vessi
boleta fácil

Documentación
API Boletas

Mayo 2023

Índice

Control de cambios

Fecha	Autor	Resumen
2023-Feb-07	JO	Borrador base
2023-Feb-14	JO	Primera versión
2023-May-18	JO	Añade esquema de respuesta
2023-May-19	JO	Añade campo uniqueId

Visión general del producto

Descripción

Vessi ofrece mediante este producto la emisión de documentos tributarios electrónicos (DTEs) mediante una API, lo cual le permite a sus clientes automatizar procesos informáticos que la requieran. La API se encuentra alojada en Amazon Web Services (AWS) lo que le otorga robustez y rapidez, aún frente a un uso intensivo.

Autenticación

Vessi le entregará a cada cliente una credencial AWS Identity and Access Management (IAM) de acceso a la API, la cual consta de dos llaves: una AccessKey y una SecretKey que deberán ser almacenadas apropiadamente y no ser expuestas públicamente. Ambas llaves se deben usar para acceder mediante el tipo de autenticación AWS Signature.

Acceso

El producto dispone de dos etapas de implementación: una para pruebas para software aún en desarrollo y otra para software productivo. En ambas etapas se debe usar la misma credencial proveída por Vessi.

Etapa de pruebas

Esta etapa es para pruebas de software. Los DTEs solicitados no serán reales y no tendrán ningún efecto tributario real.

URL: https://qgd3pr8an8.execute-api.us-west-2.amazonaws.com/qa

Etapa de producción

Esta etapa es para software de producción. Los DTEs solicitados serán reales y tendrán efectos tributarios reales. No debe usarse para hacer pruebas.

URL: https://c0hocia0ua.execute-api.us-west-2.amazonaws.com/prod

Recursos de la API

La API ofrece un sólo recurso, detallado a continuación:

`/api/dte/documentos/generar/ext/{rut-emisor}`

Recibe un POST y permite generar un DTE, cuya especificación se detalla en payload JSON de la petición. En la URL, `{rut-emisor}` deberá reemplazarse por el RUT del cliente (que es el emisor del DTE). Dependiendo del tipo de DTE, el payload deberá tener cierta estructura, detallada más adelante.

Acceso

Como se mencionó previamente, AWS Signature es el método de autenticación que permite acceder a la API. Para poder usarlo, se deben seguir los siguientes 2 pasos.

Llamada a STS

Se debe tomar la Access key y la Secret access key entregadas por Vessi, y usar el AWS Security Token Service (STS) para generar:

- Un par temporal de Access key y Secret access key.
- Token temporal de acceso.

La documentación oficial de STS se encuentra disponible en: https://docs.aws.amazon.com/STS/latest/APIReference/welcome.html.

De forma predeterminada, las llaves y token generados por STS expiran en 12 horas. Es posible reusarlas hasta que venzan. Por eficiencia, se recomienda cachearlas a fin de reusarlas hasta que ya no sea posible. En caso de falla, ahí hacer nueva llamada a STS.

Firmado de petición

El siguiente paso es firmar la petición a la API antes de realizarla, usando las llaves temporales y token temporal generados por STS en el paso anterior. La documentación para calcular manualmente la firma y armar una petición (encabezados HTTP, etc.) se encuentra aquí: https://docs.aws.amazon.com/apigateway/api-reference/signing-requests/

La firma también depende del tiempo local[1] actual, por lo que si está demasiado desfasado del tiempo real, se generará una firma incorrecta.

Servicio de tiempo

Para que el consumidor de la API se asegure de tener una hora correcta, Vessi ofrece un servicio simple que permite obtenerla en https://zehhq8aey7.execute-api.us-west-2.amazonaws.com/dev/hora.

El servicio no requiere autenticación y entrega un JSON conteniendo el campo `requestTimeEpoch`, que es el tiempo Epoch en milisegundos. Igualmente el cliente es libre de usar otro proveedor externo de tiempo (por ejemplo, usar un cliente NTP).

