????

????

官方对接地址 文件统计

- 计费记录设计文档
- 验证和错误
- Web 服务描述文档（v1.0.3） PDF文档 （1.673 KB）
- Web 服务描述文档（v1.0.3，英文版） PDF文档 （1,715 KB）
- Web 服务的 WSDL
- Web 服务示意图
- 生成记录指纹或哈希的技术规范详情（v0.1.2） PDF文档 （1.11 MB）
- 发票二维码技术规格说明文档（v0.4.7） PDF文档 （1.9 MB）
- 生成账单记录电子签名的技术规范
    - 生成账单记录电子签名的技术规范（v0.1.5） PDF文档 （1.46 MB）这个暂不开发 NO VERIFACTU
    - ZIP文件：包含账单记录签名样本的附件 ZIP压缩文件 （5.74 KB）
- 无法核实的计费记录验证服务说明文档（v0.2） PDF文档 （576 KB）这个暂不开发 NO VERIFACTU
- 开发公司常见问题解答（更新于 2025 年 9 月 19 日） PDF文档 （666 KB）
- 负责任声明示例 PDF文档 （708 KB）

?????????

发送票务地址：

https://prewww10.aeat.es/wlpl/TIKE-CONT/ws/SistemaFacturacion/VerifactuSOAP

???? ?????

发送票务地址：

https://www10.agenciatributaria.gob.es/wlpl/TIKE-CONT/ws/SistemaFacturacion/VerifactuSOAP

```
<sum1:DetalleDesglose>
<sum1:ClaveRegimen>01</sum1:ClaveRegimen>
<!-- 税务制度代码（01=普通IVA） -->
<sum1:CalificacionOperacion>S1</sum1:CalificacionOperacion>
<!-- 操作类型 -->
<sum1:TipoImpositivo>4</sum1:TipoImpositivo>
<!-- 税率 4% -->
<sum1:BaseImponibleOimporteNoSujeto>10</sum1:BaseImponibleOimporteNoSujeto>
<!-- 不含税金额 -->
<sum1:CuotaRepercutida>0.4</sum1:CuotaRepercutida>
<!-- 税额 -->
</sum1:DetalleDesglose
```

????????

节点	含义	可用值/格式	说明
`<sum1:Subsanacion>`	修正标志	`S` 或省略	`S` 表示此记录为修正（例如纠正错误或重新提交）
`<sum1:TipoFactura>`	发票类型	`F1` 普通发票 `F2` 简易发票 `R1` 作废 `R2` 更正 `R3` 替换	可以参见 TipoFactura 段落
`<sum1:ClaveRegimen>`	税制代码	`01` 普通制度 `02` 简易制度 `03` 免税	取决于发票适用税法
`<sum1:CalificacionOperacion>`	交易类型	`S1` 受IVA约束 `N1` 不受约束 `E1` 免税	常见餐饮类一般使用 `S1`
`<sum1:TipoImpositivo>`	税率	4, 10, 21 , 0 几种税率	西班牙标准IVA税率
`<sum1:TipoHuella>`	指纹类型	`01`	表示 SHA-256 Base64
`<sum1:TipoUsoPosibleSoloVerifactu>`	是否专属VeriFactu	`S` 或 `N`	如果你的系统只用于VeriFactu，则为 `S`
`<sum1:TipoUsoPosibleMultiOT>`	是否支持多义务人	`S` 或 `N`	SaaS类一般 `S`
`<sum1:IndicadorMultiplesOT>`	是否有多个义务人使用中	`S` 或 `N`	如果服务器下多家餐厅，则为 `S`
`<sum1:NumeroInstalacion>`	安装编号	数字或字符串	每个部署/门店唯一
`<sum1:Huella>`	指纹	Base64 编码字符串	SHA-256(关键字段串联)

TipoFactura ?????

Código	名称	适用情形	关键点

F1	**Factura completa** (Art. 6, 7.2 y 7.3 del RD 1619/2012)	标准完整发票，识别买方、含增值税细节。	是最常见的发票类型。必须包含: - 客户身份（NIF / NombreRazon） - 分项税额 (Desglose)
F2	**Factura simplificada** (Art. 6.1.d RD 1619/2012)	金额较小、无需识别买方的发票（小票、餐饮等）。	通常对应 **Ticket**。买方信息可选。
F3	**Factura emitida en sustitución de facturas simplificadas**	当企业把之前已申报的简易发票（F2）替换为完整发票（F1）。	属于更正性质的“补开发票”，用于“由 ticket 转发票”。这正是你现在在做的情况。
R1	**Factura rectificativa** (Art. 80.1 y 80.2 y error fundado en derecho)	纠正税基、税率或金额错误（因判决、行政原因等）。	Rectificación 依据法条 80.1 / 80.2。
R2	**Factura rectificativa** (Art. 80.3)	因退货、折扣、无效合同等导致的修正。	对应业务更改（例如取消销售）。
R3	**Factura rectificativa** (Art. 80.4)	客户破产或无力偿付引起的修正。	专用于破产情形。
R4	**Factura rectificativa (Resto)**	其他未涵盖情形的更正发票。	一般性修正用途。
R5	**Factura rectificativa simplificada**	针对简易发票的修正。	通常用于 ticket 更正。

QR ???? ????

