United Nations ECE/TRANS/WP.30/GE.1/2021/4

Economic and Social Council

Distr.: General
3 December 2020

Original: English

Economic Commission for Europe

Inland Transport Committee

Working Party on Customs Questions affecting Transport

Group of Experts on Conceptual and Technical Aspects of Computerization of the TIR Procedure

First session
Geneva, 27–29 January 2021
Item 6 (a) of the provisional agenda
eTIR international system:
Progress report on the development of the eTIR international system

eTIR web services – Overview, security and access*

Note by the secretariat

I. Introduction - Mandate

1. The Inland Transport Committee during its eighty-second session (23–28 February 2020) approved (ECE/TRANS/294, para 84[1]) the establishment of the Group of Experts on Conceptual and Technical Aspects of Computerization of the TIR Procedure (WP.30/GE.1) and endorsed its ToR[2] (ECE/TRANS/WP30/2019/9 and ECE/TRANS/WP.30/2019/9/Corr.1) pending approval by UNECE Executive Committee (EXCOM). EXCOM during its Remote informal meeting of members of the Executive Committee (20 May 2020) approved the establishment of the Group of Experts on Conceptual and Technical Aspects of Computerization of the TIR Procedure (WP.30/GE.1) until 2022, based on the terms of reference included in document ECE/TRANS/WP.30/2019/9 and Corr.1, as contained in document ECE/TRANS/294 (ECE/EX/2020/L.2, para 5(b))[3].

* This document was submitted late for processing since clearance in finalizing this document took longer than anticipated.

[1] Decision of the Inland Transport Committee para 84 / ECE/TRANS/294 www.unece.org/fileadmin/DAM/trans/doc/2020/itc/ECE-TRANS-294e.pdf

[2] Terms of reference of the newly established Group approved by the Inland Transport Committee and the Executive Committee (EXCOM) of UNECE www.unece.org/fileadmin/DAM/trans/bcf/wp30/documents/2019/ECE-TRANS-WP30-2019-09e.pdf and corrigendum www.unece.org/fileadmin/DAM/trans/bcf/wp30/documents/2019/ECE-TRANS-WP30-2019-09c1e.pdf

[3] Decision of EXCOM , ECE/EX/2020/L.2 / para 5(b) www.unece.org/fileadmin/DAM/commission/EXCOM/Agenda/2020/Remote_informal_mtg_20_05_2020/Item_4_ECE_EX_2020_L.2_ITC_Sub_bodies_E.pdf

2. The terms of reference of the Group stipulate that the Group should focus its work on preparing a new version of the eTIR specifications, pending the formal establishment of TIB. More specifically the Group should (a)prepare a new version of the technical specifications of the eTIR procedure, and amendments thereto, ensuring their alignment with the functional specifications of the eTIR procedure; (b)prepare a new version of the functional specifications of the eTIR procedure, and amendments thereto, ensuring their alignment with the conceptual specifications of the eTIR procedure; (c)prepare amendments to the conceptual specifications of the eTIR procedure, upon requests by WP.30.

3. The current document presents an overview of the eTIR web services, how to access them, the details on the security aspects of the eTIR procedure and the technical aspects of the implementation and test of the eTIR messages. This document is valid for the eTIR international system version 1.0 based on the eTIR specifications version 4.3a.

II. Purpose

4. This document describes the eTIR international system web services, in particular: the intended target audience, the system architecture, the various messages and their sequencing, the security aspects, the accesses as well as the support contacts. It does not cover the detailed implementation required for each message, which can be found in the documents dedicated to each of the eTIR messages. Rather, it describes which components and processes should be put in place in the national customs systems to effectively interconnect with the eTIR international system.

III. Target audience

5. This guide is intended for the customs ICT team in charge of TIR processes and in charge of interconnecting with their national customs systems with the eTIR international system.

IV. Prerequisites

6. This document is to be read after having an understanding of the eTIR concepts,[4] and after having read and followed the Project guidelines for customs to connect to eTIR.[5] It is most important to understand the implementation stages described in the above guidelines, and to understand where you currently stand in the implementation process.

7. In order to ensure an implementation that delivers the best value and services to the customs authorities, we highly recommend the ICT team to be accompanied by a TIR subject matter expert, as mentioned in the Project guidelines for customs to connect to eTIR.

V. eTIR documentation walkthrough

8. The eTIR international system relies on the TIR Convention, and as such the eTIR documentation relies on its articles and annexes, but also on various other key documents available online and introduced below. Some of these documents must be read and well understood while others are reference documents to be consulted when necessary. This section will help you understand what they contain and in which order we recommend reading them.

(a) Project guidelines for Customs to connect to eTIR: this document is the starting point for the customs authorities of any contracting party and should be read before any other one.

[4] www.unece.org/fileadmin/DAM/trans/bcf/adhoc/conc_tech/documents/id17-06e.pdf

[5] wiki.unece.org/download/attachments/106299939/Project Guidelines for customs to connect to the eTIR international system.pdf

If you are not familiar with the transport and border crossing facilitation problematics, we highly encourage you to read:

(b) The TIR Convention handbook[6]: this document contains all the legal elements that rule the TIR Convention (including Comments and Explanatory Notes). It also describes the processes to be followed by customs authorities, national associations and TIR Carnet holders.

(c) Annex 11 to the TIR Convention[7]: Annex 11 to the TIR Convention describes the eTIR procedure, how processes shall be adapted to be computerized and set the legal provisions for the implementation and usage of the eTIR international system.

9. Once the TIR legal context and key processes are well understood, we strongly advise to read:

(d) The introduction to the eTIR conceptual, functional and technical documentation:[8] this document introduces the conceptual, functional and technical documentation for the TIR Procedure Computerization Project in accordance with the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) Modelling Methodology. It is the first document to be read in order to approach how eTIR was envisioned as an extension of the TIR Convention.

(e) The eTIR concepts:[9] this document describes the approach and core concepts used to support the business logic, and to implement the eTIR international system. This document is also a reference document that describes all the use cases and all the business rules used for the technical implementation.

(f) The eTIR functional specifications:[10] this document is the most important to read to get a deep understanding of the mechanisms used to implement the eTIR international system. The current version (4.2) is being expanded in a working document ({etir-spec-version}) and refined as the work progresses and as feedback is received from modelling work carried out by the Informal Ad hoc Expert Group on Conceptual and Technical Aspects of Computerization of the TIR Procedure (GE.1) and all the eTIR focal points. These specifications (once called Reference Model) also refer to the following additional documents and lists:

(i) eTIR XML schemas[11]

(ii) eTIR code lists[12]

(iii) eTIR error code list[13]

(g) Finally, if you wish to contact other contracting parties during this implementation, please refer to the eTIR Focal Points page[14] to find their contact information.

