	`true` when the training mode was on. The `isTrainingMode()` method reads this field.

The endpoint is `POST /verify` at `https://api.eu-captcha.eu/v1`. An authorization is not necessary. The request identifies itself with the `secret` field.

5.4.6.8 Security

Do not set `ApiClient.setDebugging(true)` in production. In debug mode, the client writes the full JSON body to the log. `VerifyRequest.toString()` hides the secret. The transmitted body does not hide it.

Put the secret in an environment variable or in a key management system, never in the source code.

`ApiClient` is thread-safe. Make one instance and use it in the full application instead of a new instance for each request.

5.4.6.9 Full example

See Angular and Java.

5.4.7 Client from the specification

For languages without a ready-made package, generate the client yourself from the specification of the verification API. The basis is the `openapi.yaml` file according to OpenAPI 3.1.0. It describes the `POST /verify` and `POST /verify-credentials` endpoints of the `https://api.eu-captcha.eu/v1` server.

For PHP, Symfony and Laravel, Python, Django, Ruby, and Java, ready-made packages are available. Generate your own client only for languages that are not listed there. See **Integration**.

5.4.7.1 Requirements

The following requirements must be met:

Requirement	Value
Specification	The `openapi.yaml` file is available
Tool	`openapi-generator-cli` ; Myra generates its own clients with version 7.18.0

5.4.7.2 Generating a client

To generate a client, run the following command:

```
openapi-generator-cli generate -i openapi.yaml -g <generator> -o <output
directory>
```

Replace `<generator>` with the name of the generator you want. The following examples show the calls used by Myra:

Destination	Command
Spring	`openapi-generator-cli generate -i openapi.yaml -g spring -o spring`

Destination	Command
Node.js with TypeScript	`openapi-generator-cli generate -i openapi.yaml -g typescript-node -o typescript-node`

For other languages — for example Go, C#, or Rust — enter the name of the applicable generator. The list of the generators is part of the tool, not of the specification.

5.4.7.3 Aligning the naming

So that the generated packages match the packages from Myra, pass the namespace and the package name as additional properties:

```
openapi-generator-cli generate -i openapi.yaml -g python \
  --additional-properties=invokerPackage=Myrasec\
\EuCaptcha,packageName=eucaptcha,library=httpx \
  -o python
```

For Java, Myra uses the following properties:

Variant	Additional properties
`java-resttemplate`	`packageName=eucaptcha,groupId=com.myrasec,invokerPackage=com.myrasec.client,apiPackage=com.myrasec.client.api,modelPackage=com.myrasec.client.model,library=resttemplate`
`java-webclient`	`packageName=eucaptcha,groupId=com.myrasec,invokerPackage=com.myrasec.client,apiPackage=com.myrasec.client.api,modelPackage=com.myrasec.client.model,library=webclient`
`java-springboot3`	`packageName=eucaptcha,groupId=com.myrasec,invokerPackage=com.myrasec.client,apiPackage=com.myrasec.client.api,modelPackage=com.myrasec.client.model,library=webclient,useSpringBoot3=true,useJakartaEe=true`

All three variants use the `java` generator. See **Java**.

5.4.7.4 Using the generated client

For the `POST /verify` endpoint, the generator makes the `verifyClientToken` operation. The names in the destination language follow the rules of the applicable generator, but the structure of the request and of the answer stays the same:

Field of the request	Description
`sitekey`	Public sitekey from the **Details** view.
`secret`	Secret belonging to the sitekey.
`client_ip`	Actual IP address of the visitor.
`client_token`	Token from `verify.js` .
`client_user_agent`	`User-Agent` header of the request.

See **Verify a client token**.

In the answer, always also read the `train` field. If it has the `true` value, no verification occurred. See **Configuring a sitekey**.

5.4.7.5 Embedding the widget

The generated client does the server-side verification. Add the widget in the page with the form to be protected unchanged. See **HTML and JavaScript**.

5.5 Mobile apps

5.5.1 Android

The SDK for Android embeds the widget in your app and reports the condition of the challenge with a listener.

5.5.1.1 Requirements

The following requirements must be met:

Requirement	Value
Android	From version 4.1 Jelly Bean, API level 16
Language	Kotlin or Java. The SDK is written in Kotlin.
Credentials	Public sitekey from the **Details** view

5.5.1.2 Embed the SDK

Add these entries to the `build.gradle` file:

```
repositories {
    mavenCentral()
}

dependencies {
    implementation "eu.eucaptcha.android:eu-captcha-android:1.0.0"
    // Or for `build.gradle.kts`
    // implementation("eu.eucaptcha.android:eu-captcha-android:1.0.0")
}
```

5.5.1.3 Make the widget

Make the SDK and the widget in your activity:

```
import eu.eucaptcha.android.sdk.*

const val EU_CAPTCHA_SITEKEY = "EUCAPTCHA_SITE_KEY"
```

```
class MainActivity : ComponentActivity() {
    private val sdk by lazy {
        EuCaptchaSDK(context = this)
    }

    private val widget by lazy {
        sdk.createWidget(sitekey = EU_CAPTCHA_SITEKEY)
    }
}
```

5.5.1.4 Show the widget

Show the widget in your interface and read the condition:

```
val captchaResponse = remember { mutableStateOf("") }
var buttonEnabled by remember { mutableStateOf(false) }

widget.setOnStateChangeListener { event ->
    captchaResponse.value = event.response
    when (event.state) {
        "completed" -> buttonEnabled = true
        "expired"   -> buttonEnabled = false
        "error"     -> buttonEnabled = true
    }
}

AndroidView(factory = { _ -> widget.view })

Button(onClick = { widget.reset() }, enabled = buttonEnabled) {
    Text("Submit")
}
```

The listener reports these conditions:

Condition	Meaning
`completed`	The challenge passed. The `response` field contains the token.
`expired`	The token expired.
`error`	An error occurred during the verification.

The `widget.reset()` call starts a new challenge.

5.5.1.5 Verify the token

Send the token to your server and verify it there. See **Verify a client token**.

5.5.2 iOS

The SDK for iOS embeds the widget in a `WKWebView` and supplies an interface in Swift.

5.5.2.1 Requirements

The following requirements must be met:

Requirement	Value
iOS	From version 13.0
Swift	From version 5.5
Xcode	From version 13
Credentials	Public sitekey from the **Details** view

5.5.2.2 Embed the SDK

With the Swift Package Manager, add the dependency to `Package.swift`:

```
.package(url: "https://github.com/Myra-Security-GmbH/eu-captcha-ios-sdk.git",
from: "1.0.0")
```

In Xcode, you can also add the package with **File > Add Package Dependencies**.

With CocoaPods, add this line to the `Podfile` file:

```
pod 'EuCaptcha', '~> 1.0'
```

Then run `pod install`.

5.5.2.3 Make the widget

```
import EuCaptcha

let widget = EuCaptchaSDK.createWidget(
    sitekey: "EUCAPTCHA_SITE_KEY",
    theme: .auto,          // .light, .dark, or .auto
    language: "en"         // BCP 47 language code; defaults to system
```

```
    language
)
```

The `theme` value controls the appearance with `.light`, `.dark`, and `.auto`. The `language` value takes a language code as specified in BCP 47. Without a value, the language of the device applies.

5.5.2.4 Read the events

```
widget.onComplete = { event in
    // event.response is the token to verify server-side
    submitTokenToServer(event.response)
}

widget.onExpire = { _ in
    // The token has expired; the widget resets automatically
    disableLoginButton()
}

widget.onError = { _ in
    // An error occurred during the proof-of-work
    showErrorMessage()
}

widget.onStateChange = { event in
    print("State:", event.state, "Response:", event.response)
}
```

5.5.2.5 Embed the widget

```
let widgetVC = widget.viewController()
addChild(widgetVC)
view.addSubview(widgetVC.view)
widgetVC.didMove(toParent: self)

widgetVC.view.translatesAutoresizingMaskIntoConstraints = false
NSLayoutConstraint.activate([
    widgetVC.view.leadingAnchor.constraint(equalTo: view.leadingAnchor,
constant: 16),
    widgetVC.view.trailingAnchor.constraint(equalTo: view.trailingAnchor,
constant: -16),
    widgetVC.view.heightAnchor.constraint(equalToConstant:
EuCaptchaSDK.preferredHeight),
])
```

5.5.2.6 Life cycle of the widget

Method	Effect
`start()`	No effect. The widget starts automatically.
`reset()`	Discards the condition and starts a new challenge.
`destroy()`	Releases the resources. Do not use the reference again.
`getResponse()`	Gives the current token.
`getState()`	Gives the current `EuCaptchaWidgetState` condition.

5.5.2.7 Conditions of the widget

Condition	Meaning
`.initial`	The widget is set up, but not active.
`.checking`	The challenge is in operation.
`.completed`	The challenge passed. The `response` field contains the token.
`.expired`	The token expired.
`.error`	An error occurred.
`.destroyed`	The widget was removed.

5.5.2.8 Verify the token

Send the `event.response` value to your server as `client_token`. The server verifies the token:

```
POST https://api.eu-captcha.eu/v1/verify
Content-Type: application/json

{
  "sitekey":           "EUCAPTCHA_SITE_KEY",
  "secret":            "EUCAPTCHA_SECRET_KEY",
  "client_ip":         "<client IP address>",
  "client_token":      "<token from the widget>",
```

```
  "client_user_agent": "<client user-agent>"
}
```

The response is `{ "success": true, "train": false }`.

See **Verify a client token**.

5.5.3 Flutter

For Flutter, the widget embeds the EU CAPTCHA in a WebView in the app and supplies an interface in Dart.

5.5.3.1 Requirements

The following requirements must be met:

Requirement	Value
Flutter	From version 3.0
Dart SDK	From version 3.0
Operating system	Android from API level 19, iOS from version 13
Credentials	Public sitekey from the **Details** view

5.5.3.2 Enter the dependencies

Add these entries to the `pubspec.yaml` file:

```
dependencies:
  flutter_inappwebview: ^6.0.0
  http: ^1.2.0
```

Copy the `lib/eucaptcha.dart` file and the `assets/` directory into your project, and enter the files:

```
flutter:
  assets:
    - assets/euc_check.all.js
    - assets/euc_styles.css
```

5.5.3.3 Set up the platforms

For Android, add this entry to the `android/app/src/main/AndroidManifest.xml` file:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

For iOS, no more settings are necessary.

5.5.3.4 Embed the widget

```
import 'eucaptcha.dart';

EuCaptcha(
  sitekey: 'EUCAPTCHA_SITE_KEY',
  theme: 'auto',   // 'light', 'dark', or 'auto'
  lang: 'en',      // BCP 47 language code
  onComplete: (response) {
    // response is the token — submit to your server as "eu-captcha-response"
  },
  onExpire: (_) { /* CAPTCHA expired */ },
  onError:  (_) { /* error occurred  */ },
)
```

The `theme` value controls the appearance with `light`, `dark`, and `auto`. The `lang` value takes a language code as specified in BCP 47.

5.5.3.5 Reset the widget

Use a `GlobalKey<EuCaptchaState>` to call `reset()`:

```
final _captchaKey = GlobalKey<EuCaptchaState>();

EuCaptcha(key: _captchaKey, sitekey: '...', onComplete: ...)

// later:
_captchaKey.currentState?.reset();
```

5.5.3.6 Verify the token

After the `onComplete` call, send the token to your server. The server verifies the token:

```
POST https://api.eu-captcha.eu/v1/verify
Content-Type: application/json

{
  "sitekey":           "EUCAPTCHA_SITE_KEY",
  "secret":            "EUCAPTCHA_SECRET_KEY",
  "client_ip":         "<client IP>",
  "client_token":      "<token from widget>",
  "client_user_agent": "<user-agent>"
}
```

The response is `{ "success": true, "train": false }`.

See **Verify a client token**.

5.6 Infrastructure

5.6.1 Caddy

The module for Caddy puts a challenge in front of each request. A visitor who passes it gets a signed session cookie and goes to your application without a subsequent interruption until the time limit expires. The module redirects all other requests to the challenge page.