QR 尺寸 40*40 mm

以地址为准生成 QR 码，并在下方打印: Factura verificable en la sede electrónica de la AEAT

???????

https://prewww2.aeat.es/wlpl/TIKE-CONT/ValidarQR?nif=XXXXXXXXY&numserie=YYYY...YYYY&fecha=DD- DD-AAAA&importe=NNNNNNNNN.DD

???????

https://www2.agenciatributaria.gob.es/wlpl/TIKE-CONT/ValidarQR?nif=XXXXXXXXY&numserie=YYYY...YYYY&fecha=DD- DD-AAAA&importe=NNNNNNNNN.DD

XML 返回的资料进行拼接 " &" = " 链接 xia

nif= XXXXXXXXXXXX &numseria= 123456 &fecha= &importe=

- URL base: https://prewww2.aeat.es/wlpl/TIKE-CONT/ValidarQR?
- Parámetro nif: 89890001K
- Parámetro numserie: 12345678&G33
- Parámetro fecha: 01-01-2024
- Parámetro importe: 241.4

????

设置中 添加 加载电子证书按钮 + 密码框 , 解析 + 使用

1 - ???? ? ?????? ID, ???? ?

- CN : CERTIFICADO FISICA PRUEBAS -99999910G' （ - 前是公司 /个人名字 - 后面是税号）
- SN : 公司或者个人名称
- O： 有这个字段代表公司 否则是个人
- SERIALNUMBER - 后面是 税号
  C 是国家
- NotAfter 证书到期时间

参考一下代码

```
using System;
using System.Security.Cryptography.X509Certificates;
using System.Text.RegularExpressions;

namespace CertInfoExtractor
{
    public enum CertType
    {
        Unknown,
        Personal,
        Organization
    }
```

```
public class CertInfo
{
    public CertType Type { get; set; }
    public string Name { get; set; }           // 姓名或公司名
    public string TaxNumber { get; set; }      // 税号 / NIF / CIF
    public DateTime NotAfter { get; set; }     // 到期时间
    public int DaysRemaining { get; set; }     // 剩余天数
}

public class CertParser
{
    public static CertInfo Parse(string pfxPath, string password)
    {
        var cert = new X509Certificate2(pfxPath, password,
            X509KeyStorageFlags.Exportable | X509KeyStorageFlags.MachineKeySet);

        string subject = cert.Subject;
        string issuer = cert.Issuer;

        string cn = GetField(subject, "CN");
        string org = GetField(subject, "O");
        string serial = GetField(subject, "SERIALNUMBER");

        // 税号兜底正则
        if (string.IsNullOrEmpty(serial))
        {
            var m = Regex.Match(subject, @"(SERIALNUMBER|NIF|CIF|NIE|UID)\s*=\s*([A-Z0-9\-
\/]{6,20})",
                RegexOptions.IgnoreCase);
            if (m.Success)
                serial = m.Groups[2].Value.Trim();
        }

        // 判断类型
        CertType type = DetectType(org, serial, issuer);

        // 名称: 企业用 O, 个人用 CN
        string name = (type == CertType.Organization && !string.IsNullOrEmpty(org))
            ? org
```

```
                : cn;

            var notAfter = cert.NotAfter;
            int daysRemaining = (int)(notAfter - DateTime.Now).TotalDays;

            return new CertInfo
            {
                Type = type,
                Name = name,
                TaxNumber = serial,
                NotAfter = notAfter,
                DaysRemaining = daysRemaining
            };
        }

        private static CertType DetectType(string org, string serial, string issuer)
        {
            bool hasOrg = !string.IsNullOrEmpty(org);
            bool looksLikeCompanyId = Regex.IsMatch(serial ?? "", @"^[A-Z]\d{7,8}",
RegexOptions.IgnoreCase);
            bool looksLikePersonId = Regex.IsMatch(serial ?? "", @"\d{7,8}[A-Z]$",
RegexOptions.IgnoreCase);

            if (hasOrg || looksLikeCompanyId)
                return CertType.Organization;
            if (looksLikePersonId)
                return CertType.Personal;

            if (issuer.Contains("Persona Física", StringComparison.OrdinalIgnoreCase))
                return CertType.Personal;
            if (issuer.Contains("Representante", StringComparison.OrdinalIgnoreCase))
                return CertType.Organization;

            return CertType.Unknown;
        }

        private static string GetField(string subject, string key)
        {
            var m = Regex.Match(subject, key + @"\s*=\s*([^,]+)", RegexOptions.IgnoreCase);
            return m.Success ? m.Groups[1].Value.Trim() : null;
```

```
        }

        // 测试入口
        public static void Main()
        {
            string pfxPath = @"C:\certs\yourcert.pfx";
            string password = "yourPassword";

            var info = Parse(pfxPath, password);

            Console.WriteLine("□□□□□□: " + info.Type);
            Console.WriteLine("□□名称: " + info.Name);
            Console.WriteLine("□□税号: " + info.TaxNumber);
            Console.WriteLine("□□到期时间: " + info.NotAfter.ToString("yyyy-MM-dd HH:mm:ss"));
            Console.WriteLine("□ □□□□: " + info.DaysRemaining);
        }
    }
}
```