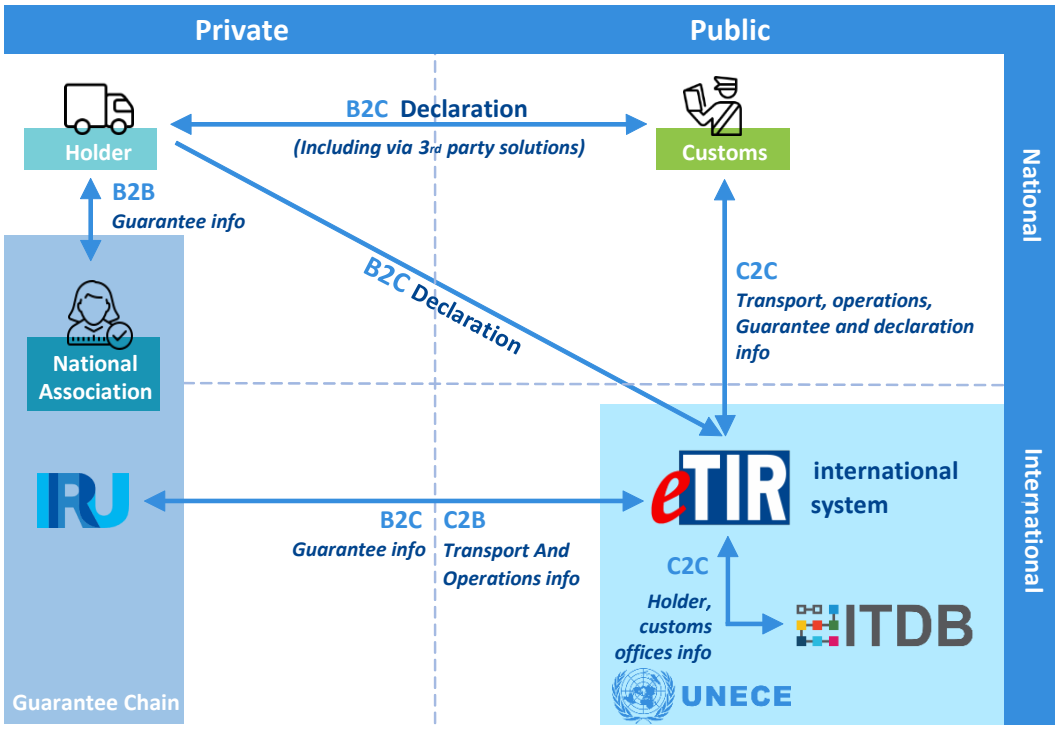
VI. eTIR web service overview

A. High level architecture

10. The eTIR international system is based on the following functional overview diagram:

[6] www.unece.org/tir/tir-hb.html
[7] www.unece.org/fileadmin/DAM/trans/bcf/ac2/documents/2020/ECE-TRANS-WP30-AC2-147e.pdf#page=12
[8] www.unece.org/fileadmin/DAM/trans/bcf/adhoc/conc_tech/documents/id17-05e.pdf
[9] www.unece.org/fileadmin/DAM/trans/bcf/adhoc/conc_tech/documents/id17-06e.pdf
[10] www.unece.org/fileadmin/DAM/trans/bcf/adhoc/conc_tech/documents/id17-07e.pdf
[11] wiki.unece.org/display/ED/Technical+artefacts
[12] www.unece.org/fileadmin/DAM/trans/bcf/eTIR/documents/CodeLists0_4.pdf
[13] wiki.unece.org/display/ED/Error+Management
[14] www.unece.org/trans/bcf/etir/focals.html

Figure I
eTIR high-level architecture



* Kindly note the following acronyms meanings:
B2B: Business to Business relationship (where business refers to the private sector).
C2B: Customs to Business relationship (where business refers to the private sector).
B2C: Business to Customs relationship (where business refers to the private sector).
C2C: Customs to Customs relationship.

11. Technology stack: the eTIR international system has been implemented using the following technologies:

- Java programming language[15]
- PostgreSQL database[16]
- Spring framework[17]
- ActiveMQ[18]
- SOAP-XML web services[19]
- Apache CFX[20]
- Apache Camel[21]

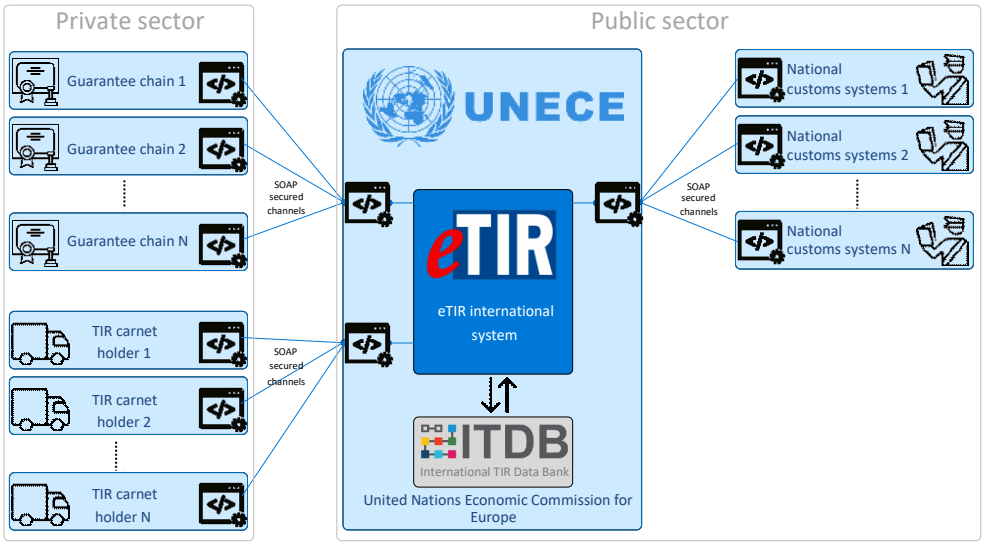
12. Although these technologies are only listed for information, if the reader has any questions on the implementation methods and or on the technology stack used to implement the eTIR international system, please contact the eTIR service desk at etir@un.org.

13. The eTIR international system exposes secured web service endpoints with both the public and the private sector partners following the diagram below:

Figure II

[15] docs.oracle.com/javase/7/docs/technotes/guides/language/
[16] www.postgresql.org/
[17] spring.io/
[18] activemq.apache.org/
[19] www.w3.org/TR/soap/
[20] cxf.apache.org/
[21] camel.apache.org/

eTIR endpoints



14. Note that at the moment there is only one guarantee chain in activity (namely the International Road Transport Union) but that the TIR Convention does not limit it to one.

B. Interface message overview

15. The list of eTIR messages are categorized internally, based on the message sender/addressee and based on message direction (eTIR centric):

- All messages codes start either:
  - by "I" for Internal (Internal to the public sector, meaning between the eTIR international system and either national customs systems or ITDB)
  - by "E" for External (External to the public sector, meaning between eTIR and either a guarantee chain or a TIR Carnet holders)
- All messages work by pair (request-response), and their codes all end either:
  - by an "odd number" for the initiating/calling/request message
  - by an "even number" for the response/acknowledgment message

16. Find below the list of all internal and external messages:

External messages	*Internal messages*
E1 - Register guarantee	**I1 - Accept guarantee**
E2 – Register results	*I2 - Acceptance results*
E3 – Cancel guarantee	**I3 - Get holder information**
E4 – Cancellation results	*I4 - Holder information*
E5 – Query guarantee	**I5 - Query guarantee**
E6 – Query results	*I6 - Query results*
E7 – Notify guarantee chain	**I7 - Record declaration data**
E8 – Notification confirmation	*I8 - Record declaration data results*
E9 - Advance TIR data	**I9 - Start TIR operation**
E10 - Advance TIR data results	*I10 - Start results*
E11 - Advance amendment data	**I11 - Terminate TIR operation**

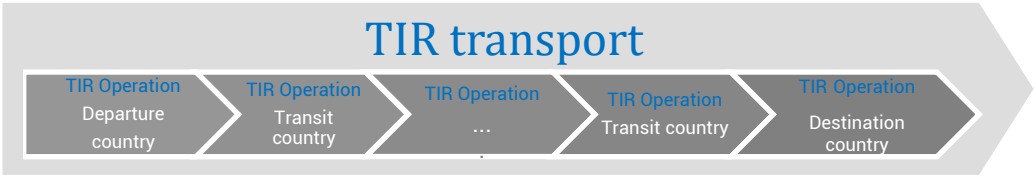
External messages	*Internal messages*
E12 - Advance amendment data results	*I12 - Termination results*
E13 - Cancel advance data	**I13 - Discharge TIR operation**
E14 - Cancel advance data results	*I14 - Discharge results*
	I15 - Notify customs
	I16 - Notification confirmation
	I17 - Refusal to start TIR operation
	I18 - Refusal results
	I19 - Check customs offices
	I20 - Customs offices validation

17. The eTIR international system aims to ensure the secure exchange of data between the national customs systems related to the international transit of goods, vehicles or containers according to the provisions of the TIR Convention and to allow customs to manage the data on guarantees, issued by guarantee chains, to holders authorized to use the TIR system.

C. TIR transport and TIR operation in eTIR

18. The diagram below highlights the important concepts of TIR transport and TIR operations and gives an example of them:

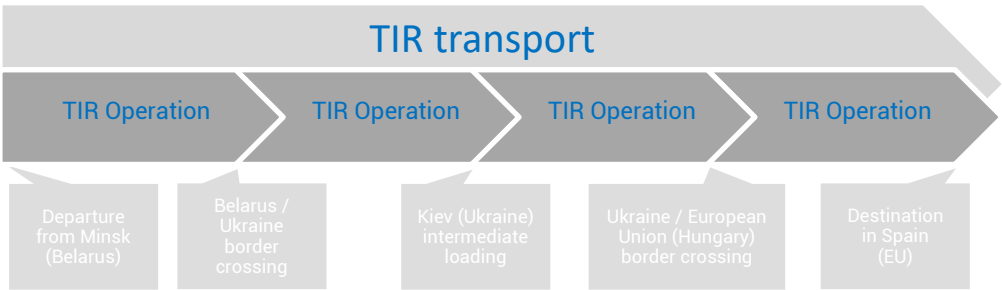
Figure III
Summary of a TIR transport



19. It is important to note that a TIR transport may have multiple loading and unloading points, and of course multiple border crossings. Each of them separates the TIR operations. Also note that the border crossings separating the TIR operations are between each customs territories (that can be a country, or a common customs territory like in case of the European Union). More details can be found in the dedicated page of the eTIR introduction document[22].

