Factura de principio a fin

Un programa completo en Python que construye, valida, envía al SII, consulta el estado y genera el PDF de una factura electrónica.

Anonymous · 05/11/2026

Factura de principio a fin

En este caso juntamos los seis pasos en un solo programa. Emite una factura afecta (`33`) en el **ambiente de certificación** del SII.

Lo que necesitas

Archivo o dato	Dónde se explica
Token de API Gateway	Primeros pasos
Certificado digital en PEM: `cert.pem` y `key.pem`	El certificado digital
CAF de facturas (`33`) del ambiente de certificación	Folios y CAF
Fecha de la resolución en certificación	Resolución de autorización

¿Aún no tienes CAF de certificación?

Puedes probar los pasos 1, 2, 3 y 6 con un CAF y un certificado de prueba. Te explicamos cómo en Probar sin el SII.

El programa

Instala `requests` (`pip install requests`) y define la variable de entorno `API_TOKEN`. Deja en la misma carpeta los archivos `cert.pem`, `key.pem` y `caf_33.xml`.

```
"""Emite una factura electrónica en el ambiente de certificación del SII."""
import base64
import os
import time

import requests

API_URL = 'https://app.apigateway.cl/api/v2/libredte'
HEADERS = {
```

```
    'Authorization': f'Token {os.environ["API_TOKEN"]}',
    'Content-Type': 'application/json',
}

AMBIENTE = 1  # 1: certificación, 0: producción.
SII_RUT = '60803000-K'

# Estados del envío que ya no cambian (ver el paso 5). Los demás significan
# que el SII sigue procesando, o que la consulta falló y hay que repetirla.
ESTADOS_FINALES = {'EPR', 'LOK', 'RPR', 'RLV', 'RCH', 'RCO', 'RFR', 'RSC', 'RCT',
'RPT', 'VOF'}
ESTADOS_ACEPTADOS = {'EPR', 'LOK', 'RPR', 'RLV'}


def libredte(operacion: str, parameters: dict) -> dict | list | None:
    """Ejecuta una operación de LibreDTE y entrega el contenido de `data`."""
    response = requests.post(
        f'{API_URL}/{operacion}',
        json={'parameters': parameters},
        headers=HEADERS,
        timeout=120,
    )
    if response.status_code != 200:
        try:
            detalle = response.json().get('detail', response.text)
        except ValueError:  # La respuesta de error no viene en JSON.
            detalle = response.text
        raise RuntimeError(f'{operacion}: HTTP {response.status_code}: {detalle}')
    return response.json()['data']


def leer_texto(ruta: str) -> str:
    """Lee un archivo de texto, como el certificado o la llave en PEM."""
    with open(ruta, encoding='ascii') as archivo:
        return archivo.read()


def leer_base64(ruta: str) -> str:
    """Lee un archivo y lo entrega codificado en base64."""
    with open(ruta, 'rb') as archivo:
        return base64.b64encode(archivo.read()).decode()


def guardar_base64(ruta: str, contenido: str) -> None:
    """Decodifica un base64 y lo guarda como archivo, sin tocar sus bytes."""
    with open(ruta, 'wb') as archivo:
        archivo.write(base64.b64decode(contenido))


# Datos que normalmente vienen de tu base de datos.
CERTIFICADO = {
```

```
    'certificate': leer_texto('cert.pem'),
    'privateKey': leer_texto('key.pem'),
}
CAF = leer_base64('caf_33.xml')
EMISOR = {
    'rut': '88888888-8',
    'razon_social': 'Mi Empresa SpA',
    'autorizacion_dte': {
        'fecha_resolucion': '2025-11-20',  # Fecha de autorización en certificación.
        'numero_resolucion': 0,            # En certificación siempre es 0.
    },
}
FOLIO = 1  # El siguiente folio libre del CAF, según tu control de folios.

# Paso 1: construir el DTE. Primero, los datos del documento.
datos = {
    'Encabezado': {
        'IdDoc': {'TipoDTE': 33, 'Folio': FOLIO, 'FmaPago': 1},
        'Emisor': {
            'RUTEmisor': EMISOR['rut'],
            'RznSoc': EMISOR['razon_social'],
            'GiroEmis': 'Venta de software',
            'Acteco': 620100,
