



## Customs & Excise

# Customs & Excise SOAP Web Service Integration Guide

*The information in this document is provided as a guide only and is not professional advice, including legal advice. It should not be assumed that the guidance is comprehensive or that it provides a definitive answer in every case.*

## Contents

Document References .....	3
Abbreviations Used in This Document .....	3
Audience .....	4
1 Introduction .....	5
2 Services specifications.....	6
3 Calling the Services .....	6
3.1 C&E SOAP Endpoints .....	6
3.2 HTTP Content-type Header Values .....	7
4 Interpreting the Responses.....	8
5 Digital Signatures .....	10
5.1 WS-Security Header .....	10
5.1.1 Binary Security Token.....	12
5.1.2 Timestamp.....	12
5.1.3 Signature .....	13
Appendix A – Extracting from a .p12 File .....	15

Version Control		
Version	Date	Change
0.1	04/10/2019	Initial document published
0.2	27/01/2022	Updated reference to AES
0.3	22/07/2025	Updated to reflect changes introduced in 06/09/2025
0.4	27/08/2025	Added section on allowed http content-type values
0.5	05/12/2025	Updated the example in the section on WS-Security Header. Added more clarifications in the Signature section.

## Document References

Reference
1. <i>"Customs &amp; Excise Web Services Common Specification"</i>
2. Full list of ROS errors.

## Abbreviations Used in This Document

Abbreviation	Description
AES	Automated Export System
AIS	Automated Import System
C&E	Customs and Excise
EDE	Excise Duty Entry
EMCS	Excise Movement Control System
EORI	Economic Operators Registration and Identification
ERV	Export Release Verification
NCTS	New Computerised Transit System
PIT	Public Interface Testing
RORO	Roll-On Roll-Off system
ROS	Revenue Online Service
UCC	Union Customs Code

## ***Audience***

This document is for any software provider who has chosen to build or update their products to consume SOAP web services provided by Revenue Customs and Excise division.

## 1 Introduction

This document provides a technical overview of how to integrate with Revenue Customs and Excise SOAP web services, including how to sign requests validly.

It is applicable to the following SOAP web services:

- Handshake
- AIS Submit
- AES Submit
- NCTS (P5) Submit
- EDE Submit
- EMCS Submit
- Mailbox Collect
- Mailbox Acknowledge
- Transaction ID
- Export Release Verification

This document details the SOAP/XML web services specification for Customs and Excise web services. This specification is a result of the modernisation work that was done as part of the UCC introduction. The modernisation includes the adoption of the latest specifications in an industry standard, such as SOAP 1.2 Specification, WS Security 1.1.1 and WSDL 2.0.

This document is designed to be read in conjunction with the SOAP/XML example files as well as the Documents specified in [Document References](#) section, which detail the technical documentation, specification, and examples for the above web services.

This document assumes familiarity with the SOAP/XML web services. The Web Service Description Language (WSDL) is a W3C standard for describing web services. Further details of each web service can be found in the supporting WSDL file. Each file references the necessary schemas and indicates the URL where the services may be accessed. The URL for each web service will use the HTTPS protocol to ensure the privacy of all communication between ROS and the web service client.

## 2 Services specifications

The SOAP web services are described through WSDL files and the schema for each message. Further details of the web services can be found in the published Customs and Excise WSDL files.

The WSDL, SOAP and WS Security version specifications we are following include:

**WSDL 2.0:** [WSDL 2.0](#)

**SOAP 1.2:** [SOAP 1.2](#)

**WS Security 1.1.1:** [WS Security 1.1.1](#)

## 3 Calling the Services

Calls to Customs and Excise web services MUST be digitally signed using a digital certificate that has been previously retrieved from ROS. The signature MUST conform to WS-Security specification, as detailed description can be found in the [Digital Signatures](#) section below.

### 3.1 C&E SOAP Endpoints

The C&E web services provide the following endpoints (Live):

Description	Http Method	Endpoint URL	Additional Information
Handshake	POST	<a href="https://www.ros.ie/customs/webservice/v1/soap/handshake">https://www.ros.ie/customs/webservice/v1/soap/handshake</a>	General handshake to verify connectivity and correctness of digital signature.
AIS Submit	POST	<a href="https://www.ros.ie/customs/webservice/v1/soap/aisSubmit">https://www.ros.ie/customs/webservice/v1/soap/aisSubmit</a>	Automated Import System
AES Submit	POST	<a href="https://www.ros.ie/customs/webservice/v1/soap/aesSubmit">https://www.ros.ie/customs/webservice/v1/soap/aesSubmit</a>	Automated Export System
NCTS (P5) Submit	POST	<a href="https://www.ros.ie/customs/webservice/v1/soap/nctsSubmit">https://www.ros.ie/customs/webservice/v1/soap/nctsSubmit</a>	New Computerised Transit System - Phase 5.