Thus, the protection applies to the full domain, not to individual forms.

5.6.1.1 Sequence

1. The module examines each request for a valid session cookie with an HMAC signature that is bound to the IP address.
2. Without a valid cookie, it redirects the request to the path of the challenge.
3. After the challenge passed, the module verifies the token on the server with the API.
4. If the verification is successful, it sets the signed cookie and redirects to the initial address.

5.6.1.2 Requirements

The following requirements must be met:

Requirement	Value
Caddy	From version 2.9
Tool	`xcaddy` to build with the module
Go	From version 1.22 to build from the sources
Credentials	Public sitekey and secret from the **Details** view

5.6.1.3 Build Caddy with the module

Build Caddy with `xcaddy`:

```
xcaddy build --with github.com/Myra-Security-GmbH/eu-captcha-caddy
```

Build from the sources as follows:

```
git clone https://github.com/Myra-Security-GmbH/eu-captcha-caddy.git
cd eu-captcha-caddy
xcaddy build --with github.com/Myra-Security-GmbH/eu-captcha-caddy=.
```

5.6.1.4 Set up the module

Add these entries to the `Caddyfile` file:

```
example.com {
    eu_captcha {
        sitekey        EUCAPTCHA_SITE_KEY
        secret         EUCAPTCHA_SECRET_KEY
        cookie_secret  <random-32-byte-hex>   # openssl rand -hex 32
        grace_period   1h
    }

    reverse_proxy localhost:3000
}
```

In the configuration as JSON, the entry is as follows:

```
{
  "handler": "eu_captcha",
  "sitekey": "EUCAPTCHA_SITE_KEY",
  "secret": "EUCAPTCHA_SECRET_KEY",
  "cookie_secret": "<random-32-byte-hex>",
  "grace_period": "1h"
}
```

5.6.1.5 Settings

These settings are available:

Setting	Necessary	Default	Effect
`sitekey`	yes	–	Public sitekey.
`secret`	yes	–	Secret key.
`cookie_secret`	yes	–	Key for the HMAC signature of the session cookies. Make it with `openssl rand -hex 32`.
`grace_period`	no	`1h`	Validity period of a passed challenge as a time value in the format of Go.
`cookie_name`	no	`__eucaptcha`	Name of the session cookie.
`challenge_path`	no	`/__captcha__`	Start of the path for the challenge and for the verification.
`verify_url`	no	`https://api.eu-captcha.eu/v1/verify`	Address of the endpoint for the verification of the token.
`credentials_check_url`	no	`https://api.eu-captcha.eu/v1/verify-credentials`	Address of the endpoint for the verification of the credentials. The module calls it at the start.

5.6.1.6 Reserved paths

The module reserves two paths of your domain. Do not use these paths in your application:

Path	Use
`{challenge_path}`	Shows the challenge page.
`{challenge_path}/verify`	Accepts the solved token with POST.

With the default value for `challenge_path`, these are the `/__captcha__` and `/__captcha__/verify` paths.

5.6.2 CrowdSec

The bouncer for CrowdSec is a standalone upstream server that connects the decisions of CrowdSec with the Myra EU CAPTCHA. IP addresses with the `captcha` decision get a challenge, IP addresses with the `ban` decision get the 403 status code. The other traffic goes to your application unchanged.

5.6.2.1 Sequence

1. The bouncer queries the decision stream of CrowdSec at `/v1/decisions/stream` and keeps the decisions in the memory.
2. For each request, it looks up the IP address of the visitor:

Decision	Behaviour
`ban`	The bouncer answers the request with the 403 Forbidden status code.
`captcha`	The bouncer redirects to `/__captcha__`, shows the widget, verifies the token on the server, sets a signed cookie, and redirects back.
no decision	The bouncer sends the request to your application.

5.6.2.2 Requirements

The following requirements must be met:

Requirement	Value
CrowdSec	Agent in operation with an available local API
Go	From version 1.22 to build from the sources
Credentials	Public sitekey and secret from the **Details** view

5.6.2.3 Install the bouncer

Download the built program for your platform from the releases page:

```
# Linux amd64
curl -L https://github.com/Myra-Security-GmbH/eu-captcha-crowdsec/releases/
latest/download/cs-eucaptcha-bouncer-linux-amd64 \
  -o cs-eucaptcha-bouncer
chmod +x cs-eucaptcha-bouncer
```

Programs are available for Linux (amd64, arm64), macOS (amd64, arm64), and Windows (amd64).

Build from the sources as follows:

```
git clone https://github.com/Myra-Security-GmbH/eu-captcha-crowdsec.git
cd eu-captcha-crowdsec
make build
# produces ./cs-eucaptcha-bouncer
```

5.6.2.4 Set up the bouncer

Proceed as follows to set up the bouncer:

- ▶ Register the bouncer with CrowdSec:

```
cscli bouncers add eu-captcha-bouncer
# Note the printed API key — you'll need it in the next step
```

  - ↳ CrowdSec shows an API key.
- ▶ Make the configuration file:

```
cp config.yaml.example config.yaml
```

- ▶ Enter the necessary values:

```
listen_addr: "0.0.0.0:8080"
upstream_url: "http://localhost:3000"   # your application
```

```
crowdsec:
  lapi_url: "http://localhost:8080"
  api_key: "<API key from cscli bouncers add>"
  update_interval: "10s"

eu_captcha:
  sitekey: "EUCAPTCHA_SITE_KEY"    # from app.eu-captcha.eu
  secret:  "EUCAPTCHA_SECRET_KEY"  # from app.eu-captcha.eu

session:
  secret: "<random hex string>"  # openssl rand -hex 32
  ttl: "1h"
```

► Start the bouncer:

```
./cs-eucaptcha-bouncer -config config.yaml
```

➔ The bouncer writes its log as structured JSON to the standard output.

Then point your load balancer or your DNS to the address from `listen_addr`.

5.6.2.5 Settings

These settings are available:

Setting	Default	Effect
`listen_addr`	`0.0.0.0:8080`	Address at which the bouncer accepts requests.
`upstream_url`	necessary	Address of your application.
`crowdsec.lapi_url`	`http://localhost:8080`	Address of the local API of CrowdSec.
`crowdsec.api_key`	necessary	API key from `cscli bouncers add`.
`crowdsec.update_interval`	`10s`	Interval between two queries of the decision stream.
`eu_captcha.sitekey`	necessary	Public sitekey.

Setting	Default	Effect
`eu_captcha.secret`	necessary	Secret key.
`eu_captcha.verify_url`	`https://api.eu-captcha.eu/v1/verify`	Address of the endpoint for the verification.
`session.secret`	necessary	Key for the HMAC signature. Make it with `openssl rand -hex 32`.
`session.cookie_name`	`__eucaptcha_pass`	Name of the session cookie.
`session.ttl`	`1h`	Validity period of a passed challenge.
`trusted_proxies`	empty	CIDR ranges whose `X-Forwarded-For` header the bouncer reads.

5.6.2.6 Operation with systemd

```
[Unit]
Description=EU Captcha CrowdSec Bouncer
After=network.target crowdsec.service

[Service]
ExecStart=/usr/local/bin/cs-eucaptcha-bouncer -config /etc/eu-captcha-bouncer/
config.yaml
Restart=on-failure
User=www-data

[Install]
WantedBy=multi-user.target
```

5.6.2.7 Operation in a container

```
FROM golang:1.22-alpine AS builder
WORKDIR /src
COPY . .
RUN go build -o cs-eucaptcha-bouncer ./cmd/cs-eucaptcha-bouncer

FROM alpine:3.19
```

```
COPY --from=builder /src/cs-eucaptcha-bouncer /usr/local/bin/
ENTRYPOINT ["cs-eucaptcha-bouncer", "-config", "/etc/bouncer/config.yaml"]
```

5.6.2.8 Reserved paths

The bouncer reserves two paths of the protected domain. Do not use these paths in your application:

Path	Use
`/__captcha__`	Shows the challenge page.
`/__captcha__/verify`	Accepts the solved token.

5.6.3 Traefik

Traefik embeds the Myra EU CAPTCHA through the **CrowdSec Bouncer Traefik Plugin**. The plugin does not come from Myra Security. It queries the decisions of CrowdSec and, for the `captcha` decision, shows a challenge of the Myra EU CAPTCHA.

5.6.3.1 Sequence

The plugin works as a middleware upstream of your service:

Decision	Behaviour
`ban`	The plugin refuses the request.
`captcha`	The plugin shows the `captcha.html` template with the widget, verifies the solved token, and lets the IP address through for the duration of the grace period.
no decision	The plugin sends the request to your service unchanged.

In contrast to **Caddy**, CrowdSec therefore decides who sees a challenge. To protect all requests of a domain, use the separate bouncer. See **CrowdSec**.

5.6.3.2 Requirements

The following requirements must be met:

Requirement	Value
Traefik	Version that supports plugins
CrowdSec	Agent in operation with an available local API
Credentials	Public sitekey and secret from the **Details** view

5.6.3.3 Registering the plugin

Proceed as follows to make the plugin known:

- ▶ Add the following to the static configuration `traefik.yml`:

```
experimental:
  plugins:
    bouncer:
      moduleName: github.com/maxlerebourg/crowdsec-bouncer-traefik-plugin
      version: v1.3.5
```

- ▶ Restart Traefik.
- ➔ Traefik downloads the plugin at startup.

5.6.3.4 Preparing the template and the bouncer

Proceed as follows to provide the template of the challenge and to register the bouncer:

- ▶ Download the template:

```
curl -o captcha.html \
  https://raw.githubusercontent.com/maxlerebourg/crowdsec-bouncer-traefik-
plugin/main/captcha.html
```

- ▶ Embed the file as a volume into the container of Traefik.
- ▶ Register the bouncer at CrowdSec:

```
docker exec crowdsec cscli bouncers add traefik-bouncer
```

  - ↳ The command outputs an API key. Make a note of it.

5.6.3.5 Setting up the middleware

To set up the middleware, add the following to the dynamic configuration:

```
http:
  middlewares:
    crowdsec:
```

```
    plugin:
      bouncer:
        crowdsecLapiKey: "<API key from cscli bouncers add>"
        crowdsecLapiHost: "crowdsec:8080"
        captchaProvider: eucaptcha
        captchaSiteKey: "EUCAPTCHA_SITE_KEY"
        captchaSecretKey: "EUCAPTCHA_SECRET_KEY"
        captchaGracePeriodSeconds: 1800
        captchaHTMLFilePath: /captcha.html
```

The following keys apply to the Myra EU CAPTCHA:

Key	Description
`captchaProvider`	Provider of the challenge. For the Myra EU CAPTCHA, the value is `eucaptcha` .
`captchaSiteKey`	Public sitekey.
`captchaSecretKey`	Secret of the sitekey.
`captchaGracePeriodSeconds`	Duration in seconds for which a passed challenge is valid.
`captchaHTMLFilePath`	Path to the `captcha.html` template in the container.

- ▶ Assign the middleware to the routes that are to be protected.

The secret belongs exclusively in the configuration on the server. Store it in an environment variable or in a service for the management of secrets.

The installation wizard in the **Integration** tab of the sitekey shows the same steps. See **Integration**.

5.7 Examples

5.7.1 React and PHP

This example protects the contact form of an application that already exists. The frontend is a React application with Vite, the backend is a PHP server. Both are already in operation. For the Myra EU CAPTCHA, you change only the files that an arrow marks below.

5.7.1.1 Requirements

The following requirements must be met:

Requirement	Value
Node.js with npm	installed
PHP	From version 8.0
Composer	installed
Sitekey	Public sitekey and secret from the **Details** view

The public sitekey belongs in the frontend, the secret only in the backend. Each visitor can read a secret that is in the delivery package of the frontend.