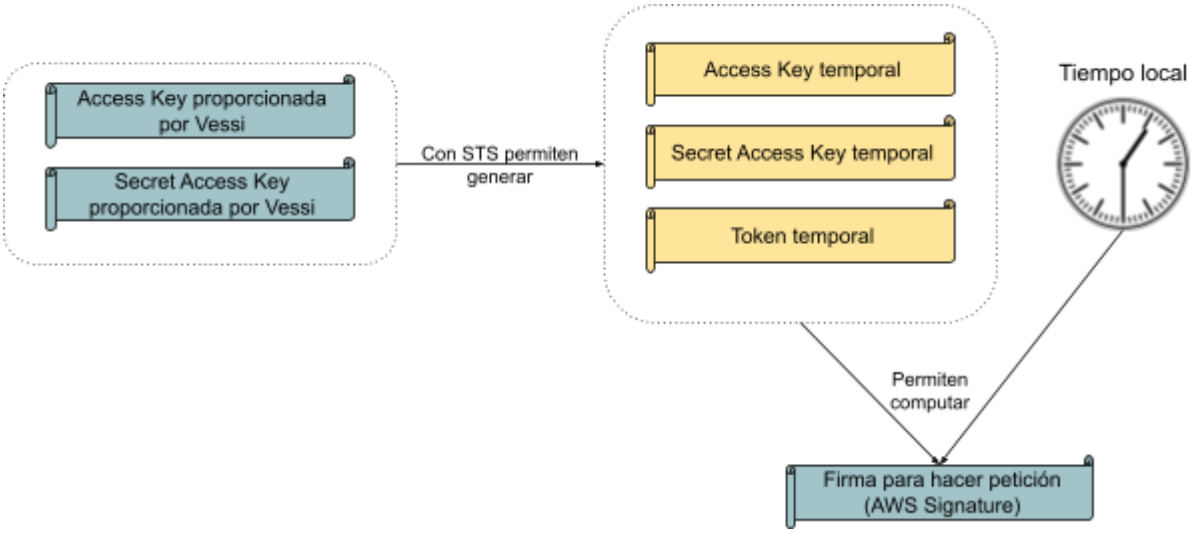
Hay dos estrategias para usar este servicio cada vez que se desee generar un DTE:

- Estrategia “naïve”. Llamar al servicio de hora, establecer localmente la hora que entregue y ahí hacer petición a la API.

[1] Local a la máquina en que se aloja el consumidor. No local geográfico.

- Estrategia eficiente. Hacer la petición a la API y sólo si falla: llamar al servicio de hora, reintentar una vez más con la hora corregida y si falla nuevamente, desistir.

Esquema resumen



Emisión de DTEs

El payload JSON necesario para el recurso de emisión de DTE de la API es distinto para cada tipo de DTE. Las estructuras de los JSONs para cada caso, se detallan a continuación junto a breves descripciones de sus campos.

Boletas

Su esquema JSON es:

```
{"Encabezado": {"IdDoc": {"TipoDTE":TIPODTE, "Folio":1},
"Emisor":{"RUTEmisor":RUTEMISOR"}, "Receptor":{"RUTRecep":RUTRECEP,
"RznSocRecep":RAZONSOCRECEP, "DirRecep":DIRRECEP}},
"Detalle":[{"NmbItem":NOMBREITEM, "QtyItem":CANT, "PrcItem":PRECIO,
"UnmdItem":UNIDADMED},...], "pago":TIPOPAGO, "extData": {"uniqueId":IDUNICO}}
```

Valor	Tipo	Descripción	Opcional
`TIPOPAGO`	`Cadena`	Tipo de medio de pago (p. ej. “Crédito” o “Débito”).	✓
Estructura `Encabezado`			
`TIPODTE`	`Entero`	Acepta los valores: 39 (Boleta) o 41 (Boleta exenta).	
Estructura `Emisor`			
`RUTEMISOR`	`Cadena`	Rut con DV del emisor.	
Estructura `Receptor`			
`RUTRECEP`	`Cadena`	Rut con DV del receptor.	
`RAZONSOCRECEP`	`Cadena`	Razón social del receptor.	
`DIRRECEP`	`Cadena`	Dirección del receptor.	✓
Estructura `Detalle`			
`NOMBREITEM`	`Cadena`	Nombre del ítem.	
`CANT`	`Número`	Cantidad del ítem. Máximo 2 decimales.	
`PRECIO`	`Entero`	Precio del ítem.	
`UNIDADMED`	`Cadena`	Unidad de medida del ítem.	✓
Estructura `extData` Es opcional			
`IDUNICO`	`Cadena`	Identificador único al documento a generar. Si ya se ocupó antes, no se generará un nuevo documento y se devolverá el previo.	

El arreglo Detalle soporta múltiples ítemes. Para mayor detalle sobre todos los campos véase https://www.sii.cl/factura_electronica/factura_mercado/descripcion_formato.htm.