EDE Submit	POST	https://www.ros.ie/customs/webservice/v1/soap/edeSubmit	Excise Duty Entry.
EMCS Submit	POST	https://www.ros.ie/customs/webservice/v1/soap/emcsSubmit	Excise Movement Control System.
Mailbox Collect	POST	https://www.ros.ie/customs/webservice/v1/soap/mailboxCollect	Collect Customs and Excise response. More details available in <i>“Customs &amp; Excise Web Services Common Specification”</i> document.
Mailbox Acknowledge	POST	https://www.ros.ie/customs/webservice/v1/soap/mailboxAcknowledge	Acknowledge the collection of Customs and Excise responses. More details available in <i>“Customs &amp; Excise Web Services Common Specification”</i> document.
Transaction ID (reliable messaging)	POST	https://www.ros.ie/customs/webservice/v1/soap/transactionID	Obtain tokens for Reliable Messaging. More details available in <i>“Customs &amp; Excise Web Services Common Specification”</i> document.
Export Release Verification	POST	https://www.ros.ie/customs/webservice/v1/soap/export/releaseVerification	Submit ERV requests.

### 3.2 HTTP Content-type Header Values

The table below specifies allowed values in HTTP content-type header for SOAP requests:

Content-type value	Purpose
application/soap+xml	All SOAP requests
application/soap+xml; charset=utf-8	All SOAP requests

## 4 Interpreting the Responses

Each web service returns a response message to the client. The type of the response depends on:

- Functional nature of the web service (e.g. AIS Submit vs Mailbox Collect)
- Success vs error response
- Type of errors (Authentication/authorisation error vs functional/validation error)

This is outlined in the table below:

Endpoint	Success Response	Error Response	Notes
Handshake	Message Acknowledgement with Status SUCCESS	Message Acknowledgement with ROS Error	
AIS, AES, NCTS, EDE, EMCS Submit	Message Acknowledgement containing Transaction ID	Message Acknowledgement with ROS Error	ROS Error is returned for authentication, authorisation and internal service errors. Functional errors in submitted messages are returned as responses in Customs Mailbox
Mailbox Collect	Mailbox Collect Response	Message Acknowledgement with ROS Error	ROS Error is returned for authentication, authorisation and internal service errors. There are no functional errors.
Mailbox Acknowledge	Mailbox Acknowledge Response	Message Acknowledgement with ROS Error	ROS Error is returned for authentication, authorisation and internal service errors. There are no functional errors.
Transaction ID (reliable messaging)	Transaction ID Response	Message Acknowledgement with ROS Error	Returned for authentication, authorisation and internal service errors.
Transaction ID (reliable messaging)	Transaction ID Response	Transaction ID Response with functional error	Returned for functional errors.
Export Release Verification	Export Release Verification Response	Message Acknowledgement with ROS Error	Returned for authentication, authorisation and internal service errors.
Export Release Verification	Export Release Verification Response	Export Release Verification Response	Returned for functional errors.

ROS Error codes possible in SOAP responses are presented in the table below.

Error Code	Description
ROS-300-02	Issue with requests media type.
ROS-300-03	Issue with parsing SOAP message. The message might be malformed.
ROS-300-10	Issue with the request's timestamp.
ROS-300-20	Issue with request's digital signature.
ROS-300-30	Issue with request's digest.
ROS-300-40	Certificate holder does not have permissions to submit this request.
ROS-100-00	Unrecognised digital certificate used.
ROS-100-10	Digital certificate used to sign the request is expired.
ROS-100-20	Digital certificate used to sign the request is revoked.
ROS-100-30	Digital certificate used to sign the request is invalid.
FRQ-100-10	Request submitted too soon after the previous one.
REL-100-10	Transaction ID request was invalid.
ROS-300-00	Unexpected error in processing the request. Please try again later.

