

TCPP Mask Node

Wie man den Masken-Node neben einem TCPP-Node betreibt und liest, was er zeigt: die Masken, die ein Node liefert, die Werte, die sie füllen, und den Fehlercode, mit dem er jede Anfrage beantwortet. Alles, was er mit dem Node gemeinsam hat, steht im [Handbuch des Nodes](#).

Drei Anfragen

`mask_output`,
`display_output` und
`ping`; alles andere wird mit
einem Fehler beantwortet.

Auto-reply

Jede Anfrage erhält eine
signierte Antwort, deren
`error_code` sagt, was
vorgefunden wurde.

Eine Binärdatei

Der Code des Nodes mit
einem Display obendrauf.
Nichts zu installieren.

ZUR SPRACHE

Die Oberfläche der Anwendung ist englisch. Bedienelemente stehen deshalb auch hier unter ihrem englischen Namen, genau so, wie sie auf dem Bildschirm und in den Abbildungen heißen.

INHALT

- | | |
|--|---|
| 01 Was ein Masken-Node ist | 06 Die Bibliothek über Neustarts hinweg |
| 02 Erste Schritte | 07 Arbeitsabläufe |
| 03 Die Ansicht Mask | 08 Kommandozeile |
| 04 Antworten und die beiden Schalter | 09 HTTP-API |
| 05 Was er annimmt und was er antwortet | 10 Anhang |

■ Was ein Masken-Node ist

`tcpp-mask` ist eine zweite ausführbare Datei, aus demselben Code gebaut wie der Node: ein TCPP-Modul, das sich wie ein Display verhält. Ein Node liefert ihm mit `mask_output` ein Layout und füllt es mit `display_output`; der Masken-Node zeichnet das Ergebnis als Gerätebildschirm und beantwortet jede Anfrage von sich aus, mit einer signierten Antwort, deren `error_code` sagt, was er vorgefunden hat. Damit hat ein Node, oder eine eigene TCPP-Implementierung, ein Gegenüber, das das Gesendete so prüft, wie ein Display es täte.

Alles Übrige an ihm ist der Node: dieselbe Identität, dasselbe Frame-Log, dieselben Peers und Signaturen, und dieselben Ansichten `Compose`, `Inbox`, `Sent`, `Peers`, `Identity` und `Log`, als Debug-Reiter unter seiner eigenen Ansicht `Mask`. Das [Handbuch des Nodes](#) beschreibt sie, und was es sagt, gilt hier unverändert. Dieses Handbuch behandelt, was der Masken-Node hinzufügt.

■ Erste Schritte

Was Sie bekommen haben

Eine ausführbare Datei, heruntergeladen als eigenes Archiv neben dem des Nodes, `tcpp-mask-...-windows-amd64.zip`. Wie der Node braucht sie keine Installation und greift auf nichts im Internet zu. Entpacken Sie sie und starten Sie sie aus einem Terminal:

```
# Windows
.\tcpp-mask.exe

# macOS und Linux
chmod +x tcpp-mask
./tcpp-mask
```

Der Download ist nicht signiert, beim ersten Start kommen also dieselben Warnungen wie beim Node; [Erste Schritte](#) im Handbuch des Nodes sagt, wie man an jeder vorbeikommt.

Neben einem Node starten

Allein nützt ein Masken-Node nichts: Er wartet darauf, dass ein Node ihm Masken schickt. Seine Standardwerte sind so gewählt, dass er zu einem Node auf dem üblichen Port passt:

EINSTELLUNG	NODE	MASKEN-NODE
<code>--port</code>	<code>8443</code>	<code>8444</code>
<code>--peer</code>	er selbst	<code>localhost:8443</code>
<code>--node-type</code>	<code>CSU</code>	<code>LDU</code> , die lokale Displayeinheit, an die sich die Maskenbeispiele richten
seine Dateien	in <code>tcpp-node-data/</code> neben der Binärdatei	in <code>tcpp-mask-data/</code> neben der Binärdatei

```
# PowerShell, zwei Fenster
.\tcpp-node.exe --peer localhost:8444
.\tcpp-mask.exe

# macOS und Linux
./tcpp-node --peer localhost:8444
./tcpp-mask
```

Öffnen Sie `https://localhost:8443/` für den Node und `https://localhost:8444/` für den Masken-Node. Jeder hat sein eigenes selbstsigniertes Zertifikat, der Browser warnt also für jeden einmal.