Facturas

Su esquema JSON es:

```
{"Encabezado": {"IdDoc": {"Folio":1 ,"TipoDTE":TIPODTE, "TermPagoGlosa":TERMPAGO,
"FchVenc":FECHAVENC, "FchEmis":FECHAEMIS, "FmaPago":FORMAPAGO}, "Emisor":
{"RUTEmisor":RUTEMISOR},
"Receptor":{"RUTRecep":RUTRECEP,"RznSocRecep":RAZONSOCRECEP,"DirRecep":DIRRECEP,"Cm
naRecep":COMUNARECEP,"Contacto":CONTACTORECEP,"GiroRecep":GIRORECEP,"CorreoRecep":C
ORREORECEP}},"Detalle":[{"NmbItem":NOMBREITEM,"QtyItem":CANT,"PrcItem":PRECIO,"Unmd
Item":UNIDADMED},...],"Referencia":[{"TpoDocRef":TIPODOCREF,"FolioRef":FOLIOREF,"Fc
hRef":FECHAREF,"RazonRef":RAZONREF,"NroLinRef":NUMEROLINREF},...], "extData":
{"uniqueId":IDUNICO}}
```

Valor	Tipo	Descripción	Opcional
TIPOPAGO	Cadena	Tipo de medio de pago (p. ej. “Crédito” o “Débito”).	✓
Estructura Encabezado			
TIPODTE	Entero	Acepta los valores: 33 (Factura) o 34 (Factura exenta).	
TERMPAGO	Cadena	Glosa de término de pago	✓
FORMAPAGO	Entero	Forma en que se pagará la factura. Acepta los valores 1 (Contado), 2 (Crédito), 3 (Sin costo).	✓
FECHAVENC	Cadena	Fecha en formato YYYY-MM-DD que indica fecha máxima en que se deberá pagar la factura. Incluir sólo si FORMAPAGO es Crédito.	✓
FECHAEMIS	Cadena	Fecha en formato YYYY-MM-DD que indica fecha en que la factura debe ser considerada para efectos contables.	
Estructura Emisor			
RUTEMISOR	Cadena	Rut con DV del emisor.	
Estructura Receptor			
RUTRECEP	Cadena	Rut con DV del receptor.	
RAZONSOCRECEP	Cadena	Razón social del receptor.	
DIRRECEP	Cadena	Dirección del receptor.	
COMUNARECEP	Cadena	Comuna de la dirección del receptor.	
CONTACTORECEP	Cadena	Información de contacto del receptor.	✓
GIRORECEP	Cadena	Giro del receptor.	
CORREORECEP	Cadena	E-mail del receptor.	✓
Estructura Detalle			
NOMBREITEM	Cadena	Nombre del ítem.	

CANT	Número	Cantidad del ítem. Máximo 2 decimales.	
PRECIO	Entero	Precio del ítem.	
UNIDADMED	Cadena	Unidad de medida del ítem.	✓
Estructura Referencia Es opcional			
TIPODOCREF	Entero	Código del tipo del documento referenciado.	
FOLIOREF	Entero	Folio del documento referenciado.	
FECHAREF	Cadena	Fecha en formato YYYY-MM-DD que indica fecha del documento referenciado. No debe ser mayor a la fecha actual.	
RAZONREF	Cadena	Razón de la referencia	✓
NUMEROLINREF	Entero	Posición de la referencia: 1 para la primera referencia, 2 para la segunda, etc.	
Estructura extData Es opcional			
IDUNICO	Cadena	Identificador único al documento a generar. Si ya se ocupó antes, no se generará un nuevo documento y se devolverá el previo.	

El arreglo Detalle soporta múltiples ítemes. Para mayor detalle sobre todos los campos véase https://www.sii.cl/factura_electronica/factura_mercado/descripcion_formato.htm.

Guía de despacho

Su esquema JSON es:

```
{"Encabezado":{"IdDoc":{"Folio":1,"TipoDTE":52,"TipoDespacho":TIPODESPACHO,"IndTraslado":INDICETRASLADO,"TermPagoGlosa":TERMINOPAGO},"Emisor":{"RUTEmisor":RUTEMISOR,"CdgSIISucur":CODIGOSUCURSAL},"Receptor":{"RUTRecep":RUTRECEP,"RznSocRecep":RAZONSOCRECEP,"DirRecep":DIRRECEP,"CmnaRecep":COMUNARECEP,"Contacto":CONTACTORECEP,"GiroRecep":GIRORECEP,"CorreoRecep":CORREORECEP},"Transporte":{"Patente":PATENTE,"RUTTrans":RUTTRANS,"Chofer":{"RUTChofer":RUTCHOFER,"NombreChofer":NOMBRECHOFER},"DirDest":DIRECCDEST,"CmnaDest":COMUNADEST}},"Detalle":[{"NmbItem":NOMBREITEM,"QtyItem":CANT,"PrcItem":PRECIO,"UnmdItem":UNIDADMED},...],"Referencia":[{"TpoDocRef":TIPODOCREF,"FolioRef":FOLIOREF,"FchRef":FECHAREF,"RazonRef":RAZONREF,"NroLinRef":NUMEROLINREF},...], "extData": {"uniqueId":IDUNICO}}
```

