

ASOG Receive Web Service Interface API Specification

Version 5.1

Document Number: ASR-UOM-05.1

March 30, 2025

**COMMERCIALLY SENSITIVE
REPRODUCTION PROHIBITED**

Copyright 2025 RazorFlow

All rights reserved. All information and data contained in this document is CONFIDENTIAL and PROPRIETARY to RazorFlow and its affiliates and intended only for use of authorized user to whom this copy is assigned by RazorFlow. The user shall not copy, reproduce, transmit, distribute, disclose or use the information or data contained in it, or any portion of it, except as expressly permitted in writing by RazorFlow nor permit anyone else to do the same. The recipient shall be held fully liable for any wrongful disclosure or use of any of the information contained in this document by him, or his officers or employees or anyone else to whom he makes the information available.

Exchange Path™ Interconnection Gateway are trademarks of RazorFlow and the Lumine Group Inc.

All other names or marks may be trademarks of their respective owners.

Revision No.	Date	Prepared By	Description of Changes
3.3	11/22/2014	Chandhini S	1. Supported services have been updated. 2. Security Requirements have been updated to include Security Header support. 3. tML Header is mandatory on the Request and the tML Header fields that are supported have changed. 4. Samples have been removed from document and will be provided on request. 5. Additional SOAP Fault Codes and Messages have been added.
4.0	12/5/2014	Chandhini S	Baseline Version.
4.1	5/9/2015	Chandhini S	1. SOAP Ack updated from http 200 to tML OK in the sequence diagrams. 2. SOAP Structure sample for ASR Response updated.
4.2	7/28/20215	Chandhini S	1. Added error codes 2. Updated tML Header requirements
5.0	7/28/2015	Chandhini S	Baseline Version
5.1	3/30/2025	Jackie Gerber	Updated for RazorFlow

TABLE OF CONTENTS

1	Introduction	4
1.1	Referenced Documents:	4
1.2	System requirements	4
1.3	Definitions.....	4
2	Supported Services.....	4
2.1	Supported ASR Order Services (Asynchronous services).....	4
2.2	Supported ASR Preorder Services (Synchronous services).....	5
2.3	Supported Access Order REQTyps	5
2.4	Supported Access Order responses	6
2.5	Security Requirements	7
2.5.1	Encryption.....	7
2.5.2	Authentication.....	7
3	ASR Receive Web Service interface architecture	8
4	WSDL and ASR FLOW.....	9
4.1	ASR Request – Receive WSDL.....	9
4.2	ASR Response – Send WSDL	11
4.3	tML Header Requirements.....	12
4.3.1	tML Header Schema	14
4.4	ASR Order Receive - Positive / Successful scenario.....	18
4.5	ASR Order Receive – Negative/fault scenario	21
4.6	ASR Order Receive – Negative/Reject Scenario.....	22
4.7	ASR Order response Send – Negative/fault scenario	24
4.8	ASR Pre-Order Receive - Positive / Successful scenario	25
4.9	ASR Preorder Receive - Negative / fault scenario.....	26
5	SOAP Structure	27
5.1	ASR Pre-Order - Synchronous request/response.....	27
5.1.1	ASR Pre-Order synchronous request	27
5.1.2	ASR Pre-Order synchronous response.....	27
5.1.3	ASR Pre-Order synchronous soap fault response.....	28
5.2	ASR Order Receive - Asynchronous request/response	28
5.2.1	ASR Order asynchronous soap request (from ILEC Customer).....	28
5.2.2	ASR Order asynchronous soap response	29
5.2.3	ASR Order Synchronous tML OK response.....	29
5.2.4	ASR Order Synchronous soap fault response	29
6	SOAP Faults and Fault messages	30

1 Introduction

The purpose of this document is to provide API specifications to interface with the Synchronoss ASOG Receive Web Service.

1.1 Referenced Documents:

ATIS Documents for UOM ASR
UOM ASR XML Schema

ILEC customers have to purchase/procure above reference documents from ATIS.

1.2 System requirements

- Web Services over HTTP/HTTPS
- SOAP version 1.2
- WSDL 1.1

1.3 Definitions

XML – Extensible Markup language
ASR – Access Service Request
ASOG – Access Services Ordering Guideline
WSDL – Web Service Definition Language
RPC – Remote procedure Call
HTTP – Hypertext Transfer protocol
SOAP – Simple Object Access Protocol
tML – Telecommunications Messaging Language

2 Supported Services

2.1 Supported ASR Order Services (Asynchronous services)

UOM ASR XML is used as payload for the ASR Order request and responses. Following order request and responses are supported. Note that order requests and responses follow ASYNCHRONOUS flow.

- ASR Order request (includes supplement orders)
- ASR Acceptance- response
- ASR Rejection response
- ASR TP Errored response.
- ASR Firm Order confirmation notification

- Error Clarification notification (CNR Error)
- Previous CNR clear notification
- Jeopardy notification
- Jeopardy with Errors notification
- ASR Provider initiated Cancellation notification
- ASR Remarks notification
- Design Layout Report Notification
- ASR Order Completion notification

2.2 Supported ASR Preorder Services (Synchronous services)

UOM ASR XML is used as payload for the ASR Preorder request and responses. Following preorder requests are supported and note that response is returned synchronously.

- CFA preorder Inquiry
- Location preorder Inquiry

2.3 Supported Access Order REQTYPES

UOM has structured order request schemas based on ASR services. Refer ASOG and UOM documentation for more details. See tab-1 for REQTYPE-Service type mapping.