Was auf der Platte entsteht

Jeder hält Schlüssel, Zertifikat, Log und Records in einem eigenen Ordner, `tcpp-node-data/` oder `tcpp-mask-data/`; deshalb lässt sich der Masken-Node in dasselbe Verzeichnis entpacken wie ein Node, ohne dass beide eine Identität teilen. Die Dateien in `tcpp-mask-data/` sind die des Nodes, aufgeführt unter [Was auf der Platte entsteht](#). Beim Start gibt er die Übersicht des Nodes aus und drei eigene Zeilen:

```
2026/10/22 11:39:51 TCPP mask node LDU / P:NODE-01D1F0C8
2026/10/22 11:39:51 listening https://localhost:8445/
2026/10/22 11:39:51 identity C:\Users\User\tcpp-lab\m\tcpp-mask-data
...
2026/10/22 11:39:51 masks 0 in the library
2026/10/22 11:39:51 accepts mask_output, display_output, ping
2026/10/22 11:39:51 auto-reply on, strict checks on
```

Das Labor dieses Handbuchs

Die Abbildungen stammen aus dem [Zwei-Node-Labor](#) des Node-Handbuchs und einer dritten Instanz daneben, **M**, gestartet mit `--port 8445`, weil Node **B** schon 8444 belegt. Node **A** hat ihr das `mask_output` des Zählerbeispiels geschickt, dann sein `display_output`, dann ein `display_output` mit einer Maske, die M nie bekommen hat, und ein `ping`.

■ Die Ansicht Mask

TCPP Mask Node schema 1.3.0 | LDU / P: NODE-01D1F8C8 | :8445 | ES384 · 01d1f8c8 | LIVE

MASKS

MASK

ANSWERS

DEBUG

Compose

Inbox 4

Sent

Peers

Identity

Log

MASK, METER.AC.L004C030
30x6 received 11:40:17 values

Every mask, output this node accepted; it starts empty. Pick one to show it; a frame that arrives switches to its mask.

MASK, METER.AC.L004C030
I: MASK, METER.AC.L004C030.1.0.0 · 30 chars × 4 Lines · received in frame f-20261022T094017-001-9f22

Region **de-DE** ☒ Field outlines and ruler

Values from display_output frame f-20261022T094017-003-7648 at 11:40:17.

Press a button or Enter in an input field on the screen to see the display_input it stands for.

FIELDS

FIELD_ID	FIELD_SIGN	FIELD_TYPE	LINE	CHAR	LENGTH	VALUE	MANDATORY	NOHID
I:1010	GENERAL.DATETIME	OUTPUT.DATETIME	1	1	20	21.09.2026 14:15:00	yes	
I:1011	GENERAL.DATETIME	OUTPUT.TEXT	1	22	6	0 h		
I:1020	METER.ACTUAL.KWH	OUTPUT.NUMERIC	2	1	20	2032.8	yes	
I:1021	METER.ACTUAL.KWH	OUTPUT.TEXT	2	22	6	kWh		
I:1030	METER.ACTUAL.TARIFF.ID	OUTPUT.STRING	3	1	20	HT	yes	
I:1031	METER.ACTUAL.TARIFF	OUTPUT.TEXT	3	22	6	Tarif		
I:1040	METER.ID	OUTPUT.STRING	4	1	20	1ESY1170046711	yes	
I:1041	METER.ID	OUTPUT.TEXT	4	22	3	ID		
I:1042	METER.ACTUAL.STATUS	OUTPUT.LIGHT	4	27	1	OK		

