

Veri*Factu ??api??

-  _____

?????

?????

████████████████████

- ██████████
- ██████
- [Web ██████████ v1.0.3 PDF](#) 1.673 KB
- [Web ██████████ v1.0.3████████ PDF](#) 1,715 KB
- [Web █████ WSDL](#)
- [Web █████](#)
- [████████████████████ v0.1.2 PDF](#) 1.11 MB
- [████████████████████ v0.4.7 PDF](#) 1.9 MB
- [████████████████████](#)
 - [████████████████████ v0.1.5 PDF](#) 1.46 MB █████ NO VERIFACTU
 - [ZIP████████████████████ ZIP](#) 5.74 KB
- [████████████████████ v0.2 PDF](#) 576 KB █████ NO VERIFACTU
- [████████████████████ 2025 9 19 PDF](#) 666 KB
- [████████ PDF](#) 708 KB

??????????

████████

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

???? ?????

████████

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

F1	Factura completa (Art. 6, 7.2 y 7.3 del RD 1619/2012)	XXXXXXXXXX XXXXXX	XXXXXXXXXX - XXXXX NIF / NombreRazon - XXXX (Desglose)
F2	Factura simplificada (Art. 6.1.d RD 1619/2012)	XXXXXXXXXX XXXXXXXXXX	XXXX Ticket XXXXXXXXXX
F3	Factura emitida en sustitución de facturas simplificadas	XXXXXXXXXX XX F2XXXXXXXXXX F1 XX	XX XXXXX "XXXX" XX "X ticket XX" XXXXXXXXXX
R1	Factura rectificativa (Art. 80.1 y 80.2 y error fundado en derecho)	XXXXXXXXXX XXXXXXXXXX	Rectificación XXXX 80.1 / 80.2
R2	Factura rectificativa (Art. 80.3)	XXXXXXXXXX XXXXX	XXXXXXXXXX XX
R3	Factura rectificativa (Art. 80.4)	XXXXXXXXXX XX	XXXXXXXXXX
R4	Factura rectificativa (Resto)	XXXXXXXXXX	XXXXXXXXXX
R5	Factura rectificativa simplificada	XXXXXXXXXX	XXXX ticket XX

QR ????? ???? ?

QR XX 40*40 mm

XXXXXXXX QR XXXXXX XXX Factura verificable en la sede electrónica de la AEAT

????????

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

????????

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

XML [] " & " = " [] xia

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

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

????

[] [] [] + [] [] + []

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

- CN [] CERTIFICADO FISICA PRUEBAS -99999910G' [] - [] / [] - []
- SN [] []
- O : [] []
- SERIALNUMBER - [] []
- C []
- NotAfter []

[]

```
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; }           // 名称
    public string TaxNumber { get; set; }     // 税号 / NIF / CIF
    public DateTime NotAfter { get; set; }    // 到期日
    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");

        // 提取序列号
        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();
        }

        // 检测证书类型
        CertType type = DetectType(org, serial, issuer);

        // 根据证书类型和 CN 生成名称
        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
public static void Main()
{
    string pfxPath = @"C:\certs\yourcert.pfx";
    string password = "yourPassword";

    var info = Parse(pfxPath, password);

    Console.WriteLine("证书类型: " + info.Type);
    Console.WriteLine("证书名称: " + info.Name);
    Console.WriteLine("证书编号: " + info.TaxNumber);
    Console.WriteLine("证书过期时间: " + info.NotAfter.ToString("yyyy-MM-dd HH:mm:ss"));
    Console.WriteLine("证书有效期: " + 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> // 001 00 2
  <sum1:TipoUsoPosibleSoloVerifactu>N</sum1:TipoUsoPosibleSoloVerifactu>
  <sum1:TipoUsoPosibleMultiOT>S</sum1:TipoUsoPosibleMultiOT>
  <sum1:IndicadorMultiplesOT>S</sum1:IndicadorMultiplesOT>
</sum1:SistemaInformatico>

```

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

?? 403 ? ???? TLS ?????????????????

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

? .NET Framework 4.0 ????????????? `ServicePointManager.SecurityProtocol` ???
TLS 1.1 ? **TLS 1.2** ???? .NET Framework **4.5** ?????????

????????????

???????????? **TLS 1.1** ? **TLS 1.2**????????????

