PROSIGN · PRODUCTO B2B

Guía de Integración
Firma Electrónica Avanzada

Documento técnico para empresas clientes que delegan en ProSign el ciclo completo de firma de documentos con validación biométrica y certificación digital.

Versión del documento

1.0 — Mayo 2026

Audiencia

Equipos de ingeniería del cliente

Servicio

cert-ms · v1

Índice

01 Presentación del servicio

ProSign expone una API REST que permite a una empresa cliente subcontratar el ciclo completo de firma electrónica avanzada (FEA) sobre un documento PDF de su propiedad. El servicio se encarga de:

- Recibir el documento por API y almacenarlo de manera segura.
- Generar una sesión de verificación biométrica única e individual por cada firmante.
- Validar la identidad de cada firmante mediante cotejo facial frente a su cédula chilena.
- Ensamblar el documento final, incorporando la página de bloques de firma y el comprobante biométrico de cada firmante.
- Orquestar la firma digital secuencial a través de eCert (PSC acreditado en Chile).
- Notificar al sistema del cliente cada transición de estado mediante webhooks firmados con HMAC.

> **Resultado para el cliente**
>
> Su sistema realiza una sola llamada al API y, mediante notificaciones por webhook, recibe el documento firmado con valor legal pleno una vez que todos los firmantes completan el proceso.

Quién debe usar este documento

Esta guía está destinada al equipo técnico de la empresa cliente que implementará la integración con ProSign. Asume familiaridad con HTTP, JSON, JWT/Bearer tokens y validación HMAC-SHA256.

02 Conceptos clave

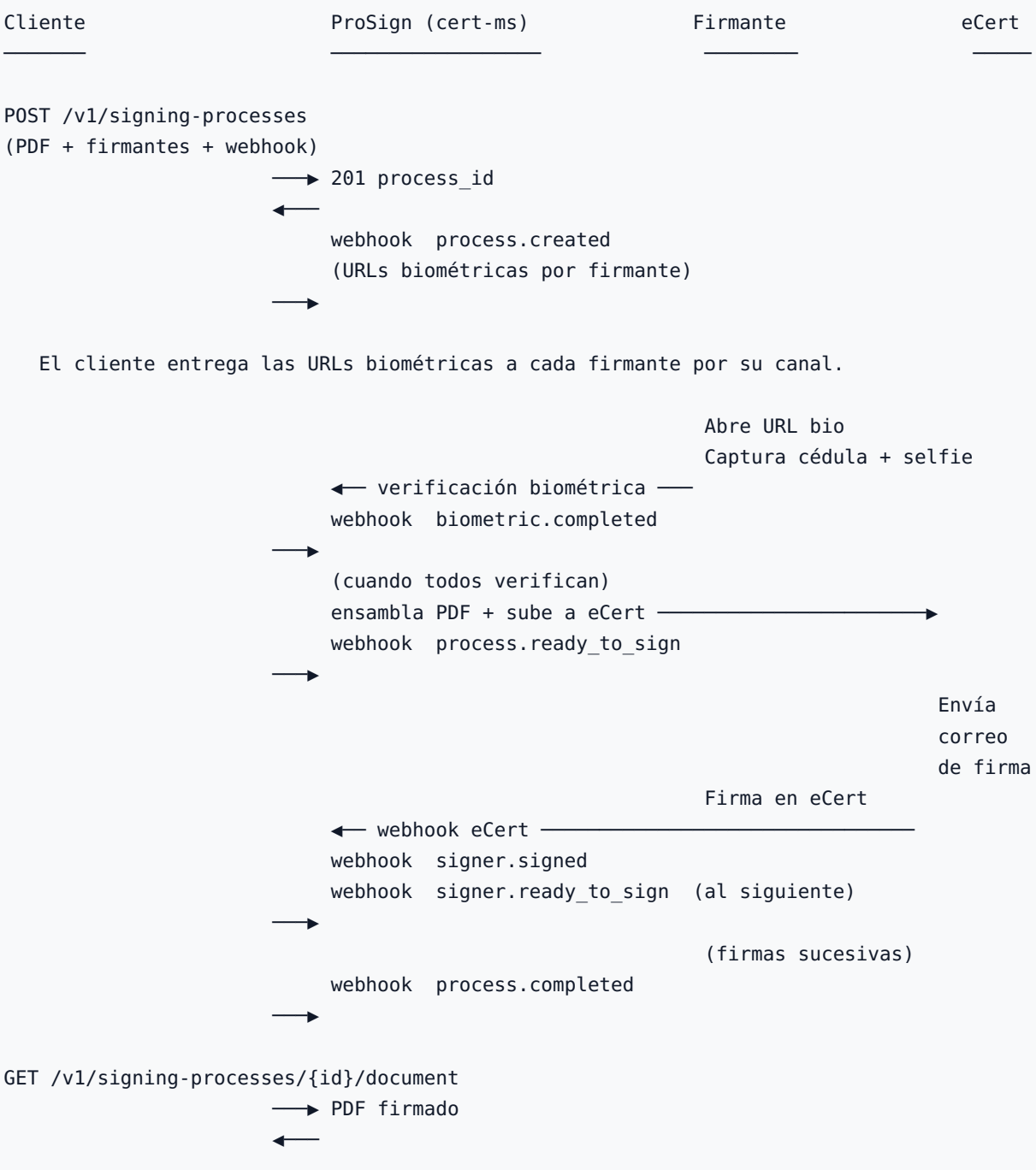
TÉRMINO	SIGNIFICADO
Proceso de firma	Unidad de trabajo creada por el cliente. Contiene un PDF, una lista de firmantes y la configuración del webhook.
Firmante	Persona natural identificada por RUT chileno que debe firmar el documento. Un proceso admite hasta veinte firmantes.
Sesión biométrica	URL única por firmante, vigente 24 horas por defecto, donde el firmante captura su cédula y un selfie/video para validar identidad.
Documento ensamblado	PDF final que se firma: documento original + página de bloques de firma + comprobantes biométricos individuales en hojas anexas.
Webhook	Notificación HTTP que ProSign envía al endpoint configurado por el cliente al ocurrir cada cambio relevante.
eCert	Prestador de Servicios de Certificación acreditado en Chile. ProSign delega la firma digital sobre el documento ensamblado.