Figure IV
Example of a TIR transport

The boundaries and names shown and the designations used on this map do not imply official endorsement or acceptance by the United Nations.



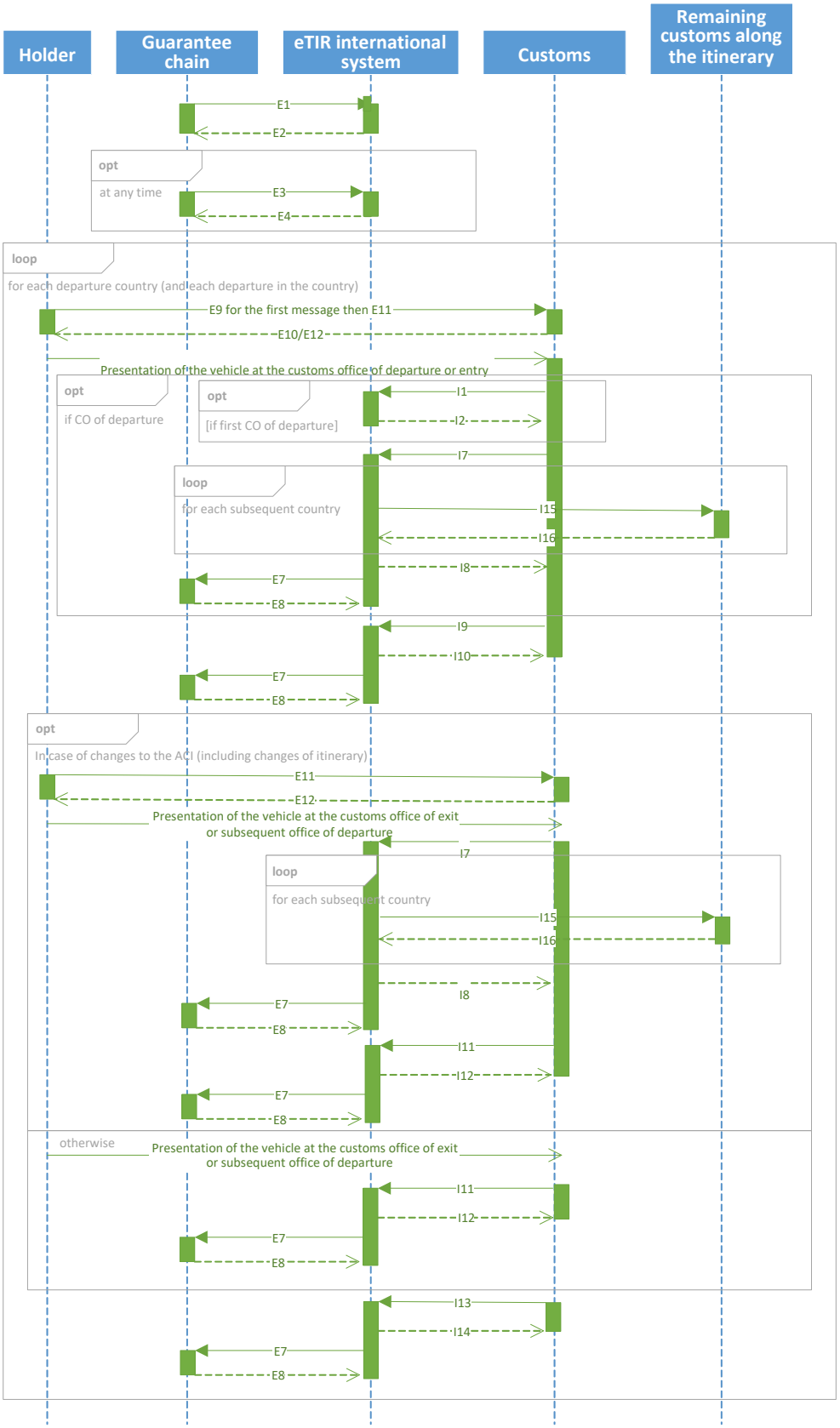
D. eTIR sequence diagrams

20. The usage of the eTIR messages is described in the following eTIR sequence diagrams for countries of departure, transit and destination.

[22] www.unece.org/fileadmin/DAM/tir/handbook/english/newtirhand/TIR-6Rev11e.pdf#page=21

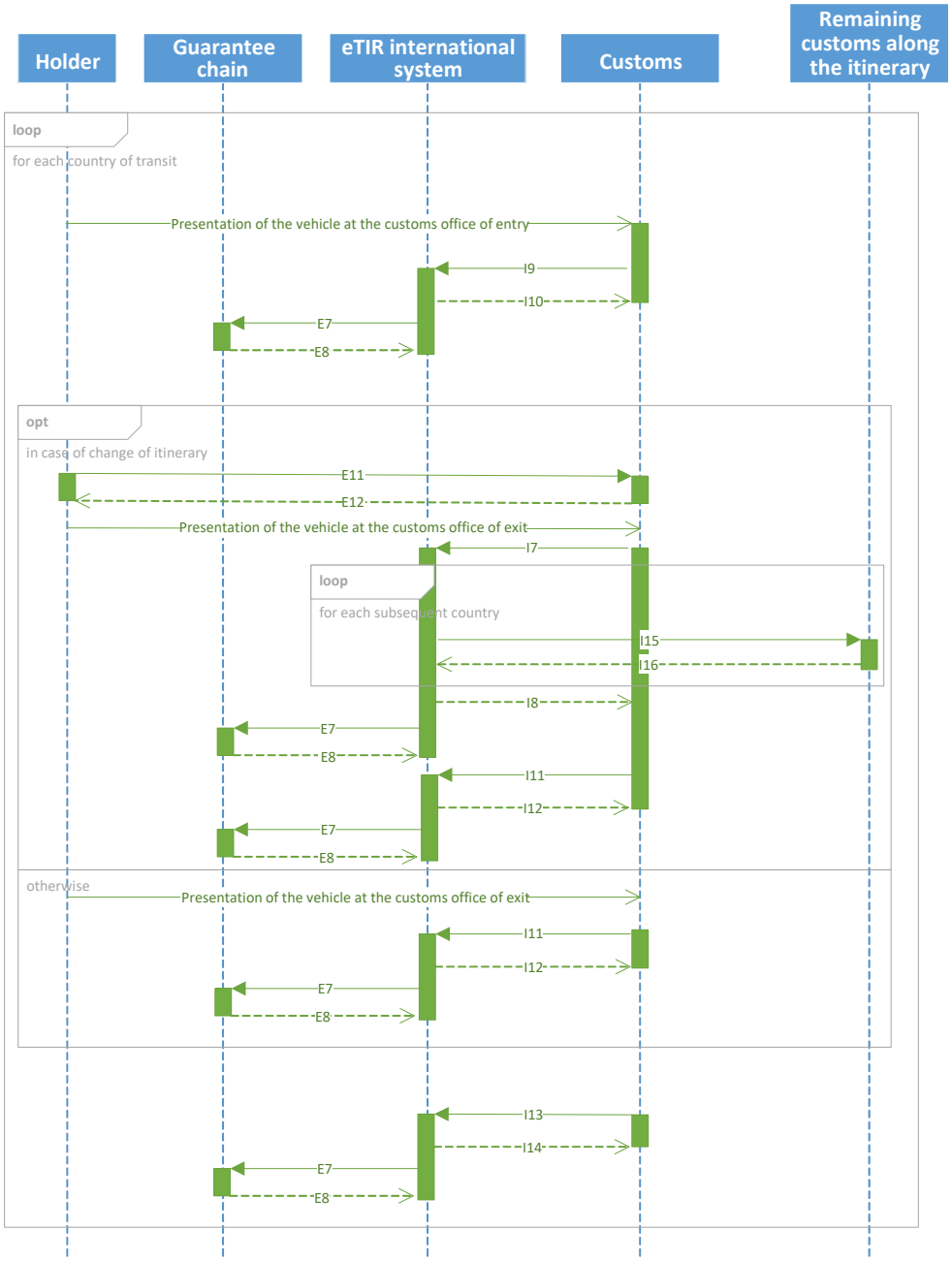
1. Message sequence for countries of departure