5.7.1.2 Project structure

These files change. All other parts of the project stay unchanged:

```
my-project/
├── frontend/                    # existing React application (Vite)
│   ├── package.json             # ← Step 1: enter the @myrasec/eu-captcha
package
│   ├── vite.config.ts
│   ├── index.html
│   └── src/
│       ├── main.tsx
│       ├── App.tsx
│       └── ContactForm.tsx      # ← Steps 1 and 2: embed the widget, send
the token
```

```
└── backend/                        # existing PHP server
    ├── composer.json               # ← Step 3: enter the myra-security-gmbh/eu-
captcha package
    ├── vendor/
    ├── .env                        # ← enter the sitekey and the secret
    └── public/
        └── index.php               # ← Step 3: put the verification at the
start of the handler
```

5.7.1.3 Step 1: Embed the widget

The form is in the `frontend/src/ContactForm.tsx` file.

Install the package:

```
npm i @myrasec/eu-captcha
```

Put the `EuCaptcha` component in the form and accept the token with the `onComplete` callback:

```
import { useState } from "react";
import { EuCaptcha, isEuCaptchaDone } from "@myrasec/eu-captcha";

const captchaSitekey = "EUCAPTCHA_SITE_KEY";

export function ContactForm() {
  const [token, setToken] = useState("");

  // handleSubmit follows in step 2

  return (
    <form onSubmit={handleSubmit}>
      <input name="email" type="email" required />
      <textarea name="message" required />

      {/* onComplete gives the token as soon as the challenge is complete */}
      <EuCaptcha sitekey={captchaSitekey} onComplete={setToken} />

      <button type="submit">Submit</button>
    </form>
  );
}
```

The component loads the `verify.js` script, makes the hidden iframe, and starts the challenge automatically.

5.7.1.4 Step 2: Send the token

Before the transmission, use `isEuCaptchaDone()` to examine if the challenge is complete. Then send the token in the `eu-captcha-response` field of the JSON body:

```
async function handleSubmit(e: React.FormEvent<HTMLFormElement>):
Promise<void> {
  e.preventDefault();
  if (!isEuCaptchaDone()) {
    // challenge not yet complete
    return;
  }
  const fields = Object.fromEntries(new FormData(e.currentTarget));
  const res = await fetch("/api/submit", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ ...fields, "eu-captcha-response": token }),
  });
  if (res.ok) {
    // success
  }
}
```

The `EuCaptcha` component has a `memo` wrapper. Thus, it does not render again for each entry in the form. The optimization is effective only for references that stay the same. The `setToken` function from `useState` stays the same. Put your own inline functions for `onComplete`, `onExpired`, or `onError` in a `useCallback` wrapper.

5.7.1.5 Step 3: Verify the token

The verification belongs in the handler on the server that accepts the form data, here `/api/submit` in the `backend/public/index.php` file. It is in the first position, before all processing.

Install the package:

```
composer require myra-security-gmbh/eu-captcha
```

Read the fields from the JSON body and put the verification at the start of the handler:

```php
<?php

require __DIR__ . '/vendor/autoload.php';

use Myrasec\EuCaptcha;

// read the fields from the JSON body
$body = json_decode(file_get_contents('php://input'), true) ?: [];

// verification before all processing
$captcha = new EuCaptcha(
    sitekey: getenv('EUCAPTCHA_SITE_KEY'),
    secret:  getenv('EUCAPTCHA_SECRET_KEY'),
);

if (!$captcha->validate()->success()) {
    http_response_code(400);
    exit('captcha verification failed');
}

// the existing logic continues from here
$email   = $body['email'];
$message = $body['message'];
mail('team@example.com', 'Contact', $message);
echo 'Thank you!';
```

The `validate()` call reads the token automatically from the `eu-captcha-response` field, from a form field and also from a JSON body. The call reads the IP address of the visitor from the headers. Keep the sitekey and the secret in the `.env` file, from which `getenv` reads them.

See **PHP**.

5.7.1.6 Verify the integration

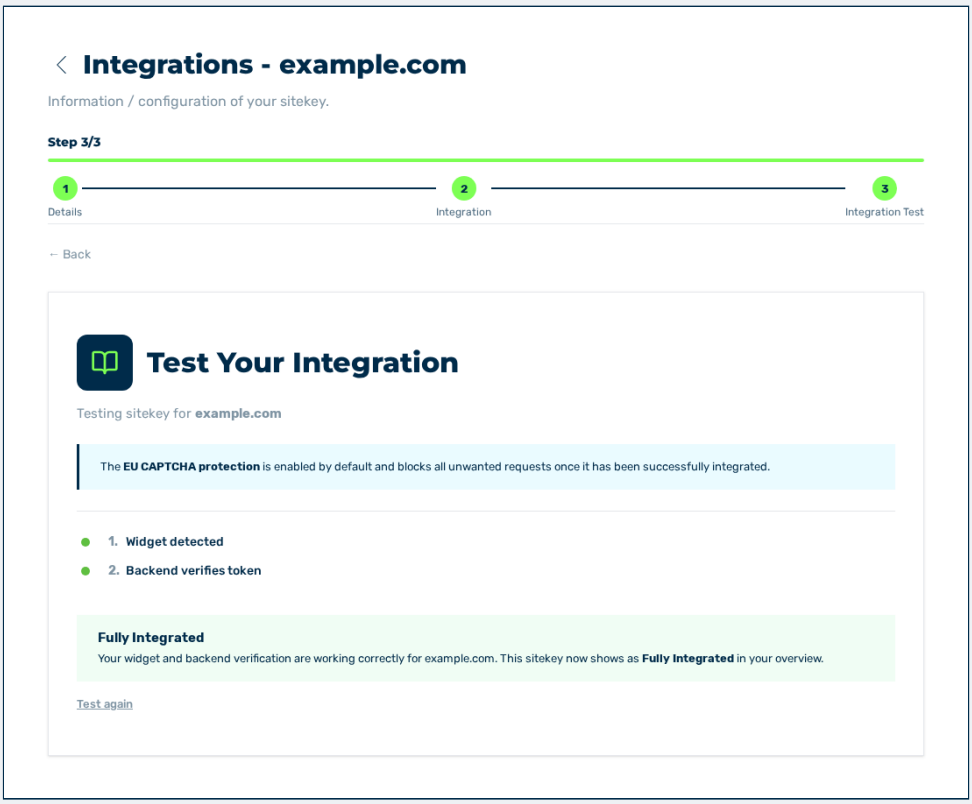


Figure 150: Integration Test view with the Fully Integrated result

At the end, use the **Integration Test** view of the sitekey to examine if the frontend and the backend operate together. Then the sitekey has the **Fully Integrated** status.

See **Testing the integration**.

5.7.2 Vue and Python

This example protects the contact form of an application that already exists. The frontend is a Vue application with Vite, the backend is a Python server with Flask. Both are already in operation. For the Myra EU CAPTCHA, you change only the files that an arrow marks below.

5.7.2.1 Requirements

The following requirements must be met:

Requirement	Value
Node.js with npm	installed
Python	Version 3 with pip
Sitekey	Public sitekey and secret from the **Details** view

The public sitekey belongs in the frontend, the secret only in the backend. Each visitor can read a secret that is in the delivery package of the frontend.

5.7.2.2 Project structure

These files change. All other parts of the project stay unchanged:

```
my-project/
├── frontend/                    # existing Vue application (Vite)
│   ├── package.json             # ← Step 1: enter the @myrasec/eu-captcha-
vue package
│   ├── vite.config.js
│   ├── index.html
│   └── src/
│       ├── main.js
│       ├── App.vue
│       └── ContactForm.vue      # ← Steps 1 and 2: embed the widget, send
the token
└── backend/                     # existing Python server (Flask)
    ├── .venv/
    ├── requirements.txt         # ← Step 3: enter the myra-eucaptcha package
    └── app.py                   # ← Step 3: put the verification at the
start of the route
```

5.7.2.3 Step 1: Embed the widget

The form is in the `frontend/src/ContactForm.vue` file.

Install the package:

```
npm i @myrasec/eu-captcha-vue
```

Put the `EuCaptcha` component in the form and accept the token with the `onComplete` callback:

```
<template>
  <form @submit.prevent="handleSubmit">
    <input name="email" type="email" required />
    <textarea name="message" required></textarea>

    <!-- onComplete gives the token as soon as the challenge is complete -->
    <EuCaptcha :sitekey="captchaSitekey" :onComplete="onComplete" />

    <button type="submit">Submit</button>
  </form>
</template>
```

The component loads the `verify.js` script, makes the hidden iframe, and starts the challenge automatically.

5.7.2.4 Step 2: Send the token

Before the transmission, use `isEuCaptchaDone()` to examine if the challenge is complete. Then send the token in the `eu-captcha-response` field of the JSON body:

```
<script setup>
import { ref } from "vue";
import { EuCaptcha, isEuCaptchaDone } from "@myrasec/eu-captcha-vue";

const captchaSitekey = "EUCAPTCHA_SITE_KEY";
const token = ref("");

function onComplete(t) {
  token.value = t;
}

async function handleSubmit(e) {
  if (!isEuCaptchaDone()) {
    // challenge not yet complete
```

```
    return;
  }
  const fields = Object.fromEntries(new FormData(e.target));
  const res = await fetch("/api/contact", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ ...fields, "eu-captcha-response": token.value }),
  });
  if (res.ok) {
    // success
  }
}
</script>
```

5.7.2.5 Step 3: Verify the token

The verification belongs in the handler on the server that accepts the form data, here the `/api/contact` route in the `backend/app.py` file. It is in the first position, before all processing.

Install the package:

```
pip install myra-eucaptcha
```

Read the fields from the JSON body and put the verification at the start of the route:

```
import os

from flask import Flask, abort, request
from myra_eucaptcha import MyraEuCaptchaClient, MyraEuCaptchaClientConfig

app = Flask(__name__)

client = MyraEuCaptchaClient(config=MyraEuCaptchaClientConfig(
    sitekey=os.environ["EUCAPTCHA_SITE_KEY"],
    secret=os.environ["EUCAPTCHA_SECRET_KEY"],
))


@app.post("/api/contact")
def contact():
    body = request.get_json(silent=True) or {}

    # verification before all processing
    result = client.validate(
        token=body.get("eu-captcha-response", ""),
        remote_addr=request.headers.get("x-real-ip", ""),
```

```
    )
    if not result.success:
        abort(400, "captcha verification failed")

    # the existing logic continues from here
    email = body.get("email")
    message = body.get("message")
    return "Thank you!"
```

The Python client needs the token explicitly. Read it from the `eu-captcha-response` field of the JSON body and give it to `validate()` together with the IP address of the visitor. Keep the sitekey and the secret in environment variables, from which `os.environ` reads them.

In training mode, `result.success` gives the `True` value. The transmission goes through. Training mode is effective for an unknown sitekey, for a wrong secret, and when the protection is off. Thus, always do the test with the true credentials from the customer portal.

See **Python**.