Por qué se trata de Firma Electrónica Avanzada

La Ley 19.799 sobre Documentos Electrónicos exige, para reconocer una firma como avanzada, que su titular sea identificado de manera unívoca al momento de firmar. ProSign cumple esta exigencia mediante:

1. Validación biométrica del firmante frente a su cédula nacional.
2. Generación de un comprobante biométrico, con score de coincidencia y artefactos resguardados.
3. Incorporación inseparable del comprobante dentro del PDF firmado.
4. Certificado digital aplicado por eCert, prestador acreditado.

03 Diagrama del flujo



04 Requisitos previos

1. El cliente debe recibir de ProSign una **API key** de tipo `cms_live_*` que utilizará para autenticar todas las llamadas. Esta key se entrega una sola vez al momento de habilitar la integración.
2. El cliente debe exponer un **endpoint HTTPS público** capaz de aceptar peticiones POST con cuerpo JSON. Este endpoint recibirá los webhooks.
3. El cliente debe configurar internamente un **secreto compartido** para validar la firma HMAC-SHA256 de los webhooks. ProSign genera este secreto al crear cada proceso, salvo que el cliente provea uno propio.
4. Los firmantes deben acceder a las URLs biométricas desde un dispositivo con cámara (idealmente un smartphone moderno). El navegador debe permitir captura de imagen y video.

05 Autenticación

Toda llamada al API se autentica enviando la API key en el header `Authorization` con el esquema `Bearer`:

```http
Authorization: Bearer cms_live_xxxxxxxxxxxx_yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy
```

El formato de la API key se compone de cuatro segmentos separados por guión bajo:

SEGMENTO	DESCRIPCIÓN
`cms`	Identificador estático del servicio ProSign.
`live`	Entorno de la key. Las llamadas con `live` tienen efectos reales (eCert, biometría).
`xxxxxxxxxxxx`	Prefijo público visible (doce caracteres). Se utiliza para identificar la key sin exponer el secreto.
`yyyy…`	Secreto de la key. Tratar como contraseña: nunca registrar en logs ni en el repositorio.

> **Importante**
>
> El secreto de la API key solo se muestra una vez al momento de su creación. Si se extravía, debe revocarse y emitirse una nueva. ProSign no puede recuperarlo.