<u>ServiceType</u>	<u>REQTYPE</u>	<u>UOM Service Class</u>
SA – 2 Point	SD	TRANSPORT_SVC
SA-Multi-Point	SD	TRANSPORT_SVC
Virtual Connection-SA	VD	TRANSPORT_VC_SVC
Transport EVC	SD	TRANSPORT_EVC_SVC
EUSA–2 Point	ED	END_USER_SA_SVC
EUSA-Multi-Point	ED	END_USER_SA_SVC
Virtual Connection-EUSA	XD	END_USER_VC_SVC
End User EVC	ED	END_USER_EVC_SVC
Feature Group A	AD	FEATURE_GROUP_A_SVC
Feature Group A-Multi-Point	AD	FEATURE_GROUP_A_SVC
Feature Group B	MD	MESSAGETRUNK_SVC

Feature Group C	MD	MESSAGETRUNK_SVC
Feature Group D	MD	MESSAGETRUNK_SVC
SACNXX	MD	MESSAGETRUNK_SVC
Forecasting	MD	MESSAGETRUNK_SVC
Local Trunking Service	MD	MESSAGETRUNK_SVC
SS7 Link CCS Links	LD	SS7_LINK_SVC
Ring Service	RD	RING_SVC
Standalone EVC	SD	STANDALONE_EVC_SVC
Standalone Trunking	MD	STANDALONE_TRUNKING_SVC
TQ	MD	MESSAGETRUNK_SVC
WAL	WD	WATS_SVC
WAL Ext	WD	WATS_SVC
CANCEL	(Any of the reotypes)	CANCEL_SVC
Transport Switched Ethernet	SD	SWITCHED_ETHERNET_SVC
Transport Switched Ethernet EVC	SD	SWITCHED_ETHERNET_EVC_SVC
End User Switched Ethernet	ED	SWITCHED_ETHERNET_SVC
End User Switched Ethernet EVC	ED	SWITCHED_ETHERNET_EVC_SVC

Table -1

Each of the services in turn contains all the required/optional forms; goes with that service.

2.4 Supported Access Order responses

Following table provides list of response and corresponding schema mapping sections.

RESPONSE NAME	SCHEMA NAME	FORM NAME/PARENT TAG
ACCEPT/REJECT	UOM-ASR_XX.xsd	
1. Accept Response		ASR_SERVICE_RESPONSE

2. Reject Response		ASR_SERVICE_RESPONSE/REJECTION
ERROR	UOM-ASR_XX.xsd	
1. Error		ASR_SERVICE_RESPONSE/ERROR
2. TP Error		ASR_SERVICE_RESPONSE/ERROR
CONFIRMATION (FOC)	UOM-ASRNotify_XX.xsd	
1. Firm Order Confirmation		CONFIRMATION
CLARIFICATION/NOTIFICATION	UOM-ASRNotify_XX.xsd	
1. CNR - ASR Completion		ORDER INFO NOTIFY
2. CNR – Jeopardy		JEOPARY NOTIFY
3. CNR - Previous C/NR Clear		ERROR CLARIFICATION OR JEOPARY NOTIFY
4. CNR – Errors		ERROR CLARIFICATION
5. CNR - Jeopardy with Errors		JEOPARY NOTIFY
6. CNR - Provider Initiated cancellation		ORDER INFO NOTIFY
7. CNR – Remarks		ORDER INFO NOTIFY
DLR NOTIFICATION	UOM-ASRNotify_XX.xsd	DESGIN NOTIFY

2.5 Security Requirements

2.5.1 Encryption

HTTPs is supported. Only one-way SSL is supported on inbound transactions. One-way and two-way SSL on outbound transactions; CLEC can define their preference of one-way OR two-way SSL during the connectivity process.

2.5.2 Authentication

2.5.2.1 *Request Security Header*

If the ILEC has enabled the Security Header, then the CLEC must send Security Header in the SOAP Header with the Username and Password, i.e., with the Security Credentials assigned by the ILEC. The Webservice Component will

validate the username and password against preconfigured values. If they do not match, a SOAP fault will be returned to the CLEC.

2.5.2.2 Response Security Header

If the CLEC requires security header on responses, the ILEC's ASR Receive Web Service client can be configured to support it. If configured, the ILEC's ASR Receive Web Service client will send the Security Header in response SOAP Header, with the predefined security username and password provided by the CLEC.

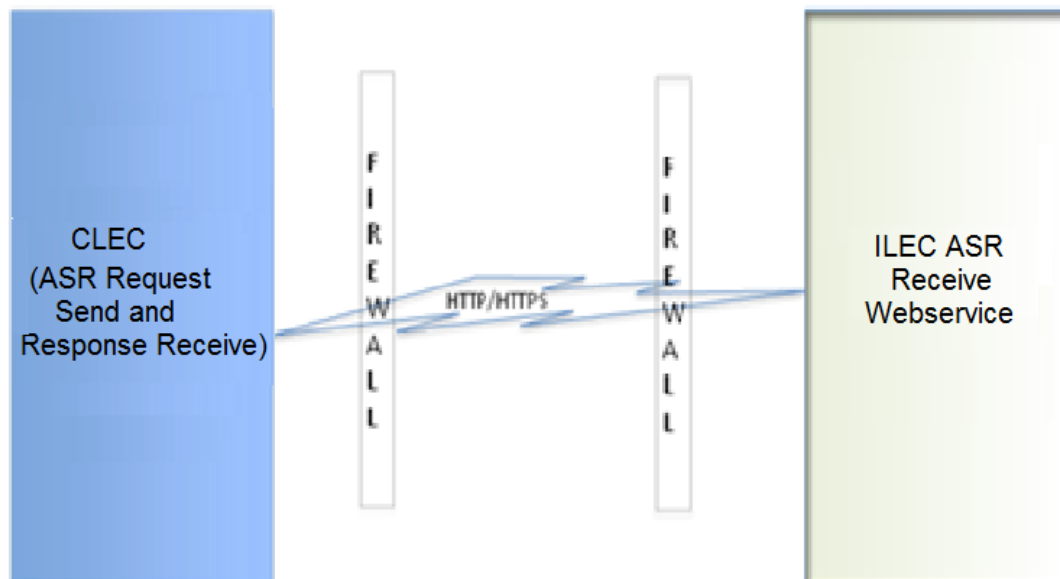
2.5.2.3 Security Header format

```
<wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext" soapenv:actor=""
soapenv:mustUnderstand="0">
  <wsse:UsernameToken>
    <wsse:Username>clecusername</wsse:Username>
    <wsse:Password>userpass1234</wsse:Password>
  </wsse:UsernameToken>
</wsse:Security>
```