ANSWERS

☒ **Auto-reply**
answer every request with its error code

☒ **Strict checks**
reject masks off the grid or overlapping, and values for unknown fields

ping REQUEST **000000** 11:40:17
from CSU-node answered - 202 accepted

display_output REQUEST **3FFFFFFF** 11:40:17
from CSU-node answered - 202 accepted
display_mask_id I: MASK, METER.AC.L004C030.1.0.0
names no mask this node has; send its mask_output first

display_output REQUEST **000000** 11:40:17
from CSU-node answered - 202 accepted

mask_output REQUEST **000000** 11:40:17
from CSU-node answered - 202 accepted

frames 8

ABBILDUNG 01 · Der Masken-Node öffnet mit **Mask**. Die Ansichten des Nodes stehen darunter, unter *Debug*.

- **Masks**, links, ist die Bibliothek: jede Maske, die der Node angenommen hat, jede mit ihrer Größe in Zeichen × Zeilen, ihrer Herkunft, ihren Layout-Befunden und ob schon Werte für sie eingegangen sind. Ein Klick zeigt sie an. Ein Frame, der für eine Maske eingeht, schaltet die Ansicht auf diese Maske um.
- Die mittlere Spalte ist die Maske selbst: Kennzeichen, Id, Größe und der Frame, mit dem sie kam, der Gerätebildschirm, der Frame, aus dem die Werte stammen, und die Tabelle **Fields**, eine Zeile je Feld mit Id, Kennzeichen, Typ, Position, Länge, aktuellem Wert und Merkmalen.
- **Answers**, rechts, listet jede empfangene Anfrage mit dem Code, mit dem sie beantwortet wurde.

Der Bildschirm

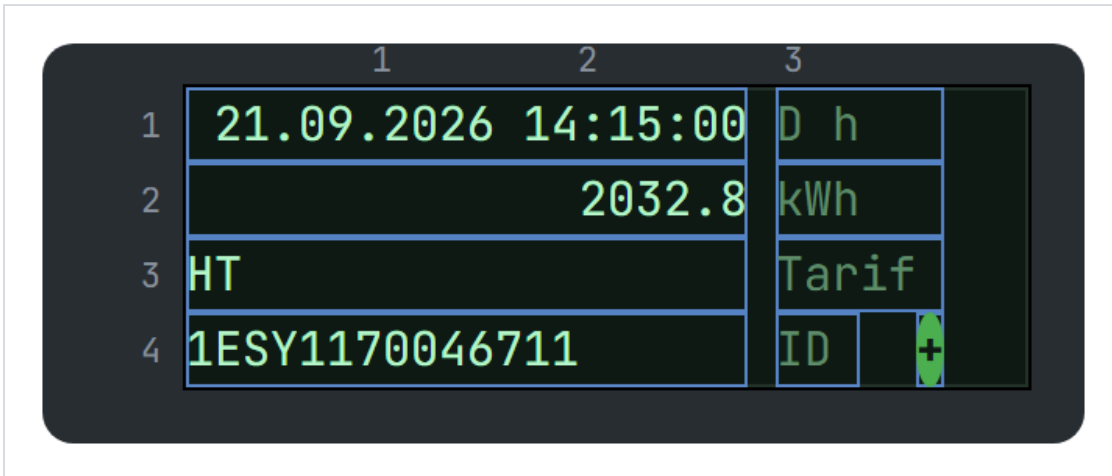


ABBILDUNG 02 • Der Bildschirm mit eingeschaltetem `Field outlines and ruler`.

Der Bildschirm wird auf dem Zeichenraster der Maske gezeichnet. `Region` wählt, in welcher Region der Maske (etwa `de-DE`) die Beschriftungen erscheinen.

`Field outlines and ruler` rahmt jedes Feld und nummeriert die Zeilen und jede zehnte Spalte; so fällt ein Feld, das um eins verrutscht ist, am schnellsten auf. Ein Druck auf eine Taste oder Enter in einem Eingabefeld des Bildschirms zeigt unter dem Bildschirm das `display_input`, für das er steht; gesendet wird es nicht.