?????? SistemaInformatico XML

```
               <sum1:SistemaInformatico>
                  <sum1:NombreRazon>JIECHENG INFORMATICA SL</sum1:NombreRazon>

                  <sum1:NIF>B67287789</sum1:NIF>
                  <sum1:NombreSistemaInformatico>JIECHENG TPV</sum1:NombreSistemaInformatico>
                  <sum1:IdSistemaInformatico>J2</sum1:IdSistemaInformatico>
                  <sum1:Version>1.0.03</sum1:Version>
                  <sum1:NumeroInstalacion>383</sum1:NumeroInstalacion>   // 主机1 副机 2
                  <sum1:TipoUsoPosibleSoloVerifactu>N</sum1:TipoUsoPosibleSoloVerifactu>
                  <sum1:TipoUsoPosibleMultiOT>S</sum1:TipoUsoPosibleMultiOT>
                  <sum1:IndicadorMultiplesOT>S</sum1:IndicadorMultiplesOT>
               </sum1:SistemaInformatico>
```

? ?? .NET Framework 4.0 ?????

要解决 **403** 错误（极有可能是 TLS 版本不兼容），您需要执行以下两个关键步骤：

1. ???? TLS 1.2???????

在 .NET Framework 4.0 中，您需要通过代码显式设置 `ServicePointManager.SecurityProtocol`。但是，**TLS 1.1** 和 **TLS 1.2** 枚举值在 .NET Framework **4.5** 版本中才被正式引入。

解决方案（通过类型转换）：

您必须使用其底层整数值来引用 **TLS 1.1** 和 **TLS 1.2**，以绕过编译时缺少枚举的问题。

C#

```
using System.Net;

// 检查您的项目是否已经引用了 System.Net.dll

// 1. 定义或使用 TLS 1.2 和 TLS 1.1 的数值
// Tls12 = 3072 (十六进制 0xC00)
// Tls11 = 768 (十六进制 0x300)
const SslProtocols Tls12 = (SslProtocols)3072;
const SslProtocols Tls11 = (SslProtocols)768;
const SslProtocols SystemDefault = (SslProtocols)0; // SystemDefault 是 .NET 4.6.2+ 才有的，在 4.0 中忽略

public static void EnableTls12()
{
    // 强制全局使用 Tls12 和 Tls11 (和旧的 Tls/Ssl3 兼容)
    ServicePointManager.SecurityProtocol =
        (SecurityProtocolType)Tls12 |
        (SecurityProtocolType)Tls11 |
        SecurityProtocolType.Tls;
        // SecurityProtocolType.Tls 对应 TLS 1.0
}

// 在应用程序启动或发送请求前调用此方法
EnableTls12();
```

注意：您必须确保您的运行环境（运行您的程序的机器）已经安装了 **.NET Framework 4.0** 的最新更新，以确保底层操作系统（Windows）支持 TLS 1.2。

2. ???????? WebRequest ?

由于 .NET Framework 4.0 没有 `HttpClient` 类（它是在 4.5 引入的），您通常会使用 **`HttpWebRequest`** 类来发送请求。

证书加载代码（与之前类似）：

C#

```
using System.Security.Cryptography.X509Certificates;
using System.Net;

// ... (您的 CertParser.Parse 方法) ...

// 假设这是您获取证书的方法
X509Certificate2 clientCert = new X509Certificate2(
    pfxPath,
    password,
    // 再次强调，使用 UserKeySet 确保私钥在当前用户下可访问
    X509KeyStorageFlags.Exportable |
    X509KeyStorageFlags.PersistKeySet |
    X509KeyStorageFlags.UserKeySet
);

if (!clientCert.HasPrivateKey)
{
    throw new InvalidOperationException("Private key is missing/inaccessible.");
}

// ----------------------------------------------------------------

// HTTP 请求发送代码
public void SendVerifactuRequest(string url, X509Certificate2 cert)
{
```

```
    // 确保 TLS 1.2 已启用 (见上一步)
    EnableTls12();

    HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
    request.Method = "POST";
    request.ContentType = "application/json"; // 或 VeriFactu 要求的其他类型

    // □□ □□□□□□□□□□□□□□□□□□
    request.ClientCertificates.Add(cert);

    try
    {
        // 写入请求体 (alta factura 数据)
        using (var stream = request.GetRequestStream())
        {
            // stream.Write(...);
            // 写入您的 factura XML/JSON 数据
        }

        using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())
        {
            if (response.StatusCode == HttpStatusCode.OK)
            {
                // 处理成功响应
                // ...
            }
            // ...
        }
    }
    catch (WebException ex)
    {
        // 处理 Web 异常, 特别是 403 错误
        if (ex.Response is HttpWebResponse errorResponse && errorResponse.StatusCode ==
HttpStatusCode.Forbidden)
        {
            // 错误可能是 TLS 或证书问题, 记录详情
            Console.WriteLine("Error 403 Forbidden: Check TLS version and certificate
chain.");
        }
        else
```

```
        {
            // 其他网络错误
        }
    }
}
```

请优先尝试 第 **1** 步（强制启用 TLS 1.2）。如果服务器只支持 TLS 1.2，这是您在 .NET 4.0 中解决 403 错误的关键。

您是否想尝试使用这些修改后的代码，或者您需要我帮助您检查证书链的完整性？

2- ????? ???????? ??????????????