## 5 Digital Signatures

Any ROS web service request that either returns confidential information or accepts submission of information MUST be digitally signed. This MUST be done using a digital certificate that has been previously retrieved from ROS.

The digital signature MUST be applied to the message in accordance with the WS-Security specification as specified in [Section 5.1](#) below.

The digital signature ensures the integrity of the document. By signing the document, we can ensure that no malicious intruder has altered the document in any way. It can also be used for non-repudiation purposes.

If a valid digital signature is not attached, a “Message Acknowledgement” response containing the relevant ROS Error will be returned. The error code will indicate possible cause of the problem.

### 5.1 WS-Security Header

The security header contains three elements:

1. The Binary Security Token
2. The Timestamp
3. The Signature.

The valid approach for this is using Oasis standards:

- The WS-Security namespace should be:  
*<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd>*
- The WSU namespace should be:  
*<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd>*
- All Id references should now be of the form *wsu:Id* E.g.:  
`<x:myElement wsu:Id="ID1" xmlns="..." xmlns:wsu="..." />`
- The XML Digital Signature namespace should be:  
*<http://www.w3.org/2000/09/xmldsig#>*

Below is a sample SOAP Envelope showing the signature header.

```
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd">
      <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" wsu:Id="X509-id-of-security-
token">MIIFKjCCA3qgAwIBAgIUFP4b... // truncated </wsse:BinarySecurityToken>
      <ds:Signature Id="id-of-signature" xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
            <ec:InclusiveNamespaces PrefixList="soap" xmlns:ec="http://www.w3.org/2001/10/xml-
exc-c14n#" />
          </ds:CanonicalizationMethod>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha512" />
          <ds:Reference URI="#id-of-timestamp">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <ec:InclusiveNamespaces PrefixList="wsse soap"
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" />
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#sha512" />
            <ds:DigestValue>c6nYGWwmu3n... // truncated </ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#id-of-body">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <ec:InclusiveNamespaces PrefixList="" xmlns:ec="http://www.w3.org/2001/10/xml-
exc-c14n#" />
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#sha512" />
            <ds:DigestValue>PktSh+Gh1dTF... // truncated </ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>fp2o4M4Fka70gbC1z... // truncated </ds:SignatureValue>
        <ds:KeyInfo Id="id-of-key-info">
          <wsse:SecurityTokenReference wsu:Id="id-of-security-token-ref">
            <wsse:Reference URI="#X509-id-of-security-token" ValueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </wsse:Security>
    </soap:Header>
  </soap:Envelope>
```

```

    </wsse:SecurityTokenReference>
  </ds:KeyInfo>
</ds:Signature>
<wsu:Timestamp wsu:Id="id-of-timestamp">
  <wsu:Created>2025-12-05T10:32:05.465Z</wsu:Created>
  <wsu:Expires>2025-12-05T12:02:05.465Z</wsu:Expires>
</wsu:Timestamp>
</wsse:Security>
</soap:Header>
<soap:Body wsu:Id="id-of-body" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd">
  <ns2:HandshakeRequest xmlns:ns2="http://www.ros.ie/schemas/customs/handshake"/>
</soap:Body>
</soap:Envelope>

```

### 5.1.1 Binary Security Token

The X509 certificate used to sign the message should be included in the message as a Base64 encoded *BinarySecurityToken* element (*Envelope/Header/Security/BinarySecurityToken*).

The *EncodingType* attribute of the *BinarySecurityToken* should have a value of <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary>.

The *ValueType* attribute should have a value of <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3>.

### 5.1.2 Timestamp

The timestamp element (*Envelope/Header/Security/Timestamp*) must contain:

1. The Created date
2. The Expired date.

Both dates must conform to the following Oasis standard:

[https://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-SOAPMessageSecurity.htm#\\_Toc118717167](https://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-SOAPMessageSecurity.htm#_Toc118717167)  
 and the expired date must be after the created date.

For SOAP requests the maximum time between timestamp creation and expiration is 60 seconds.

### 5.1.3 Signature

This is the signature that is calculated using your ROS digital certificates private key. The signature contains three elements:

1. The SignedInfo element
2. The SignatureValue element
3. The KeyInfo element.

The parts of the SOAP message that must be signed are:

- <Timestamp> element inside the SOAP Header.
- SOAP <Body> element.

Both must be formatted using Exclusive Canonicalization (see [Canonicalization Algorithm](#)) before the digest and signature are generated.