■ Antworten und die beiden Schalter

ANSWERS		
☒ Auto-reply	answer every request with its error code	
☒ Strict checks	reject masks off the grid or overlapping, and values for unknown fields	
ping REQUEST	000000	11:40:17
from CSU-node answered · 202 accepted		
display_output REQUEST	3FFFFFF	11:40:17
from CSU-node answered · 202 accepted		
display_mask_id I:MASK.METER.DC.L004C030.1.0.0		
names no mask this node has; send its		
mask_output first		
display_output REQUEST	000000	11:40:17
from CSU-node answered · 202 accepted		
mask_output REQUEST	000000	11:40:17
from CSU-node answered · 202 accepted		

ABBILDUNG 03 · Vier Anfragen von Node A, jede von selbst beantwortet. Die mit 3FFFFFF nannte eine Maske, die der Node nie erhalten hat.

Jede Zeile nennt die Anfrage, den Code der Antwort, den Peer und ob die Antwort zugestellt wurde. Ein anderer Code als 000000 kommt mit dem Grund, den die Antwort auch in error_text trägt.

SCHALTER	FLAG	AUS HEISST
Auto-reply	--auto-reply	Die Antworten werden weiter ermittelt und gelistet, aber nicht gesendet: Der Frame wartet in der Inbox auf eine Antwort von Hand, wie bei einem Node.
Strict checks	--strict-layout	Eine Maske mit Layout-Befunden und Werte für Felder, die die Maske nicht hat, werden angenommen; die Befunde werden angezeigt statt beantwortet.

Beide sind standardmäßig an. Die Flags setzen sie beim Start, etwa --strict-layout=false; die Schalter ändern sie im laufenden Betrieb.

■ Was er annimmt und was er antwortet

- `mask_output` liefert ein Layout. Angenommen, kommt es in die Bibliothek, ersetzt dort eines mit derselben `mask_id` und wird angezeigt.
- `display_output` füllt eine Maske mit Werten. Die Maske wird über `display_mask_id` benannt, jeder Wert über `field_id` an sein Feld gebunden.
- `ping` wird mit dem Record des Beispiels `ping_response` beantwortet.

Jede andere Anfrage erhält eine Fehlerantwort. Antworten werden nie beantwortet, daher können sich zwei antwortende Nodes nicht gegenseitig aufschaukeln. Die Codes stammen aus `all_error_codes`, mit `FFF` als Modulstellen, wie in den Beispielen der Spezifikation selbst:

BEFUND	ERROR_CODE
Frame ist kein JWS / Nutzlast kein JSON / kein Head	<code>103FFF</code> / <code>201FFF</code> / <code>202FFF</code> , angezeigt, aber nicht beantwortet: Es gibt keinen Head, auf den sich antworten ließe
Signatur passt nicht zum gespeicherten Peer-Schlüssel	<code>104FFF</code>
ein Head-Element verfehlt sein Schema	<code>211</code> – <code>232</code> je Element (<code>213FFF</code> <code>record_udt</code> , <code>222FFF</code> <code>origin_id</code> , ...), sonst <code>203FFF</code>
<code>record_type</code> nicht einer der drei	<code>211FFF</code>
<code>record_direction</code> nicht request	<code>212FFF</code>
Record fehlt oder ist leer	<code>301FFF</code>
Pflichtelement fehlt / Pflichtwert null / falscher Typ, falsches Muster, falsche Länge	<code>302FFF</code> / <code>303FFF</code> / <code>4FFFFF</code>
Maske außerhalb des Rasters, überlappend oder mit doppelter <code>field_id</code> (strikt)	<code>303FFF</code>
<code>display_mask_id</code> nennt keine bekannte Maske, oder ein Wert kein Feld von ihr (strikt)	<code>3FFFFF</code>
nichts davon	<code>000000</code>

`parsed` trägt den Code am Pfad jedes Befunds, etwa `record/mask/mask_fields/14`, und weiter oben an `record/mask` und `record`. `error_text` sagt, was falsch war, gekürzt auf die Zeichen, die das Muster des Heads zulässt. Ein Head-Element, das seine eigene Prüfung nicht bestanden hat, wird nicht zurückgespiegelt: Typ und Id des Nodes selbst, oder das `X-TCPP-Node` des Absenders, treten an seine Stelle.

