

IT Requirements - enmacc

Info: Please send this fact sheet to your IT department prior to using the enmacc platform in order to avoid technical problems

Outline

[About the platform](#)

[Browser support](#)

[Firewall & Web Security configuration](#)

[Verifying availability of the pusher websocket](#)

[Platform update strategy](#)

[Mailing](#)

[Billing](#)

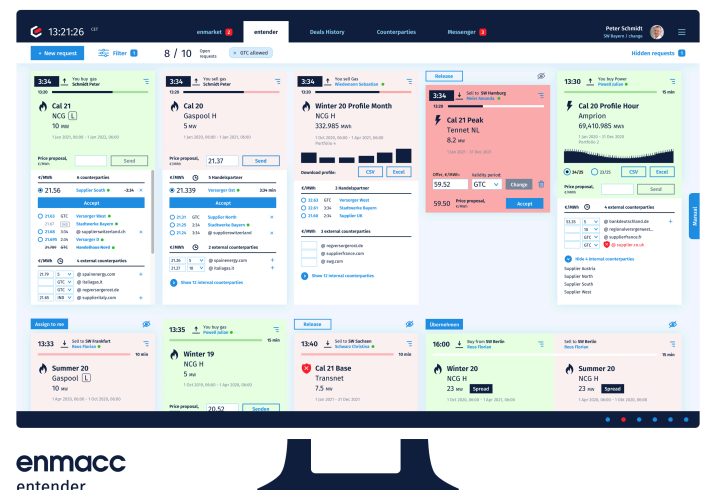
[API](#)

[Contact](#)

About the platform

enmacc is an OTC energy trading platform for Power, Gas, and environmental products. It is used via a web browser or mobile application ([iOS](#) & [Android](#)).

- Price negotiations for standards and non-standards in real-time
- Trading mechanisms:
Request For Quotes (RFQ) or Central Limit Order Book (CLOB)
- User permission management
- Credit risk management
- Optional anonymous trading
- Spread requests
- Internal chat messenger
- API for a variety of use-cases



Enmacc is a cloud-native platform and runs on Amazon Web Services located in Frankfurt, Germany.

More information about the platform is available at <https://enmacc.com>

Browser support

The browser-based platform can be used without the installation of applications or plugins. enmacc uses **HTML5 notifications** and the **HTML5 local storage**.

Enmacc supports the following browsers:

- **Google Chrome: v. 107 and newer**
- **Mozilla Firefox: v. 107 and newer**
- **Microsoft Edge: v. 107 and newer**

Be aware that browser plugins and extensions, like translators or ad blockers can impact functionality of the platform. We recommend not to use those on the platform.

Info: Support for Internet Explorer 11 has been discontinued due to security and feature issues. The browsers Safari and Opera are also not supported.

Firewall & Web Security configuration

The platform uses **Server Side Events (SSE)** and pusher.com to push updates to the browsers and uses encrypted communication via **HTTPS (TLS 1.2 & TLS 1.3) and WSS**.

Configure firewalls and proxies to allow traffic from the enmacc.com domain with all possible sub-domains (*.enmacc.com) for both protocols, HTTPS and WSS. Additional subdomains will be introduced in the future. A full list is available below.

Info: Please make sure to permit traffic for all HTTP request methods: GET, HEAD, POST, PUT, DELETE, OPTIONS, PATCH

HTTPS protocol (*tcp, Port 443*):

Enmacc domains	Third Party Domains
enmacc.com <ul style="list-style-type: none"> https://enmacc.com *.enmacc.com <ul style="list-style-type: none"> https://www.enmacc.com https://trading.enmacc.com https://gw.trading.enmacc.com https://chat.trading.enmacc.com https://sse.trading.enmacc.com https://ffp.trading.enmacc.com https://dap.trading.enmacc.com https://hermes.trading.enmacc.com https://enlytics.enmacc.com https://connect.enmacc.com https://ensights.trading.enmacc.com https://auth.trading.enmacc.com https://ld-ff.trading.enmacc.com 	*.hotjar.com <ul style="list-style-type: none"> https://script.hotjar.com *.hotjar.io <ul style="list-style-type: none"> https://content.hotjar.com *.fidectus.com <ul style="list-style-type: none"> https://gen.fidectus.com *.amazonaws.com <ul style="list-style-type: none"> https://s3.eu-central-1.amazonaws.com