Valor	Tipo	Descripción	Opcional
Estructura Encabezado			
TIPODESPACHO	Entero	Acepta los valores: 1 (Despacha receptor), 2 (Despacha emisor a instalaciones de cliente), 3 (Despacha emisor a otras instalaciones).	

INDICETRASLADO	Entero	Acepta los valores: 1 (Operación constituye venta), 2 (Ventas por efectuar), 3 (Consignaciones), 4 (Entrega gratuita), 5 (Traslados internos), 6 (Otros traslados no venta), 7 (Guía de devolución), 8 (Traslado exportación no venta), 9 (Venta exportación).	
TERMINOPAGO	Cadena	Glosa de término de pago	✓
Estructura Emisor			
RUTEMISOR	Cadena	Rut con DV del emisor.	
CODIGOSUCURSAL	Entero	Código SII de la sucursal.	
Estructura Receptor			
RUTRECEP	Cadena	Rut con DV del receptor.	
RAZONSOCRECEP	Cadena	Razón social del receptor.	
DIRRECEP	Cadena	Dirección del receptor.	
COMUNARECEP	Cadena	Comuna de la dirección del receptor.	
CONTACTORECEP	Cadena	Información de contacto del receptor.	✓
GIRORECEP	Cadena	Giro del receptor.	
CORREORECEP	Cadena	E-mail del receptor.	✓
Estructura Transporte			
PATENTE	Cadena	Patente del vehículo del transportista. Obligatorio para tipo de despacho por cuenta del emisor (2 o 3).	
RUTTRANS	Cadena	Rut con DV del transportista. Obligatorio para tipo de despacho por cuenta del emisor (2 o 3).	
RUTCHOFER	Cadena	Rut con DV del chofer.	
NOMBRECHOFER	Cadena	Nombre del chofer.	
DIRECCDEST	Cadena	Dirección de destino	
COMUNADEST	Cadena	Comuna de destino	
Estructura Detalle			
NOMBREITEM	Cadena	Nombre del ítem.	
CANT	Número	Cantidad del ítem. Máximo 2 decimales.	
PRECIO	Entero	Precio del ítem.	
UNIDADMED	Cadena	Unidad de medida del ítem.	✓
Estructura Referencia Es opcional			

TIPODOCREF	Entero	Código del tipo del documento referenciado.	
FOLIOREF	Entero	Folio del documento referenciado.	
FECHAREF	Cadena	Fecha en formato YYYY-MM-DD que indica fecha del documento referenciado. No debe ser mayor a la fecha actual.	
RAZONREF	Cadena	Razón de la referencia	✓
NUMEROLINREF	Entero	Posición de la referencia: 1 para la primera referencia, 2 para la segunda, etc.	
Estructura extData Es opcional			
IDUNICO	Cadena	Identificador único al documento a generar. Si ya se ocupó antes, no se generará un nuevo documento y se devolverá el previo.	

El arreglo Detalle y Referencias soporta múltiples ítemes. Para mayor detalle sobre todos los campos véase https://www.sii.cl/factura_electronica/factura_mercado/descripcion_formato.htm.

Nota de crédito

Su esquema JSON es:

```
{"Encabezado":{"IdDoc":{"Folio":1,"TipoDTE":61},"Emisor":{"RUTEmisor":RUTEMISOR},"Receptor":{"RUTRecep":RUTRECEP,"RznSocRecep":RAZONSOCRECEP,"GiroRecep":GIRORECEP}},"Detalle":[{"NmbItem":RAZONREF,"QtyItem":1,"PrcItem":PRECIO}],"Referencia":[{"TpoDocRef":TIPODOCREF,"FolioRef":FOLIOREF,"FchRef":FECHAREF,"RazonRef":RAZONREF,"NroLinRef":NUMEROLINREF,"CodRef":CODIGOREF}], "extData": {"uniqueId":IDUNICO}}
```

Valor	Tipo	Descripción	Opcional
Estructura Emisor			
RUTEMISOR	Cadena	Rut con DV del emisor.	
Estructura Receptor			
RUTRECEP	Cadena	Rut con DV del receptor. Usar "66666666-6" si no se desea especificar.	
RAZONSOCRECEP	Cadena	Razón social del receptor. Usar "SII Boleta" si no se desea especificar.	
GIRORECEP	Cadena	Giro del receptor. Usar "PERSONA NATURAL" para personas naturales.	
Estructura Detalle			
PRECIO	Entero	Monto de la nota de crédito. En caso de anulación, sólo puede ser por el monto total.	

