

Verae API reference — parameters, types, sample calls

Hosts: **customer.verae.ai** (portal + /v1 timestamp registry), **peergos.verae.ai** (DataCube), **sign.verae.ai** (humanIdentification). Passwords stored as password_hash; API keys as api_tokens / api_key_hash (not plain api_key column).

POST https://customer.verae.ai/portal/api/signup

Auth: none · id: portal.signup
Screens: DC-01, DC-02, DC-A02, DC-M02, DC-05N, ZP-01
Functions: VeraePortalAuth.signup · portal/app.js signup · verae-customer-auth.signup

Param	Type	Req	Description
name	string	yes	Display name
email	string (email)	yes	Account email = Verae username
password	string (≥8)	yes	Portal password
acceptTos	boolean	yes	Must be true

Sample call

```
curl -sS -X POST https://customer.verae.ai/portal/api/signup \
-H 'content-type: application/json' \
-d '{"name":"Ada", "email":"ada@example.com", "password":"CorrectHorse1!", "acceptTos":true}'

# 201
{"token":"<session>", "apiKey":"xVerae-...", "customer":{"id":"cust_...", "email":"ada@example.com"},
"veraeProvisionDeferred":true}
Returns: 201 { token, apiKey?, customer, veraeProvisionDeferred? }
```

POST https://customer.verae.ai/portal/api/login

Auth: none · id: portal.login
Screens: DC-01, DC-03, DC-A1, DC-M4, ZP-01
Functions: VeraePortalAuth.login · portal login · SavedLogins Face ID → login

Param	Type	Req	Description
email	string	yes	Account email
password	string	yes	Portal password (password_hash in DB)
totpCode	string (6 digits)	no	Required if totpEnabled

Sample call

```
curl -sS -X POST https://customer.verae.ai/portal/api/login \
-H 'content-type: application/json' \
-d '{"email": "ada@example.com", "password": "CorrectHorse1!", "totpCode": "482193"}'

# 200 {"token": "<session>", "customer": {...}}
# 401 {"error": "invalid email or password"} | {"error": "totp_required"}
Returns: 200 { token, customer } · 401 invalid / totp_required
```

GET/PATCH https://customer.verae.ai/portal/api/profile

Auth: Bearer · id: portal.profile
Screens: DC-13, DC-A13, DC-14, ZP-02, ZP-07
Functions: HttpBillingApi.updateProfile · portal profile UI

Param	Type	Req	Description
Authorization	header string	yes	Bearer session token
name, companyName, groupName, ...	string	no	PATCH body fields
billingType	'stripe' 'purchase_order'	no	Payment method type
phone	string E.164	no	+1...

Sample call

curl -sS https://customer.verae.ai/portal/api/profile \
-H "Authorization: Bearer \$TOKEN"

curl -sS -X POST https://customer.verae.ai/portal/api/profile \
-H "Authorization: Bearer \$TOKEN" -H 'content-type: application/json' \
-d '{"companyName":"Brightline","billingType":"stripe"}'
Returns: customer public profile (no password_hash)

GET/POST/DELETE https://customer.verae.ai/portal/api/2fa/*

Auth: Bearer · id: portal.2fa
Screens: DC-03, DC-04, DC-A03, DC-A04, DC-M03, DC-M04
Functions: two_factor_screen · authenticator_setup · portal Security

Param	Type	Req	Description
setup	POST {}	no	Returns secret + otpauth URI
code	string	yes	6-digit TOTP to enable
totpCode	string	yes	On login when enabled

Sample call

```
curl -sS -X POST https://customer.verae.ai/portal/api/2fa/setup \  
-H "Authorization: Bearer $TOKEN" -H 'content-type: application/json' -d '{}'  
  
curl -sS -X POST https://customer.verae.ai/portal/api/2fa/enable \  
-H "Authorization: Bearer $TOKEN" -H 'content-type: application/json' \  
-d '{"code":"482193"}'  
Returns: status / secret / backup codes remaining
```