5.7.2.6 Verify the integration

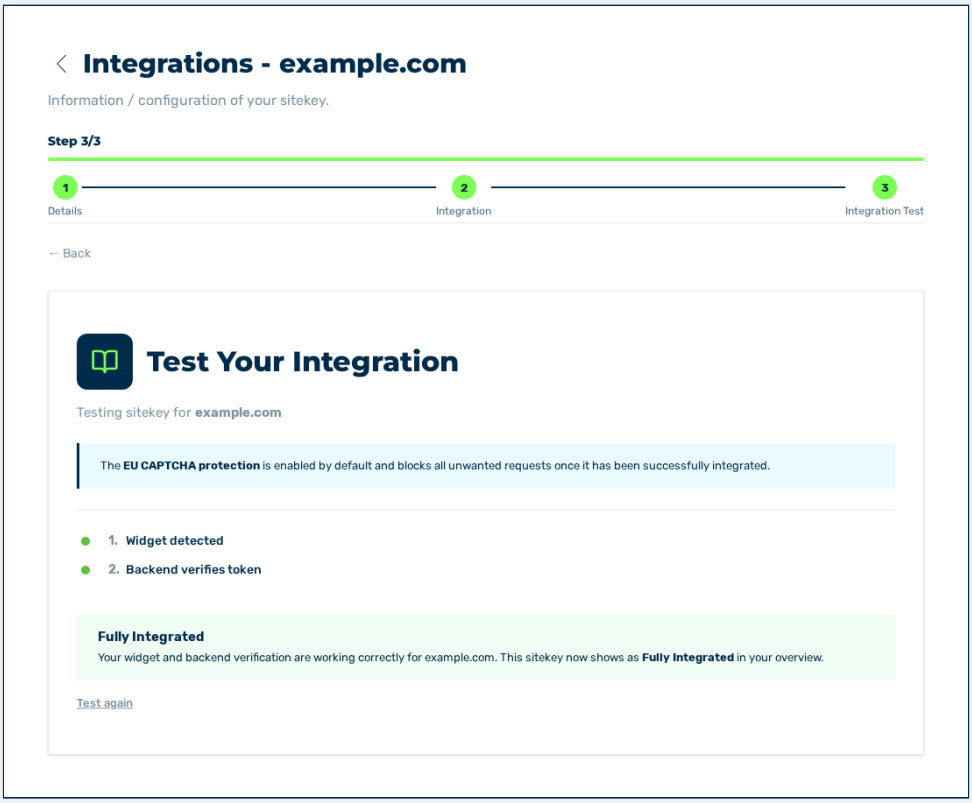


Figure 151: Integration Test view with the Fully Integrated result

At the end, use the **Integration Test** view of the sitekey to examine if the frontend and the backend operate together. Then the sitekey has the **Fully Integrated** status.

See **Testing the integration**.

5.7.3 Angular and Java

This example protects the contact form of an application that already exists. The frontend is an Angular application, the backend is a Java server with Spring Boot. Both are already in operation. For the Myra EU CAPTCHA, you change only the files that an arrow marks below.

5.7.3.1 Requirements

The following requirements must be met:

Requirement	Value
Node.js with npm	installed
JDK	Version 17 with Maven or Gradle
Sitekey	Public sitekey and secret from the **Details** view

The public sitekey belongs in the frontend, the secret only in the backend. Each visitor can read a secret that is in the delivery package of the frontend.

5.7.3.2 Project structure

These files change. All other parts of the project stay unchanged:

```
my-project/
├── frontend/                          # existing Angular application
│   ├── package.json                   # ← Step 1: enter the @myrasec/eu-
captcha-angular20 package
│   ├── angular.json
│   └── src/app/
│       ├── app.component.ts
│       └── contact-form.component.ts  # ← Steps 1 and 2: embed the widget,
send the token
└── backend/                           # existing Java server (Spring Boot)
    ├── pom.xml                        # ← Step 3: enter the dependency
    └── src/main/java/com/example/
        ├── Application.java
        └── ContactController.java     # ← Step 3: put the verification at
the start of the method
```

5.7.3.3 Step 1: Embed the widget

The form is in the `frontend/src/app/contact-form.component.ts` file.

Install the package for your Angular version, here Angular 20:

```
npm i @myrasec/eu-captcha-angular20
```

Put the component in the form and accept the token with the `completed` event:

```
import { Component } from '@angular/core';
import { EuCaptchaComponent, isEuCaptchaDone } from '@myrasec/eu-captcha-
angular20';

@Component({
  standalone: true,
  imports: [EuCaptchaComponent],
  template: `
    <form (submit)="handleSubmit($event)">
      <input name="email" type="email" required />
      <textarea name="message" required></textarea>

      <!-- (completed) gives the token as soon as the challenge is complete --
>
      <eu-captcha sitekey="EUCAPTCHA_SITE_KEY" (completed)="token = $event" />

      <button type="submit">Submit</button>
    </form>
  `,
})
export class ContactFormComponent {
  token = '';
  // handleSubmit follows in step 2
}
```

The `<eu-captcha>` selector loads the `verify.js` script, makes the hidden iframe, and starts the challenge automatically.

An overview of the packages for each Angular version is given in **Angular**.

5.7.3.4 Step 2: Send the token

Before the transmission, use `isEuCaptchaDone()` to examine if the challenge is complete. Then send the token in the `eu-captcha-response` field of the JSON body:

```
async handleSubmit(event: Event): Promise<void> {
  event.preventDefault();
  if (!isEuCaptchaDone()) {
    // challenge not yet complete
    return;
  }
  const fields = Object.fromEntries(new FormData(event.target as
HTMLFormElement));
  const res = await fetch('/api/contact', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ ...fields, 'eu-captcha-response': this.token }),
  });
  if (res.ok) {
    // success
  }
}
```

5.7.3.5 Step 3: Verify the token

The verification belongs in the handler on the server that accepts the form data, here the method for the `/api/contact` route in the `backend/src/main/java/com/example/ContactController.java` file. It is in the first position, before all processing.

Enter the dependency in the `pom.xml` file:

```
<dependency>
  <groupId>com.myrasec</groupId>
  <artifactId>eu-captcha-java-resttemplate</artifactId>
  <version>1.0.0</version>
  <scope>compile</scope>
</dependency>
```

Read the JSON body as a `Map` and put the verification at the start of the method:

```
import com.myrasec.client.ApiClient;
import com.myrasec.client.api.EuCaptchaApi;
import com.myrasec.client.model.VerifyRequest;
import com.myrasec.client.model.VerifyResponse;
```

```
@PostMapping("/api/contact")
public ResponseEntity<String> contact(
        @RequestBody Map<String, String> body,
        HttpServletRequest req) {

    // verification before all processing
    EuCaptchaApi api = new EuCaptchaApi(new ApiClient());

    VerifyRequest request = new VerifyRequest()
            .sitekey(System.getenv("EUCAPTCHA_SITE_KEY"))
            .secret(System.getenv("EUCAPTCHA_SECRET_KEY"))
            .clientIp(req.getRemoteAddr())
            .clientToken(body.get("eu-captcha-response"))
            .clientUserAgent(req.getHeader("User-Agent"));

    VerifyResponse response = api.verifyClientToken(request);
    if (response.isTrainingMode() || !
Boolean.TRUE.equals(response.getSuccess())) {
        return ResponseEntity.badRequest().body("captcha verification
failed");
    }

    // the existing logic continues from here
    String email = body.get("email");
    String message = body.get("message");
    return ResponseEntity.ok("Thank you!");
}
```

The value of the `eu-captcha-response` field goes into the request as `clientToken`. Keep the sitekey and the secret in environment variables, from which `System.getenv` reads them.

Training mode is effective for an unknown sitekey, for a wrong secret, and for a sitekey with the `train` setting. In this condition, the example rejects the transmission. Thus, a wrong configuration does not go through without a verification.

See **Java**.

5.7.3.6 Verify the integration

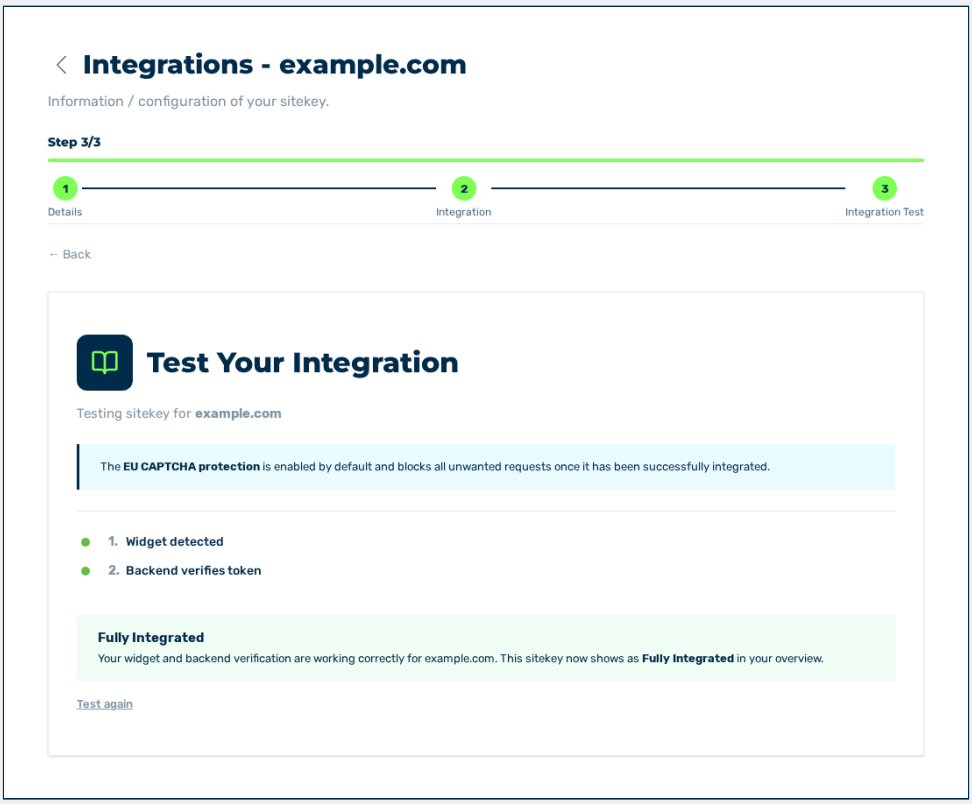


Figure 152: Integration Test view with the Fully Integrated result

At the end, use the **Integration Test** view of the sitekey to examine if the frontend and the backend operate together. Then the sitekey has the **Fully Integrated** status.

See **Testing the integration**.

5.7.4 Svelte and Ruby

This example protects the contact form of an application that already exists. The frontend is a Svelte application with Vite, the backend is a Ruby server with Sinatra. Both are already in operation. For the Myra EU CAPTCHA, you change only the files that an arrow marks below.

5.7.4.1 Requirements

The following requirements must be met:

Requirement	Value
Node.js with npm	installed
Svelte	Version 5
Ruby	From version 2.7
Sitekey	Public sitekey and secret from the **Details** view

The public sitekey belongs in the frontend, the secret only in the backend. Each visitor can read a secret that is in the delivery package of the frontend.

5.7.4.2 Project structure

These files change. All other parts of the project stay unchanged:

```
my-project/
├── frontend/                    # existing Svelte application (Vite)
│   ├── package.json             # ← Step 1: enter the @myrasec/eu-captcha-
svelte package
│   ├── vite.config.js
│   └── src/
│       ├── main.js
│       ├── App.svelte
│       └── ContactForm.svelte   # ← Steps 1 and 2: embed the widget, send
the token
└── backend/                     # existing Ruby server (Sinatra)
    ├── Gemfile                  # ← Step 3: enter the eu_captcha gem
    └── app.rb                   # ← Step 3: put the verification at the
start of the route
```

5.7.4.3 Step 1: Embed the widget

The form is in the `frontend/src/ContactForm.svelte` file.

Install the package:

```
npm i @myrasec/eu-captcha-svelte
```

Put the `EuCaptcha` component in the form and accept the token with the `onComplete` callback:

```
<form onsubmit={handleSubmit}>
  <input name="email" type="email" required />
  <textarea name="message" required></textarea>

  <!-- onComplete gives the token as soon as the challenge is complete -->
  <EuCaptcha sitekey={captchaSitekey} onComplete={(t) => (token = t)} />

  <button type="submit">Submit</button>
</form>
```

The component loads the `verify.js` script, makes the hidden iframe, and starts the challenge automatically.

5.7.4.4 Step 2: Send the token

Before the transmission, use `isEuCaptchaDone()` to examine if the challenge is complete. Then send the token in the `eu-captcha-response` field of the JSON body:

```
<script>
  import { EuCaptcha, isEuCaptchaDone } from "@myrasec/eu-captcha-svelte";

  const captchaSitekey = "EUCAPTCHA_SITE_KEY";
  let token = "";

  async function handleSubmit(e) {
    e.preventDefault();
    if (!isEuCaptchaDone()) {
      // challenge not yet complete
      return;
    }
    const fields = Object.fromEntries(new FormData(e.target));
    const res = await fetch("/api/contact", {
      method: "POST",
      headers: { "Content-Type": "application/json" },
```

```
      body: JSON.stringify({ ...fields, "eu-captcha-response": token }),
    });
    if (res.ok) {
      // success
    }
  }
</script>
```

5.7.4.5 Step 3: Verify the token

The verification belongs in the handler on the server that accepts the form data, here the `/api/contact` route in the `backend/app.rb` file. It is in the first position, before all processing.