Estructura `Referencia`			
`TIPODOCREF`	`Entero`	Código del tipo del documento referenciado.	
`FOLIOREF`	`Entero`	Folio del documento referenciado.	
`FECHAREF`	`Cadena`	Fecha en formato YYYY-MM-DD que indica fecha del documento referenciado. No debe ser mayor a la fecha actual.	
`RAZONREF`	`Cadena`	Razón de la referencia	
`NUMEROLINREF`	`Entero`	Posición de la referencia: 1 para la primera referencia, 2 para la segunda, etc.	
`CODIGOREF`	`Entero`	Acepta los valores: 1 (anular), 3 (corregir monto). En caso de anulación, todos los tipos de documentos referenciados deberán ser iguales. En caso de corrección monto, solo se acepta 1 documento referenciado. Siempre todos los documentos deben tener al mismo receptor.	
Estructura `extData` Es opcional			
`IDUNICO`	`Cadena`	Identificador único al documento a generar. Si ya se ocupó antes, no se generará un nuevo documento y se devolverá el previo.	

El arreglo Detalle y Referencias soporta múltiples ítemes. Para mayor detalle sobre todos los campos véase https://www.sii.cl/factura_electronica/factura_mercado/descripcion_formato.htm.

Nota de débito

Su esquema JSON es:

```
{"Encabezado":{"IdDoc":{"Folio":1,"TipoDTE":56},"Emisor":{"RUTEmisor":RUTEMISOR},"Receptor":{"RUTRecep":RUTRECEP,"RznSocRecep":RAZONSOCRECEP,"GiroRecep":GIRORECEP}},"Detalle":[{"NmbItem":RAZONREF,"QtyItem":1,"PrcItem":PRECIO}],"Referencia":[{"TpoDocRef":TIPODOCREF,"FolioRef":FOLIOREF,"FchRef":FECHAREF,"RazonRef":RAZONREF,"NroLinRef":NUMEROLINREF,"CodRef":CODIGOREF}], "extData": {"uniqueId":IDUNICO}}
```

Valor	Tipo	Descripción	Opcional
Estructura `Emisor`			
`RUTEMISOR`	`Cadena`	Rut con DV del emisor.	
Estructura `Receptor`			
`RUTRECEP`	`Cadena`	Rut con DV del receptor. Usar "66666666-6" si no se desea especificar.	
`RAZONSOCRECEP`	`Cadena`	Razón social del receptor. Usar "SII Boleta" si no se desea especificar.	

GIRORECEP	Cadena	Giro del receptor. Usar "PERSONA NATURAL" para personas naturales.	
Estructura Detalle			
PRECIO	Entero	Monto de la nota de crédito. En caso de anulación, sólo puede ser por el monto total.	
Estructura Referencia			
TIPODOCREF	Entero	Código del tipo del documento referenciado.	
FOLIOREF	Entero	Folio del documento referenciado.	
FECHAREF	Cadena	Fecha en formato YYYY-MM-DD que indica fecha del documento referenciado. No debe ser mayor a la fecha actual.	
RAZONREF	Cadena	Razón de la referencia	
NUMEROLINREF	Entero	Posición de la referencia: 1 para la primera referencia, 2 para la segunda, etc.	
CODIGOREF	Entero	Acepta los valores: 1 (anular), 3 (corregir monto). En caso de anulación, se pueden referenciar 1 o más documentos. En caso de corrección monto, solo se acepta 1 documento referenciado. Siempre todos los documentos deben tener al mismo receptor, y ser de tipo 61 (Nota de crédito).	
Estructura extData Es opcional			
IDUNICO	Cadena	Identificador único al documento a generar. Si ya se ocupó antes, no se generará un nuevo documento y se devolverá el previo.	

El arreglo Detalle y Referencias soporta múltiples ítemes. Para mayor detalle sobre todos los campos véase https://www.sii.cl/factura_electronica/factura_mercado/descripcion_formato.htm.

