El certificado digital

Qué es el certificado digital, en qué formatos enviarlo a la API, cómo verificarlo y cómo protegerlo.

Anonymous · 05/11/2026

El certificado digital

El **certificado digital** (o firma electrónica) es lo que permite firmar los documentos. Pertenece a una **persona**, no a la empresa: es la persona que el SII autorizó para firmar los documentos de la empresa emisora.

Se usa en tres momentos:

1. Para **firmar el DTE** al construirlo.
2. Para **firmar el sobre** que se envía al SII.
3. Para **autenticarse en el SII** al enviar y al consultar el estado.

El RUT de la firma no es el de la empresa

El certificado tiene el RUT de la persona (por ejemplo `11222333-9`), no el de la empresa (`88888888-8`). En el sobre aparece como `RutEnvia`: quien envía el documento en nombre de la empresa. Esa persona debe estar autorizada en el SII para la empresa emisora.

Formatos que acepta la API

En las operaciones de LibreDTE el certificado va en un objeto `certificate`, en **una** de estas dos formas. En todos los ejemplos del curso usamos PEM, porque es el mismo formato que usan los recursos de consulta al SII.

Opción 1: certificado y llave privada en PEM (la que usamos en este curso)

```
"certificate": {
    "certificate": "-----BEGIN CERTIFICATE-----\nMIIEcTCCA1mgAwIBAgIBRTAN...\n-----END
CERTIFICATE-----\n",
    "privateKey": "-----BEGIN PRIVATE KEY-----\nMIIEvgIBADANBgkqhkiG9w0B...\n-----END
PRIVATE KEY-----\n"
}
```

- `certificate`: el certificado público en formato PEM (texto, **sin** base64).
- `privateKey`: la llave privada en formato PEM (texto, **sin** base64).

Normalmente el certificado se entrega como archivo `.p12` o `.pfx`. Para obtener los PEM a partir de ese archivo, cárgalo con la operación `system/certificate/loader/load`: envía el archivo en base64 y su contraseña, como en la opción 2:

```
curl -X POST https://app.apigateway.cl/api/v2/libredte/system/certificate/loader/load
\
  -H "Authorization: Token $API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "parameters": {
        "certificate": {
            "data": "MIIKkQIBAzCCClcGCSqGSIb3DQEHAaCCCkgEggpEMIIKQDCCBPcGCSqG...",
            "password": "contraseña_del_archivo"
        }
    }
}'
```

La respuesta trae el certificado en `cert` y la llave privada en `pkey`, los dos en PEM:

```
{
    "data": {
        "id": "11222333-9",
        "name": "Daniel Bot",
        "cert": "-----BEGIN CERTIFICATE-----\nMIIEcTCCA1mgAwIBAgIBRTAN...\n-----END
CERTIFICATE-----\n",
        "pkey": "-----BEGIN PRIVATE KEY-----\nMIIEvgIBADANBgkqhkiG9w0B...\n-----END
PRIVATE KEY-----\n",
        "...": "..."
    }
}
```

Guarda `cert` como `cert.pem` y `pkey` como `key.pem`. Haz esta conversión una sola vez y guarda los PEM como secretos: no la repitas en cada emisión.

Los saltos de línea

En JSON, los saltos de línea de un PEM se escriben como `\n`. Si armas el JSON con un lenguaje de programación (leyendo el archivo y serializándolo), esto ocurre solo. Si lo pegas a mano, revisa que no quede todo en una línea sin `\n`.

Opción 2: archivo `.p12` / `.pfx` en base64

```
"certificate": {
    "data": "MIIKkQIBAzCCClcGCSqGSIb3DQEHAaCCCkgEggpEMIIKQDCCBPcGCSqG...",
    "password": "contraseña_del_archivo"
}
```

- `data`: el archivo `.p12` o `.pfx` completo, codificado en base64 en una sola línea.
- `password`: la contraseña del archivo.

```
base64 -i firma.p12 | tr -d '\n'
```

No es el mismo formato que en las consultas al SII

Los recursos de consulta al SII (por ejemplo, los de folios en `/api/v2/sii/dte/caf/...`) reciben el certificado en el bloque `auth.cert` con los campos `cert-data` y `pkey-data` (o `file-data` y `file-pass`). Los recursos de LibreDTE (`/api/v2/libredte/...`) lo reciben en `certificate` con `certificate` y `privateKey` (o `data` y `password`). Son el mismo certificado, con distinto nombre de campos.