发送发票 /小票 hash 计算

a) Datos de campos a utilizar en el caso de registros de facturación de alta (y la “ruta” de su localización dentro del registro):
1. IDEmisorFactura (RegistroAlta/IDFactura/IDEmisorFactura)
2. NumSerieFactura (RegistroAlta/IDFactura/NumSerieFactura)
3. FechaExpedicionFactura (RegistroAlta/IDFactura/FechaExpedicionFactura)
4. TipoFactura (RegistroAlta/TipoFactura)
5. CuotaTotal (RegistroAlta/CuotaTotal)
6. ImporteTotal (RegistroAlta/ImporteTotal)
7. Huella (RegistroAlta/Encadenamiento/RegistroAnterior/Huella)
8. FechaHoraHusoGenRegistro (RegistroAlta/FechaHoraHusoGenRegistro)

数据需要以以上顺序生成 string 进行连接

如 nombreCampo1=valorCampo1&nombreCampo2=valorCampo2&nombreCampoN=valorCampoN

可参考一下代码 （官方代码 翻译 可能需要操作 合并 xml ）

```
using System;
using System.Security.Cryptography;
using System.Text;
```

```
public class GeneradorHuella
{
    /// <summary>
    /// 计算 SHA-256 哈希并输出 Base64（与 Java 逻辑完全一致）
    /// </summary>
    public static string GetHashVerifactu(string msg)
    {
        try
        {
            using (var sha = SHA256.Create())
            {
                byte[] hash = sha.ComputeHash(Encoding.UTF8.GetBytes(msg));
                return Convert.ToBase64String(hash);
            }
        }
        catch (Exception e)
        {
            throw new Exception("生成 SHA 哈希时发生错误", e);
        }
    }

    /// <summary>
    /// 生成 RegistroAlta 记录对应的字段拼接字符串
    /// 顺序与 Java 完全相同
    /// </summary>
    public static string GetReferenciaRegistroAlta(
        string nifEmisor,
        string numFacturaSerie,
        string fechaExpedicion,
        string tipoFactura,
        string cuotaTotal,
        string importeTotal,
        string huellaAnterior,
        string fechaHoraUsoRegistro)
    {
        var sb = new StringBuilder();

        sb.Append(GetValorCampo("IDEmisorFactura", nifEmisor, true))
          .Append(GetValorCampo("NumSerieFactura", numFacturaSerie, true))
```

```
            .Append(GetValorCampo("FechaExpedicionFactura", fechaExpedicion, true))
            .Append(GetValorCampo("TipoFactura", tipoFactura, true))
            .Append(GetValorCampo("CuotaTotal", cuotaTotal, true))
            .Append(GetValorCampo("ImporteTotal", importeTotal, true))
            .Append(GetValorCampo("Huella", huellaAnterior, true))
            .Append(GetValorCampo("FechaHoraUsoRegistro", fechaHoraUsoRegistro, false)); //
□□□□□□□ &

        return sb.ToString();
    }

    /// <summary>
    /// □□□□□□□□□□□Nombre=Valor □ Nombre=Valor&
    /// </summary>
    public static string GetValorCampo(string nombre, string valor, bool separador)
    {
        string campo = nombre + "=" + (valor == null ? "" : valor.Trim());

        if (separador)
            return campo + "&";
        else
            return campo;
    }

    /// <summary>
    /// □□ Alta □□□ huella□Base64 SHA-256□
    /// </summary>
    public static string CalcularHuellaAlta(
        string nifEmisor,
        string numFacturaSerie,
        string fechaExpedicion,
        string tipoFactura,
        string cuotaTotal,
        string importeTotal,
        string huellaAnterior,
        string fechaHoraUsoRegistro)
    {
        string referencia = GetReferenciaRegistroAlta(
            nifEmisor,
            numFacturaSerie,
```

```
            fechaExpedicion,
            tipoFactura,
            cuotaTotal,
            importeTotal,
            huellaAnterior,
            fechaHoraUsoRegistro);

        return GetHashVerifactu(referencia);
    }
}
```

□□□□□

? 1. ?????????????

```
string xml = File.ReadAllText("miFactura.xml");
string huella = VerifactuHashGenerator.CalcularHuellaDesdeXml(xml);

Console.WriteLine("Huella generada = " + huella);
```