            'DirOrigen': 'Av. Siempre Viva 742',
            'CmnaOrigen': 'Providencia',
        },
        'Receptor': {
            'RUTRecep': '77777777-7',
            'RznSocRecep': 'Cliente Ejemplo Ltda.',
            'GiroRecep': 'Comercio',
            'DirRecep': 'Calle Falsa 123',
            'CmnaRecep': 'Santiago',
        },
    },
    'Detalle': [
        {'NmbItem': 'Servicio de desarrollo', 'QtyItem': 1, 'PrcItem': 100000},
        {'NmbItem': 'Soporte técnico (horas)', 'QtyItem': 3, 'PrcItem': 25000},
    ],
}

# Con los datos, el CAF y el certificado se obtiene el DTE timbrado y firmado.
dte = libredte('billing/document/builder/build', {
    'bag': {'parsedData': datos, 'caf': CAF, 'certificate': CERTIFICADO},
})
xml_dte = dte['document_xml']
guardar_base64(f'dte_33_{FOLIO}.xml', xml_dte)
totales = dte['document_normalized']['Encabezado']['Totales']
print('Documento construido:', dte['document_id'], 'total', totales['MntTotal'])

# Paso 2: validar el esquema antes de enviar.
libredte('billing/document/validator/validateSchema', {'source': xml_dte})
```

```
# Paso 3: armar el sobre dirigido al SII.
sobre = libredte('billing/document/dispatcher/create', {
    'bag': {
        'xmlDocument': xml_dte,
        'certificate': CERTIFICADO,
        'emisor': EMISOR,
        'receptor': {'rut': SII_RUT},
    },
})
guardar_base64(f'sobre_sii_33_{FOLIO}.xml', sobre['xml'])

# Paso 4: enviar al SII.
request_sii = {'certificate': CERTIFICADO, 'options': {'environment': AMBIENTE}}
envio = libredte('billing/integration/sii_dte/sendXmlDocument', {
    'request': request_sii,
    'doc': sobre['xml'],
    'company': EMISOR['rut'],
})
track_id = envio['track_id']
print('Enviado al SII, Track ID:', track_id)  # ¡Guárdalo en tu base de datos!

# Paso 5: consultar el estado hasta tener un resultado final.
estado = None
for espera in [30, 60, 120, 300, 600]:
    time.sleep(espera)
    try:
        estado = libredte('billing/integration/sii_dte/checkXmlDocumentSentStatus', {
            'request': request_sii,
            'trackId': track_id,
            'company': EMISOR['rut'],
        })
    except RuntimeError as error:
        # La consulta falló (por ejemplo, el SII no respondió): se reintenta.
        print('Falló la consulta, se reintenta:', error)
        continue
    print('Estado:', estado['status'], estado['description'])
    if estado['status'] in ESTADOS_FINALES:
        break

resumen = estado['resume'] if estado else {}
if not estado or estado['status'] not in ESTADOS_FINALES:
    print('Sin estado final todavía. Vuelve a consultar más tarde con el Track ID',
track_id)
elif estado['status'] == 'RPT':
    print('El SII ya había recibido este sobre: consulta el estado con el Track ID del
primer envío.')
elif estado['status'] == 'VOF':
    print('El SII no encontró el archivo: envía de nuevo el mismo sobre (no lo
construyas de nuevo).')
elif estado['status'] in ESTADOS_ACEPTADOS and resumen.get('rejected', 0) == 0:
    resultado = 'aceptado con reparos' if resumen.get('repairs') else 'aceptado'
```

```
    print('Resultado:', resultado)

    # Paso 6: generar el PDF.
    pdf = libredte('billing/document/renderer/render', {
        'bag': {
            'xmlDocument': xml_dte,
            'emisor': EMISOR,  # Con la resolución, que se imprime en el PDF.
            'options': {'renderer': {'format': 'pdf', 'renderings': {'tributaria': 1,
'cedible': 1}}},
        },
    })
    for archivo in pdf['renderings']:
        guardar_base64(archivo['filename'], archivo['content'])
        print('PDF generado:', archivo['filename'])
else:
    # Para saber el motivo, pide el detalle por correo (ver el paso 5).
    print('El documento fue rechazado:', estado['status'], estado['description'])
```