C#
■

```
using System.Net;

// ????????????????? System.Net.dll

// 1. ????? TLS 1.2 ? TLS 1.1 ???
// Tls12 = 3072 (???? 0xC00)
// Tls11 = 768 (???? 0x300)
const SslProtocols Tls12 = (SslProtocols)3072;
const SslProtocols Tls11 = (SslProtocols)768;
const SslProtocols SystemDefault = (SslProtocols)0; // SystemDefault ? .NET 4.6.2+ ????? 4.0
????

public static void EnableTls12()
{
    // ????? Tls12 ? Tls11???? Tls/Ssl3 ???
    ServicePointManager.SecurityProtocol =
        (SecurityProtocolType)Tls12 |
        (SecurityProtocolType)Tls11 |
        SecurityProtocolType.Tls;
    // SecurityProtocolType.Tls ?? TLS 1.0
}

// ?????????????????
EnableTls12();
```



```

// 启用 TLS 1.2 支持 (强制)
EnableTls12();

HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
request.Method = "POST";
request.ContentType = "application/json"; // 使用 VeriFactu 证书

// 添加客户端证书
request.ClientCertificates.Add(cert);

try
{
    // 发送 alta factura 请求
    using (var stream = request.GetRequestStream())
    {
        // stream.Write(...);
        // 发送 factura XML/JSON 数据
    }

    using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())
    {
        if (response.StatusCode == HttpStatusCode.OK)
        {
            // 成功
            // ...
        }
        // ...
    }
}
catch (WebException ex)
{
    // 检查 Web 错误 403 是否
    if (ex.Response is HttpWebResponse errorResponse && errorResponse.StatusCode ==
HttpStatusCode.Forbidden)
    {
        // 检查 TLS 证书链
        Console.WriteLine("Error 403 Forbidden: Check TLS version and certificate
chain.");
    }
    else

```

```
    {
        // 
    }
}
```

1 TLS 1.2 .NET 4.0 403

2- [????? ?????????? ??????????????????????](#)

/ hash

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)

string

nombreCampo1=valorCampo1&nombreCampo2=valorCampo2&nombreCampoN=valorCampoN

xml

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

```

public class GeneradorHuella
{
    /// <summary>
    ///   SHA-256   Base64   Java
    /// </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("SHA   ", e);
        }
    }

    /// <summary>
    ///   RegistroAlta
    ///   Java
    /// </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();
}
}

```

XXXXXXXXXX

(AEAT) Verifactu SOAP XXXXXXXXXXXX

XML XXXXXXXXXXXX

??????

XXXX

Veri*factu AEAT tikeV1.0

XML C#

XXXX

- p12
- HttpClient
- SOAP XML
- AEAT /
-
- Veri*factu Alta RegistroAlta Envio Consulta

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();

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

        // 忽略 SSL 警告
        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 REGISTRO**██████

✓ ███

██████████

██ Operativa█

- ████ █

██ Condiciones█

- AEAT ██████████

██ ███

███████

██ ███

- AEAT ███ → **OK** █ **1**█
 - AEAT ███ Alta → **ERROR** █ **2**█
 - AEAT █ Anulación → **ERROR** █ **2**█
-

██ **ALTA POR RECHAZO**██████████

██████████ AEAT ██████████ “Alta por rechazo”█

██ Operativa█

- = S
- = X

██ Condiciones█

- ████ AEAT ███
- AEAT ██████████

██ ███

- AEAT ███ → **Alta OK**
 - AEAT ███ Alta → **ERROR** █ **2**█
 - AEAT █ Anulación → **ERROR** █ **2**█
-

00 ALTA DE SUBSANACIÓN 00000000

0000000000

00 1 00000000000000

- AEAT 00000000

000

- AEAT 000 → **ERROR** 3
- AEAT 0 Alta → **OK** 4
- AEAT 0 Anulación → **OK** 5

00 2 00000000000000

00000000

- 00 VERI*FACTU 00000000
- 00000000
- 00000000 "0000" 00

Operativa 0000

- Subsanación = S
- RechazoPrevio = X 0000

000

- AEAT 000 → **OK** 1
- AEAT 0 Alta → **ERROR** 2
- AEAT 0 Anulación → **ERROR** 2

00 ALTA POR RECHAZO DE SUBSANACIÓN 0000000000

0000 "0000" 00 AEAT 00000000

Operativa 0

- Subsanación = S
- RechazoPrevio = X

000

- 00000000 AEAT

0000