□□ 2. VerifactuHashGenerator.cs□□□□□□

```
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Security.Cryptography;
using System.Text;
using System.Xml;

public static class VerifactuHashGenerator
{
    // ============================
    // PUBLIC ENTRY
    // ============================
    public static string CalcularHuellaDesdeXml(string xml)
    {
        XmlDocument doc = new XmlDocument();
        doc.LoadXml(xml);
```

```
        // Detectar tipo de registro (Alta / Evento / Anulación)
        string tipo = DetectarTipoRegistro(doc);

        // Obtener campos según el tipo
        var campos = tipo switch
        {
            "RegistroAlta" => ExtraerCamposRegistroAlta(doc),
            "RegistroEvento" => ExtraerCamposRegistroEvento(doc),
            "RegistroAnulacion" => ExtraerCamposRegistroAnulacion(doc),
            _ => throw new Exception("Tipo de registro no reconocido")
        };

        // Construir cadena ordenada
        string cadena = ConstruirCadena(campos);

        // Calcular hash HEX
        return Sha256Hex(cadena);
    }

    // ==============================
    // DETECCIÓN DE TIPO DE REGISTRO
    // ==============================
    private static string DetectarTipoRegistro(XmlDocument doc)
    {
        if (doc.GetElementsByTagName("RegistroAlta").Count > 0) return "RegistroAlta";
        if (doc.GetElementsByTagName("RegistroEvento").Count > 0) return "RegistroEvento";
        if (doc.GetElementsByTagName("RegistroAnulacion").Count > 0) return
"RegistroAnulacion";

        throw new Exception("No se encontró nodo RegistroAlta / RegistroEvento /
RegistroAnulacion");
    }

    // ==============================
    // CAMPO EXTRACTION (ALTA)
    // ==============================
    private static List<(string, string)> ExtraerCamposRegistroAlta(XmlDocument doc)
    {
        var lista = new List<(string, string)>();
```

```
    lista.Add(("IDEmisorFactura", Get(doc, "IDEmisorFactura")));
    lista.Add(("NumSerieFactura", Get(doc, "NumSerieFactura")));
    lista.Add(("FechaExpedicionFactura", Get(doc, "FechaExpedicionFactura")));
    lista.Add(("TipoFactura", Get(doc, "TipoFactura")));
    lista.Add(("CuotaTotal", NormalizarNumero(Get(doc, "CuotaTotal"))));
    lista.Add(("ImporteTotal", NormalizarNumero(Get(doc, "ImporteTotal"))));
    lista.Add(("Huella", Get(doc, "Huella")));
    lista.Add(("FechaHoraUsoRegistro", Get(doc, "FechaHoraUsoRegistro")));

    return lista;
}

// ==============================
// CAMPO EXTRACTION (EVENTO)
// ==============================
private static List<(string, string)> ExtraerCamposRegistroEvento(XmlDocument doc)
{
    var lista = new List<(string, string)>();

    lista.Add(("IDEmisorFactura", Get(doc, "IDEmisorFactura")));
    lista.Add(("NumSerieFactura", Get(doc, "NumSerieFactura")));
    lista.Add(("TipoEvento", Get(doc, "TipoEvento")));
    lista.Add(("DescripcionEvento", Get(doc, "DescripcionEvento")));
    lista.Add(("Huella", Get(doc, "Huella")));
    lista.Add(("FechaHoraUsoRegistro", Get(doc, "FechaHoraUsoRegistro")));

    return lista;
}

// ==============================
// CAMPO EXTRACTION (ANULACION)
// ==============================
private static List<(string, string)> ExtraerCamposRegistroAnulacion(XmlDocument doc)
{
    var lista = new List<(string, string)>();

    lista.Add(("IDEmisorFactura", Get(doc, "IDEmisorFactura")));
    lista.Add(("NumSerieFactura", Get(doc, "NumSerieFactura")));
    lista.Add(("FechaExpedicionFactura", Get(doc, "FechaExpedicionFactura")));
    lista.Add(("TipoFactura", Get(doc, "TipoFactura")));
```

```
    lista.Add(("MotivoAnulacion", Get(doc, "MotivoAnulacion")));
    lista.Add(("Huella", Get(doc, "Huella")));
    lista.Add(("FechaHoraUsoRegistro", Get(doc, "FechaHoraUsoRegistro")));

    return lista;
}

// ==============================
// XML GETTER
// ==============================
private static string Get(XmlDocument doc, string tag)
{
    var list = doc.GetElementsByTagName(tag);
    if (list.Count == 0) return "";
    return list[0].InnerText.Trim();
}

// ==============================
// STRING BUILDING
// ==============================
private static string ConstruirCadena(List<(string campo, string valor)> lista)
{
    var sb = new StringBuilder();

    for (int i = 0; i < lista.Count; i++)
    {
        bool addAmp = i < lista.Count - 1;
        sb.Append(lista[i].campo)
          .Append("=")
          .Append(lista[i].valor);

        if (addAmp) sb.Append("&");
    }

    return sb.ToString();
}

// ==============================
// NORMALIZACIÓN DE NUMEROS
// ==============================
```

```
    private static string NormalizarNumero(string raw)
    {
        if (decimal.TryParse(raw, NumberStyles.Any, CultureInfo.InvariantCulture, out var d))
            return d.ToString("G29", CultureInfo.InvariantCulture);

        return raw?.Trim();
    }

    // ============================
    // SHA256 → HEX
    // ============================
    private static string Sha256Hex(string input)
    {
        using var sha = SHA256.Create();
        byte[] bytes = sha.ComputeHash(Encoding.UTF8.GetBytes(input));
        var sb = new StringBuilder();

        foreach (var b in bytes)
            sb.Append(b.ToString("X2"));

        return sb.ToString();
    }
}
```

以下是针对西班牙税务局 (AEAT) Verifactu SOAP 服务的完整实现，使用 XML 和客户端证书发送请求：

?????