Figure V
Time sequence diagram – countries of departure



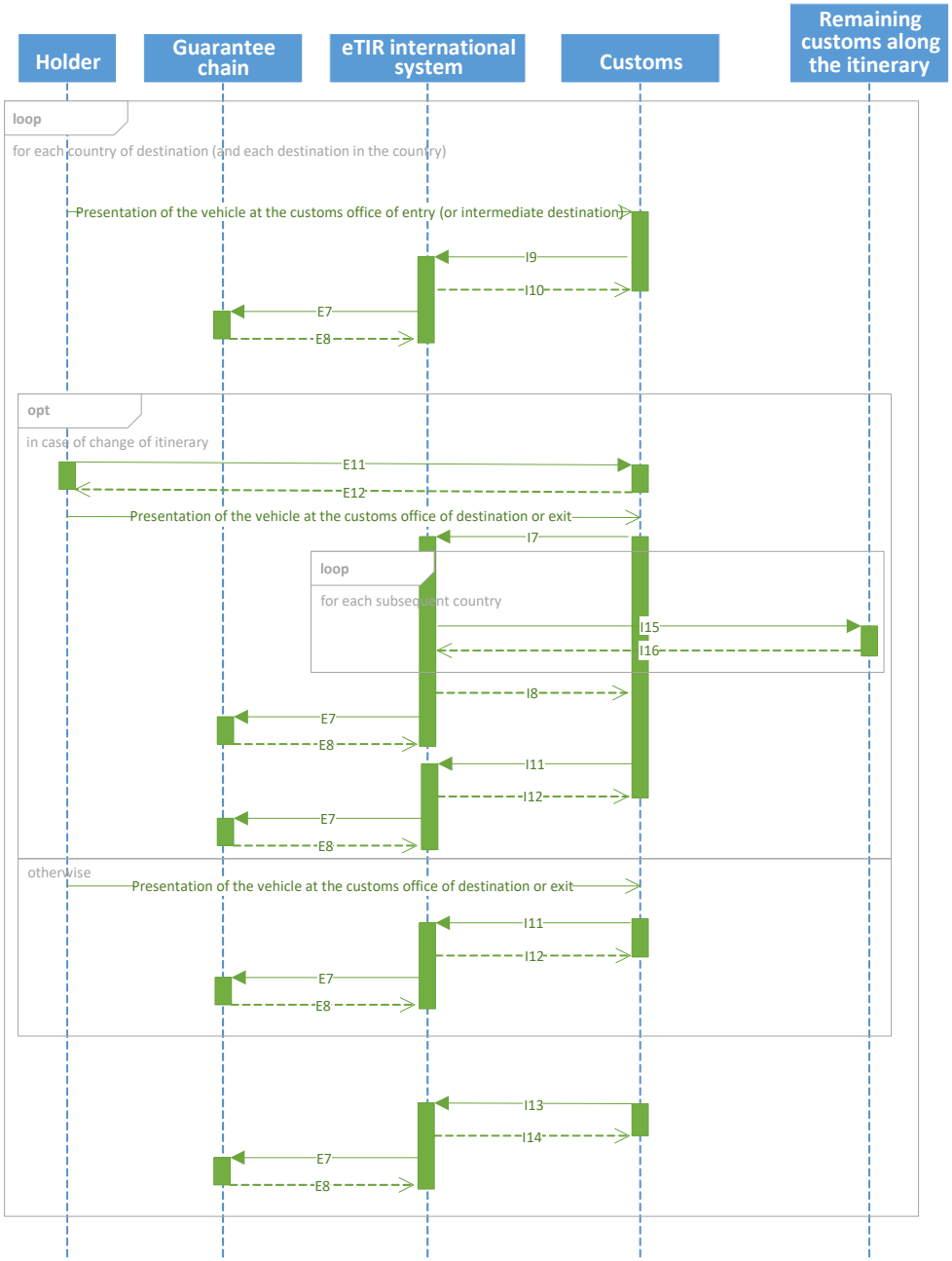
2. Message sequence for countries of transit

Figure VI
Time sequence diagram – countries of transit



3. Message sequence for countries of destination

Figure VII
Time sequence diagram – countries of destination



VII. Security aspects

21. The eTIR international system web services uses WS-Security to electronically sign the messages. WS-Security is not used to provide encryption of the message. This framework has been released as a full industry-recognized recommendation in March 2004.

22. Digital signature, using X.509 certificates (version 3), are used to identify the caller to the web service and provide non-repudiation capability. An encryption protocol (TLS v1.2 or v1.3) is used to send the messages over HTTPS to ensure the confidentiality of the information exchanged in the messages. It should be noted that the digital signature and the encryption protocol (TLS) use different asymmetric key pairs.

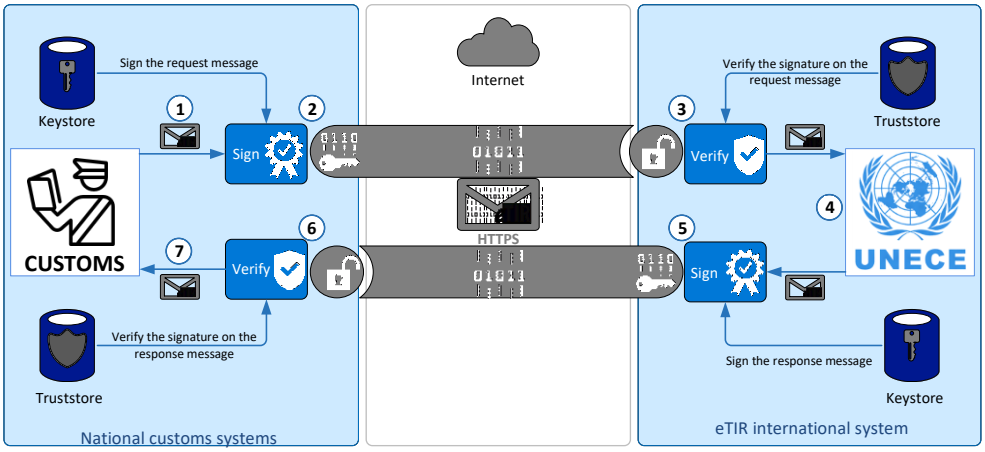
23. The next paragraphs describe the web services, security related terms and concepts that are used throughout this document. They often refer on the terms "Web Services Security

(WS-Security)", "X.509 token", "keystore" and "truststore" which are explained in detail in the Glossary.

A. eTIR security model

24. The figure below depicts the WS-Security model and this section provides an overview illustrating how this security model works.

Figure VIII
eTIR security model



25. Kindly note that if this diagram represents most of the use cases, in a few other cases the eTIR international system actually play the role of client, and the national customs systems, the role of service provider.

26. In the example above, as a first step, the X.509 certificate of the national customs systems will be installed in the eTIR international system truststore. Similarly, the eTIR international system certificate will be installed in the national customs systems truststore. This mandatory initial step allows the validation of the digital signatures that are transferred as security tokens in all SOAP messages exchanged in the context of the eTIR procedure.

27. Kindly find below a description of how a request is sent by the national customs systems to the eTIR international system, and how the related response is sent back:

(a) The national customs system generates a request message to be sent to the eTIR international system web service.

(b) The request message is signed with the private key of the national customs systems' X.509 certificate (stored in the keystore) and sent using an encryption protocol (TLS) over HTTPS.

(c) The eTIR international system receives the request message, verifies the signature of the message using the public key of the sender to authenticate it and to confirm its integrity.

(d) The eTIR international system processes the request message and generates a response message in return.

(e) The response message is signed with the private key of the eTIR international system (stored in the keystore) and sent using an encryption protocol (TLS) over HTTPS.

(f) The national customs system receives the response message, and verifies the signature of the message using the public key of the service provider to authenticate it and to confirm its integrity.