06 Endpoints disponibles

Servidor base de producción: `https://cert.autosign.cl`

Crear un proceso de firma

POST `/v1/signing-processes`

Cuerpo en formato `multipart/form-data` con dos partes:

- **document**: archivo PDF a firmar (máximo 25 MB y 50 páginas).
- **metadata**: cadena JSON con la configuración del proceso.

Estructura del objeto `metadata`:

```json
{
  "external_ref": "cliente-orden-2026-00123",
  "webhook_url": "https://cliente.cl/hooks/firma",
  "webhook_secret": "opcional; si se omite, ProSign genera uno",
  "expires_in_hours": 168,
  "biometric_url_ttl_hours": 24,
  "signers": [
    {
      "rut": "12345678-9",
      "full_name": "Juan Pérez González",
      "email": "juan.perez@empresa.cl",
      "phone": "+56912345678",
      "signature_role": "vendedor",
      "order_index": 1,
      "represented_name": "Comercial Andes SpA",
      "represented_rut": "76123456-7"
    }
  ]
}
```

CAMPO	TIPO	OBLIGATORIO	DESCRIPCIÓN
`external_ref`	string	No	Identificador del cliente. Único por API key.
`webhook_url`	string	Sí	URL HTTPS donde se entregan los eventos.

`webhook_secret`	string	No	Secreto HMAC. Si se omite, ProSign genera uno y lo retorna en la respuesta.
`expires_in_hours`	int	No	Vigencia del proceso. Por defecto 168 (siete días). Máximo 720.
`biometric_url_ttl_hours`	int	No	Vigencia de cada URL biométrica. Por defecto 24. Máximo 168.
`signers`	array	Sí	De uno a veinte firmantes. El orden del array define el orden de firma.

Respuesta exitosa `201 Created`:

```json
{
  "process_id": "p_2VqK7m4nLxJ8FaB1CdEfGh",
  "status": "pending_biometrics",
  "created_at": "2026-05-29T14:32:11Z",
  "webhook_secret": "Rk5L_aB1cDeF2gH3iJ4k…"
}
```

Inmediatamente después ProSign emite el webhook `process.created` que contiene la URL biométrica de cada firmante.

Consultar el estado de un proceso

`GET` `/v1/signing-processes/{process_id}`

Devuelve el estado completo del proceso, incluyendo todos los firmantes con sus estados de verificación biométrica, firma, fechas y URLs vigentes.

Descargar el documento

`GET` `/v1/signing-processes/{process_id}/document`

Retorna el PDF en la etapa más avanzada disponible. El header `X-Document-Stage` indica:

- `source`: PDF original sin modificar.
- `assembled`: PDF previo a la firma de eCert (documento + bloques + comprobantes biométricos).
- `signed`: PDF final firmado por todos los firmantes.

Cancelar un proceso

`POST` `/v1/signing-processes/{process_id}/cancel`

Permitido únicamente si ningún firmante ha firmado. Responde `204 No Content` y emite el webhook `process.cancelled`.

Consultar un firmante específico

`GET` `/v1/signing-processes/{process_id}/signers/{signer_id}`

Regenerar URL biométrica

`POST` `/v1/signing-processes/{process_id}/signers/{signer_id}/regenerate-biometric-url`

Resetea el contador de intentos del firmante y emite una nueva URL. Útil cuando la URL anterior expira o el firmante queda bloqueado por exceder los intentos permitidos.

Auditoría de webhooks

`GET` `/v1/signing-processes/{process_id}/webhook-deliveries`

`POST` `/v1/signing-processes/{process_id}/webhook-deliveries/{delivery_id}/redeliver`

El primer endpoint lista todas las entregas de webhook intentadas para el proceso, con su estado actual y los códigos HTTP devueltos. El segundo fuerza el reenvío inmediato de una entrega puntual.

