

SIGUIENTES PASOS >  WEB SERVICES

Factura electrónica

Requisitos previos

Para poder usar esta guía, primero necesitarás:

- [Integrar Afip SDK en tu proyecto](#)

La especificación de este Web Service se encuentra disponible en <https://www.afip.gob.ar/ws/documentacion/manuales/manual-desarrollador-ARCA-COMPG-v4-0.pdf> ↗



> Identificador `wsfe`



Aquí hablaremos de comprobante indistintamente si es una factura, nota de crédito, etc.

Si es tu primera vez conectándote con la facturación electrónica de AFIP/ARCA, te recomiendo leer esta guía básica sobre su funcionamiento general:



Conectar tu sistema con la facturación electrónica de AFIP



Crear PDF

[Crear PDF](#)

Obtener número del último comprobante creado

Node

PHP

Ruby

Python

API

```
// Numero de punto de venta
const puntoDeVenta = 1;

// Tipo de comprobante
const tipoDeComprobante = 6; // 6 = Factura B

const lastVoucher = await
afip.ElectronicBilling.getLastVoucher(puntoDeVenta,
tipoDeComprobante);
```



Crear y asignar CAE a un comprobante

Node

PHP

Ruby

Python

API

Debemos utilizar el método `createVoucher` pasándole como parámetro un Objeto con los detalles del comprobante y si queremos tener la respuesta completa enviada por el WS debemos pasarle como segundo parámetro `true`, en caso de no enviarle el segundo parámetro nos devolverá como respuesta `{ CAE : CAE asignado el comprobante, CAEFchVto : Fecha de vencimiento del CAE (yyyy-mm-dd) }`.



```
// Devolver respuesta completa del web service
const returnFullResponse = false;


const date = new Date(Date.now() - ((new
Date()).getTimezoneOffset() * 60000)).toISOString().split('T')
[0];

// Info del comprobante
let data = {
  'CantReg' : 1, // Cantidad de comprobantes a registrar
  'PtoVta' : 1, // Punto de venta
  'CbteTipo' : 6, // Tipo de comprobante (ver tipos
disponibles)
  'Concepto' : 1, // Concepto del Comprobante:
(1)Productos, (2)Servicios, (3)Productos y Servicios
  'DocTipo' : 99, // Tipo de documento del comprador (99
consumidor final, ver tipos disponibles)
  'DocNro' : 0, // Número de documento del comprador (0
consumidor final)
  'CbteDesde' : 1, // Número de comprobante o numero
del primer comprobante en caso de ser mas de uno
  'CbteHasta' : 1, // Número de comprobante o numero
del último comprobante en caso de ser mas de uno
  'CbteFch' : parseInt(date.replace(/-/g, '')), //
(Opcional) Fecha del comprobante (yyyymmdd) o fecha actual si
es nulo
  'ImpTotal' : 121, // Importe total del comprobante
  'ImpTotConc' : 0, // Importe neto no gravado
  'ImpNeto' : 100, // Importe neto gravado
  'ImpOpEx' : 0, // Importe exento de IVA
  'ImpIVA' : 21, //Importe total de IVA
  'ImpTrib' : 0, //Importe total de tributos
  'MonId' : 'PES', //Tipo de moneda usada en el
comprobante (ver tipos disponibles)('PES' para pesos
argentinos)
  'MonCotiz' : 1, // Cotización de la moneda usada (1
para pesos argentinos)
  'Iva' : [ // (Opcional) Alícuotas asociadas al
comprobante
    {
      'Id' : 5, // Id del tipo de IVA (5 para
21%)(ver tipos disponibles)
      'BaseImp' : 100, // Base imponible
      'Importe' : 21 // Importe
    }
  ],
};
```



```
const res = await afip.ElectronicBilling.createVoucher(data,
returnFullResponse);

res['CAE']; //CAE asignado el comprobante
res['CAEFchVto']; //Fecha de vencimiento del CAE (yyyy-mm-dd)
```

 Este método acepta mas parámetros, puedes encontrar todos los parámetros disponibles [En este ejemplo ↗](#)

Crear y asignar CAE a siguiente comprobante

Node

PHP

Ruby

Python

API

Debemos utilizar el método `createNextVoucher` pasándole como parámetro un Objeto con los detalles del comprobante al igual que el método `createVoucher`, nos devolverá como respuesta `{ CAE : CAE asignado al comprobante, CAEFchVto : Fecha de vencimiento del CAE (yyyy-mm-dd), voucher_number : Número asignado al comprobante }`.



```
const res = await
afip.ElectronicBilling.createNextVoucher(data);

res['CAE']; //CAE asignado el comprobante
res['CAEFchVto']; //Fecha de vencimiento del CAE (yyyy-mm-dd)
res['voucher_number']; //Número asignado al comprobante
```

Obtener información de un comprobante ya emitido

Con este método podemos obtener toda la información relacionada a un comprobante o simplemente saber si el comprobante existe.