GET/POST/DELETE https://customer.verae.ai/portal/api/tokens

Auth: Bearer · id: portal.tokens
Screens: DC-14, ZP-07, signup response apiKey
Functions: ApiTokenRepo · VeraeRequestAuth.apiKey · portal Security keys

Param	Type	Req	Description
name	string	no	Token label on create
id	string	yes	Token id on DELETE

Sample call

```
curl -sS -X POST https://customer.verae.ai/portal/api/tokens \  
-H "Authorization: Bearer $TOKEN" -H 'content-type: application/json' \  
-d '{"name":"iphone"}'  
  
# 201 {"token":"xVerae-...", "id":"...", "prefix":"xVerae-..."}  
# Store once – server keeps api_key_hash only (never plain in customers table).  
Returns: plaintext xVerae- once; DB stores hash/prefix
```

GET https://customer.verae.ai/portal/api/usage

Auth: Bearer or x-api-key · id: portal.usage

Screens: DC-14, DC-A14, DC-M14, ZP-03

Functions: HttpBillingApi.usage · UsageBillingScreen

Param	Type	Req	Description
peergosUsername	string?	no	Filter storage lines
from, to	ISO date?	no	Period bounds

Sample call

```

curl -sS 'https://customer.verae.ai/portal/api/usage' \
-H "Authorization: Bearer $TOKEN"
# or -H "x-api-key: $XVERAE"
Returns: usage aggregates / line items

```

GET/POST https://customer.verae.ai/portal/api/billing/*

Auth: Bearer · id: portal.billing
Screens: DC-14, ZP-04, ZP-05, ZP-06
Functions: billing screens · portal Billing tab

Param	Type	Req	Description
methods	GET	no	Stripe cards + PO metadata
reload	POST {amountCents:number}	no	Stripe PaymentIntent
po-request	POST {poNumber,terms}	no	Purchase Order billing

Sample call

```
curl -sS -X POST https://customer.verae.ai/portal/api/billing/reload \
-H "Authorization: Bearer $TOKEN" -H 'content-type: application/json' \
-d '{"amountCents":5000}'

curl -sS -X POST https://customer.verae.ai/portal/api/billing/po-request \
-H "Authorization: Bearer $TOKEN" -H 'content-type: application/json' \
-d '{"poNumber":"PO-2026-091","terms":"Net-30"}'

Returns: Stripe client secret or PO status
```

POST https://customer.verae.ai/v1/timestamp

Auth: Bearer xVerae-... or x-api-key: id: v1.timestamp
Screens: DC-07, DC-07D, DC-07R, DC-08, DC-08R, DC-12, PT-*, MT-*

Functions: timestamps_ui · auto_stamp · edge /v1/timestamp · never JetStream job-complete

Param	Type	Req	Description
sha256	string (64 hex)	yes	Content digest only
restamp	boolean	no	default false; true links restampOf
restampOf	string jobId?	no	Parent stamp when restamp
metadata	object?	no	Optional non-file metadata (digested)

Sample call

```
curl -sS -X POST https://customer.verae.ai/v1/timestamp \
-H "Authorization: Bearer $XVERAE" -H 'content-type: application/json' \
-d '{"sha256":"e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855"}'
```

restamp with additional JSONL line digest:

```
curl -sS -X POST https://customer.verae.ai/v1/timestamp \
-H "Authorization: Bearer $XVERAE" -H 'content-type: application/json' \
-d '{"sha256":"<jsonl_line_sha256>","restamp":true,"restampOf":"<originalJobId>"}'
```

Returns: jobId, certificate/CID, status pending|included|final

GET https://customer.verae.ai/v1/timestamp/status

Auth: xVerae- or public policy · id: v1.timestamp.status
Screens: DC-08, DC-09, DC-10, DC-12, ZP-V*
Functions: receipt detail · verify · chain badges

Param	Type	Req	Description
sha256	string (64 hex)	yes	query param
jobId	string?	no	alternate lookup

Sample call

```
curl -sS 'https://customer.verae.ai/v1/timestamp/status?sha256=e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855' \
-H "Authorization: Bearer $XVERAE"
```