Install the gem:

```
gem install eu_captcha
```

Read the fields from the JSON body and put the verification at the start of the route:

```
require 'eu_captcha'
require 'json'

captcha = EuCaptcha::Client.new(
  sitekey: ENV['EUCAPTCHA_SITE_KEY'],
  secret:  ENV['EUCAPTCHA_SECRET_KEY']
)

post '/api/contact' do
  payload = JSON.parse(request.body.read)

  # verification before all processing
  result = captcha.validate(
    token:       payload['eu-captcha-response'],
    remote_addr: request.ip
  )
  halt 400, 'captcha verification failed' unless result.success?

  # the existing logic continues from here
  email   = payload['email']
  message = payload['message']
  'Thank you!'
end
```

The Ruby client needs the token explicitly. Read it from the `eu-captcha-response` field of the JSON body and give it to `validate` together with the IP address of the visitor. Keep the sitekey and the secret in environment variables, from which `ENV` reads them.

In training mode, the `success?` method gives the `false` value. Thus, a wrong configuration does not go through without a verification.

See **Ruby**.

5.7.4.6 Verify the integration

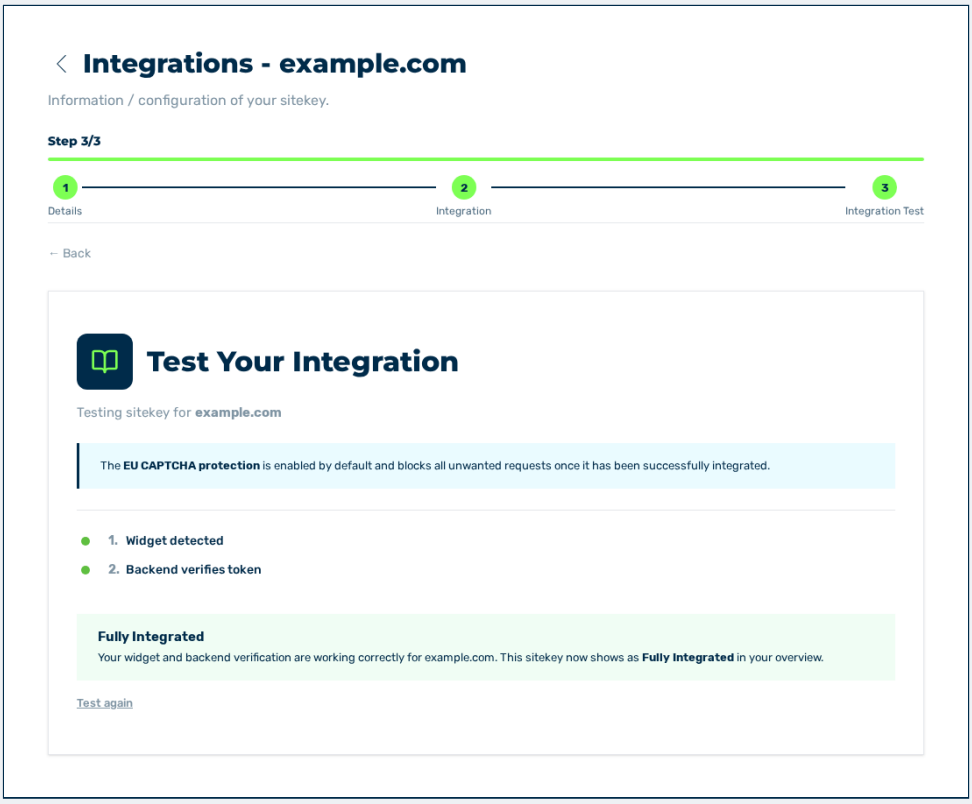


Figure 153: Integration Test view with the Fully Integrated result

At the end, use the **Integration Test** view of the sitekey to examine if the frontend and the backend operate together. Then the sitekey has the **Fully Integrated** status.

See **Testing the integration**.

5.7.5 HTML and Django

This example protects the contact form of an application that already exists and has no frontend framework. The page is delivered as a Django template. The form sends a usual POST. The widget makes the hidden `eu-captcha-response` field itself, which goes with the form.

> In a single-page application, you accept the token with a callback and send it as JSON instead. See **React and PHP**.

5.7.5.1 Requirements

The following requirements must be met:

Requirement	Value
Django project	in operation and delivers the page
Python	Version 3 with pip
Sitekey	Public sitekey and secret from the **Details** view

> The public sitekey belongs in the page, the secret only on the server. Each visitor can read a secret that is in a template.

5.7.5.2 Project structure

These files change. All other parts of the project stay unchanged:

```
my-project/                      # existing Django project
├── manage.py
├── templates/
│   └── contact.html             # ← Steps 1 and 2: embed the script and the
widget
└── server/
    ├── settings.py              # ← Step 3: enter the app and the
credentials
    ├── urls.py                  # ← Step 3: register the /contact/ route
    └── views.py                 # ← Step 3: put the verification at the
start of the view
```

5.7.5.3 Step 1: Embed the widget

The page is in the `templates/contact.html` file.

Load the `verify.js` script from the Myra CDN and put the widget element in the form:

```
<script src="https://cdn.eu-captcha.eu/verify.js" async defer></script>

<form action="/contact/" method="post">
  <input name="email" type="email" required />
  <textarea name="message" required></textarea>

  <div class="eu-captcha" data-sitekey="EUCAPTCHA_SITE_KEY"></div>

  <button type="submit">Submit</button>
</form>
```

The script makes the hidden iframe, starts the challenge automatically, and puts the hidden `eu-captcha-response` field in the form. The field goes with the transmission.

5.7.5.4 Step 2: Release the transmission

If necessary, disable the button and release it only after the challenge. To do this, read the `euCaptchaCompleted` window message. Examine the origin of the message, because each script and each extension in the browser can send messages:

```
<script>
  const CAPTCHA_ORIGIN = "https://cdn.eu-captcha.eu";
  const btn = document.querySelector('button[type="submit"]');
  btn.disabled = true;

  window.addEventListener("message", (msg) => {
    if (msg.origin !== CAPTCHA_ORIGIN) return;
    const data = msg.data ?? {};
    if (data.type === "euCaptchaCompleted") {
      btn.disabled = false;
    }
  });
</script>
```

The disabled button in the browser controls only the operation. The verification of the token on the server is always necessary.

5.7.5.5 Step 3: Verify the token

The verification belongs in the view that accepts the POST of the form, here the `/contact/` route in the `server/views.py` file. It is in the first position, before all processing.

Install the app:

```
pip install myra-eucaptcha-django
```

Enter the app and the credentials in the `settings.py` file:

```
import os

INSTALLED_APPS = [
    # ...
    "myra_eucaptcha_django",
]

EUCAPTCHA_SITEKEY = os.environ["EUCAPTCHA_SITE_KEY"]
EUCAPTCHA_SECRET = os.environ["EUCAPTCHA_SECRET_KEY"]
```

Make the tables of the app:

```
python manage.py migrate
```

Put the verification at the start of the view:

```
from django.core.exceptions import ValidationError
from django.http import HttpResponse, HttpResponseBadRequest
from myra_eucaptcha_django import validate_captcha


def contact(request):
    if request.method == "POST":
        # verification before all processing
        try:
            validate_captcha(
                token=request.POST.get("eu-captcha-response", ""),
                remote_addr=request.META.get("REMOTE_ADDR"),
            )
        except ValidationError:
            return HttpResponseBadRequest("captcha verification failed")
```

```
        # the existing logic continues from here
        email = request.POST["email"]
        message = request.POST["message"]
        return HttpResponse("Thank you!")
```

The widget puts the token in a form field. Thus, it is in `request.POST`. Keep the sitekey and the secret in environment variables. As an alternative, keep the credentials in the administration area of Django at **Captcha Configurations**. The configuration that is set as the default has precedence over the values from `settings.py`.

In training mode, `validate_captcha` causes no `ValidationError`. The transmission goes through. Training mode is effective for an unknown sitekey, for a wrong secret, and when the protection is off. Thus, always do the test with the true credentials from the customer portal.

See **Django**.

5.7.5.6 Verify the integration

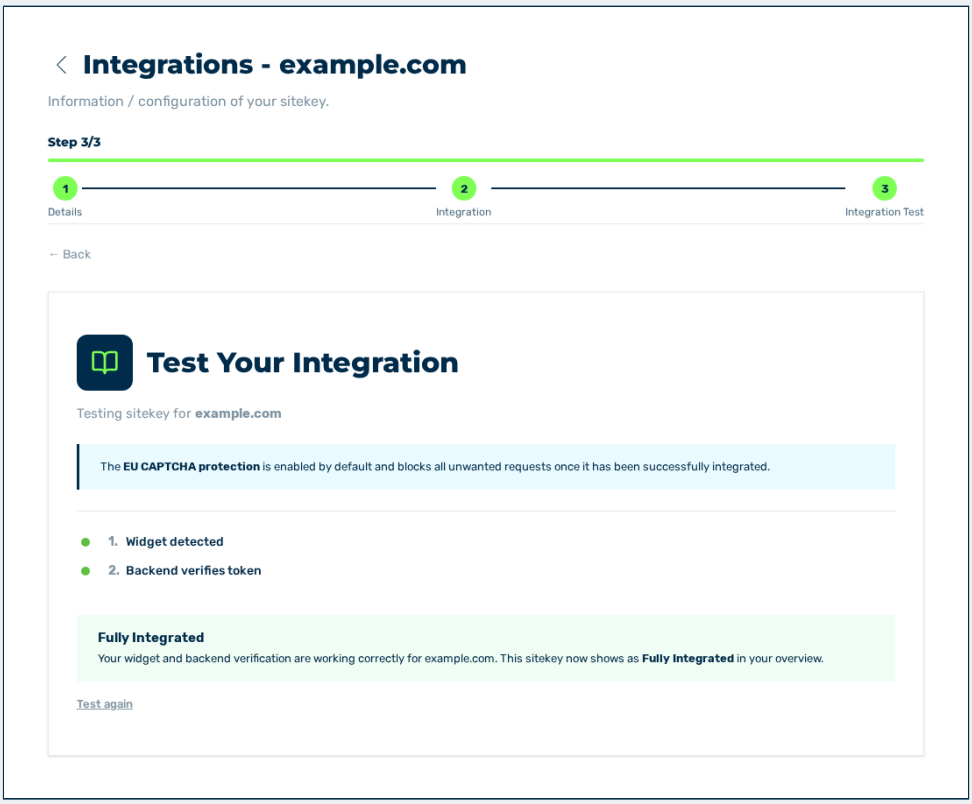


Figure 154: Integration Test view with the Fully Integrated result

At the end, use the **Integration Test** view of the sitekey to examine if the frontend and the backend operate together. Then the sitekey has the **Fully Integrated** status.

See **Testing the integration**.

6. FAQ and troubleshooting

6.1 FAQ

This page answers the questions that occur most frequently before and during an integration. Each answer refers to the section with the full description.

6.1.1 Must the visitor solve a puzzle?

No. The widget collects signals of the browser, and the browser solves a calculation task in the background. The visitor only sees a small Myra logo. There are no puzzles, no image selections, and no entries.

See **How EU CAPTCHA works**.

6.1.2 Does the website need a cookie banner for the widget?

No. Myra EU CAPTCHA operates without HTTP cookies and without permanent browser storage.

See **Introduction**.

6.1.3 How long does a challenge take?

The duration agrees with the difficulty level. There are eight levels from 0 to 7. Level 0 is solved after approximately 600 milliseconds, level 7 after approximately 30 minutes. Legitimate human visitors almost always get level 0 or level 1.

See **A challenge takes an unusually long time**.

6.1.4 Can I set the difficulty level myself?

You set an initial value for each sitekey. The setting contains the levels 0 to 3. The risk detection sets the higher levels itself as soon as suspicious behaviour occurs.