The wsse security standard (as shown in the above sample) is the format supported by the ILEC.

3 ASR Receive Web Service interface architecture

ASR Receive Web Service interface follows standard business-to-business integration mode between two trading partners.



- ASR Receive Web Service is implemented in RPC style

- CLECs need to implement Web Service client as per the ASR Receive WSDL to send synchronous and asynchronous ASR order/preorder requests to ILEC's ASR Receive Web Service.
- CLECs also need to develop and host RPC style Web Service as per the ASR Response WSDL to receive asynchronous ASR order responses from the ILEC. CLECs need to initiate SOAP tML request with XML payload (UOM ASR XML) over HTTPS (one-way SSL handshaking)
- ILEC firewall ensures request is coming from authorized source.
- After validations, the ILEC ASR receive Web Service interface extracts the payload, forwards it to back end system and sends back the SOAP response to the CLEC.(for orders it is the ACK or NACK, for preorders it is the preorder response)

4 WSDL and ASR FLOW

ILEC's ASR Receive Web Services supports RPC style Web services. The service is synchronous for ASR Preorder and asynchronous for ASR order. Only one input argument will be taken per service, and it is in xml format. The CLEC must escape the Request payload, i.e., the characters listed below must be replaced by their corresponding equivalent. In addition, if the payload data contains any of these special characters, they must be double escaped.

Character	Equivalent
&	&
'	'
"	"
>	>
<	<

The response is also in the format of xml string and the payload will be escaped.

Payload samples corresponding to the current ASOG version will be provided on request.

4.1 ASR Request – Receive WSDL

ASR Receive Web Service implemented on the ILEC side based on the following WSDL. ILEC's customers need to implement Web Service client based on this WSDL to send preorder and order ASR requests.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions targetNamespace="java:asr.webservice.wisor.com"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:tns="java:asr.webservice.wisor.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <message name="request">
    <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
type="partns:string" name="string"/>
  </message>
  <message name="response">
    <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
type="partns:string" name="response"/>
  </message>
  <portType name="asrport">
    <operation name="processAsyncRequest">
      <input message="tns:request"/>
      <output message="tns:response"/>
    </operation>
    <operation name="processSyncRequest">
      <input message="tns:request"/>
      <output message="tns:response"/>
    </operation>
  </portType>
  <binding type="tns:asrport" name="asrport">
    <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="processAsyncRequest">
      <soap:operation soapAction=""/>
      <input>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="java:asr.webservice.wisor.com" use="literal"/>
      </input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="java:asr.webservice.wisor.com" use="literal"/>
      </output>
    </operation>
    <operation name="processSyncRequest">
      <soap:operation soapAction=""/>
      <input>
        <soap:body
```

```
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="java:asr.webservice.wisor.com" use="literal"/>
    </input>
    <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="java:asr.webservice.wisor.com" use="literal"/>
        </output>
    </operation>
</binding>
<service name="asrservice">
    <port binding="tns:asrport" name="asrport">
        <soap:address location="Wisor_Service_URL"/>
    </port>
</service>
</definitions>
```

4.2 ASR Response – Send WSDL

CLECs need to develop and deploy a Web Service based on this WSDL to receive asynchronous ASR order responses (notifications) from ILEC ASR service.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions targetNamespace="java:webservice.asr.customer.wisor.com"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:tns="java:webservice.asr.customer.wisor.com"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <message name="response">
        <part xmlns:partns="http://www.w3.org/2001/XMLSchema"
type="partns:string" name="string"/>
    </message>
    <message name="result"/>
    <portType name="asrport">
        <operation name="processAsyncResponse">
            <input message="tns:response"/>
        </operation>
    </portType>
    <binding type="tns:asrport" name="asrport">
        <soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="processAsyncResponse">
```

```

        <soap:operation style="rpc" soapAction=""/>
        <input>
            <soap:body
namespace="java:webservice.asr.customer.wisor.com"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="literal"/>
        </input>
    </operation>
</binding>
<service name="asrservice">
    <port name="asrport" binding="tns:asrport">
        <soap:address location="http://clec.com"/>
    </port>
</service>
</definitions>

```

4.3 tML Header Requirements

The CLEC must always send tML Header on order and preorder requests. The ILEC's ASR Receive Webservice will also send tML Header on order and preorder responses back to the CLEC.

The tML Header will not be sent on positive and negative acknowledgements to the CLEC in the Request Flow and is not expected from the CLEC on positive and negative acknowledgements in the Response flow.

The ASR Receive Webservice will validate the tML Header on the Request for the characteristics mentioned in the table below. If any of tML Header field validations fail, a SOAP fault with fault message "Error while validating the tML Header. Please correct the tML Header and retry again" will be returned.