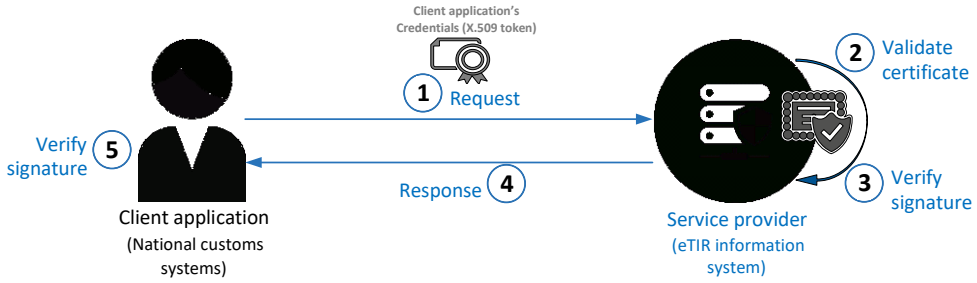
28. The completion of this process illustrates the implementation of four security aspects that the eTIR international system wishes to achieve with this security model: authentication, integrity, confidentiality and non-repudiation.

B. Authentication, Integrity and Non-repudiation

29. Authentication is used to ensure that the different parties of a transaction are really who they claim to be; thus, a proof of identity is required. This proof can be claimed in various ways. One simple example is to provide a user ID together with a secret password. A more secured approach is to use an X.509 certificate issued from a trusted Certificate Authority (later referred as "CA" in this section). The X.509 certificate identify the end user. In addition to the authentication feature, the private key to the X.509 certificate is also used to sign SOAP messages. This electronic signature not only ensures the identity of the sender but also guarantees that the message content has not been tampered during the transmission, thus ensuring integrity. The sender uses his private key to sign the message (the hash of the message to be more precise) and the recipient of the message uses the sender's public key to verify the signature, thus providing proofs of non-repudiation.

1. Authentication steps using an X.509 certificate

Figure IX
Authentication steps



- The client application (national customs system) sends a message to the service provider (eTIR international system - Step #1). The message includes the client application's credentials, signed with the private key that is paired with the public key in the client application's X.509 certificate. The service provider validates the certificate by performing a number of checks (Step #2), including:
- Verifying that the certificate has not expired. If the expiration date of the certificate is past due, then the certificate is no longer valid.
- Verifying that the certificate is internally consistent. The service checks that the data in the certificate has not been tampered with by verifying the certificate contents against the signature of the issuing CA.
- Verifying the issuing CA of the client's X.509 certificate. This is done by comparing the issuer's signature on the client application's X.509 certificate with the X.509 certificate of the issuing CA. For this step to work properly, the CA that issued the client's X.509 certificate must be trusted by both the client application and the service provider.
- Verifying that the issuing CA has not revoked the certificate. The service provider checks this by making sure that the X.509 certificate does not appear on a Certificate Revocation List (CRL) published by the issuing CA. The service can check the revocation status of the certificate by directly accessing it from the CA or by checking against a CRL that was previously downloaded from the issuing CA to the certificate repository used by the service to look up X.509 certificates.
- The service provider uses the public key in the client application's X.509 certificate to verify the client application's signature (Step #3). This allows the service provider to authenticate the client application and ensure that the signed data has not been tampered with after the message was signed.

30. If the authentication is unsuccessful, the service provider then sends back an HTTP 500 error message. If it is successful, it will process the content of the request and produce

the appropriate response message following the eTIR specifications (Step #4) that will, upon reception, require the same checks described above (Step #5).

2. Key pairs generation

31. Customs can obtain a public and private key pair from a trusted certificate authority or create self-signed ones. They should generate an RSA key pair and provide to the UNECE TIR secretariat team the public part (X.509 certificate) of the generated key and keep well protected in their keystore the private key. See in the Annexes the description of the two ways to generate an RSA key pair.

C. Confidentiality

32. The HyperText Transfer Protocol Secure (HTTPS), used to access the eTIR international system endpoints, is an extension of the HyperText Transfer Protocol (HTTP) where communication is encrypted using Transport Layer Security (TLS), a cryptographic protocol designed to provide communications security over public networks like the Internet. The principal reason for using HTTPS is the need to ensure confidentiality and integrity of the exchanged data while in transit. It protects against information security threats like the Man-in-the-middle attack. The bidirectional encryption of communications between a client and server protects against from eavesdropping and tampering of the communication. Therefore, communication between the eTIR international system and the national customs systems is secured by the bidirectional encryption using TLS (at least v1.2).

VIII. Access to eTIR web services

A. Prerequisites

1. Network requirements

33. The national customs systems must have access to the Internet and all the domains of the User Acceptance Testing and Production environments described in the section below should be white-listed by the Network engineers of the customs authorities. In addition, in the eTIR international system context, SOAP is implemented to use HTTPS on port TCP/8083, therefore this protocol and port should be opened in the firewalls of the customs authorities to be usable by the national customs systems.

2. Certificates

34. As explained in the Security aspects section, the connection between the eTIR international system and the national customs systems requires both parties to share their X.509 certificates and to mutually store them in their truststores.

35. This procedure must be completed for each client-service environment pair during the implementation phase, the national customs systems User acceptance Testing (UAT) environment should be connected to the eTIR international system UAT environment. Once ready for production, both parties Production environments should exchange their dedicated Production X.509 certificates to allow for communication between them and usage of the eTIR procedure when the launch is confirmed.

36. Both systems (the eTIR international system and national customs systems) should use different X.509 certificates for their respective UAT and Production environments. Also note that if this documentation refers to methods to generate locally certificates using an open source application that may be useful for test and development purposes, the certificates generated and shared by the customs ICT team is entirely up to their choice so long as it respects the security requirements and aspects mentioned in this document.

37. In addition to the generated X.509 certificate, the ECE server certificate (unece.org.cer) will be used in the encryption mechanism (HTTPS) of the message.

Depending on your case, you may need to retrieve and store it in your truststore or your application may retrieve it automatically (preferred option).

38. During the certificate generation process explained in the KeyStore: step by step generation of key pairs section, the email field should be correctly filled as it will be used by the eTIR international system to send email notifications.

3. Whitelisting servers

39. The servers of the national customs systems first need to be whitelisted by the ECE IT security team to allow the connection to the servers of the eTIR international system. To do this, the IT experts of the customs authorities need to provide the TIR secretariat with all the IP addresses of the servers of the national customs systems which will send request to the eTIR international system. Similarly, the TIR secretariat stands ready to provide the IP addresses of its servers to perform the same operation on the customs authorities side.

B. Test and production URLs

40. The list below provides the information about the web service end-points both for the production and UAT environments as well as the accompanying URLs for the Web Service Description Language (WSDL) files:

- UAT web service base URL

  etir-uat-01.unece.org/etir/v4.3/customs

- WSDL for Customs Authorities

  etir-uat-01.unece.org/etir/v4.3/customs?wsdl

- WSDL for Guarantee Chains

  etir-uat-01.unece.org/etir/v4.3/guaranateeChain?wsdl

IX. Implementation and test of eTIR messages

A. Recommended general approach

1. Purpose

41. In this section, the TIR secretariat wishes to share the development processes followed to ensure a timely and qualitative delivery of the eTIR international system, hoping that by doing so, the reader may in turn be able to benefit from our lessons learned.

2. General approach

42. The TIR secretariat adopted an Agile approach following the principles of the Agile manifesto. The members of the IT team meets every week to review the work accomplished during the previous week, to discuss about work to be done during the current week and to mention any potential impediment on which other team members could bring their help. To support the work of the TIR secretariat, an internal Knowledge Management System (KMS) features the following components:

- An issue-tracking system which manages all tasks to be done, which offers excellent traceability and accountability;
- A documentation platform which hosts all development, managerial and operational aspects of the eTIR project;
- A source code management system which hosts the code repositories of the TIR information systems.

3. Continuous integration

43. The TIR secretariat also adopted best practices from the DevOps community. In particular, we recognize the important added value of automating as many processes as possible in the software development lifecycle to relieve human beings from mundane tasks, increase reliability by reducing the probability of human errors, and drastically increasing productivity.

44. In order to have a system as reliable as possible, we deploy regularly our new code through our continuous integration (CI) pipeline. We also rebuild our entire code base every time new code is committed to our code repository, to ensure that at any given time, every completed functionality works as intended.

4. Extensive testing

45. We use automated a static code analysis tool to check on a daily basis our source code against sets of rules and best practices created by the IT industry community. This allows to ensure a high quality of the source code, and a continuous training of the members of the IT team.