可直接用于 **Veri*factu**（**AEAT tikeV1.0**）发送 **XML** 的 **C#** 代码模板，包括：

- 加载 **p12** 客户端证书
- 创建 **HttpClient**
- 构造 **SOAP XML**
- 发送到 **AEAT**（测试 / 生产）
- 读取返回结果
- 兼容 **Veri*factu Alta**（**RegistroAlta**）、**Envio**、**Consulta** 等所有操作

1. ??? HttpClient?????

```
using System.Net.Http;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
using System.Text;

public class VeriFactuClient
{
    private readonly HttpClient _client;

    public VeriFactuClient(string certPath, string certPassword, bool ignoreSsl = false)
    {
        var handler = new HttpClientHandler();

        // □□ p12□□
        var cert = new X509Certificate2(certPath, certPassword,
X509KeyStorageFlags.MachineKeySet);
        handler.ClientCertificates.Add(cert);

        // □□ SSL□□□□□□□
        if (ignoreSsl)
        {
            handler.ServerCertificateCustomValidationCallback =
                HttpClientHandler.DangerousAcceptAnyServerCertificateValidator;
        }

        _client = new HttpClient(handler);
    }
```

HASH sha-256 ????

```
using System;
using System.Security.Cryptography;
using System.Text;
```

```
public class GeneradorHuella
{
    // ========================
    // SHA256 HEX (Base16)
    // ========================
    public static string GetHashVerifactu(string msg)
    {
        try
        {
            using (SHA256 sha = SHA256.Create())
            {
                byte[] digest = sha.ComputeHash(Encoding.UTF8.GetBytes(msg));
                return BytesToHex(digest); // □□ Base16
            }
        }
        catch (Exception e)
        {
            throw new ArgumentException("Error al generar la huella SHA", e);
        }
    }

    private static string BytesToHex(byte[] data)
    {
        StringBuilder sb = new StringBuilder(data.Length * 2);
        foreach (byte b in data)
            sb.AppendFormat("{0:x2}", b);
        return sb.ToString();
    }

    // ========================
    // □□ RegistroAlta □□□□□
    // ========================
    public static string GetReferenciaRegistroAlta(
        string nifEmisor,
        string numFacturaSerie,
        string fechaExpedicion,
        string tipoFactura,
        string cuotaTotal,
        string importeTotal,
        string huellaAnterior,
```

```
    string fechaHoraUsoRegistro)
{
    StringBuilder sb = new StringBuilder();

    sb.Append(GetValorCampo("IDEmisorFactura", nifEmisor, true))
      .Append(GetValorCampo("NumSerieFactura", numFacturaSerie, true))
      .Append(GetValorCampo("FechaExpedicionFactura", fechaExpedicion, true))
      .Append(GetValorCampo("TipoFactura", tipoFactura, true))
      .Append(GetValorCampo("CuotaTotal", cuotaTotal, true))
      .Append(GetValorCampo("ImporteTotal", importeTotal, true))
      .Append(GetValorCampo("Huella", huellaAnterior, true))
      .Append(GetValorCampo("FechaHoraUsoGenRegistro", fechaHoraUsoRegistro, false));

    return sb.ToString();
}

// ==========================
// □□ X="value";
// ==========================
public static string GetValorCampo(string nombre, string valor, bool separador)
{
    string campo = nombre + "=" + (valor == null ? "" : valor.Trim());
    return separador ? campo + ";" : campo;
}

// ==========================
// □□□ URL □□
// ==========================
public static string GetValorCampoEncoded(string nombre, string valor, bool separador)
{
    string campo = nombre + "=" + Uri.EscapeDataString(valor ?? "");
    return separador ? campo + ";" : campo;
}

// ==========================
// □□ huellaAlta
// ==========================
public static string CalcularHuellaAlta(
    string nifEmisor,
    string numFacturaSerie,
```

```
        DateTime fechaExpedicion,
        string tipoFactura,
        string cuotaTotal,
        string importeTotal,
        string huellaAnterior,
        string fechaHoraUsoRegistro)
    {
        string fecha = fechaExpedicion.ToString("yyyy-MM-dd"); // Java □ formatea()

        string refStr = GetReferenciaRegistroAlta(
            nifEmisor,
            numFacturaSerie,
            fecha,
            tipoFactura,
            cuotaTotal,
            importeTotal,
            huellaAnterior,
            fechaHoraUsoRegistro
        );

        return GetHashVerifactu(refStr);
    }
}
```

Excel ?? ?????????? ???? ?