Only the fields mentioned in the table below are supported. The corresponding tML Header schema has been provided in this section.

tML Header Field	Characteristics	Usage
From	Required AN(64)	This field identifies the CLEC. It must be sent by CLECs on the request. The static value to be sent will be predefined by the ILEC for that CLEC and shared along with other connectivity information. On asynchronous responses, this value will be returned under <To> tag of the tML Header.

To	Required AN(64)	This field identifies the ILEC on the request flow. This will be sent by CLECs on the request. The static value to be sent will be predefined by the ILEC for that CLEC and shared along with other connectivity information. On asynchronous responses, this value will be returned under <From> tag of the tML Header.
ApplicationType	Optional AN(32)	This field identifies the Application sending the transaction and is optional on the request. If the CLEC chooses to populate the ApplicationType, they must share with the ILEC their predefined ApplicationType values(one or more) that will be sent on Requests. These predefined ApplicationType values must be configured against the corresponding From-To combination. When a request is received, if the ApplicationType is populated, the webservice will validate the ApplicationType on the request with the predefined value(s) configured for that From-To combination and return a SOAP fault if there is no match. In the response flow, If the Response Application Type is configured for each root node, the Webservice will return the configured Application Type on Asynchronous responses. The webservice will not validate the ApplicationType on the acknowledgement (ACK/NACK)

		from the CLEC for the asynchronous response.
--	--	--

4.3.1 tML Header Schema



Transport.xsd

4.3.1.1 Text Version of the schema

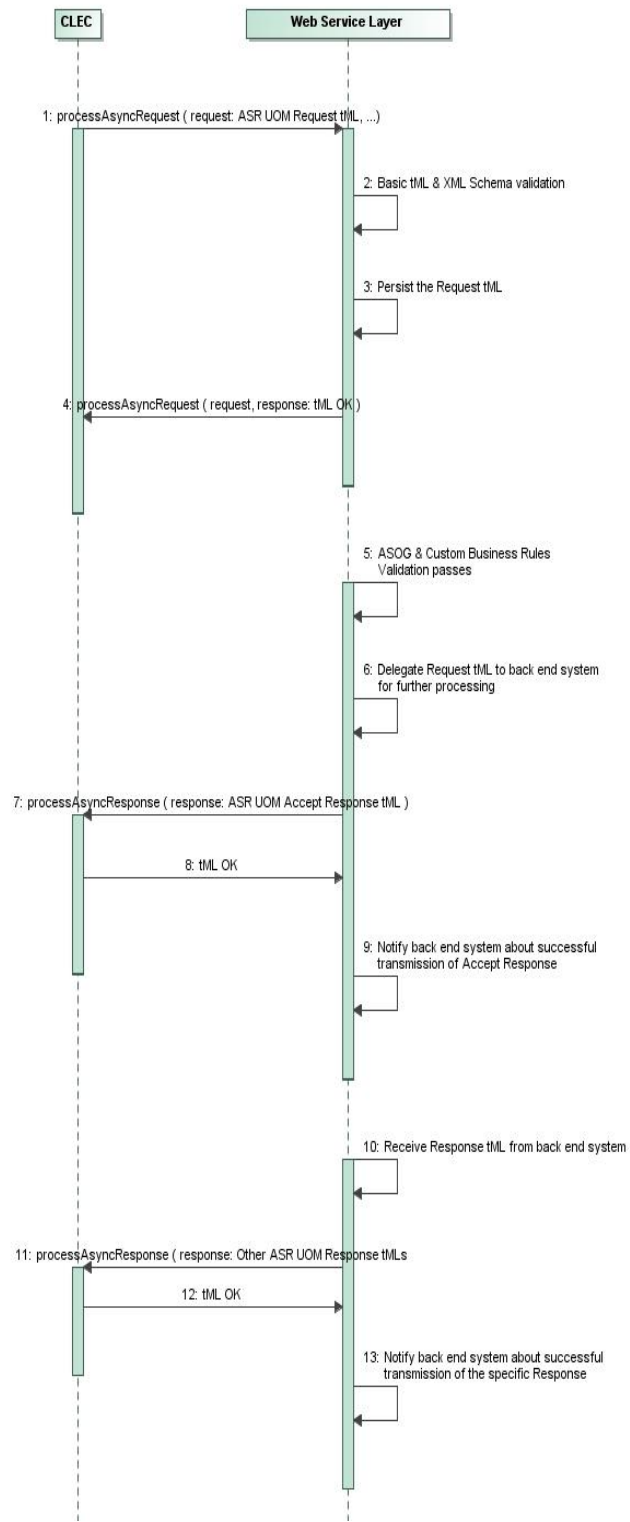
```
<?xml version="1.0" encoding="UTF-8"?>
<!--Generated by Turbo XML 2.3.1.100. Conforms to w3c http://www.w3.org/2001/XMLSchema-->
<!-- Modified by Madhusudhan K S on 8/26/2014 added ApplicationVersion element and removed
sequence in complex element and changed it to all -->
<xs:schema targetNamespace="http://tml.t1m1.org/tML.Transport.xsd"
xmlns="http://tml.t1m1.org/tML.Transport.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <!--Gleaned from TRANSPORT tML.doc, Appendix D -->
  <!-- HEADER FIELDS -->
  <xs:element name="TransportID">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="64"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="CorrelationID">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="64"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <xs:element name="TrackID">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="64"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
  <!-- Dan added this as a guess -->
  <xs:element name="ResultID">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="64"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:element>
</xs:schema>
```

```
</xs:restriction>
</xs:simpleType>
</xs:element>
<!-- end of guess -->
<xs:element name="ApplicationType">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="32"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="From">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="64"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="To">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="64"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="SendTimestamp" type="xs:dateTime"/>
<xs:element name="RetryCount" type="xs:integer"/>
<xs:element name="ApplicationVersion">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="32"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="tMLHeader">
  <xs:complexType>
    <xs:all>
      <xs:element ref="TransportID" minOccurs="0"/>
      <xs:element ref="CorrelationID" minOccurs="0"/>
      <xs:element ref="TrackID" minOccurs="0"/>
      <xs:element ref="ApplicationType" minOccurs="0"/>
      <xs:element ref="From"/>
      <xs:element ref="To"/>
      <xs:element ref="SendTimestamp" minOccurs="0"/>
      <xs:element ref="RetryCount" minOccurs="0"/>
      <xs:element ref="ApplicationVersion" minOccurs="0"/>
    </xs:all>
  </xs:complexType>
</xs:element>
<!-- OK MESSAGE -->
<xs:element name="OK">
```

```
<xs:complexType/>
</xs:element>
<!-- REQUEST ACKNOWLEDGEMENT FIELDS -->
<xs:element name="RequestAcknowledgement">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Acknowledgement" type="xs:boolean"/>
      <xs:element name="Reason" type="xs:string" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!-- REQUEST NOTIFICATION FIELDS -->
<xs:element name="Result">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ResultID"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="ResultWaitingNotification">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="Result" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!-- GET RESULT FIELDS -->
<xs:element name="GetResult">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ResultID"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!-- RESULT ACKNOWLEDGEMENT FIELDS -->
<xs:element name="ResultAcknowledgement">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="ResultID"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<!-- EXCEPTION DESCRIPTION FIELDS -->
<xs:element name="ExceptionDescription" type="ExceptionDescription_Type"/>
<xs:complexType name="ExceptionDescription_Type">
  <xs:sequence>
    <xs:element name="Type" type="ExceptionType_Type"/>
    <xs:element ref="ExceptionDetails" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="ExceptionDetails" type="ExceptionDetails"/>
<xs:complexType name="ExceptionDetails">
  <xs:sequence>
    <xs:element name="Code" minOccurs="0">
      <xs:simpleType>
```

```
<xs:restriction base="xs:string">
  <xs:minLength value="1"/>
  <xs:maxLength value="64"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="Description">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="64"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
<xs:element name="AdditionalInfo" minOccurs="0">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:minLength value="1"/>
      <xs:maxLength value="240"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:simpleType name="ExceptionType_Type">
  <xs:restriction base="xs:string">
    <xs:enumeration value="GENERAL_FAILURE"/>
    <xs:enumeration value="GENERAL_ERROR_RETRYABLE"/>
    <xs:enumeration value="TRANSPORT_FAILURE"/>
    <xs:enumeration value="TRANSPORT_ERROR_RETRYABLE"/>
    <xs:enumeration value="SECURITY_PROBLEM"/>
    <xs:enumeration value="RESOURCE_PROBLEM_RETRYABLE"/>
    <xs:enumeration value="FUNCTION_NOT_SUPPORTED"/>
    <xs:enumeration value="INVALID_MESSAGE_FORMAT"/>
    <xs:enumeration value="RETURN_RESULT_PROBLEM"/>
    <xs:enumeration value="DUPLICATE_INVOCATION"/>
    <xs:enumeration value="INVALID_ATTRIBUTE"/>
    <xs:enumeration value="MISSING_ATTRIBUTE"/>
    <xs:enumeration value="INVALID_DATA_VALUE"/>
  </xs:restriction>
</xs:simpleType>
</xs:schema>
```