Digest must be calculated for each of the signed elements. The Digest Algorithm must be SHA512 - <http://www.w3.org/2001/04/xmlenc#sha512>.

<b>SignedInfo</b>	<p>The <b>SignedInfo</b> element contains a <b>CanonicalizationMethod</b>, <b>SignatureMethod</b> and two <b>Reference</b> elements.</p> <p><b>Canonicalization:</b> XML Canonicalization is used to format the XML before calculating the digest values. The Canonicalization Algorithm should be Exclusive Canonicalization XML-EXC-C14N with identifier “<a href="http://www.w3.org/2001/10/xml-exc-c14n#">http://www.w3.org/2001/10/xml-exc-c14n#</a>” - <a href="#">Canonicalization Algorithm</a></p> <p><b>Signature Method:</b> The Signature Algorithm should be SHA512withRSA - <a href="http://www.w3.org/2001/04/xmldsig-more#rsa-sha512">http://www.w3.org/2001/04/xmldsig-more#rsa-sha512</a></p> <p>This type of signature is known as a detached signature.</p> <p><b>References:</b> There must be two Reference elements (Envelope/Header/Security/Signature/SignedInfo/Reference) in the SignedInfo element, corresponding to the signed parts of the SOAP message.</p> <ul style="list-style-type: none"> <li>• <b>Body reference</b> – the Reference corresponding to the SOAP Body should have a URI attribute which value is the same as the Id attribute of the SOAP &lt;Body&gt; element.</li> <li>• <b>Timestamp reference</b> – the Reference corresponding to the Timestamp should have a URI attribute which value is the same as the Id attribute of the &lt;Timestamp&gt; element inside the SOAP Header.</li> </ul> <p>The References should specify:</p>
-------------------	---

	<ul style="list-style-type: none"> <li>• Canonicalization algorithm (exclusive canonicalization) in their &lt;Transform&gt; elements. This must also specify lists of namespace prefixes included in the signed element, as per Exclusive XML Canonicalization specification: <a href="https://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/#def-InclusiveNamespaces-PrefixList">https://www.w3.org/TR/2002/REC-xml-exc-c14n-20020718/#def-InclusiveNamespaces-PrefixList</a>.</li> <li>• Digest algorithm should (SHA512) in their &lt;DigestMethod&gt; elements.</li> <li>• Calculated digest value in their &lt;DigestValue&gt; elements.</li> </ul>
<b>SignatureValue</b>	<p>For the SignatureValue (Envelope/Header/Security/Signature/SignatureValue) extract the private key from the .p12 file. Please see <a href="#">Appendix A – Extracting from a .p12 File</a> for instructions on how to do this.</p>
<b>KeyInfo</b>	<p>The KeyInfo contains a SecurityTokenReference element which contains a Reference corresponding to the BinarySecurityToken.          (Envelope/Header/Security/Signature/KeyInfo/SecurityTokenReference/Reference).</p> <p>The URI attribute of the Reference should reference the Id attribute of the BinarySecurityToken element. For example, if the Id attribute of the BinarySecurityToken is “X509Token”, the URI attribute of the Reference sub-element should be “#X509Token”.</p>

## Appendix A – Extracting from a .p12 File

Each customer of ROS will have a digital certificate and private key stored in an industry standard PKCS#12 file.

In order to create a digital signature, the private key of the customer must be accessed. A password is required to retrieve the private key from the P12 file. This password can be obtained by prompting the user for their password.

The password on the P12 is not the same as the password entered by the customer. It is in fact the MD5 hash of that password, followed by the Base64-encoding of the resultant bytes.

To calculate the hashed password, follow these steps:

1. First get the bytes of the original password, assuming a "Latin-1" encoding. For the password "Baltimore1," these bytes are: 66 97 108 116 105 109 111 114 101 49 44 (i.e. the value of "B" is 66, "a" is 97, etc.).
2. Then get the MD5 hash of these bytes. MD5 is a standard, public algorithm. Once again, for the password "Baltimore1," these bytes work out as: 223 238 161 24 62 121 39 143 115 167 51 163 245 231 226 94.
3. Finally, create the new password by Base64-encoding the bytes from the previous step. For example, the password, "Baltimore1," this is "3+6hGD55J49zpzOj9efiXg==".

This new password can then be used to open a standard ROS P12 file.