See **Configuring a sitekey**.

6.1.5 How many times can a token be used?

Each token is valid one time. A second verification of the same token responds with the `timeout-or-duplicate` error code. The same code also stands for an expired challenge.

See **Verify a client token**.

6.1.6 Why does the endpoint report a success although there was no verification?

In this case, the response contains `success: true` together with `train: true`. The `train: true` value means that there was no real verification. Each transmission then counts as successful. This occurs with an unknown sitekey, with a secret that does not agree, with a sitekey whose protection is off, and with each other malfunction of the verification.

In production, always examine the two fields. With `train: true`, examine the values of `sitekey` and `secret` immediately.

See **Verify a client token**.

6.1.7 What counts as an assessment?

An assessment counts as soon as the widget starts the examination in the background. The examination in the background and the server-side call count together as **one** assessment. The verification mode does not change this.

See **How EU CAPTCHA works**.

6.1.8 What occurs if I go above the monthly limit?

The monthly limit for assessments is not enforced at this time. If you go above the limit, the widget continues to operate.

See **Plans tab**.

6.1.9 How many sitekeys does a plan contain?

All plans contain an unlimited count of sitekeys. One sitekey for each domain is intended.

See **Plans tab**.

6.1.10 Can one sitekey be used for more than one domain?

One sitekey for each domain is intended. The system does not examine that the domain is unique. Thus two sitekeys for the same domain occur without a notice.

See **Two sitekeys for the same domain**.

6.1.11 What occurs at the end of the trial period?

The widget returns the `TRIAL_EXPIRED` error code and lets the requests through. Visitors are not blocked, and your forms stay available. The protection against bots ends. After you book a plan, the protection becomes active again in 60 seconds after the successful payment.

See **Trial period expired**.

6.1.12 Are my forms blocked if the API has a malfunction?

No. With a malfunction, the widget lets the requests through. The forms stay available to visitors. But the protection is not effective during this time.

See **The API is not reachable**.

6.1.13 Where are the sitekey and the secret given?

The two values are given in the **Details** view of the sitekey.

See **Details view**.

6.1.14 Is the secret permitted in the source code of the page?

No. The secret stays on your server. Call the `/verify` endpoint only from your server. The source code of the page contains only the public sitekey.

See **Verify a client token**.

6.1.15 How do I examine if the integration is complete?

The **Integration Test** view examines the two parts separately: the load of the widget and the server-side verification of the token. The **Fully Integrated** result confirms a complete integration.

See **Testing the integration**.

6.1.16 Which addresses must my Content Security Policy permit?

The `script-src` and `frame-src` directives must contain the `https://cdn.eu-captcha.eu` address, and the `connect-src` directive must contain the `https://api.eu-captcha.eu` address. If one of these permissions is missing, then the browser blocks the widget, although the service is reachable.

See **Content Security Policy**.

6.2 Trial period expired

The widget returns the `TRIAL_EXPIRED` error code. The protection is not effective any more.

6.2.1 Cause

The trial period of your account ended. No plan with costs is active.

6.2.2 Behaviour

In this case, the widget lets the requests through. Visitors are not blocked, and your forms stay available. But the active protection against bots stops.

6.2.3 Solution

- ▶ Book a plan. See **Book a plan**.
- ➔ The protection becomes active again in 60 seconds after the successful payment.

6.3 A challenge takes an unusually long time

A visitor waits an unusually long time before the form can be sent.

6.3.1 Cause

Myra EU CAPTCHA has no failed challenge. A challenge is solved, or its solution takes longer. A long duration indicates traffic with a high risk, or an incorrect classification of a legitimate visitor.

6.3.2 Solution

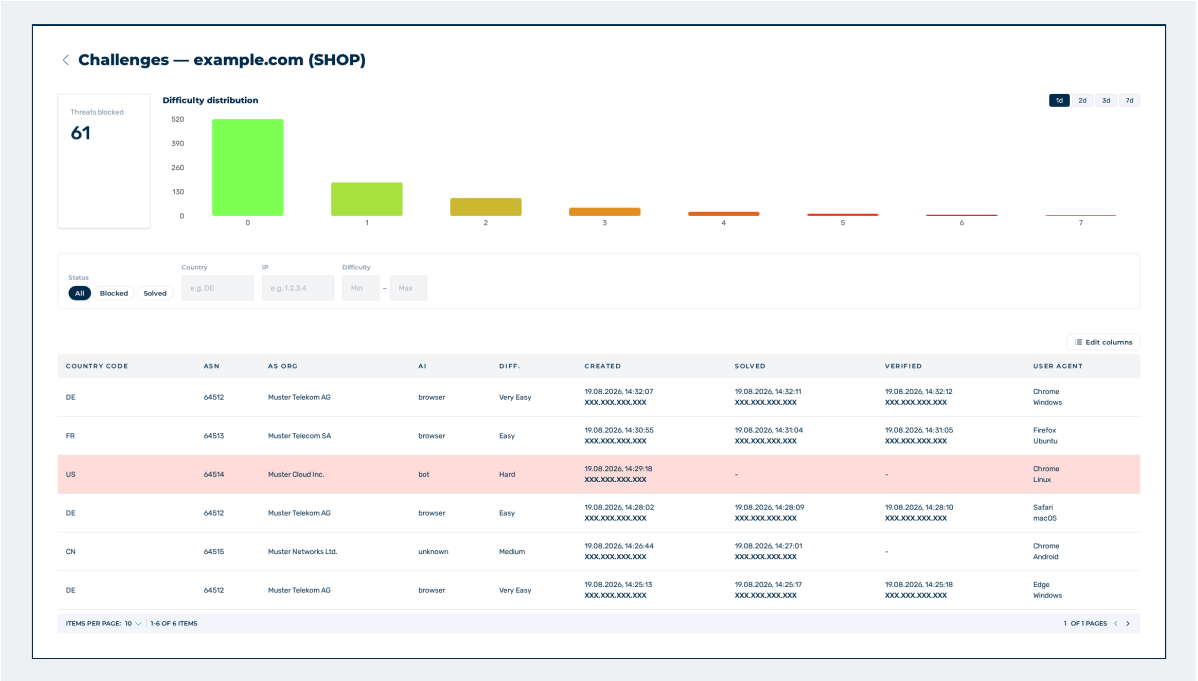


Figure 155: Challenges view

- In the **Challenges** view, examine the difficulties of the applicable requests. See Challenges view.
- Set the **Initial difficulty value per challenge 0-3** value to 0. See Configuring a sitekey.
- If a legitimate visitor continues to be blocked, speak to the support with the data of the session.

6.4 The cause of a block is not visible

The interface and the API do not tell why a request was blocked.

6.4.1 Cause

The causes of a block are not given intentionally. A person who knows the detection logic can bypass it.

6.4.2 Solution

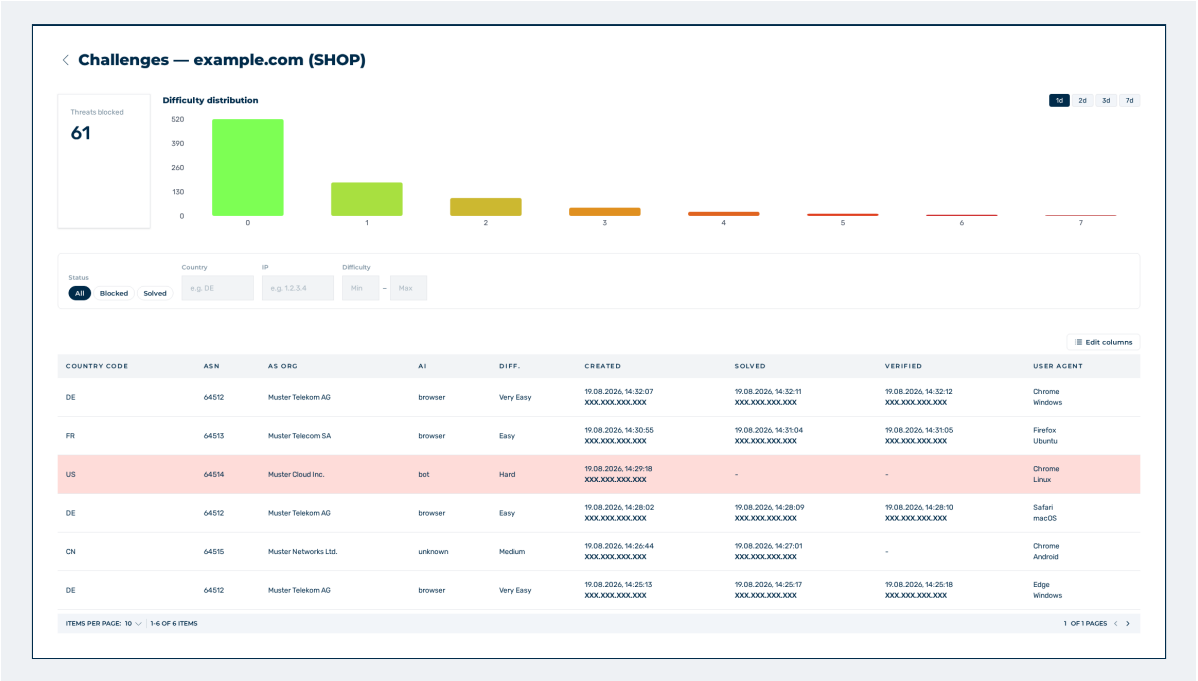


Figure 156: Challenges view

- In the **Challenges** view, examine the characteristics of the blocked requests. Give special attention to the **ASN**, **Country Code**, and **User Agent** columns. See **Challenges view**.
- If legitimate visitors are blocked regularly, speak to the support.

6.5 The login with a passkey fails

During the login or during the creation of a passkey, the portal shows **Invalid attestation object. Unexpected object.**

6.5.1 Cause

A known defect in the processing of the passkey. The defect is recorded and is not corrected yet.

6.5.2 Solution

► Log in with the e-mail address and the password. See **Log in**.

As an alternative, these methods are available:

- login with Google
- login with Apple
- login with GitHub
- login with Okta
- login with SAML and SSO. See **Set up SAML**.

6.6 Two sitekeys for the same domain

More than one sitekey exists for one domain.

6.6.1 Cause

One sitekey for each domain is intended. But the system does not examine that the domain is unique. The defect is recorded. A correction is planned.

6.6.2 Solution

Figure 157: Sitekey List area

- Open the **Sitekey List** area and use the search field to find the domain. See **Sitekey List area**.
- Find which sitekey is embedded in your website. The value is given in the `data-sitekey` attribute of the widget element.
- Delete the other sitekeys. See **Deleting a sitekey**.

> Do not delete a sitekey that is still embedded. If you do, the widget does not get a valid token.

6.7 The API is not reachable

The server-side verification does not reach the endpoint, or the widget does not get a challenge.

6.7.1 Cause

The API of Myra EU CAPTCHA is temporarily not reachable.

6.7.2 Behaviour

In this case, the widget lets the requests through. The forms stay available to visitors. The protection is not effective.

6.7.3 Solution

- Examine if your server reaches the `cdn.eu-captcha.eu` and `api.eu-captcha.eu` addresses.
- Examine the rules of your firewall for outgoing connections.
- If the addresses are reachable and the error continues, speak to the support.