Esquema común de respuesta

Para todos los tipos de DTE. Si hay éxito, entregará una respuesta con código **HTTP 200 OK** con un payload de la siguiente forma:

```
{"pdf_public_url":URLPDF,
"b64encoded_pdf":B64PDF,
"dte":DTE,
"folio":FOLIO,
"emisor":EMISOR,
"receptor":RECEPTOR,
"fecha":FECHA,
"total":TOTAL,
"exento":EXENTO,
```

```
"iva":IVA,
"neto":NETO}
```

Dependiendo del tipo de DTE emitido, vendrán unos u otros atributos como se detalla en la siguiente tabla. Los atributos marcados con ✓ siempre se entregan. Los marcados con * se entregan sólo bajo ciertas condiciones.

Valor	Tipo	Descripción	Tipo de DTE					
			39	41	33	34	52	61
URLPDF	Cadena	URL pública del PDF que contiene el DTE emitido.	✓	✓	✓	✓	✓	✓
B64PDF	Cadena	Contenido PDF codificado en Base64.	✓	✓	✓	✓	✓	✓
DTE	Entero	Tipo del DTE emitido. Será **39** si es Boleta o **41** si es Boleta exenta o **52** si es Guía de despacho o **33** si es Factura electrónica afecta o **34** si es Factura electrónica exenta o **61** si es Nota de crédito.	✓	✓	✓	✓	✓	✓
FOLIO	Entero	Folio del DTE emitido.	✓	✓	✓	✓	✓	✓
EMISOR	Entero	Rut sin DV del emisor.	✓	✓	✓	✓	✓	✓
RECEPTOR	Entero	Rut sin DV del receptor.	✓	✓	✓	✓	✓	✓
FECHA	Cadena	Fecha en formato YYYY-MM-DD del DTE emitido.	✓	✓	✓	✓	✓	✓
TOTAL	Entero	Monto total del DTE.	✓	✓	✓	✓	✓	✓
EXENTO	Entero	Monto exento del DTE.	*	✓	*	✓	*	*
IVA	Entero	Monto de IVA del DTE.	✓		✓		✓	✓
NETO	Entero	Monto neto del DTE.	✓		✓		✓	✓

En caso de error, entregará el mensaje de error correspondiente y un código de estado **HTTP 4XX o 5XX**.

Desarrollo de integración

A continuación se listan utilidades y documentación para desarrollar una integración con la API para los lenguajes de programación Java, Python y Javascript.

AWS ofrece su kit de desarrollo (SDK) para varios lenguajes, en particular para **Java**, **Python** y **Javascript**: https://aws.amazon.com/es/tools/.

Java

- Utilidad para **STS**: https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/prog-services-sts.html
- Utilidad para **firmar**: https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/amazonaws/auth/AWS4Signer.html

Python

- Utilidad para **STS**: https://docs.aws.amazon.com/code-samples/latest/catalog/code-catalog-python-example_code-sts.html
- Instrucciones para **firmar manualmente**: https://docs.aws.amazon.com/general/latest/gr/sigv4-signed-request-examples.html

Javascript

- Utilidad para **STS**: https://www.npmjs.com/package/aws-sdk, úsese la clase STS.
- Utilidad para **firmar**: https://www.npmjs.com/package/@aws-amplify/core, úsese la clase Signer.

Para este lenguaje se entrega un ejemplo de código en la siguiente sección.

Integración minimal Javascript

Finalmente, se ofrece un código de integración minimal para Javascript/NodeJS.

Preámbulo

Se deberán instalar los paquetes de AWS:

- **`aws-sdk`**: El SDK de AWS. Ofrece utilidad para usar **STS**.
- **`@aws-amplify/core`**: Ofrece utilidad **que facilita el cálculo de la firma**.

Y los auxiliares:

- **`moment`**: Facilita el manejo, representación y aritmética de fechas.
- **`axios`**: Facilita la creación y ejecución de peticiones HTTP.

> **Nota**: Al final de este documento se ofrece un **código mínimo** que implementa lo explicado aquí.

Siguiendo el flujo ya descrito, con las llaves proporcionadas por Vessi se deberá usar **STS** de **aws-sdk** para obtener llaves y token temporal. Si esa solicitud falla, ver si la razón fue por fecha/hora errónea. Si es así, invocar un servicio de hora remota para obtener el tiempo actual correcto.