ALTA DE REGISTROS								
Operación	Descripción	Operativa	Tipo SIF al que aplica	Condiciones	Consecuencias	Situación en la AEAT (sólo aplica a sistemas VERI*FACTU)		
						No existe registro de facturación	Existe registro de alta	Existe registro de anulación
ALTA DE REGISTRO								
ALTA	· Alta inicial ("normal") del registro de facturación. · Es el alta habitual de un registro de facturación.	· No informar <Subsanacion> o informarlo con valor N · No informar <RechazoPrevio> o informarlo con valor N	· VERI*FACTU · NO VERI*FACTU	El registro de facturación no debe existir previamente en SIF del obligado / AEAT.	Alta del registro de facturación con los nuevos datos.	OK(1)	ERROR (2)	ERROR (2)
ALTA POR RECHAZO	· Alta por rechazo del registro de facturación de alta inicial (y que, por tanto, no existe aún en la AEAT). · Es la subsanación de datos de un registro de facturación de alta inicial, cuando no se exige la emisión de una factura rectificativa (u otro mecanismo contemplado en el Reglamento de Facturación).	· <Subsanacion>=S · <RechazoPrevio>=X	· VERI*FACTU	· La clave única del registro de facturación no debe existir previamente en la AEAT. · El alta previa del registro de facturación fue rechazada.	Alta del registro de facturación con los nuevos datos.			
ALTA DE REGISTRO POR SUBSANACIÓN DE ERRORES								
ALTA DE SUBSANACIÓN	· Alta para la subsanación de un registro de facturación ya generado/remitido anteriormente. · Es la subsanación habitual de un registro de facturación cuando no se exige la emisión de una factura rectificativa (u otro mecanismo contemplado en el Reglamento de Facturación).	· <Subsanacion>=S · No informar <RechazoPrevio> o informarlo con valor N	· VERI*FACTU · NO VERI*FACTU	El registro de facturación debe existir previamente en SIF del obligado / AEAT.	Con él se deja constancia de los nuevos datos que deben ser tenidos en cuenta.	ERROR(3)	OK(4)	OK(5)
ALTA POR RECHAZO DE SUBSANACIÓN	· Alta para la subsanación de un registro de facturación que ya está registrado en la AEAT y cuya subsanación posterior fue rechazada. · Es la subsanación de datos cuando no se exige la emisión de una factura rectificativa (u otro mecanismo contemplado en el Reglamento de Facturación).	· <Subsanacion>=S · <RechazoPrevio>=S	· VERI*FACTU	· La clave única del registro de facturación debe existir previamente en la AEAT. · El alta con subsanación del registro de facturación fue rechazada.	Con él se deja constancia de los nuevos datos que deben ser tenidos en cuenta.			
ALTA DE REGISTRO POR SUBSANACIÓN CUANDO EL REGISTRO ORIGINAL NO EXISTE								
ALTA DE SUBSANACIÓN SIN REGISTRO PREVIO	· Alta para la subsanación de un registro de facturación existente en SIF del obligado (a expedir factura) pero que NO existe en la AEAT porque no se remitió en su momento, por ejemplo, porque funcionaba como NO VERI*FACTU y cambió a VERI*FACTU, que es como funciona ahora; o bien, dependiendo de la implementación del SIF y si así lo necesita, cuando no consta el registro anterior a subsanar en el SIF donde se da de alta la subsanación. · Es la subsanación especial de un registro de facturación cuando no se exige la emisión de una factura rectificativa (u otro mecanismo contemplado en el Reglamento de Facturación).	· <Subsanacion>=S · <RechazoPrevio>=X	· VERI*FACTU · NO VERI*FACTU (opcional)	El registro de facturación no debe existir previamente en la AEAT (o bien, en su caso, en SIF del obligado).	Alta del registro de facturación con los nuevos datos	OK(1)	ERROR (2)	ERROR (2)
ALTA POR RECHAZO DE SUBSANACIÓN SIN REGISTRO PREVIO	· Alta para la subsanación de un registro de facturación que existe en SIF del obligado (a expedir factura) pero que NO existe en la AEAT porque no se remitió en su momento, por ejemplo, porque el SIF funcionaba como NO VERI*FACTU y cambió a VERI*FACTU, que es como funciona ahora, y cuya subsanación se rechazó en la AEAT. · Es la subsanación especial de datos cuando no se exige la emisión de una factura rectificativa (u otro mecanismo contemplado en el Reglamento de Facturación).	· <Subsanacion>=S · <RechazoPrevio>=X	· VERI*FACTU	· La clave única del registro de facturación no debe existir previamente en la AEAT.	Alta del registro de facturación con los nuevos datos			

LEYENDA
OPERATIVA HABITUAL (fondo blanco).
OPERATIVA ESPECIAL (fondo gris).
OK (1) = Se admite el registro de facturación de alta
ERROR (2) = No puede recibirse un registro de facturación de alta de esas características existiendo ya algún registro de facturación de esa factura (bien sea de alta o de anulación).
ERROR (3) = Debería existir ya un registro de facturación para esa factura. Si esta situación de error se produce debido a una secuencia de envío incorrecta, se deberá volver a remitir, sin más, una vez haya entrado —por la vía que sea— el registro de facturación.
OK (4) = El registro remitido sustituye completamente al registro de facturación con los nuevos datos recibidos
OK (5) = Reactiva (vuelve a dejar de alta y en vigor) el registro de facturación anulado existente, sustituyendo completamente el registro de facturación registrado con los nuevos datos recibidos