6.8 Widget not detected

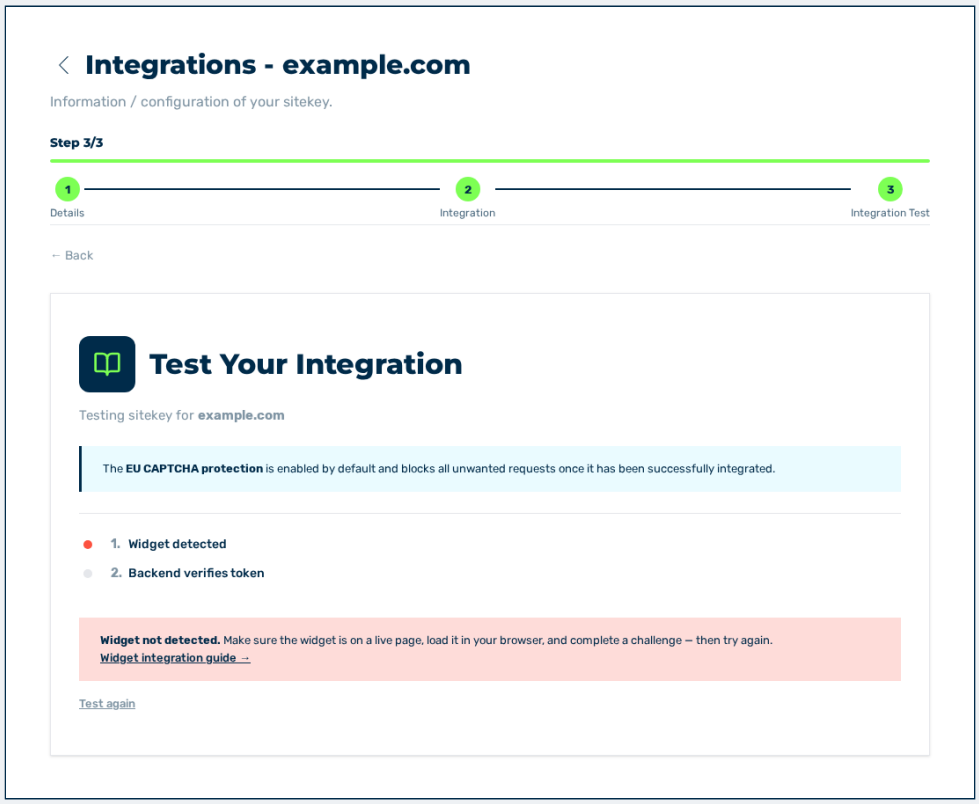


Figure 158: Integration Test view with the Widget not detected. message

The **Integration Test** view shows **Widget not detected.** The **Sitekey List** area shows the **Not Started** status.

6.8.1 Cause

The widget does not load on the protected page, or it does not find its element. The causes that follow are possible:

- The script element or the widget element is missing on the page.
- The value in the `data-sitekey` attribute is related to a different sitekey.
- A Content Security Policy of your website blocks the addresses of the service. See **Content Security Policy**.

6.8.2 Solution

- Examine if the script element is given in the `<head>` area of the page:

```
<script src="https://cdn.eu-captcha.eu/verify.js" async defer></script>
```

- Examine if the widget element is given on the page and if it has the `eu-captcha` class:

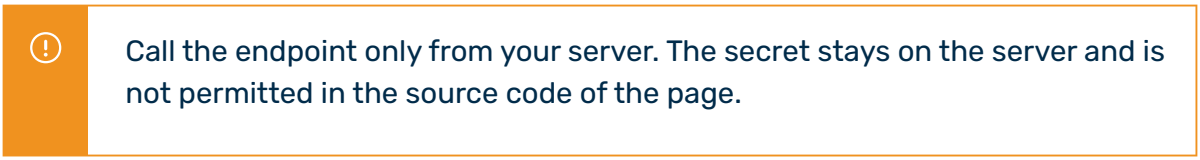
```
<div class="eu-captcha" data-sitekey="a1b2c3d4-0000-0000-0000-000000000000"></
div>
```

- Examine if the value in the `data-sitekey` attribute agrees with the public key of the sitekey. See **Details view**.
- Open the protected page and examine in the developer console of the browser if the `verify.js` file loads.
- In the developer console, examine if a message gives a Content Security Policy. See **Content Security Policy**.
- Do the test again. See **Testing the integration**.

If the test shows **Widget is working but your server didn't verify the token.**, the server-side verification is missing. See **Embedding the widget**.

7. API

7.1 Verify a client token



Call the endpoint only from your server. The secret stays on the server and is not permitted in the source code of the page.

Send a `POST` request to the `/verify` endpoint at `https://api.eu-captcha.eu/v1`.

7.1.1 Description

The endpoint examines the proof-of-work token that `verify.js` made in the browser of the visitor. The verification finds if the calculations were done correctly, if the token was not used before, and if the sitekey and the secret are valid.

The `success` field of the response names the result of the verification. Always also examine the `train` field. See the description of the field in **Response fields**.

The endpoint does not need an authorization in a header. The authentication occurs with the `secret` field in the body of the request.

7.1.2 Request

The body of the request contains these fields:

Field	Type	Necessary	Description
`sitekey`	`string`	yes	Public sitekey of the domain that the widget is embedded in. The value is shown in the **Details** view of the sitekey.
`secret`	`string`	yes	Secret that belongs to the sitekey. The value is shown in the **Details** view of the sitekey and stays on the server.
`client_ip`	`string`	yes	IPv4 or IPv6 address of the visitor. Send the address of the visitor, not the address of a proxy or a CDN in front of it. To get it, examine the `X-Forwarded-For` or `X-Client-IP` headers.
`client_token`	`string`	yes	Token from `verify.js` . The value comes to your server in the `eu-captcha-response` form field. The value is empty if the widget did not complete, for example when JavaScript is off.
`client_user_agent`	`string`	yes	`User-Agent` header of the request of the visitor. The value identifies the type of the client if no token was calculated.

Always send the `client_token` field, also with an empty value. From incomplete and empty tokens, the service learns across all verifications, also the failed ones, and thus detects attacks more accurately.

See **Details view**.

7.1.3 Example

Body of the request:

```
{
  "sitekey": "1c87e240-0000-0000-0000-23ac9f99da68",
  "secret": "LqFgQA••••",
  "client_ip": "XXX.XXX.XXX.XXX",
  "client_token": "XVtHUQEx••••",
  "client_user_agent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/114.0.0.0 Safari/537.36"
}
```

7.1.4 Responses

The endpoint supplies these status codes:

Status code	Description
`200`	Result of the verification. Examine `success` and `train`.
`400`	Necessary fields are missing in the body of the request, or the body is incorrect.
`429`	The count of the requests is exceeded. Wait the count of seconds that is given in the `Retry-After` header, then send the request again.
`500`	Unexpected error on the server.

7.1.4.1 Response fields

The response with the `200` status code contains these fields:

Field	Type	Description
`success`	`boolean`	`true` if the token passed the verification, if not `false`. Always also examine the `train` field, because `success` gets the `true` value when `train` is `true`.
`train`	`boolean` or `null`	Tells if a true verification occurred. The `false` and `null` values are the usual operation. The `true` value is a verification that was skipped.
`error-codes`	`array`	Available only with `success: false`. Names the causes of the failure.

A response with `train: true` means that the request was not examined and that each transmission counts as successful. This occurs with an unknown sitekey, with a secret that does not agree, and when the protection of the sitekey is off. It also occurs with each other malfunction of the verification. In production, examine the values of `sitekey` and `secret` immediately.

7.1.4.2 Error codes

The `error-codes` field contains these values. The format agrees with reCAPTCHA, hCaptcha, and Turnstile.

Value	Meaning
`invalid-input-secret`	The secret does not agree with the sitekey.
`invalid-input-sitekey`	The sitekey is unknown.
`invalid-input-response`	The `client_token` field was available, but it did not contain a valid proof. The value was incorrect, not decodable, unsolved, or empty.
`timeout-or-duplicate`	The token was used before, or the challenge is expired. Each token is valid one time.
`missing-input-secret`	The `secret` field is missing or is not a string.
`missing-input-sitekey`	The `sitekey` field is missing or is not a string.
`missing-input-response`	The `client_token` field is missing or is not a string. An empty string counts as available and causes `invalid-input-response`.
	The `client_ip` field is missing or is not a string.

Value	Meaning
`missing-input-remoteip`	

7.1.4.3 Examples of the response

Token valid:

```
{
  "success": true,
  "train": false
}
```

Token invalid or used before:

```
{
  "success": false,
  "train": false
}
```

Verification skipped, credentials incorrect:

```
{
  "success": true,
  "train": true
}
```

Error response with the `400`, `429`, and `500` status codes:

```
{
  "error": "missing_field",
  "message": "Required field 'sitekey' is missing."
}
```

7.1.5 Verify the credentials

To examine the sitekey and the secret without a token, see **Verify the sitekey and the secret**.

7.2 Verify the sitekey and the secret

Call the endpoint only from your server. The secret stays on the server and is not permitted in the source code of the page.

Send a `POST` request to the `/verify-credentials` endpoint at `https://api.eu-captcha.eu/v1`.

7.2.1 Description

The endpoint examines if a pair of a sitekey and a secret is valid and active. A token of the visitor is not necessary for this. Use the call at the start of your application or during the setup, to detect incorrect credentials early.

Different from `/verify`, the endpoint does not use a token and does not do a proof-of-work verification. It finds only if the credentials that were sent are known and released.

The endpoint is available from version 1.1 of the API.

See **Verify a client token**.

7.2.2 Request

The body of the request contains these fields:

Field	Type	Necessary	Description
`sitekey`	`string`	yes	Public sitekey of the domain that the widget is embedded in. The value is shown in the **Details** view of the sitekey.
`secret`	`string`	yes	Secret that belongs to the sitekey. The value is shown in the **Details** view of the sitekey and stays on the server.

See **Details view**.

7.2.3 Example

Body of the request:

```
{
  "sitekey": "1c87e240-0000-0000-0000-23ac9f99da68",
  "secret": "LqFgQA••••"
}
```

7.2.4 Responses

The endpoint supplies these status codes:

Status code	Description
`200`	Result of the verification of the credentials.
`400`	Necessary fields are missing in the body of the request, or the body is incorrect.
`429`	The count of the requests is exceeded. Wait the count of seconds that is given in the `Retry-After` header, then send the request again.
`500`	Unexpected error on the server.

7.2.4.1 Response fields

The response with the `200` status code contains this field:

Field	Type	Description
`valid`	`boolean`	`true` if the sitekey is available and the secret agrees with it, if not `false`.

7.2.4.2 Examples of the response

Credentials valid:

```
{
  "valid": true
}
```

Sitekey unknown or secret does not agree:

```
{
  "valid": false
}
```

Error response with the `400`, `429`, and `500` status codes:

```
{
  "error": "missing_field",
  "message": "Required field 'sitekey' is missing."
}
```

7.3 Managing sitekeys

Besides the verification API, the Myra EU CAPTCHA operates a second interface: the API of the dashboard. Through it, you create sitekeys, read their settings, and change them. The dashboard itself uses this interface only.

The base URL is:

```
https://api-app.eu-captcha.eu/myra-auto-app-api
```

This interface serves the dashboard and is not part of version 1.1 of the public API. It has no assurance of stability and no description that can be called up in the browser. For the verification of tokens, use `https://api.eu-captcha.eu/v1` only. See **Verify a client token**.

7.3.1 Log in

All calls require a token according to JSON Web Token. You receive it through the `/login` endpoint:

```
curl -X POST https://api-app.eu-captcha.eu/myra-auto-app-api/login \
  -H "Content-Type: application/json" \
  -d '{
        "email": "max.mustermann@example.com",
        "password": "••••••••"
      }'
```

The response contains the token:

```
{
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9••••"
}
```

If two-factor authentication is switched on for the account, the endpoint responds with the `totp_required` field and a `preAuthToken` value instead. The token follows only after the confirmation of the one-time password. See **Set up two-factor authentication**.

Pass the token in the header with every further call:

```
Authorization: Bearer <token>
```

7.3.2 Listing sitekeys

```
GET /captcha_sitekeys
```

The endpoint returns the sitekeys that belong to your account or for which you hold a permission. The following parameters are available:

Parameter	Description
`page`	Page of the result list.
`itemsPerPage`	Number of entries per page.
`light`	With `light=1` , the expensive metrics on the usage are omitted. The response then contains only the master data.

```
curl "https://api-app.eu-captcha.eu/myra-auto-app-api/captcha_sitekeys?
page=1&itemsPerPage=10" \
  -H "Authorization: Bearer <token>"
```

The response contains the entries in the `member` field and the total number in the `totalItems` field.