Additionally, enmacc makes use of the **secure Websocket protocol WSS** for real-time updates. Please permit the websocket protocol by your firewall and web security tools for the following domains:

Secure Websocket protocol (*wss, Port 443*):

Enmacc domains	Third Party Domains
*.enmacc.com <ul style="list-style-type: none"> wss://ws-chat.trading.enmacc.com wss://dap.trading.enmacc.com 	*.pusher.com <ul style="list-style-type: none"> wss://ws-eu.pusher.com *.hotjar.com <ul style="list-style-type: none"> wss://ws.hotjar.com

The domain **enmacc.de** is not used anymore by the enmacc platform and a redirect is implemented (301 Moved Permanently) to the **enmacc.com** domain.

→ *trading.enmacc.de* forwards to *trading.enmacc.com*

Verifying availability of the pusher websocket

The availability of the **pusher.com** service is critical for the usage of the enmacc platform. The pusher test website can be used to verify if the service is available.

1. Go to <https://test.pusher.com/?env=eu>
2. Choose the **EU cluster**
3. Select only the transport mode: **ws**
4. Click **Disconnect** then **Connect** then **Trigger (AJAX)**.
 - a. The message should be successfully sent
 - b. The connection needs to be established via **"transport": "ws"**
5. If you encounter an error verify your settings according to the section above for ***.pusher.com (wss, Port 443)**

The screenshot displays the Pusher Test interface. At the top, there are buttons for 'Trigger (AJAX)', 'Trigger (Client)', 'Connect', 'Disconnect', and 'SSL on'. The 'connected' status is shown. Below these buttons is the 'Event log' section, which contains a list of events. A red box highlights the 'Trigger (AJAX)' button and the 'Event log' section. A red arrow points from the 'Trigger (AJAX)' button to the 'Event log' section. Another red arrow points from the 'Event log' section to the 'Choose cluster' dropdown menu. The 'Choose cluster' dropdown menu is set to 'eu'. Below the dropdown menu is the 'Select transports' section, which has checkboxes for 'ws', 'xhr_streaming', 'xdr_streaming', 'xhr_polling', 'xdr_polling', and 'sockjs'. The 'ws' checkbox is checked. Below the 'Select transports' section is the 'Filter logs' section, which has checkboxes for 'status', 'message', 'error', and 'debug'. The 'status', 'message', and 'error' checkboxes are checked. Below the 'Filter logs' section is the 'Files' section, which lists 'pusher.js', 'pusher.min.js', 'json2.js', 'json2.min.js', and 'sockjs.js'. The 'Event log' section shows a message event: '15:01:04 message {"data":"hello"}'. Below this is a debug event: '15:01:04 debug Pusher : ["Event recd",{"event":"event","channel":"presence-channel","data":{"data":"hello"}}]'. Below this is another debug event: '15:01:04 debug Pusher : [{"No callbacks on presence-channel for pusher:subscription_succeeded"}]'. Below this is a debug event: '15:01:04 debug Pusher : [{"Event recd",{"event":"pusher_internal:subscription_succeeded","channel":"presence-channel","data":{"presence":{"count":2,"ids":["63","651"],"hash":{"63":null,"651":null}}}}]'. Below this is a debug event: '15:01:04 debug Pusher : [{"Event sent",{"event":"pusher:subscribe","data":{"auth":{"9ff50474ed7d2a9238bf:64d34cc219c69bce4669ab3ca3d64bebc742f7d44202f5e018e2aa7f5fad074"},"channel_data":{"user_id":"63"},"channel":"presence-channel"}}}]'. Below this is a status event: '15:01:04 status connected'. Below this is a debug event: '15:01:04 debug Pusher : [{"State changed","connecting -> connected with new socket ID 180132.3487573"}]'. Below this is a debug event: '15:01:04 debug Pusher : [{"Connecting",{"transport":"ws","url":"wss://ws-eu.pusher.com:443/app/9ff50474ed7d2a9238bf?protocol=7&client=js&version=8.0.1&flash=false"}}]'. Below this is a status event: '15:01:04 status connecting'. Below this is a debug event: '15:01:04 debug Pusher : [{"State changed","disconnected -> connecting"}]'. Below this is a status event: '15:01:03 status disconnected'. Below this is a debug event: '15:01:03 debug Pusher : [{"State changed","connected -> disconnected"}]'. The 'Event log' section is scrollable, and the 'Choose cluster' dropdown menu is also scrollable.