07 Eventos de webhook

Cada evento se entrega como una petición POST con cuerpo JSON. El conjunto de headers es siempre el mismo:

```http
Content-Type: application/json
X-CertMs-Event: process.created
X-CertMs-Event-Id: evt_8KpL3aB1cDeF2gH3iJ4k5L
X-CertMs-Delivery: wd_4a5b6c7d8e9f0a1b
X-CertMs-Signature: t=1748522103,v1=4f3a…
```

EVENTO	SIGNIFICADO
`process.created`	El proceso fue aceptado. Contiene las URLs biométricas por firmante.
`biometric.completed`	Un firmante completó su verificación biométrica satisfactoriamente.
`biometric.failed`	Un firmante agotó sus intentos (por defecto tres) y quedó bloqueado.
`biometric.url_regenerated`	El cliente o un operador ProSign generó una nueva URL biométrica para un firmante.
`biometric.url_expired`	La URL biométrica de un firmante expiró sin uso.
`process.ready_to_sign`	Todos los firmantes verificaron biométricamente. El documento fue ensamblado y subido a eCert.
`signer.ready_to_sign`	eCert habilitó la firma del siguiente firmante en la cadena secuencial.
`signer.signed`	Un firmante completó su firma digital.
`signer.rejected`	Un firmante rechazó la firma en eCert.
`process.completed`	Todos los firmantes firmaron. El PDF final está disponible para descargar.
`process.rejected`	Un firmante rechazó. El proceso queda en estado terminal de rechazo.

`process.cancelled`	El proceso fue cancelado por el cliente.
`process.expired`	El proceso superó su fecha de expiración sin completarse.
`process.failed`	Error irrecuperable durante el procesamiento.

Deduplicación

El cliente debe deduplicar los eventos utilizando el header `X-CertMs-Event-Id`, que es estable entre reintentos. El header `X-CertMs-Delivery` cambia en cada intento de entrega y sirve únicamente como identificador de la entrega.

08 Verificación de firmas y reintentos

Formato de la firma

El header `X-CertMs-Signature` contiene dos valores separados por coma:

```http
X-CertMs-Signature: t=1748522103,v1=4f3a8c0b7d2e9f1a…
```

- `t`: timestamp Unix (UTC) en que se generó la firma.
- `v1`: HMAC-SHA256 del valor `"{t}.{cuerpo_crudo}"`, codificado en hexadecimal. La clave es el `webhook_secret` del proceso.

Implementación de referencia

```python
import hmac
import hashlib
import time


def verify_webhook(raw_body: bytes, signature_header: str, secret: str) -> bool:
    """Valida un webhook entrante de ProSign FEA."""
    try:
        parts = dict(piece.split("=", 1) for piece in signature_header.split(","))
        timestamp = parts["t"]
        received = parts["v1"]
    except (KeyError, ValueError):
        return False

    payload = (timestamp + ".").encode("utf-8") + raw_body
    expected = hmac.new(
        secret.encode("utf-8"),
        payload,
        hashlib.sha256,
    ).hexdigest()

    if not hmac.compare_digest(expected, received):
        return False

    # Rechazar firmas con más de 5 minutos de antigüedad para evitar replays.
    return abs(int(time.time()) - int(timestamp)) <= 300
```

Política de reintentos

Si el endpoint del cliente responde con un código fuera del rango 200-299, o si se produce un timeout (10 segundos), ProSign reintenta la entrega con backoff exponencial:

INTENTO 1	INTENTO 2	INTENTO 3	INTENTO 4
30 s	**2 min**	**10 min**	**1 h**

INTENTO 5	INTENTO 6-8	TOTAL	TIMEOUT
6 h	**24 h c/u**	**~3 días**	**10 s**

Si tras ocho intentos la entrega sigue sin tener éxito, el estado pasa a `giving_up`. El cliente puede consultar `GET /v1/signing-processes/{id}/webhook-deliveries` para auditar y, si corresponde, forzar un reintento manual con el endpoint `redeliver`.

09 Manejo de errores

Todas las respuestas de error siguen el mismo formato:

```json
{
  "error": {
    "code": "validation_error",
    "message": "signers[0].rut tiene formato inválido (use NNNNNNNN-V, sin puntos).",
    "details": { "field": "signers[0].rut" },
    "request_id": "req_8a9b0c1d2e3f"
  }
}
```

CÓDIGO	HTTP	SIGNIFICADO
`unauthorized`	401	API key ausente, inválida o revocada.
`forbidden`	403	API key válida pero sin permisos sobre el recurso.
`validation_error`	422	El cuerpo o los parámetros no superaron la validación.
`idempotency_conflict`	409	Mismo `Idempotency-Key` reutilizado con cuerpo distinto.
`not_found`	404	El recurso solicitado no existe para esta API key.
`process_not_cancellable`	409	Cancelación rechazada porque al menos un firmante ya firmó.
`biometric_expired`	409	Operación rechazada porque la URL biométrica ya expiró.
`biometric_blocked`	409	El firmante quedó bloqueado tras tres intentos fallidos.
`document_too_large`	413	El PDF supera los 25 MB o las 50 páginas.
`quota_exceeded`	429	Se excedió la cuota de peticiones por minuto. Incluye `Retry-After`.
`ecert_unavailable`	502	eCert no respondió tras los reintentos internos.
`internal_error`	500	Error inesperado del servicio. El `request_id` es la referencia a entregar al equipo de soporte.