■ Die Bibliothek über Neustarts hinweg

Die Bibliothek folgt dem Frame-Log, wie der Posteingang. Sie beginnt leer — Masken kommen nur von einem Peer, die mitgelieferten Beispiele sind für den Node, der sie sendet —, und beim Start werden die empfangenen Frames vom ältesten an wieder eingespielt. Wo das Log die Antwort enthält, die hinausging, zählt diese Antwort: Eine Maske, die bei ausgeschalteten strikten Prüfungen angenommen wurde, bleibt auch nach einem Neustart mit eingeschalteten angenommen.

Um mit einer leeren Bibliothek neu anzufangen, halten Sie den Masken-Node an und entfernen `tcpp-frames.jsonl` aus `tcpp-mask-data/`. Schlüssel und Peers bleiben, für jeden, der ihn prüft, ist er also derselbe Node.

■ Arbeitsabläufe

A. Eine Maske von einem Node aus steuern

- 01 Starten Sie einen Node mit `--peer localhost:8444` und den Masken-Node daneben, und öffnen Sie beide im Browser.
- 02 Öffnen Sie auf dem Node **Compose**, wählen Sie `mask_output`, drücken Sie **Load...** → `example_meter` und **Sign & Send**. Der Masken-Node zeigt die Zählermaske, noch ohne Werte.
- 03 Wählen Sie `display_output`, laden Sie `example_meter` und senden Sie es. Der Bildschirm füllt sich mit den Werten des Zählers.
- 04 Beide Antworten treffen in der **Inbox** des Nodes ein, ihren Anfragen zugeordnet, und der Masken-Node listet sie unter **Answers**.
- 05 Ändern Sie eine `field_id` oder die `display_mask_id` und senden Sie erneut: So sehen Sie, mit welchem Code ein Display einen fehlerhaften Frame beantwortet.

EIN ÄLTERER NODE HAT ÄLTERE BEISPIELE

Die Beispiele zu `mask_output` und `display_output` wurden nach dem Build vom 21. Oktober 2026 überarbeitet, sodass jedes `display_output` die Maske derselben Variante füllt. Ein Node-Verzeichnis aus diesem oder einem früheren Build behält seine `records/mask_output_request/` und `records/display_output_request/`, wie sie waren, weil ein vorhandenes Schema-Verzeichnis nie neu angelegt wird: Löschen Sie beide in `tcpp-node-data/` und starten Sie den Node neu, dann werden die aktuellen Beispiele geschrieben.

B. Einen Frame von Hand beantworten

- 01 Schalten Sie **Auto-reply** aus, oder starten Sie den Masken-Node mit `--auto-reply=false`.

- 02 Senden Sie ihm vom Node eine Anfrage. Er zeigt die Maske weiterhin und listet den Code, mit dem er geantwortet hätte, sendet aber nichts.
- 03 Öffnen Sie auf dem Masken-Node den Debug-Reiter **Inbox** und wählen Sie den Frame: Das Antwortformular ist das des Nodes, aus der Anfrage vorbefüllt.
- 04 Setzen Sie `error_code` und `parsed` so, wie Ihr Test es braucht, und drücken Sie **Sign & Send response** .