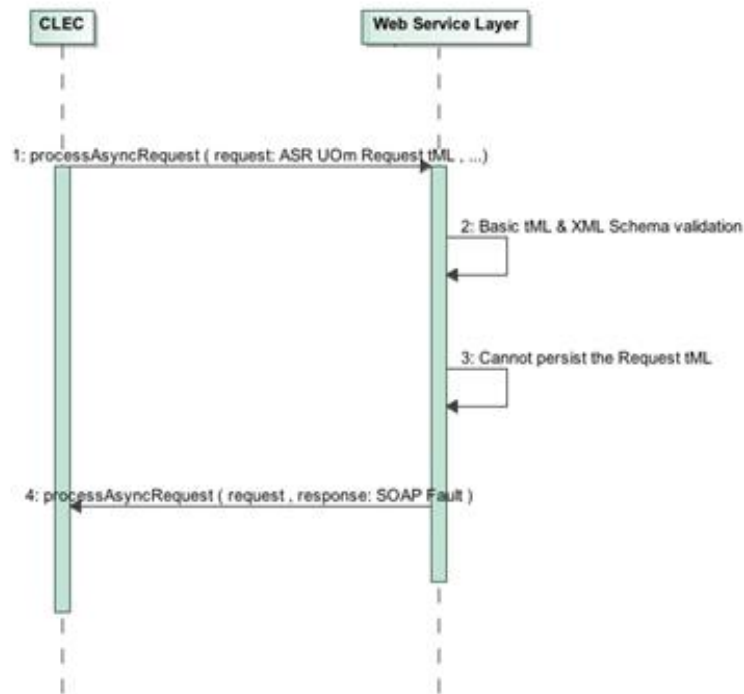
4.4 ASR Order Receive - Positive / Successful scenario