[illegible]

[illegible] “[illegible] **ALTA**[illegible] ” [illegible] SIF[illegible] Sistema Informático de Facturación[illegible] AEAT [illegible] “Alta”[illegible] “Alta por rechazo”[illegible] “Alta de subsanación”[illegible]

[illegible]

- **Operación**[illegible] Alta[illegible] Alta por rechazo[illegible] Alta de subsanación...[illegible]
- **Descripción**[illegible]
- **Operativa**[illegible] Subsanación / RechazoPrevio[illegible]
- **Tipo SIF**[illegible] VERI*FACTU* [illegible] *VERI*FACTU [illegible]
- **Condiciones**[illegible]
- **Consecuencias**[illegible]
- **Situación en la AEAT**[illegible] AEAT [illegible] → [illegible] OK [illegible] ERROR

[illegible] [illegible] [illegible] VERI*FACTU[illegible]

1. **No existe registro de facturación**[illegible] AEAT [illegible]
2. **Existe registro de alta**[illegible] AEAT [illegible] Alta [illegible]
3. **Existe registro de anulación**[illegible] AEAT [illegible] Anulación [illegible]

底部有 **LEYENDA**（图例），说明 OK(1)、ERROR(2)… 等含义。

第一部分：**ALTA DE REGISTRO**（普通新增）

✔ 用途：

正常开票时的新增登记。

Operativa：

- 不应发送 `<Subsanacion>` 或 `<RechazoPrevio>`

Condiciones：

- AEAT 中不应已有该发票的记录。

结果：

新增发票记录。

决策：

- AEAT 无记录 → **OK**（**1**）
- AEAT 已有 Alta → **ERROR**（**2**）
- AEAT 有 Anulación → **ERROR**（**2**）

ALTA POR RECHAZO（因被拒重新提交）

当你之前发送的记录被 AEAT 拒绝后，需要重新发送，此时为 “Alta por rechazo”。

Operativa：

- `<Subsanacion> = S`
- `<RechazoPrevio> = X`

Condiciones：

- 之前确实被 AEAT 拒绝
- AEAT 中不能存在已注册记录

决策：

- AEAT 无记录 → **Alta OK**
- AEAT 已有 Alta → **ERROR**（**2**）
- AEAT 有 Anulación → **ERROR**（**2**）

□□ ALTA DE SUBSANACIÓN（错误更正的新增）

用于更正已有记录的情况。

类型 1：有原记录且需更正（正常场景）

- AEAT 中应已有此发票记录

决策：

- AEAT 无记录 → **ERROR**（**3**）
- AEAT 有 Alta → **OK**（**4**）
- AEAT 有 Anulación → **OK**（**5**）

类型 2：原记录不存在且允许更正（特殊情况）

用于以下情形，例如：

- 在非 VERI*FACTU 状态下发票未按要求发送
- 系统切换后需补发
- 之前未上报但现在需要以“更正方式”上报

Operativa 可能包含：

- Subsanación = S
- RechazoPrevio = X（可选）

决策：

- AEAT 无记录 → **OK**（**1**）
- AEAT 有 Alta → **ERROR**（**2**）
- AEAT 有 Anulación → **ERROR**（**2**）

□□ ALTA POR RECHAZO DE SUBSANACIÓN（更正也被拒绝后再次提交）

当你发送“更正记录”也被 AEAT 拒绝后，要再次发送。

Operativa：

- Subsanación = S
- RechazoPrevio = X

条件：

- 原记录不应存在于 AEAT

决策同前：

- AEAT无记录 → OK（1）
- AEAT有 Alta → ERROR（2）
- AEAT有 Anulación → ERROR（2）

□□ LEYENDA（图例解释）

代码	含义
OK(1)	可正常新增记录
ERROR(2)	不应新增，AEAT已有 Alta 或 Anulación
ERROR(3)	应该已有记录但不存在（系统顺序错误）
OK(4)	有 Alta → 用收到的新数据更正记录
OK(5)	有 Anulación → 记录恢复为有效并替换数据

□□ 总结（简单理解）

操作类型	**AEAT无记录**	**AEAT有 Alta**	**AEAT有 Anulación**
正常新增	✔	□	□
拒绝后重发	✔	□	□
更正（正常）	□	✔（更正）	✔（恢复+替换）
更正（特殊情况，无原记录）	✔	□	□
更正被拒后再次提交	✔	□	□

如果你需要，我可以为你将此表格转换为程序逻辑（**if/else** 或 **flowchart**），用于实现 Veri*factu 的自动决策。

Revision #29
Created 7 November 2025 09:03:48 by JC TPV
Updated 9 December 2025 14:09:37 by JC TPV