46. We also have a target of 70% of our functional code base covered by unit tests, as well as a comprehensive suite of functional non-regression tests written with Apache JMeter. By combining these two types of tests, we ensure that every part of the eTIR international system works as intended, and we protect ourselves from possible regressions when adding new code to our code base.

B. Tool kit: list of what we believe is useful

47. The list below represents the various softwares that are believed to be useful for the implementation of a client application to the eTIR international system and for which the TIR secretariat has experience with:

- SOAP-UI testing suite: a useful tool for performing ad-hoc tests on SOAP messages (note that you can find in annex the SOAP UI quick guide);
- JMeter testing suite: a useful tool for automating message testing and covering a large scale of scenarios;
- GIT version control system: the industry leading version control system, ensuring that all eTIR international system code and other configuration items are properly versioned;
- IntelliJ IDEA: one of the industry leading Integrated Development Environment (IDE) fitting the eTIR international system technology stack, that proved to enhance our productivity on various aspects;
- SonarQube static code analysis tool: software used for continuous inspection of the code quality that performs automatic reviews with static analysis of the code to detect bugs, so-called "code smells" (which are coding bad practices), and security vulnerabilities.

48. Kindly note that installing and using these applications is not required. However, those tools have proven to be useful to the TIR secretariat team during the implementation of the eTIR international system.

C. SOAP Header security generation

49. The eTIR international system relies on an exchange of web messages using the SOAP protocol v1.2. As a SOAP extension, WS-Security provides the SOAP header element titled "Security", which is designed to act as a container to store all security related information for SOAP request and response messages. Here is how the **Security** element is defined in the WS-Security schema.

1. Security Elements

50. The wsse:Security header element provides a mechanism for attaching security-related information targeted at a specific recipient in the form of a SOAP actor/role. This element represents the signing steps the message producer took to create the message. This prepending rule ensures that the receiving application can process sub-elements in the order they appear in the wsse:Security header element, because there will be no forward dependency among the sub-elements.

Signature element

The **Signature** element is the root element of an XML Signature

Schema Definition

```
<element name="Signature" type="ds:SignatureType"/>
  <complexType name="SignatureType">
    <sequence>
      <element ref="ds:SignedInfo"/>
      <element ref="ds:SignatureValue"/>
      <element ref="ds:KeyInfo" minOccurs="0"/>
      <element ref="ds:Object" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
</complexType>
```

In message example

```
<ds:Signature Id="SIG-AD473EF9595256C9D115409733594021 73"
        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-ENV cus urn"
              xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          </ds:CanonicalizationMethod>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          <ds:Reference URI="#id-AD473EF9595256C9D115409733594011 72">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <ec:InclusiveNamespaces PrefixList="cus urn"
                  xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>H2Ai···</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>bY65hsJdkxh40···
        </ds:SignatureValue>
        <ds:KeyInfo Id="KI-AD473EF9595256C9D115409733594011 70">
          <wsse:SecurityTokenReference wsu:Id="STR-AD473EF9595256C9D115409733594011 71">
            <wsse:KeyIdentifier 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">
              G9w0BAQsFAAN...
            </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
```

SignatureValue element

51. The SignatureValue element contains the information about the sender's public key which is needed for the eTIR international system to validate the digital signature of the caller. It is always encoded using base64.

Schema Definition

```
<element name="SignatureValue" type="ds:SignatureValueType" />

  <complexType name="SignatureValueType">
    <simpleContent>
      <extension base="base64Binary">
        <attribute name="Id" type="ID" use="optional"/>
      </extension>
    </simpleContent>
  </complexType>
```

In message example

```
<ds:SignatureValue>
    bY65hsJdkxh40⋯
</ds:SignatureValue>
```

52. The SignatureValue element contains a signature that only covers the SignedInfo element: only the content of this SignedInfo element is included into the signature digest.

SignedInfo element

53. The SignedInfo element describes which part of the message was used to generate the signature. It has an attribute Id pointing to the SOAP body part of the signed message and we will explain it in more detail in the next element (Reference). The structure of the SignedInfo element includes the canonicalization algorithm, a signature algorithm, and one or more references. The content of the SignedInfo element can be divided into two parts: information about the SignatureValue and information about the application content, as we can see in the following XML Schema fragment:

Schema definition

```
<element name="SignedInfo" type="ds:SignedInfoType"/>

  <complexType name="SignedInfoType">
    <sequence>
      <element ref="ds:CanonicalizationMethod"/>
      <element ref="ds:SignatureMethod"/>
      <element ref="ds:Reference" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
  </complexType>
```

In message example

```
<ds:SignedInfo>
  <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
    <ec:InclusiveNamespaces PrefixList="SOAP-ENV cus urn"
      xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
  </ds:CanonicalizationMethod>
  <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
  <ds:Reference URI="#id-AD473EF9595256C9D11540973359401172">
    <ds:Transforms>
      <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
        <ec:InclusiveNamespaces PrefixList="cus urn"
          xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      </ds:Transform>
    </ds:Transforms>
    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
    <ds:DigestValue>H2Ai⋯</ds:DigestValue>
  </ds:Reference>
</ds:SignedInfo>
```

54. Canonicalization, or C14N (exclusive XML Canonicalization), is the process of picking one path through all the possible output options, so that sender and receiver can generate the exact same byte value, no matter what intermediate XML software might be involved. The SignedInfo/CanonicalizationMethod element specifies how to reconstruct the exact byte stream. The SignedInfo/SignatureMethod element specifies what type of algorithm (e.g., Kerberos or RSA) is used to generate the signature. Together, these two elements describe how to create the digest, and how to protect it from any modification.

55. In a java-based client, this can be configured as following:

```
org.apache.wss4j.dom.message.WSSecSignature wsSecSignature = new WSSecSignature();
//http://www.w3.org/2001/10/xml-exc-c14n#
wsSecSignature.setSigCanonicalization(WSConstants.C14N_EXCL_OMIT_COMMENTS);
//http://www.w3.org/2000/09/xmldsig#rsa-sha1
wsSecSignature.setSignatureAlgorithm(WSConstants.RSA_SHA1);
```

Reference element

56. The Reference element is used to refer to another content. It contains a digest of the content, a description of how that digest was generated (e.g. SHA1), and a specification of how the content should be transformed before the digest is generated. The transformations provide flexibility on how the XML signature is built.

Schema definition

```
<element name="Reference" type="ds:ReferenceType"/>

  <complexType name="ReferenceType">
    <sequence>
      <element ref="ds:Transforms" minOccurs="0"/>
      <element ref="ds:DigestMethod"/>
      <element ref="ds:DigestValue"/>
    </sequence>
    <attribute name="Id" type="ID" use="optional"/>
    <attribute name="URI" type="anyURI" use="optional"/>
    <attribute name="Type" type="anyURI" use="optional"/>
  </complexType>
```

In message example

```
<ds:Reference URI="#id-AD473EF9595256C9D11540973359401172">
  <ds:Transforms>
    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
      <ec:InclusiveNamespaces PrefixList="cus urn"
        xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
    </ds:Transform>
  </ds:Transforms>
  <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
  <ds:DigestValue>H2Ai…</ds:DigestValue>
</ds:Reference>
```

57. The DigestMethod element specifies the hashing algorithm, and the DigestValue element is the value of the hash of the content encoded in base64. The key part of the Reference element is the set of transforms that may be used. The Transforms element is a list of Transform elements, each of which specifies a transformation step.