Pusher Test

Trigger (AJAX) Trigger (Client) Connect Disconnect SSL on connected

Event log

15:01:04 message {"data":"hello"}

15:01:04 debug Pusher : ["Event recd",{"event":"event","channel":"presence-channel","data":{"data":"hello"}}]

15:01:04 debug Pusher : [{"No callbacks on presence-channel for pusher:subscription_succeeded"}]

15:01:04 debug Pusher : [{"Event recd",{"event":"pusher_internal:subscription_succeeded","channel":"presence-channel","data":{"presence":{"count":2,"ids":["63","651"],"hash":{"63":null,"651":null}}}}]

15:01:04 debug Pusher : [{"Event sent",{"event":"pusher:subscribe","data":{"auth":{"9ff50474ed7d2a9238bf:64d34cc219c69bce4669ab3ca3d64bebc742f7d44202f5e018e2aa7f5fad074"},"channel_data":{"user_id":"63"},"channel":"presence-channel"}}}]

15:01:04 status connected

15:01:04 debug Pusher : [{"State changed","connecting -> connected with new socket ID 180132.3487573"}]

15:01:04 debug Pusher : [{"Connecting",{"transport":"ws","url":"wss://ws-eu.pusher.com:443/app/9ff50474ed7d2a9238bf?protocol=7&client=js&version=8.0.1&flash=false"}}]

15:01:04 status connecting

15:01:04 debug Pusher : [{"State changed","disconnected -> connecting"}]

15:01:03 status disconnected

15:01:03 debug Pusher : [{"State changed","connected -> disconnected"}]

Using version 8.0.1

Connecting using WSS/HTTPS.

Choose cluster

eu

Select transports

☒ ws

☐ xhr_streaming

☐ xdr_streaming

☐ xhr_polling

☐ xdr_polling

☐ sockjs

Filter logs

☒ status

☒ message

☒ error

☒ debug

Files

- pusher.js
- pusher.min.js
- json2.js
- json2.min.js
- sockjs.js

Platform update strategy

enmacc rolls out platform updates on a weekly schedule. To guarantee the best experience the browser must run the latest available platform version.

New versions are loaded through a simple browser refresh or through the automatic version update strategy:

1. The client periodically triggers HTTP HEAD requests to <https://trading.enmacc.com/index.html>
2. The response includes a **x-enmacc-tag** header, e.g. **version-25cw23-36c4f815**
3. If the client detects a version change the **version update banner** is triggered that allows the user to refresh the application

Appropriate caching strategies are in place to allow browser based caching.

Caution: Avoid proxying or caching strategies that circumvent enmaccs' caching configuration, to allow the automatic platform update to work.

Mailing

The enmacc platform sends transactional and information mails through an AWS hosted email server. The following well-known security best practices are implemented by enmacc to guarantee secure email delivery:

- Domain Keys (DKIM) signature
- Sender Policy Framework (SPF)
- Domain-based Message Authentication, Reporting, and Conformance (DMARC)
- Brand Indicators for Message Identification (BIMI)

Whitelist or update your SPAM settings to allow incoming mails from **enmacc.com**:

- no-reply@enmacc.com
- tradingplatform@enmacc.com
- billing@enmacc.com
- info@enmacc.com

Billing

Our invoices are sent as **PDFs** in an email via the mailbox billing@enmacc.com. Please adjust your filter settings accordingly.

API

Enmacc offers a modular HTTPS API (Application Programming Interface) to automate processes between enmacc and your trading systems. It enables, for example, an information exchange with your Portfolio Management System (PFM), Risk Management System, Pricing Tools or other (proprietary) systems to digitize and automate your energy trading processes.

The use of APIs is not a prerequisite for the successful use of enmacc. More information about the enmacc connect API are available here:

- <https://enmacc.com/energy-trading/automation>
- <https://developer.enmacc.com>

Info: Besides the enmacc connect API, enmacc also provides pre-configured deal-capturing integrations to major PFM systems or customizable integrations via (S)FTP or mail. Please contact connect@enmacc.com and ask about existing “Connect Config” setups.

Contact

Please don't hesitate to contact info@enmacc.com, or call **+49 (0) 89 215 415 00**, if you have any further questions. We can also support you with a demo environment access to check for correctly configured firewall and web security tools.

Additional information security related topics can be found in the **enmacc trust center**:
→ <https://enmacc.com/trust>