Usar **moment** para tomar la hora remota y la hora local para calcular su diferencia. Inyectar la diferencia de hora tanto a **aws-sdk** como a **@aws-amplify/core** para que compensen la hora local y así sus futuras peticiones no fallen por ese motivo. Luego, reintentar la petición a STS, la cual debería tener éxito.

Ya obtenidas las llaves y token temporales, tomar los datos de la boleta y usar **Signer** de **@aws-amplify/core** para generar una petición firmada. Para ser usada con **axios**, se deberá descartar el header "`host`" de la petición generada, y finalmente usar axios para despacharla a la URL de la API de Boletas.

Uso en Java y Python

El código Javascript minimal aún podrá usarse con otros lenguajes de programación si se invoca en el marco de un subproceso creado "on-the-fly", entregándole parámetros mediante su escritura en la entrada estándar y leyendo su salida estándar para obtener los resultados de la emisión del DTE:

- En Python, se puede usar **subprocess**: https://docs.python.org/3/library/subprocess.html
- En Java, se puede usar **ProcessBuilder** y **Process**: https://docs.oracle.com/javase/8/docs/api/java/lang/Process.html

Debe advertirse que, si bien resuelve el problema, esta estrategia de levantar un subproceso por cada DTE es muy poco eficiente debido al “overhead” de cada levantamiento en el sistema operativo. Sólo debe considerarse como última opción o como algo meramente provisorio.

Código minimal

Este código reúne todo lo ya explicado para integrarse con la API para así usar el producto.

Para la eventualidad de hora demasiado desfasada, la estrategia que usa en caso de falla es: invocar al servicio de hora, calcular el “delta” de diferencia de tiempo e inyectarlo en aws-sdk y amplify-core para que compensen internamente la hora usada y así generar una firma válida, sin alterar la hora local de la máquina.