Returns: stage: pending|provisional|included|final + receipt refs

POST https://sign.verae.ai/sign/v1/requests

Auth: xVerae- owner · id: sign.requests
Screens: SG-W1, SG-I2, SG-M1, humanIdentification
Functions: SignApi.createRequest · sign-router

Param	Type	Req	Description
documentSha256	string 64 hex	yes	PDF digest
documentName	string	yes	Filename
signers	array<{email:string,name?:string}>	yes	Signer list
title	string?	no	Request title
message	string?	no	Note to signers
signInOrder	boolean	no	default true
requireOtp	boolean	no	default true

Sample call

```
curl -sS -X POST https://sign.verae.ai/sign/v1/requests \
-H "Authorization: Bearer $XVERAE" -H 'content-type: application/json' \
-d '{"documentSha256":"<64hex>","documentName":"msa.pdf",
"signers":[{"email":"grace@example.com","name":"Grace"}],"requireOtp":true}'
```

Returns: 201 request + publicToken

POST https://sign.verae.ai/sign/v1/public/:token/sign

Auth: public token + OTP · id: sign.public.sign
Screens: SG-W3, SG-W4, SG-I1, SG-A1
Functions: SignApi.submitSignature · HumanSignScreen

Param	Type	Req	Description
token	path string	yes	publicToken
email	string	yes	Signer email
otp	string	no	if requireOtp
signatureSha256	string 64 hex	yes	Hash of ink/PNG — never upload bytes
kind	'draw' 'type' 'upload'	yes	humanIdentification kind
consent	boolean	yes	must be true
name	string?	no	Display name

Sample call

```
curl -sS -X POST https://sign.verae.ai/sign/v1/public/$TOKEN/sign \
-H 'content-type: application/json' \
-d '{"email": "grace@example.com", "otp": "123456", "signatureSha256": "<64hex>", "kind": "draw", "consent": true}'
```

Returns: ok, eventSha256, stampRef?, status

— peergos.verae.ai (Peergos HTTP/API + local rust)

Auth: Peergos user session · id: peergos.cube
Screens: DC-05, DC-05N, DC-06, DC-07, DC-07D, DC-A2, DC-M1
Functions: RustLib session · file_browser · datacube_ui · destination Upload toggle

Param	Type	Req	Description
username	string	yes	DataCube account
password	string	yes	Encryption password (may equal Verae pw)
path	string	yes	Cube path / file path
bytes	bytes	yes	Upload — E2E encrypted client-side

Sample call

```
# App uses rust bridge (not raw curl). Conceptual:
# session.login(server, user, pass)
# session.createCube("Contracts 2026")
# session.upload(cubeId, "Lease.pdf", bytes)
# sha256 = sha256(bytes) # then optional POST /v1/timestamp
Returns: cube id, file multihash, chain blocks
```

local + APIs App add flow (DC-07D)

Auth: session + optional xVerae- · id: add.destination
 Screens: DC-07D, DC-07DM, DC-A07D, DC-M07D, DC-JL1, DC-06T
 Functions: file_browser add · JSONL builder · tags · auto_stamp

Param	Type	Req	Description
uploadPeergos	boolean	yes	Icon toggle default false
addTags	boolean	yes	Icon toggle default false
timestampRegistry	boolean	yes	Icon toggle default false
tags	string[]	no	On tags + previous selectable
metadataJsonl	string (JSONL)	no	Nested builder output
copyLastMetadata	boolean	no	Per-file metadata sheet

Sample call

Pseudocode after user confirms DC-07D:
 ops = {upload: false, tags: false, timestamp: false} # defaults OFF
 for file in batch:
 meta = file.meta or (copy_last_meta() if user.tapped_copy else {})
 if ops.tags: apply_tags(file, meta.tags | previous_selected)
 if ops.upload: peergos.upload(cube, file.bytes)
 digest = sha256(file.bytes) # always local
 if ops.timestamp:
 POST /v1/timestamp {sha256: digest}
 if meta.jsonl_lines:
 for line in meta.jsonl_lines:
 POST /v1/timestamp {sha256: sha256(line), restamp: true, restampOf: jobId}
 Returns: per-file progress; receipts on registry when timestamp on