10 Restricciones operativas

RECURSO	LÍMITE
Tamaño máximo del PDF	25 MB
Páginas máximas del PDF	50
Firmantes por proceso	1 a 20
Vigencia del proceso	1 a 720 horas (por defecto 168)
Vigencia de cada URL biométrica	1 a 168 horas (por defecto 24)
Intentos biométricos antes de bloqueo	3
Modo de firma	Secuencial (eCert no soporta paralelo)
Cuota por defecto	60 peticiones de escritura por minuto, 600 de lectura
Timeout de webhook	10 segundos por intento

11 Ejemplo completo de integración

Crear un proceso (curl)

```bash
curl -X POST https://cert.autosign.cl/v1/signing-processes \
  -H "Authorization: Bearer cms_live_xxxxxxxxxxxx_yyy…" \
  -F "document=@contrato.pdf" \
  -F 'metadata={
    "external_ref": "venta-2026-0421",
    "webhook_url": "https://api.empresa.cl/hooks/autosign",
    "signers": [
      {
        "rut": "12345678-9",
        "full_name": "Juan Pérez González",
        "email": "juan.perez@empresa.cl",
        "signature_role": "vendedor",
        "order_index": 1
      },
      {
        "rut": "98765432-1",
        "full_name": "María García Soto",
        "email": "maria.garcia@cliente.cl",
        "signature_role": "comprador",
        "order_index": 2
      }
    ]
  }'
```

Recibir y procesar un webhook (Python con Flask)

```python
from flask import Flask, request, abort
import hmac
import hashlib
import time

app = Flask(__name__)
WEBHOOK_SECRET = "el-secreto-recibido-al-crear-el-proceso"


def verify(raw_body: bytes, signature_header: str) -> bool:
    try:
        parts = dict(p.split("=", 1) for p in signature_header.split(","))
        timestamp, received = parts["t"], parts["v1"]
    except (KeyError, ValueError):
        return False
    payload = (timestamp + ".").encode() + raw_body
    expected = hmac.new(
        WEBHOOK_SECRET.encode(), payload, hashlib.sha256
    ).hexdigest()
    if not hmac.compare_digest(expected, received):
        return False
    return abs(int(time.time()) - int(timestamp)) <= 300


@app.post("/hooks/autosign")
def autosign_webhook():
    raw = request.get_data()
    if not verify(raw, request.headers.get("X-CertMs-Signature", "")):
        abort(401)

    event = request.headers["X-CertMs-Event"]
    event_id = request.headers["X-CertMs-Event-Id"]
    payload = request.get_json()

    # Deduplicar por event_id en una tabla propia.
    if already_processed(event_id):
        return "", 200

    if event == "process.created":
        guardar_urls_biometricas(payload["signers"])
    elif event == "biometric.completed":
        marcar_firmante_verificado(payload["signer"])
    elif event == "process.completed":
        descargar_pdf_final(payload["process"]["process_id"])
    elif event == "process.rejected":
        notificar_rechazo(payload["process"], payload.get("reason"))

    registrar_procesado(event_id)
    return "", 200
```

Descargar el PDF firmado

```bash
curl -O -J \
  -H "Authorization: Bearer cms_live_xxxxxxxxxxxx_yyy…" \
  https://cert.autosign.cl/v1/signing-processes/p_2VqK7m4nLxJ8/document
```

12 Soporte

Para reportar incidentes, solicitar nuevas API keys o resolver dudas técnicas, contactar al equipo de ProSign indicando, cuando corresponda, el `process_id`, el `event_id` o el `request_id` involucrado.

CANAL	DIRECCIÓN
Correo de soporte técnico	integraciones@autosign.cl
Estado del servicio	https://status.autosign.cl
Sitio principal	https://autosign.cl