■ Kommandozeile

Der Masken-Node versteht jedes Flag des Nodes, beschrieben unter [Kommandozeile und Dateien](#) in dessen Handbuch. Vier davon haben andere Standardwerte, zwei sind seine eigenen:

FLAG	STANDARD	BEDEUTUNG
<code>--port</code>	<code>8444</code>	TLS-Port, auf dem gelauscht wird.
<code>--peer</code>	<code>localhost:8443</code>	Standardziel, und wo ein Node auf dem üblichen Port lauscht.
<code>--node-type</code>	<code>LDU</code> , oder der gespeicherte Wert	Drei Großbuchstaben. Nach dem ersten Start gespeichert.
<code>--data-dir</code>	<code>tcpp-mask-data/</code> neben der Binärdatei	Wo Schlüssel, Zertifikat, Log, Peers und Records liegen.
<code>--auto-reply</code>	<code>true</code>	Empfangene Anfragen ohne den Bediener beantworten. Im Browser umschaltbar.
<code>--strict-layout</code>	<code>true</code>	Masken, die sich nicht sauber auslegen lassen, und Werte für unbekannte Felder zurückweisen. Im Browser umschaltbar.

Der Datenordner heißt nicht `tcpp-mask/`, weil unter macOS und Linux die Binärdatei selbst so heißt. Er läuft und hält an wie der Node; ein gescheiterter Start gibt `tcppmask: <Grund>` aus und endet mit `1`.

■ HTTP-API

Der Masken-Node bedient jede Route des Nodes, aufgeführt unter [HTTP-API](#) in dessen Handbuch, ebenso ohne Authentifizierung. Die Ansicht Mask nutzt drei weitere:

ROUTE	BEDEUTUNG
<code>GET /api/mask/state</code>	Die Bibliothek, die angezeigte Maske, ihre Werte und Layout-Befunde, die gegebenen Antworten und die beiden Einstellungen.
<code>POST /api/mask/active</code>	Eine andere Maske anzeigen: <code>{"mask_id":"..."}</code> . <code>404</code> , wenn die Bibliothek keine solche Maske hat.
<code>PUT /api/mask/settings</code>	Einen der Schalter ändern: <code>{"auto_reply":true,"strict":false}</code> ; ein weggelassenes Element behält seinen Wert. Gibt beide Einstellungen zurück.

```
# strikte Prüfungen an einem laufenden Masken-Node abschalten
curl -k -X PUT https://localhost:8444/api/mask/settings \
  -H 'content-type: application/json' -d '{"strict":false}'
```

■ Anhang

Glossar

Maske

Ein Bildschirmlayout, geliefert mit `mask_output`: ein Raster aus Zeichen × Zeilen und die darauf platzierten Felder, benannt durch seine `mask_id`.

Feld

Eine Stelle auf einer Maske, benannt durch ihre `field_id`, in die ein Wert aus `display_output` geschrieben wird.

Region

Eine Sprachfassung der Beschriftungen einer Maske, etwa `de-DE`.

Masken-Node

`tcpp-mask`, ein Node, der Masken anzeigt und jede Anfrage mit einem Fehlercode beantwortet, wie es ein Display täte.

Frame, Peer, Record

Wie im [Handbuch des Nodes](#).

Über die Abbildungen

Jede Abbildung in diesem Handbuch stammt von einem laufenden Masken-Node neben dem Zwei-Node-Labor des Node-Handbuchs, keine ist gezeichnet; was Sie hier sehen, ist also, was die Anwendung tut. Aufgenommen werden sie vom selben Skript wie die Abbildungen des Nodes, das einen Browser ohne Anzeige durch die Schritte eines Bedieners führt; sie werden neu aufgenommen, wann immer sich die Oberfläche ändert. Die Anwendung ist englisch, deshalb zeigen auch die Abbildungen dieser Ausgabe die englische Oberfläche.

`tcpp.digital` Deutsches Handbuch zum Masken-Node • Schema 1.3.0

Abbildungen von einem Masken-Node (LDU auf 8445) neben einem Zwei-Node-Labor, aufgenommen am 22. Oktober 2026. Teile dieser Dokumente wurden mit KI-Unterstützung erstellt.