Node

PHP

Ruby

Python

API

```
// Numero de comprobante
const numeroDeComprobante = 1;

// Numero de punto de venta
const puntoDeVenta = 1;

// Tipo de comprobante
const tipoDeComprobante = 6; // 6 = Factura B

const voucherInfo = await
afip.ElectronicBilling.getVoucherInfo(numeroDeComprobante,
puntoDeVenta, tipoDeComprobante);

if(voucherInfo === null){
    console.log('El comprobante no existe');
}
else{
    console.log('Esta es la información del comprobante:');
    console.log(voucherInfo);
}
```



Crear CAEA

Node

PHP

Ruby

Python

API

```
// Periodo del CAEA. (yyyymm)
const periodo = 202307;

// Orden del CAEA dentro del periodo. Quincena 1, Quincena 2
const orden = 1;

const caeaInfo = await
afip.ElectronicBilling.createCAEA(periodo, orden);
```

Obtener información de CAEA ya emitido

Node

PHP

Ruby

Python

API

```
// Periodo del CAEA. (yyyymm)
const periodo = 202307;

// Orden del CAEA dentro del periodo. Quincena 1, Quincena 2
const orden = 1;

const caeaInfo = await afip.ElectronicBilling.getCAEA(periodo,
orden);
```

Obtener puntos de venta disponibles

⚠ Es normal recibir un error en testing ya que no existen puntos de venta para testing (siempre se usa `1`)

Node	PHP	Ruby	Python	API
<pre>const salesPoints = await afip.ElectronicBilling.getSalesPoints()</pre>				



Obtener tipos de comprobantes disponibles

Node	PHP	Ruby	Python	API
<pre>const voucherTypes = await afip.ElectronicBilling.getVoucherTypes();</pre>				

Obtener tipos de conceptos disponibles

Node	PHP	Ruby	Python	API
<pre>const conceptTypes = await afip.ElectronicBilling.getConceptTypes();</pre>				

Obtener tipos de documentos disponibles

Node

PHP

Ruby

Python

API

```
const documentTypes = await
afip.ElectronicBilling.getDocumentTypes();
```

Obtener tipos de alícuotas disponibles

Node

PHP

Ruby

Python

API

```
const aloquotTypes = await
afip.ElectronicBilling.getAliquotTypes();
```



Obtener tipos de monedas disponibles

Node

PHP

Ruby

Python

API

```
const currenciesTypes = await
afip.ElectronicBilling.getCurrenciesTypes();
```

Obtener cotización de moneda

Node

PHP

Ruby

Python

API


```
// ID de la moneda a consultar (El ID se obtiene del método
para obtener los tipos de monedas)
const currencyID = 'DOL'

// Fecha de la cotización en formato aaaammdd | Por ej.
20250221 = 21 de febrero de 2025
const rateDate = '20250221';

const exchangeRate = await
afip.ElectronicBilling.executeRequest('FEParamGetCotizacion',
{
  MonId: currencyID,
  FchCotiz: rateDate
})
```

Obtener tipos de opciones disponibles para el comprobante



Node

PHP

Ruby

Python

API

```
const optionTypes = await
afip.ElectronicBilling.getOptionsTypes();
```

Obtener tipos de tributos disponibles

Node

PHP

Ruby

Python

API

```
const taxTypes = await afip.ElectronicBilling.getTaxTypes();
```

Obtener tipos de condiciones frente al IVA disponibles

Node

PHP

Ruby

Python

API

```
const taxConditionTypes = await
afip.ElectronicBilling.executeRequest('FEParamGetCondicionIvaR
eceptor')
```

Transformar formato de fecha que utiliza AFIP

De a

Node

PHP

Ruby

Python

API

```
const date = afip.ElectronicBilling.formatDate('19970508');
//Nos devuelve 1997-05-08
```

Obtener estado del servidor



No te confíes mucho de este método, ARCA siempre devuelve que esta todo bien incluso cuando hay algo fallando.

Node

PHP

Ruby

Python

API

```
const serverStatus = await
afip.ElectronicBilling.getServerStatus();

console.log('Este es el estado del servidor:');
console.log(serverStatus);
```

Anterior
Web services

Siguiente
Crear PDF

Última actualización hace 19 días

¿Te fue útil?