Recurso	Campo	PEM	`.p12` en base64
`/api/v2/libredte/...`	`certificate`	`certificate` + `privateKey`	`data` + `password`
`/api/v2/sii/...`	`auth.cert`	`cert-data` + `pkey-data`	`file-data` + `file-pass`

Verificar el certificado

Antes de usarlo para firmar, puedes comprobar que la API lo lee bien y cuándo vence:

```
curl -X POST https://app.apigateway.cl/api/v2/libredte/system/certificate/loader/load
\
  -H "Authorization: Token $API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "parameters": {
        "certificate": {
            "certificate": "-----BEGIN CERTIFICATE-----\nMIIG...\n-----END
CERTIFICATE-----",
            "privateKey": "-----BEGIN PRIVATE KEY-----\nMIIE...\n-----END PRIVATE KEY-
----"
        }
    }
}'
```

La respuesta trae, entre otros datos:

```
{
    "data": {
        "id": "11222333-9",
        "name": "Daniel Bot",
        "email": "daniel.bot@example.com",
        "from": "2026-10-29T11:38:23",
```

```
        "to": "2027-10-29T11:38:23",
        "expirationDays": 364,
        "isActive": true,
        "issuer": "Derafu Test Certificate Authority"
    }
}
```

Campo	Qué revisar
`id`	El RUT de la persona dueña del certificado.
`to` / `expirationDays`	Cuándo vence. Un certificado vencido no puede firmar.
`isActive`	Debe ser `true`.

Si falta un dato, la API responde con un error como:

```
{
    "code": 422,
    "detail": "Se requieren los datos del certificado (data y password) o sus llaves
(certificate y privateKey)."
}
```

Seguridad

El certificado permite firmar documentos tributarios **a nombre de la empresa**. Trátalo como una contraseña:

- Guárdalo cifrado o en un almacén de secretos (variables de entorno del servidor, un vault, el gestor de secretos de tu nube).
- Nunca lo subas a un repositorio ni lo envíes desde el navegador o una app móvil: las llamadas a la API deben salir de **tu servidor**.
- No guardes en tus logs las solicitudes ni las respuestas completas: incluyen el certificado.
- Controla su vencimiento y renuévalo antes de que expire.

Un certificado de prueba

Mientras sigues el curso puedes construir documentos con un certificado de prueba: la API firmará el XML igual. Lo que no podrás hacer sin un certificado real autorizado es **enviar al SII**, porque el SII rechazará la autenticación.

La API genera uno con esta operación:

```
curl -X POST
https://app.apigateway.cl/api/v2/libredte/billing/trading_parties/mandatario_manager/c
reateFakeCertificate \
```

```
  -H "Authorization: Token $API_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "parameters": {
        "mandatario": {
            "run": "11222333-9",
            "nombre": "Daniel Bot",
            "email": "daniel.bot@example.com"
        }
    }
}'
```

La respuesta (resumida) trae el certificado y su llave privada en PEM, válidos por un año:

```
{
    "data": {
        "id": "11222333-9",
        "name": "Daniel Bot",
        "email": "daniel.bot@example.com",
        "from": "2026-11-04T14:07:03",
        "to": "2027-11-04T14:07:03",
        "issuer": "Derafu Test Certificate Authority",
        "cert": "-----BEGIN CERTIFICATE-----\nMIIE...\n-----END CERTIFICATE-----\n",
        "pkey": "-----BEGIN PRIVATE KEY-----\nMIIE...\n-----END PRIVATE KEY-----\n",
        "...": "..."
    }
}
```

Guarda `cert` como `cert.pem` y `pkey` como `key.pem`, y úsalos como cualquier certificado en PEM: `"certificate": {"certificate": <cert>, "privateKey": <pkey>}`.

Última actualización el 05/11/2026