Transforms element

58. The schema defines an array of transforms, with one XPath element that has a defined structure:

Schema definition

```
<element name="Transforms" type="ds:TransformsType"/>
  <complexType name="TransformsType">
    <sequence>
      <element ref="ds:Transform" maxOccurs="unbounded"/>
    </sequence>
  </complexType>

<element name="Transform" type="ds:TransformType"/>
  <complexType name="TransformType" mixed="true">
    <choice minOccurs="0" maxOccurs="unbounded">
      <any namespace="##other" processContents="lax"/>
      <element name="XPath" type="string"/>
    </choice>
    <attribute name="Algorithm" type="anyURI" use="required"/>
  </complexType>
```

59. The content in a Transform element will depend on the Algorithm attribute. For example, if simple XML is being signed, then there will most likely be a single Transform element that specifies a C14N algorithm:

In message example

```
<ds:Transforms>
  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
    <ec:InclusiveNamespaces PrefixList="cus urn"
      xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
  </ds:Transform>
</ds:Transforms>
```

KeyInfo element

60. The last step is to identify the signer, or at least the key that generated the signature (the key that protects the digest from being modified). This task is ensured by the KeyInfo element:

Schema definition

```
<element name="KeyInfo" type="ds:KeyInfoType"/>

  <complexType name="KeyInfoType" mixed="true">
    <choice maxOccurs="unbounded">
      <element ref="ds:KeyName"/>
      <element ref="ds:KeyValue"/>
      <element ref="ds:RetrievalMethod"/>
      <element ref="ds:X509Data"/>
      <element ref="ds:PGPData"/>
      <element ref="ds:SPKIData"/>
      <element ref="ds:MgmtData"/>
      <any processContents="lax" namespace="##other"/>
    </choice>
    <attribute name="Id" type="ID" use="optional"/>
  </complexType>
```

In message example

```
<ds:KeyInfo Id="KI-AD473EF9595256C9D11540973359401170">
  <wsse:SecurityTokenReference wsu:Id="STR-AD473EF9595256C9D11540973359401171">
    <wsse:KeyIdentifier 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">
              G9w0BAQsFAAN...
    </wsse:KeyIdentifier>
  </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

SecurityTokenReference element

61. The digital signature operation requires that a key be specified. The element containing the key in question may be located elsewhere in the message or completely outside the message. The SecurityTokenReference element provides a mechanism for referencing security tokens and other key bearing elements. The SecurityTokenReference element provides an open content model for referencing key bearing elements. It should contain either a keyIdentifier element or a X509Data element. If we don't specify the KeyIdentifier configurations by default wsse:SecurityTokenReference will contain a X509Data element and both are accepted.

62. The following configuration will set the keyIdentifier element instead of using a X509Data element:

```
org.apache.wss4j.dom.message.WSSecSignature wsSecSignature = new WSSecSignature();
wsSecSignature.setKeyIdentifierType(WSConstants.X509_KEY_IDENTIFIER);
```

KeyIdentifier element

63. A token should be referenced using a KeyIdentifier element. The following table list both encoding and value types.

URI Fragment	URL
#Base64Binary	*http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary*
#X509v3	*http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3*

In message example

```
<ds:KeyInfo Id="KI-AD473EF9595256C9D11540973359401170">
  <wsse:SecurityTokenReference wsu:Id="STR-AD473EF9595256C9D11540973359401171">
    <wsse:KeyIdentifier 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">
        G9w0BAQsFAAN...
    </wsse:KeyIdentifier>
  </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

X509Data element

Schema definition

```
<complexType name="X509IssuerSerialType">
  <sequence>
    <element name="X509IssuerName" type="string"/>
    <element name="X509SerialNumber" type="integer"/>
  </sequence>
</complexType>
```

64. X.509 certificates are supported through the ds:X509Data element. This element allows the signer to embed its certificate (encoded in Base64), or any other alternative methods to verify the certificate: a subject's name, the issuer's name and serial number, the key identifier, or another format. The signer can also include a current copy of the Certificate Revocation List (CRL), to show that the signer's identity was valid at the time the document was signed.

2. SOAP eTIR message example

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:cus="etir:v4.3:customs" xmlns:etir=
"etir:I5:v4.3">
  <soap:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <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">
      <ds:Signature Id="SIG-AD473EF9595256C9D11540973359402173"
        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 wsa cus etir"
              xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          </ds:CanonicalizationMethod>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          <ds:Reference URI="#id-AD473EF9595256C9D11540973359401172">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <ec:InclusiveNamespaces PrefixList="cus etir"
                  xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
              </ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>H2Ai···</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>bY65hsJdkxh40···
        </ds:SignatureValue>
        <ds:KeyInfo Id="KI-AD473EF9595256C9D11540973359401170">
          <wsse:SecurityTokenReference wsu:Id="STR-AD473EF9595256C9D11540973359401171">
            <wsse:KeyIdentifier 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">
              G9w0BAQsFAAN...
            </wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
    <wsa:Action>etir:v4.3:customs/queryGuarantee</wsa:Action>
    <wsa:MessageID>uuid:8a20af11-8170-495d-9563-6a89b32ef745</wsa:MessageID>
  </soap:Header>
  <soap:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id=
"id-942cd702-1de3-4f27-bfcd-6628ab5d3143">
    <cus:queryGuarantee>
      <etir:InterGov>
        <etir:FunctionCode>9</etir:FunctionCode>
        <etir:ID>656dbce8-e810-44ec-8f3d-5f8a9b425d99</etir:ID>
        <etir:TypeCode>I5</etir:TypeCode>
        <etir:ReplyTypeCode>00</etir:ReplyTypeCode>
        <etir:ObligationGuarantee>
          <etir:ReferenceID>XC95xxxxxx</etir:ReferenceID>
        </etir:ObligationGuarantee>
      </etir:InterGov>
    </cus:queryGuarantee>
  </soap:Body>
</soap:Envelope>
```

D. Implementation of the Message Identifier / Functional Reference

1. General

65. All messages sent and received are uniquely identified using the Message Identifier field. This field must be set by the Sender in the request message. The Receiver will set another unique value for the Message Identifier field of the response message. In addition to that, the Receiver will also set the Functional Reference field of the response message with the value of the Message Identifier field of the related request message. This method allows a full traceability of the Request/Response messages.

2. Request message

66. The Receiver must set the Message Identifier field of the message using the following format:

```
SenderID:UUID
```

67. Where:

- SenderID: a unique value identifying the sending entity. This value should be the same as the one set in the Sender identification field in the message.
- UUID: a universally unique identifier v4 as detailed in RFC 4122. Version 4 is based on pseudo-random numbers hence the need to keep the SenderID to decrease the probabilities of collisions between messages sent by various eTIR stakeholders.

68. The main programming languages provide native helper classes to generate a UUID v4:

In Java:

```
java.util.UUID.randomUUID();
```

In C#:

```
System.Guid.NewGuid();
```

69. Here are some samples of valid values for the **Message Identifier** field:

Sample 1:

```
<urn:ID>CustomsCountryA:6aca5f82-2285-4f00-b4ae-36269d4cc865</urn:ID>
```

Sample 2:

```
<urn:ID>eTIRInternationalSystem:1486e5b7-c6ae-4d27-b794-44c4bf545fb3</urn:ID>
```

3. Response message

70. The Receiver receives the request message from the Sender and stores the value of the Message Identifier field. When preparing the response message, the Receiver will set a new value for the Message Identifier field of that message following the same way as described in the section above. Then the stored value of the Message Identifier field of the request message will be set to the Functional reference of the response message being prepared.

E. Implementation of the date fields

71. The eTIR messages contain several fields in which dates have to be entered. For some of these fields, the format only includes a date and for some others, it also includes the time. This section explains more in detail how to properly populate these fields.

1. Fields with dates only

72. In the XML Schema Definition (XSD) files, this type is named EtirDateType and is defined as follows, along with the types it inherits from.

EtirDateType definition

```
<xs:complexType name="EtirDateType">
  <xs:simpleContent>
    <xs:extension base="ds:DateTimeType_102_S">
      <xs:attribute name="formatCode" use="optional">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="102"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:simpleType name="DateTimeType_102_S">
  <xs:restriction base="ds:DateTimeType_S">
    <xs:pattern value="[1-9][0-9][0-9][0-9](([0][1|3|5|7|8])([0][1-9]|[1-2][0-9]|[3][0-1])|([0][4|6|9])([0][1-9]|[1-
2][0-9]|[3][0])|([0][2])([0][1-9]|[1-2][0-9])|([1][0|2])([0][1-9]|[1-2][0-9]|[3][0-1])|([1][1])([0][1-9]|[1-2][0-
9]|[3][0]))"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="DateTimeType_S">
  <xs:restriction base="xs:string">
    <xs:pattern value=".{1,35}"/>
  </xs:restriction>
</xs:simpleType>
```

73. The format of this type of date is aligned on the UN/EDIFACT format code 102: CCYYMMDD

With:

- CCYY: the year on four digits. Examples: 1979, 2020;
- MM: the month on two digits from 01 to 12 starting with 01 for January;
- DD: the day of the month on two digits from 01 to 31.

Samples:

- 01 January 1970 is coded as "19700101";
- 29 February 2020 is coded as "20200229";
- 31 December 2045 is coded as "20451231".

74. The date field also contains an optional attribute named formatCode which value is therefore always "102". With this format, there is no notion of time zone and the date has to be regarded as valid in all time zones.

The following XML code show a sample date only field.

Expiration of a guarantee on 01 August 2024