1. ILEC's ASR Send customers (referred to as CLEC here) must send a soap request with ASR UOM Order XML as a string *request* parameter over the operation *processAsyncRequest* with a place holder for the *response* string parameter.
2. If the Security Header Authentication is mandated by the ILEC, the ILEC's ASR Receive Web Service will validate the Username and Password in the Security Header against the preconfigured values and raise a SOAP Fault if they do not match. The assumption for this flow is that the validation is successful.
3. ILEC's ASR Receive Web Service performs tML SOAP header validation and ASOG XML Schema validation upon receipt of the Order Request. The Webservice will also validate the CCNA-ICSC combination. The assumption in this flow is that the validations pass.
4. Order Request is persisted in the database before acknowledging the CLEC in order to guarantee further processing.
5. tML OK response is sent back on the *response* message of the *processAsyncRequest* web service call to the CLEC as an acknowledgement for receiving the Order Request.
6. ILEC systems continue processing the persisted Order Request by validating its contents against the ASOG & Custom Business Rules. In this flow the assumption is all rule validations pass without any errors.
7. The validated Order Request is sent to the back end system for further processing.
8. ASOG UOM Accept Response tML is sent to the CLEC on *response* message of the *processAsyncResponse* web service call.
9. If CLEC is able to handle the response successfully, the web service call must be acknowledged with a tML OK, which enables ILEC to understand successful transmission of the Response.
10. ILEC notifies the back end system about successful transmission of the acceptance response to the CLEC.
11. During further processing and provisioning life cycle of the Order, the back end system will send further responses, like FOC/DLR/etc... that needs to be notified to the CLEC
12. The Web Service Layer notifies the response to the CLEC as a tML in the *response* message of the *processAsyncResponse* web service call.

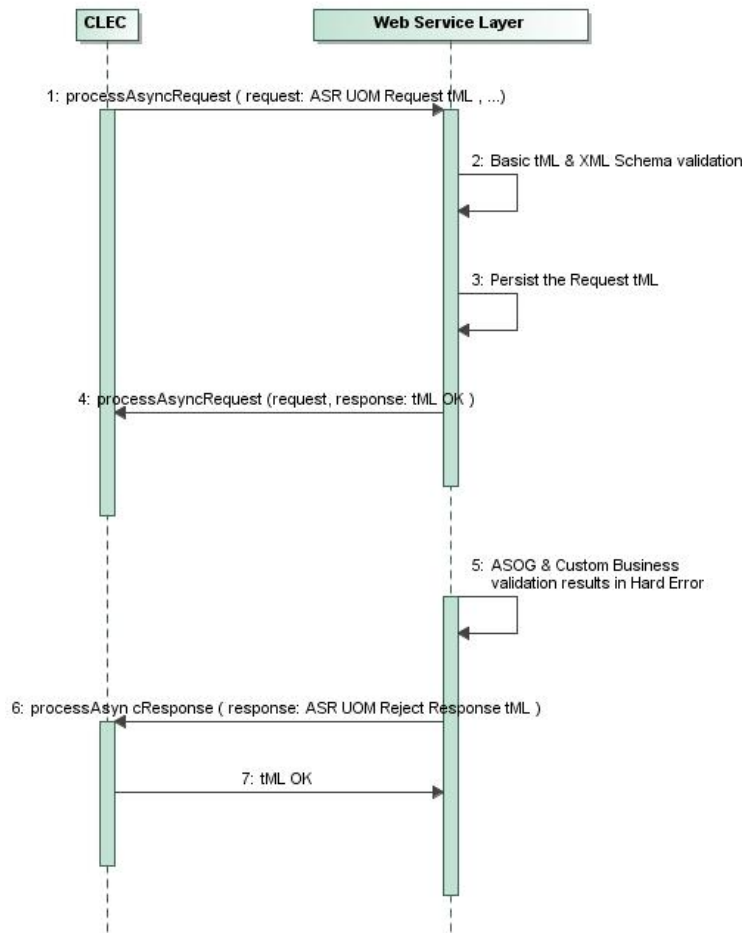
13. If during the connectivity discussions the CLEC has requested for Security Header in the Response flow, then the Web Service Layer will send the Security Header on the response SOAP Header to the CLEC. The response security header will be independent of request security header section.
14. CLEC acknowledges the response with tML OK.
15. Web Service Layer again notifies the successful transmission of the corresponding response.

4.5 ASR Order Receive – Negative/fault scenario



1. ILEC's ASR Send customers (referred to as CLEC here) must send a soap request with ASR UOM Order XML as a string *request* parameter over the operation *processAsyncRequest* with a place holder for the *response* string parameter.
2. ILEC ASR Receive Web Service performs Security Header validation (if mandated by ILEC), basic tML SOAP header validation, ASOG XML Schema validation and CCNA-ICSC validation upon receipt of the Order Request.
3. Either any of the validations in step #2 fail OR validation succeeds but persistence/further handling of the Order Request fails.
4. A SOAP Fault response is sent back on the *response* message of the *processAsyncRequest* web service call to the CLEC as a negative acknowledgement for the received Order Request. At this point, the Order Request is no longer in ILEC's system. Depending on the error in the SOAP Fault, the CLEC needs to either rectify the issue and resubmit the Order or contact ILEC support.

4.6 ASR Order Receive – Negative/Reject Scenario



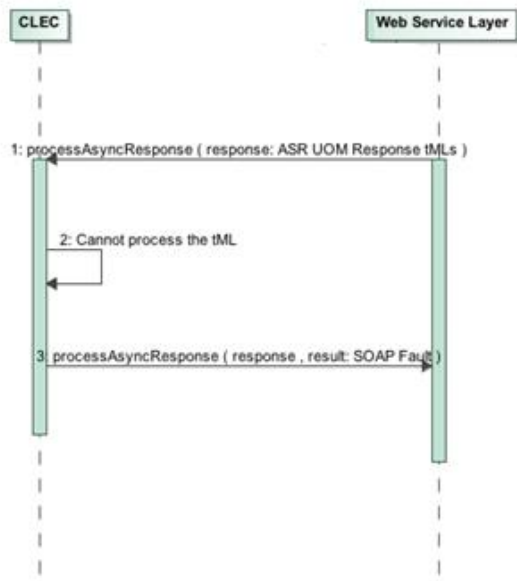
1. ILEC's ASR Send customers (referred to as CLEC here) must send a soap request with ASR UOM Order XML as a string *request* parameter over the operation *processAsyncRequest* with a place holder for the *response* string parameter.