Advertencia: Este código se provee tal cual, sin garantía de ningún tipo. Es meramente de ejemplo: se asegura su correctitud, no así su mantenibilidad. **No constituye una librería oficial de Vessi**.

```
////////////////////////////////////////////////////////////////////////////////////////////
// Importaciones
const moment = require('moment');
const axios = require('axios');
const awsAmplifyCore = require('@aws-amplify/core');
const awsSdk = require('aws-sdk');

////////////////////////////////////////////////////////////////////////////////////////////
// Constantes
const URL_API_BOLETAS = 'https://qgd3pr8an8.execute-api.us-west-2.amazonaws.com/qa';
const URL_API_HORA = 'https://zehhq8aey7.execute-api.us-west-2.amazonaws.com/dev/hora';
const ACCESS_KEY_ID = 'XXXXXXXXXXXXXXXXXXXX'; // Proporcionada por Vessi
const SECRET_ACCESS_KEY = 'YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY'; // Proporcionada por Vessi
const AWS_REGION = 'us-east-1';

////////////////////////////////////////////////////////////////////////////////////////////
// Ejemplo de datos de boleta
const DATOS_BOLETA_EJEMPLO = {
  "Encabezado":  {
    "IdDoc": {
      "TipoDTE": 39,
      "Folio": 1
    },
    "Emisor": {
      "RUTEmisor": "76431892-7"
    },
    "Receptor": {
      "RUTRecep": "2-7",
      "RznSocRecep": "DOS SIETE SPA",
      "DirRecep": "AV DEL SOL 76"
    }
  },
```

```
  "Detalle": [
    {
      "NmbItem": "CLIPS",
      "QtyItem":2,
      "PrcItem":300
    },
    {
      "NmbItem": "MARRAQUETA",
      "QtyItem": 0.5,
      "PrcItem": 600,
      "UnmdItem": "KG"
    }
  ],
  "pago": "DEBITO"
};

////////////////////////////////////////////////////////////////////////////////////////
// Funciones auxiliares
const getClockOffset = async function() {
  let clockOffset;
  try {
    let response = await axios.get(URL_API_HORA);
    let requestTimeEpoch = response.data.requestTimeEpoch;
    // Obtener hora remota y local
    let remoteTimeMillisEpoch = Number(moment(requestTimeEpoch).format('x'));
    let localTimeMillisEpoch = Number(moment().format('x'));
    // Calcular offset: calcular diferencia
    clockOffset = remoteTimeMillisEpoch - localTimeMillisEpoch;
  }
  catch(e) {
    throw 'Error al obtener hora remota';
  }
  return clockOffset;
};

const executeJSONSignedRequest = async function(method, url, data) {
  // Hace lo mismo que executeSignedRequest, pero aplica JSON.stringify a data antes.
  let stringifiedData = (data !== undefined) ? JSON.stringify(data) : undefined;
  let response;

  try {
    response = await executeSignedRequest(method, url, stringifiedData);
  }
  catch(e) {
    throw e;
  }

  return response;
};

const executeSignedRequest = async function(method, url, data) {
  // De ser necesario, data puede ser undefined (por ejemplo, en un GET)
  awsSdk.config.update({
    accessKeyId: ACCESS_KEY_ID,
    secretAccessKey: SECRET_ACCESS_KEY,
    region: AWS_REGION
```

```
  });
  let stsData;
  let sts;
  try {
    sts = new awsSdk.STS();
    stsData = await sts.getSessionToken().promise();
  }
  catch(e) {
    // Ver si fue error por firma fuera de rango temporal
    const expiredSignatureRegex = /^SignatureDoesNotMatch\: Signature not yet current\: .+ is
still later than .+$/;
    const notYetCurrentSignatureRegex = /^SignatureDoesNotMatch\: Signature expired\: .+ is
now earlier than .+$/;
    let isExpiredSignatureError = expiredSignatureRegex.test(e.toString());
    let isNotYetCurrentSignatureError = notYetCurrentSignatureRegex.test(e.toString());
    // Añadir offset y reintentar una vez más.
    if(isExpiredSignatureError || isNotYetCurrentSignatureError) {
      try {
        // Obtener diferencia entre hora remota y local. Puede ser tanto positiva como
negativa.
        let clockOffset = await getClockOffset();

        // Corregir hora en Amplify Core.
        awsAmplifyCore.DateUtils.setClockOffset(clockOffset);

        // Corregir hora en AWS SDK.
        awsSdk.config.update({
          correctClockSkew: true,
          systemClockOffset: clockOffset,
        });

        // Reintentar petición para obtener datos de STS.
        sts = new awsSdk.STS(); // Reinstanciar STS, ahora con tiempo corregido
        stsData = await sts.getSessionToken().promise();
      }
      catch(e) {
        // Obtención de offset falló, no queda más que fallar.
        throw 'Error al hacer petición a API. Posibles causas: problema de red, credenciales
inválidas u hora local incorrecta.';
      }
    }
    else {
      throw e;
    }
  }
  let newCreds = sts.credentialsFrom(stsData);
  let sessionToken = newCreds.sessionToken;
  let accessKeyId = newCreds.accessKeyId;
  let secretAccessKey = newCreds.secretAccessKey;

  // Usar Signer para firmar petición. Para el proceso se usa la hora LOCAL.
  // Véase https://aws-amplify.github.io/amplify-js/api/classes/signer.html
  let request = {
    method,
    url,
    data
```

```
  };

  let accessInfo = {
    access_key: accessKeyId,
    secret_key: secretAccessKey,
    session_token: sessionToken,
  };

  let serviceInfo = null;
  // https://stackoverflow.com/questions/60554364/authenticate-to-aws-with-axios-and-aws4
  let signedRequest = awsAmplifyCore.Signer.sign(request, accessInfo, serviceInfo);
  delete signedRequest.headers['host'];
  let axiosInstance = axios.create();

  let response;
  try {
    response = await axiosInstance(signedRequest);
  }
  catch(e) {
    throw e;
  }
  return response;
};

const rutDv2Rut = function(rutDv) {
  const rutDvRegex = /^(\d+)\-(\d|k|K)$/;
  let rutDvParts = rutDv.match(rutDvRegex);
  if(rutDvParts === null) {
    throw `"${rutDv}" no calza con patrón de rut con dígito verificador`;
  }
  let rut = rutDvParts[1];
  return rut;
};

////////////////////////////////////////////////////////////////////////////////////////////
// Generador de boleta
const generarBoleta = async function(datosBoleta) {
  console.log('Generando boleta...\n', datosBoleta);
  let resultData;
  try {
    let rutEmisor = rutDv2Rut(datosBoleta.Encabezado.Emisor.RUTEmisor);
    let method = 'POST';
    let url = `${URL_API_BOLETAS}/api/dte/documentos/generar/ext/${rutEmisor}`;
    let payload = datosBoleta;
    let response = await executeJSONSignedRequest(method, url, payload);
    resultData = response.data; // Atributo pdf_public_url contendrá URL de PDF de boleta
generada
  }
  catch(error) {
      resultData = (error.response !== undefined) ? error.response.data : error;
  }
  console.log('Resultado:\n', resultData);
};

// Generar boleta con datos de prueba
generarBoleta(DATOS_BOLETA_EJEMPLO);
```