Probar sin el SII

Si todavía no tienes un CAF y un certificado reales del ambiente de certificación, puedes probar los pasos 1, 2, 3 y 6 con datos de prueba. Los pasos 4 y 5 no funcionan con ellos: el SII rechaza la autenticación y el programa se detiene en el envío.

1. **El certificado:** genera uno de prueba (ver Un certificado de prueba) y guarda `cert` como `cert.pem` y `pkey` como `key.pem`.
2. **El CAF:** genera uno de prueba y guárdalo como `caf_33.xml`. Ejecuta esto una vez, con las funciones `libredte()` y `guardar_base64()` del programa. El CAF viene en base64 en el campo `xml`, y `guardar_base64()` lo decodifica al guardarlo:

```
caf = libredte('billing/identifier/caf_faker/create', {
    'emisor': {'rut': '88888888-8', 'razon_social': 'Mi Empresa SpA'},
    'codigoDocumento': 33,
    'folioDesde': 1,
    'folioHasta': 100,
})
guardar_base64('caf_33.xml', caf['xml'])
```

3. **El programa:** borra todo lo que sigue a `# Paso 4: enviar al SII.` y pon en su lugar solo el paso 6, sin la condición de que el SII haya aceptado:

```
# Paso 6: generar el PDF, sin pasar por el SII.
pdf = libredte('billing/document/renderer/render', {
    'bag': {
        'xmlDocument': xml_dte,
        'emisor': EMISOR,
        'options': {'renderer': {'format': 'pdf', 'renderings': {'tributaria': 1,
```

```
'cedible': 1}}},
    },
})
for archivo in pdf['renderings']:
    guardar_base64(archivo['filename'], archivo['content'])
    print('PDF generado:', archivo['filename'])
```

El documento que obtienes no tiene validez tributaria: el CAF es ficticio. Sirve para revisar los datos, los totales y el PDF.

Qué hace, en orden

Paso	Llamada	Qué guarda
1	`builder/build`	`dte_33_1.xml`: el documento timbrado y firmado
2	`validator/validateSchema`	(nada; falla si el XML no es válido)
3	`dispatcher/create` con `receptor` del SII	`sobre_sii_33_1.xml`
4	`sii_dte/sendXmlDocument`	El Track ID
5	`sii_dte/checkXmlDocumentSentStatus`	El estado final
6	`renderer/render`	Los PDF tributario y cedible

Para llevarlo a tu sistema

Este programa hace todo seguido para que lo entiendas de una vez. En un sistema real conviene separarlo:

1. **Al emitir** (cuando el usuario confirma la venta): reservar el folio, construir, validar, armar el sobre, enviar y guardar el Track ID. Responder al usuario.
2. **En segundo plano** (un proceso programado): consultar el estado de los envíos pendientes y actualizar cada documento.
3. **Cuando el documento es aceptado**: generar el PDF y entregarlo al cliente. Si el cliente es una empresa que emite documentos electrónicos, enviarle además el XML a su correo de intercambio (ver Entregar el documento al receptor).

Así ningún usuario espera al SII, y si algo falla en un paso, sabes exactamente desde dónde continuar.

El mismo flujo en otros lenguajes

Son solo solicitudes HTTP con JSON. Puedes traducir este programa a PHP, JavaScript, Java, C# o cualquier otro lenguaje usando su cliente HTTP habitual. Si usas un asistente de IA, cópiale esta página (ver Cómo usar esta documentación).

Última actualización el 05/11/2026