2. ILEC ASR Receive Web Service performs Security Header validation (if mandated by ILEC), basic tML SOAP header validation, ASOG XML Schema validation and CCNA-ICSC validation upon receipt of the Order Request. The assumption in this flow is that the validations pass.

3. Order Request is persisted in the database before acknowledging the CLEC in order to guarantee further processing.

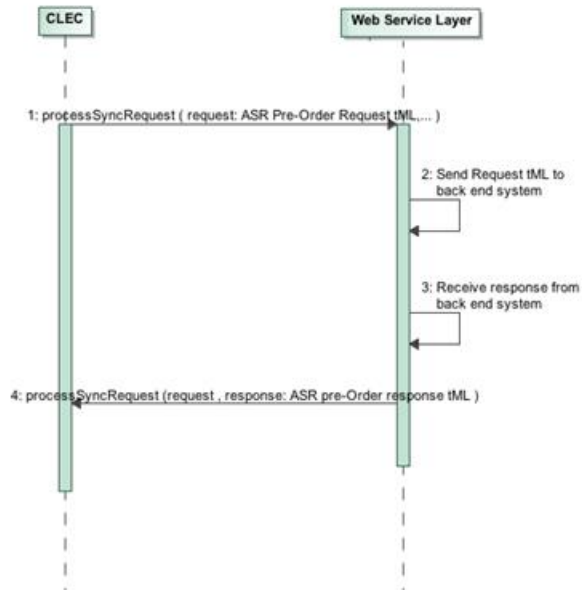
4. tML OK response is sent back on the *response* message of the *processAsyncRequest* web service call to the CLEC as an acknowledgement for receiving the Order Request.
5. ILEC systems continue processing the persisted Order Request by validating its contents against the ASOG & Custom Business Rules. In this flow the assumption is validation fails and the errors are identified as Hard errors.
6. ASOG UOM Reject Response tML is sent to the CLEC on *response* message of the *processAsyncResponse* web service call with the details. The validated Order Request is not processed further.

4.7 ASR Order response Send – Negative/fault scenario



1. The Web Service Layer notifies the response to the CLEC as a tML in the *response* message of the *processAsyncResponse* web service call.
2. The assumption in this flow is that the CLEC is not able to handle the response tML.
3. CLEC returns back the SOAP Fault on the *result* message of the *processAsyncResponse* web service call.

4.8 ASR Pre-Order Receive - Positive / Successful scenario

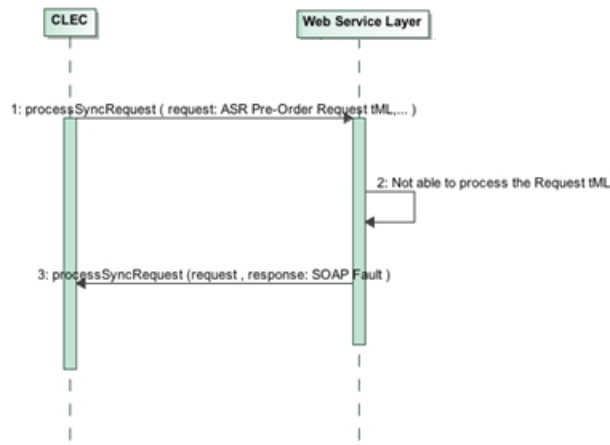


1. ILEC ASR Send customers (referred to as CLEC here) must send a soap request with ASR UOM Pre-Order XML as a string *request* parameter over the operation *processSyncRequest* with a place holder for the *response* string parameter.

2. ILEC ASR Receive Web Service performs Security Header validation (if mandated by ILEC), basic tML SOAP header validation and CCNA-ICSC validation upon receipt of the PreOrder Request. For this flow the assumption is that both the validations pass. The Pre-Order Request is sent to the back end system for further processing while still holding on to the web service connection with the CLEC.

3. When response data is received from the back end system, the Web Service Layer notifies the response to the CLEC as a tML in the *response* message of the same *processSyncRequest* web service call.

4.9 ASR Preorder Receive - Negative / fault scenario



1. ILEC ASR Send customers (referred to as CLEC here) must send a soap request with ASR UOM Pre-Order XML as a string *request* parameter over the operation *processSyncRequest* with a place holder for the *response* string parameter.
2. ILEC ASR Receive Web Service performs Security Header validation (if mandated by ILEC), basic tML SOAP header validation and CCNA-ICSC validation upon receipt of the PreOrder Request. If any of the validations fail or ILEC system is not able to handle the Pre-Order Request, it is a failure scenario.
3. A SOAP Fault is sent to the CLEC in the *response* message of the same *processSyncRequest* web service call.