```
<ExpirationDateTime formatCode="102">20240801</ExpirationDateTime>
```

2. Fields with dates and times

75. In the XML Schema Definition (XSD) files, this type is named EtirDateTimeType and is defined as follows, along with the types it inherits from.

EtirDateTimeType definition

```
<xs:complexType name="EtirDateTimeType">
  <xs:simpleContent>
    <xs:extension base="ds:DateTimeType_208_S">
      <xs:attribute name="formatCode" use="optional">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="208"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:simpleType name="DateTimeType_208_S">
  <xs:restriction base="ds:DateTimeType_S">
    <xs:pattern value="[1-9][0-9][0-9][0-9](([0][1|3|5|7|8])([0][1-9]|[1-2][0-9]|[3][0-1])|([0][4|6|9])([0][1-9]|[1-
2][0-9]|[3][0])|([0][2])([0][1-9]|[1-2][0-9])|([1][0|2])([0][1-9]|[1-2][0-9]|[3][0-1])|([1][1])([0][1-9]|[1-2][0-
9]|[3][0]))(([0-1][0-9])|(2[0-3]))[0-5][0-9](([0-5][0-9]|60))[\-+]((0[0-9])|(1[0-4]))[0-5][0-9]"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="DateTimeType_S">
  <xs:restriction base="xs:string">
    <xs:pattern value=".{1,35}"/>
  </xs:restriction>
</xs:simpleType>
```

76. The format of this type of date and time is aligned on the UN/EDIFACT format code 208: CCYYMMDDHHMMSSZHHMM

With, defined sequentially:

- CCYY: the year on four digits. Examples: 1979, 2020;
- MM: the month on two digits from 01 to 12 starting with 01 for January;
- DD: the day of the month on two digits from 01 to 31;
- HH: the hour of the day on two digits from 00 (for midnight) to 23 (for eleven PM);
- MM: the minutes of the day on two digits from 00 to 59;
- SS: the seconds of the day on two digits from 00 to 59. 60 is also allowed in the case of a leap second;
- Z: the introduction of the time zone with either a '' or a '-'. If the time zone has no offset, either '-' or '' can be used;
- HH: the hours of the offset of the time zone from 01 to 14;
- MM: the minutes of the offset of the time zone from 00 to 59.

Samples:

- 01 January 1970 00:00:00 in London (Time offset: +00:00) is coded as "19700101000000+0000";
- 29 February 2020 09:45:36 in New York (Time offset: -05:00) is coded as "20200229094536-0500";
- 31 December 2045 22:06:59 in South Tarawa, Kiribati (Time offset: +14:00) is coded as "20451231220659+1400".

77. The date and time field also contains an optional attribute named **formatCode** which value is therefore always "208".

The following XML code show a sample date and time field.

Acceptance of a guarantee on 01 July 2021 10:03:42 in Istanbul (Time offset +03:00).

```
<ExpirationDateTime formatCode="102">20210701100342+0300</ExpirationDateTime>
```

F. Error management

1. Introduction to error management

78. When the eTIR international system receives and processes a message, it performs a series of validations on the message itself, in the context of the related guarantee, holder or transport and issues a response to the system which has sent the message in the first place. If anything goes wrong during these validation and processing steps, a list of errors is sent back in the response. Each of these errors is presented as an Error code with a Pointer which can be used to point towards a specific XML element of the message. The list of all the error codes is available on the Error code (eTIR) page.

2. Presentation of error codes

79. The list of error codes is specific to eTIR as it allows IT teams to better understand errors while implementing the interconnection to the eTIR international system. This should result in a faster implementation overall and more accurate processing of the errors from the system sending messages to the eTIR international system. Furthermore, a detailed error code system will also greatly simplify the communication between the stakeholders and the eTIR service desk, in case of an incident, to identify and correct the underlying problem. The list of error codes is based on the best practices from the IT industry. Like the list of HTTP status codes, all error codes have three digits, and the first digit of the status code defines the type of errors:

- **1XX - Validation**: validation of the message and if its parameters;
- **2XX - Workflow**: workflow related problems;
- **3XX - Functional**: other functional problems;
- **4XX - Internal**: eTIR international system internal problems.

80. Each type of error has a default error code which indicates, at least, the type of the error if the system cannot send a more explicit error.

3. Sample errors

81. Below is an example of how a single error is returned:

Missing required element.

```
<ns14:Error>
   <ns14:ValidationCode>101</ns14:ValidationCode>
   <ns14:Pointer>
      <ns14:SequenceNumeric>1</ns14:SequenceNumeric>
      <ns14:Location>/InterGov/ObligationGuarantee/ReferenceID</ns14:Location>
   </ns14:Pointer>
</ns14:Error>
```

82. Here the ValidationCode is the error code and the DocumentSectionCode element inside the Pointer element points towards the problematic element of the request using the XPath syntax.

83. When multiple errors of the same type are returned, the eTIR international system returns a single error element, with a list of pointer elements.

Example of missing multiple required element.

```
<ns14:Error>
   <ns14:ValidationCode>101</ns14:ValidationCode>
   <ns14:Pointer>
      <ns14:SequenceNumeric>1</ns14:SequenceNumeric>
      <ns14:Location>/InterGov/ObligationGuarantee/ReferenceID</ns14:Location>
   </ns14:Pointer>
   <ns14:Pointer>
      <ns14:SequenceNumeric>2</ns14:SequenceNumeric>
      <ns14:Location>/InterGov/ObligationGuarantee/Surety/ID</ns14:Location>
   </ns14:Pointer>
</ns14:Error>
```

84. Finally, as an example, please find below a complete response to an I1 - Accept guarantee message containing multiple errors:

Example of a complete I2 - Acceptance results message body with multiple errors returned

```
<soap:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
          wsu:Id="id-30706360-7e18-4764-b080-05364eb55892">
    <ns3:acceptanceResults xmlns:ns4="etir:I2:v4.3" xmlns:ns5="etir:MetaData_DS:v4.3" >
        <ns4:InterGov>
            <ns4:FunctionCode>27</ns4:FunctionCode>
            <ns4:FunctionalReferenceID>bc26c1b8-7392-4d44-9899-317fd72206eb</ns4:FunctionCode>
            <ns4:TypeCode>I2</ns4:TypeCode>
            <ns4:Error>
               <ns5:ValidationCode>102</ns5:ValidationCode>
               <ns5:Pointer>
                  <ns5:SequenceNumeric>1</ns5:SequenceNumeric>
                  <ns5:Location>/InterGov/FunctionCode</ns5:Location>
               </ns5:Pointer>
               <ns5:Pointer>
                  <ns5:SequenceNumeric>2</ns5:SequenceNumeric>
                  <ns5:Location>/InterGov/TypeCode</ns5:Location>
               </ns5:Pointer>
            </ns4:Error>
            <ns4:Error>
               <ns5:ValidationCode>101</ns5:ValidationCode>
               <ns5:Pointer>
                  <ns5:SequenceNumeric>3</ns5:SequenceNumeric>
                  <ns5:Location>/InterGov/ObligationGuarantee/ReferenceID</ns5:Location>
               </ns5:Pointer>
               <ns5:Pointer>
                  <ns5:SequenceNumeric>4</ns5:SequenceNumeric>
                  <ns5:Location>/InterGov/ObligationGuarantee/Surety/ID</ns5:Location>
               </ns5:Pointer>
            </ns4:Error>
        </ns4:InterGov>
    </ns3:acceptanceResults>
</soap:Body>
```

4. Error handling

85. Each national customs system which interconnects with the eTIR international system needs to properly handle the errors returned in the response message. When implementing the various pairs of eTIR messages, developers will find it convenient to refer to the eTIR error codes page, especially to the 'Observations' column in the tables, to see which error codes could be raised. When receiving messages from the eTIR international system, errors should be handled in such a way that the relevant information is transmitted to the relevant national customs systems. As all errors are critical and mean a failure to process the message, the appropriate follow-up actions should be performed by the users of the national customs systems.

86. The same goes for the national customs systems which may send back errors to the eTIR international system when receiving requests that they cannot process (this might be the case for the following pairs of messages: I15/I16, E9/E10, E11/E12 and E13/E14). The national customs systems should follow the same specifications as detailed above when errors have to be reported to the eTIR international system when sending response messages.