7.3.3 Reading a single sitekey

```
GET /captcha_sitekeys/{id}
```

`{id}` is the internal number of the sitekey, not its UUID.

7.3.4 Creating a sitekey

```
POST /captcha_sitekeys
```

The request uses the `application/ld+json` content type:

```
curl -X POST https://api-app.eu-captcha.eu/myra-auto-app-api/captcha_sitekeys
\
  -H "Authorization: Bearer <token>" \
  -H "Content-Type: application/ld+json" \
  -d '{
        "domain": "example.com",
        "label": "Contact form"
      }'
```

The server creates the sitekey and the secret. The sitekey is a UUID, the secret a random value of 32 bytes in Base64. Both values are given in the response.

7.3.5 Changing a sitekey

```
PATCH /captcha_sitekeys/{id}
```

The request uses the `application/merge-patch+json` content type and contains only the fields that change:

```
curl -X PATCH https://api-app.eu-captcha.eu/myra-auto-app-api/
captcha_sitekeys/42 \
  -H "Authorization: Bearer <token>" \
  -H "Content-Type: application/merge-patch+json" \
  -d '{ "initialDifficulty": 2 }'
```

A sitekey is not removed but deactivated through the `deleted` field:

```
{ "deleted": true }
```

See **Deleting a sitekey**.

7.3.6 Fields

You set the following fields when creating and when changing:

Field	Create	Edit	Value range
`domain`	yes	no	Valid host name. The domain of an existing sitekey is not changed.
`label`	yes	yes	255 characters at most.
`train`	yes	yes	`true` or `false`. In training mode, the verification always responds successfully.
`maxParallel`	yes	yes	1 to 10.
`initialDelay`	yes	yes	3 to 30 seconds.
`initialDifficulty`	yes	yes	0 to 3.
`wizardCategory`	yes	no	Category from the installation wizard, 64 characters at most.
`wizardTechnology`	yes	no	Technology from the installation wizard, 64 characters at most.
`deleted`	no	yes	`true` deactivates the sitekey.

If a value is outside of the range, the server responds with an error and names the affected field. The section **Configuring a sitekey** describes the meaning of the settings.

7.3.7 Additional fields of the response

Besides the stored values, the response contains calculated fields:

Field	Meaning
`permissionLevel`	Your access to the sitekey: `owner`, `write`, `read`, or `none`.
`challengesServed`	Number of the challenges delivered in the most recent period.
`installationStatus`	`not_started`, `widget_installed`, or `fully_integrated`. The value only moves forward.
`recentActivity`	`active` or `inactive`.

The last three fields are omitted for a call with `light=1`.

7.3.8 Permissions

The access depends on the permission of the sitekey:

Permission	Read	Edit	Secret in the response
Owner	yes	yes	yes
`write`	yes	yes	yes
`read`	yes	no	no, the field is empty

You manage the permissions in the dashboard. See **Grant access to a sitekey**.

7.4 Retrieving statistics

The figures that the dashboard shows in the **Dashboard** and **Challenges** views come from the API of the dashboard. Through the same endpoints, you retrieve the values for your own analyses.

The base URL is:

```
https://api-app.eu-captcha.eu/myra-auto-app-api
```

All calls use the `GET` method and require a token according to JSON Web Token in the `Authorization` header. See **Managing sitekeys**.

This interface serves the dashboard and is not part of version 1.1 of the public API. It has no assurance of stability and no description that can be called up in the browser.

7.4.1 Endpoints

The following endpoints are available:

Endpoint	Contents
`/captcha-stats-user`	Actions across all sitekeys to which you have access.
`/captcha-stats-sitekey`	Actions of a single sitekey.
`/captcha-challenges-per-user`	Number of the challenges across all accessible sitekeys.
`/recent-captcha-challenges-for-sitekey`	The most recent challenges of a sitekey as a list.
`/captcha-difficulty-distribution`	Distribution of the levels of difficulty of a sitekey.
`/captcha-threats-blocked`	Number of the blocked accesses of a sitekey.

A sitekey to which you have no access returns the response `{"success": false}`. The same applies if the `sitekey` parameter is missing, provided that the endpoint requires it.

7.4.2 Selecting the period

The endpoints on the actions and the challenges use the same parameters for the period:

Parameter	Default	Description
`sitekey`	empty	UUID of the sitekey. If the value remains empty, all accessible sitekeys apply.
`max_days`	`5`	Length of the period in days.
`start_date`	empty	First day of the period in the `YYYY-MM-DD` format. Without an entry, the period ends on the current day.
`per_day`	`0`	With `per_day=1`, `/captcha-stats-user` returns the values per day instead of as a total.

```
curl "https://api-app.eu-captcha.eu/myra-auto-app-api/captcha-stats-user?
max_days=30&per_day=1" \
  -H "Authorization: Bearer <token>"
```

The response contains the type of the action, the number and — for `per_day=1` — the date per line:

```
[
  { "action_type": "ACC", "date_at": "2026-08-01", "count": 1284 },
  { "action_type": "BIT", "date_at": "2026-08-01", "count": 37 }
]
```

7.4.3 Types of actions

The `action_type` field distinguishes the following values:

Value	Meaning
`VFC`	A challenge was delivered.
`ASC`	A challenge was solved.
`VSC`	The client has reported the challenge without success. The value counts as neutral, not as a block.
`ACC`	The verification of the token was successful, the access is permitted.
`BCI`	The proof for the challenge was wrong.
`BIC`	The client has submitted a challenge other than the one assigned to it.
`BCR`	The token was already used or had expired.
`BIT`	The request did not contain a token that could be evaluated.

All values that begin with `B` are blocked accesses.

The dashboard summarises the values as follows:

Key figure	Included actions
Verified	all `ACC` actions
Blocked	all actions whose value begins with `B`
Challenges solved	the `ASC` and `VSC` actions
Block rate	Share of the blocked accesses in the total of the blocked and the permitted accesses, as a percentage

See **Dashboard**.

7.4.4 Most recent challenges

```
GET /recent-captcha-challenges-for-sitekey
```

The endpoint uses the following parameters:

Parameter	Default	Description
`sitekey`	—	UUID of the sitekey. The parameter is required.
`max_results`	`30`	Number of entries per page.
`page`	`0`	Page of the result list.

The response contains the entries in the `member` field, the total number in the `totalItems` field as well as the domain and the label of the sitekey.

The `created_ai_result` field gives the assessment of the request. It has the values `browser`, `bot`, and `unknown`. See **Challenges**.

7.4.5 Distribution of the difficulty

```
GET /captcha-difficulty-distribution
```

The endpoint uses the following parameters:

Parameter	Default	Description
`sitekey`	—	UUID of the sitekey. The parameter is required.
`days`	1	Period in days, 1 to 7. Higher values count as 7.

The response gives the number of the challenges for each level of difficulty. The levels of the display go further than the initial value of a sitekey, because the Myra EU CAPTCHA raises the difficulty automatically on suspicion. See **Configuring a sitekey**.

7.4.6 Blocked accesses

```
GET /captcha-threats-blocked
```

The endpoint uses the same parameters as the distribution of the difficulty and responds with a single number:

```
{ "count": 214 }
```

8. Reference

8.1 Glossary

This page explains the terms that occur in this documentation.

8.1.1 Terms

The following terms are in use:

Term	Meaning
Assessment	A verified request. The verification in the background and the corresponding server-side verification together count as one assessment. The number of the assessments each month determines the plan. See **Book a plan**.
Bouncer	Standalone upstream server that connects the decisions of CrowdSec with the Myra EU CAPTCHA. See **CrowdSec**.
Challenge	Computing task that the browser of the visitor solves in the background. The token comes out of the solution. See **How EU CAPTCHA works**.
Content Security Policy	Policy of a website that specifies to the browser the sources from which it can load content. It must permit the addresses of the Myra EU CAPTCHA. See **Content Security Policy**.
Difficulty	Effort of a challenge. The analysis shows the levels 0 to 7. The initial value for each sitekey contains the levels 0 to 3. See **Configuring a sitekey**.
Org Admin	Role in an organization. It permits the administration of the members and of the login through SAML. See **Assign and revoke the Org Admin role**.
Passkey	Key on the device of the users that replaces the password at the login. See **Create a passkey**.
Proof of work	Procedure in which the client solves a computing task before a request is accepted. For single users, the effort is unnoticeable; for mass access through bots, it becomes expensive. See **How EU CAPTCHA works**.
SAML	Procedure for the login through the identity provider of a company. See **Set up SAML**.
Secret	Secret value of a sitekey. It belongs on the server only and identifies the calls of the API. See **Details**.
Sitekey	Public value that marks a protected domain. It is located in the widget of the page. See **Creating a sitekey**.

Term	Meaning
Token	Result of a solved challenge. The widget puts it into the `eu-captcha-response` form field. The server verifies it through the `/verify` endpoint. Each token is valid one time only. See **Verify a client token**.
train mode	Condition in which the API skips the verification and sets `success` permanently to `true`. The response then contains `train: true`. See **Configuring a sitekey**.
verify.js	Script of the widget. It creates the challenge in the browser and puts the token into the form. See **Embedding the widget**.
Widget	Component that you embed into your form. It runs the challenge and hands over the token. See **Embedding the widget**.

8.2 Content Security Policy

A Content Security Policy (CSP) is a policy with which a website specifies the sources from which the browser can load content. If your website sends such a policy, then the browser loads the widget of Myra EU CAPTCHA only if the policy permits the addresses of the service.

8.2.1 Necessary permissions

The widget uses two addresses. The table that follows gives the directives that must contain these addresses:

Directive	Address	Function
`script-src`	`https://cdn.eu-captcha.eu`	Loads the `verify.js` file into the protected page.
`frame-src`	`https://cdn.eu-captcha.eu`	Loads the `check.html` file into the hidden iframe in which the verification occurs.
`connect-src`	`https://api.eu-captcha.eu`	Requests a value for the Private Access Token procedure before the verification. Currently, only Safari supports this procedure.

`frame-src` is as necessary as `script-src`. The `verify.js` file makes the iframe at run time. Thus a missing permission does not become apparent during the integration, but in operation.

8.2.2 Example of a policy

The header that follows permits everything that the widget needs:

```
Content-Security-Policy: script-src 'self' https://cdn.eu-captcha.eu; frame-src 'self' https://cdn.eu-captcha.eu; connect-src 'self' https://api.eu-captcha.eu
```

The `default-src` directive applies as a fallback for all directives that your policy does not give individually. If your policy contains only `default-src`, then add the two addresses there.

8.2.3 Effect of a missing permission

A missing permission has a different effect for each directive:

Missing directive	Effect
`script-src`	The browser does not load `verify.js` . No widget shows on the page, and the form does not contain an `eu-captcha-response` field.
`frame-src`	The `verify.js` file loads, but the iframe stays empty. The widget does not make a token.
`connect-src`	The widget continues to operate. In Safari, the Private Access Token procedure does not occur.

The browser blocks the file before it sends the request. Therefore the request shows in the network area of the developer tools without a status code. This result looks like a malfunction of the CDN, although the service is reachable and answers. Refer to **The API is not reachable**.

8.2.4 Appearance of the widget

The widget sets its size and the position of the Myra logo with style attributes on the elements that it makes. If your policy contains a `style-src` or `style-src-attr` directive without the `'unsafe-inline'` value, then the widget shows in the wrong position or in the wrong size. The verification itself continues.

8.2.5 Examine the policy

To examine the policy of your website, do the steps that follow:

- Open the protected page in a browser.
- Open the developer tools of the browser.
- In the console, examine if a message gives one of the three addresses.
- In the network analysis, examine the `Content-Security-Policy` response header of the page.

➔ The messages of the console give the directive that refused the access.

A policy is either in the `Content-Security-Policy` response header or in a `<meta http-equiv="Content-Security-Policy">` element in the `<head>` area of the page. The two forms operate together: the browser applies each policy individually, and thus the permission must be in each of them.

Refer to **The widget is not detected** and **How it works**.