5 SOAP Structure

5.1 ASR Pre-Order - Synchronous request/response

5.1.1 ASR Pre-Order synchronous request

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext"
      soapenv:actor="" soapenv:mustUnderstand="0">
      <wsse:UsernameToken>
        <wsse:Username>clecusername</wsse:Username>
        <wsse:Password>userpass1234</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
    <tMLHeader xmlns="http://tml.t1m1.org/tML.Transport.xsd">
      <From>SUPPFROM</From>
      <To>SUPTO</To>
    </tMLHeader>
  </soapenv:Header>
  <soapenv:Body>
    <processSyncRequest xmlns="java:asr.webservice.wisor.com">
      <string xmlns="">XML GOES HERE!!!</string>
    </processSyncRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

5.1.2 ASR Pre-Order synchronous response

```
<?xml version="1.0" encoding="utf-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <tMLHeader xmlns="http://tml.t1m1.org/tML.Transport.xsd">
      <From>SUPPFROM</From>
      <To>SUPTO</To>
    </tMLHeader>
  </soapenv:Header>
  <soapenv:Body>
    <processSyncRequestResponse
      xmlns="java:webservice.asr.customer.wisor.com">
      <response xmlns=""> XML GOES HERE!!!</response>
    </processSyncRequestResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

5.1.3 ASR Pre-Order synchronous soap fault response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>WG1602</faultcode>
      <faultstring>There is no interface configured, contact Wisor
support</faultstring>
      <detail>
        <ns1:hostname xmlns:ns1="http://xml.apache.org/axis/">in-
bandevs04</ns1:hostname>
      </detail>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

5.2 ASR Order Receive - Asynchronous request/response

5.2.1 ASR Order asynchronous soap request (from ILEC Customer)

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext"
      soapenv:actor="" soapenv:mustUnderstand="0">
      <wsse:UsernameToken>
        <wsse:Username>clecusername</wsse:Username>
        <wsse:Password>userpass1234</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
    <tMLHeader xmlns="http://tml.t1m1.org/tML.Transport.xsd">
      <From>SUPPFROM</From>
      <To>SUPTO</To>
    </tMLHeader>
  </soapenv:Header>
  <soapenv:Body>
    <processAsyncRequest xmlns="java:asr.webservice.wisor.com">
      <string xmlns=""> XML GOES HERE!!! </string>
    </processAsyncRequest>
  </soapenv:Body>
</soapenv:Envelope>
```

5.2.2 ASR Order asynchronous soap response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <wsse:Security xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext"
      \soapenv:actor="" soapenv:mustUnderstand="0">
      <wsse:UsernameToken>
        <wsse:Username>clecusername</wsse:Username>
        <wsse:Password>userpass1234</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
    <ns1:tMLHeader xmlns:ns1="http://tml.t1m1.org/tML.Transport.xsd">
      <ns1:From>SUPTO</ns1:From>
      <ns1:To>SUPPFROM</ns1:To>
    </ns1:tMLHeader>
  </soapenv:Header>
  <soapenv:Body>
    <ns2:processAsyncResponse
      soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      xmlns:ns2="java:webservice.asr.customer.wisor.com">
      <string xsi:type="xsd:string"> XML GOES HERE!!!</string>
    </ns2:processAsyncResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

5.2.3 ASR Order Synchronous tML OK response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <processAsyncRequestResponse
      xmlns="java:webservice.asr.customer.wisor.com">
      <response xmlns=""> XML GOES HERE!!!</response>
    </processAsyncRequestResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

5.2.4 ASR Order Synchronous soap fault response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```

    <soapenv:Body>
      <soapenv:Fault>
        <faultcode>WG1602</faultcode>
        <faultstring>There is no interface configured, contact Wisor
support</faultstring>
        <detail>
          <ns1:hostname xmlns:ns1="http://xml.apache.org/axis/">in-
bandevs04</ns1:hostname>
        </detail>
      </soapenv:Fault>
    </soapenv:Body>
  </soapenv:Envelope>

```

6 SOAP Faults and Fault messages

The ILEC SOAP faults and fault messages have been listed below.

Fault Codes	Fault Messages
WG0204	Empty message received from internal system
WG0503	Gateway is busy. Please retry after sometime.
WG1601	There is no interface configured for Pre Order, Please contact support.
WG1602	There is no interface configured. Please contact Synchronoss support
WG1604	Interface has been Suspended. Please contact Synchronoss support.
WG0641	Host is not reachable.Please check the URL Configured in the Interface.
WG0642	Server busy, retry request after some time. (This error occurs when backend OSS is not responding in case of preorder)
WG1697	Translator is deactivated. Please contact Synchronoss support.
WSASRTML001	Gateway not reachable. Please contact Synchronoss support.
WSASRTML002	Internal server error. Please contact Synchronoss support.
WSASRTML003	Server busy, retry request after some time.
WSASRTML004	Unknown error, Please contact Synchronoss support (This will be sent when EP returns null or blank instead of WG0200)
WSASRTML005	Schema Validation failed (the actual validation error will go here)
WSASRTML006	Error while validating the tML Header. Please correct the tML Header and retry again.
WSASRTML007	Error while Transformation.
WSASRTML008	Customer with given From/To is not configured.
WSASRTML009	Username/Password in Security Header is invalid.
WSASRTML010	No Username/Password configured for the CLEC, Please contact Support.
WSASRTML011	CCNA-ICSC combination is not valid.
WSASRTML012	Invalid Application Type in tML header
WSASRTML013	ApplicationType/Application Version not configured for the given

	From/To, Please contact support
WSASRTML014	No EndPoint URL found for Response webservice in the system.

The CLEC must provide the ILEC with the complete list of their SOAP fault codes and fault messages, along with the distinction of what action is to be taken by the ILEC for that SOAP fault. The two possible ILEC actions are:

- a) Suspension - Suspend the transaction and the interface so that no further transactions will be sent to the CLEC.
- b) Failure - Mark the transaction as failure and continue to transmit other transactions to the CLEC. The error in the failed transaction to be reviewed manually by the ILEC.

Any new SOAP fault code that was not provided initially will automatically result in Suspension.