PTV API Documentation

A Step-by-Step Guide to Going Live

Last update 09-20-2022

CONTENTS

Overview

WENO Exchange is constantly trying to make the complex simple and that's not so easy. It is in that spirit that we created our documentation in the step-by-step format. If followed in order, you will be in a position to go live with the least amount of effort.

Each step is a pre-requisite for the next to get you to the point you are able to send/receive quality eRx messages for your unique end user's needs. Resist the urge to jump ahead and you will be rewarded with a fast go live!

We support the latest US government ePrescribing standards: NCPDP 20170715.

Some messages in ePrescribing are minimally required to participate; and others add value. You control which to support. After you go live, and as long as your account is in good standing, you will continue to have access to your certification dashboard for any future testing needs you may have.

Step 1 awaits!

Step 1: Sign Up & Attend Kick Off Meeting

Repeat this step for live environment: NO

If you have not done so, go to our website https://wenoexchange.com/get-started-v2/ to sign up for the API you need.

This triggers your certification manager to register your company for the certification dashboard and help you schedule your kick off meeting!

Tell your certification manager if you need to manage a common ePrescribing set up or if you need to manage a common as well as custom set ups. We offer PTVs 2 types of dashboards, specific to your needs.

a. ***PTV dashboard*** – for managing one common ePrescribing set up for all of your pharmacies
b. ***MPTV dashboard*** (MainPTV) – allows for one common set up for certain pharmacies (if this is needed) and the ability to add an unlimited number of **SubPTV** dashboards for you or your downstream clients to test and manage their own custom ePrescribing set up.

Step 2: Set Up Your Certification Dashboard

Repeat this step for live environment: YES

1. Manage team: Log in to your dashboard at https://cert.wenoexchange.com -> Manage Users

2. Find partner ID and partner password > Account Details (top right of page).
   Certification & live credentials are not the same.

Step 3: Manage Your ePrescribing Participants

Repeat this step for live environment: YES

For All PTV Types

1. Go to Manage Account ->Manage Pharmacies and add at least 1 test pharmacy.

You can continue to do this manually if you have a few, or you can use our Manage API if you have many. You are required to keep this information current at all times.

To Manage via API, you will find the information you will need here:

cert.wenoexchange.com dashboard ->Testing Tools ->Manage Account APIs

Contact us if you get an error registering any NCPDP. This means it is or was registered by another PTV and we will need to resolve the conflict.

Provide as much information about the pharmacy as you can, as the WENO directory is used by prescribers and their patients. So, the 24-hour flag, delivery, etc. can be important drivers in the patient's decision-making process.

For Main PTV Types ONLY

2. After you tackle your common ePrescribing set up as shown in #1 of this step, you will need to tackle the SubPTV set up:

   a. Add at least 1 SubPTV account dashboard to your Main PTV account to understand how to manage a custom ePrescribing set up. Go to Manage Account ->Manage SubPTV Accounts The SubPTV is not a pharmacy, the SubPTV is a technology. For example: maybe it is a custom version for Client A, or maybe it is a technology your company acquired, or a version of your technology that is soon to be converted but it has not been converted yet. Each SubPTV will register the pharmacies that use their technology.

   b. Users on your MPTV account can access the SubPTV's dashboard via a log in icon in the Manage SubPTV Account page of your dashboard.

   c. They should add any users that need to specifically access the SubPTV account (without having access to your MPTV account. Maybe this is a client's custom software's developers or a certain team within your team that handles the technology.

   d. The Subaccount users must add at least one test pharmacy to the SubPTV in Manage Account ->Manage Pharmacies.

Step 4: Incorporate Drug DB

Repeat this step for live environment: YES

Skip this step, it is for EHR use only.

Step 5: Access Directory Data

Repeat this step for live environment: YES

Decide if you, or applicable MainPTV/SubPTVs, will be doing the quick certification method or the full method (handle sending and receiving various SCRIPT messages like RxRenewals, Cancel, Change, etc.).

QUICK method: Skip this step. You will receive NewRxs and responding with either a real time success or real time error. WENO will handle the Verify Message back to prescribers that flag the NewRx with return receipt when you respond with a real time success.

FULL method: For handling one or more SCRIPT message types to and from WENO prescribers, you will need to manage WENO's Prescriber Directory. WENO provides you with 2 ways to receive prescriber data. You should look at both and decide which is best for your use or use both.

I. To access our Prescriber Directory, learn how from your dashboard -> Testing Tools – Directories

II. We can also push to your designated endpoint a daily update and a weekly full prescriber file. To understand the specs and set this up go to: dashboard ->Manage->File Destinations

Step 6: Understand the Script Header

Repeat this step for live environment: NO

NCPDP SCRIPT 20170715 is XML. All XML messages all have a Header and a Body.

This means the SCRIPT Header is common to all SCRIPT messages.

There are over 40 different SCRIPT body types. The industry nicknames the SCRIPT messages by the name of the body. For example, the NewRx Message includes a Header and the NewRx Body, but we call it the NewRx.

Same goes for the other body types like RxRenewal, Status, Error, Verify, RxRenewalResponse, to name a few.

We created a page that explains the SCRIPT Header very well. Become familiar with the fields and how to populate them accurately before you begin to tackle various body types. To complete this step:

Go to your dashboard ->Testing Tools -> SCRIPT – Header Explained

Step 7: SCRIPT Body & Samples

Repeat this step for live environment: NO

It's very likely you already have been receiving the latest NCPDP SCRIPT message types you plan to support. WENO does not depart from the standards except we require some fields that are optional. In other words, all of the messages will validate against the minimally required fields of the standards for interoperability but some make good sense to include.

SCHEMAS: You can access the schemas you will need here: cert dashboard ->Testing Tools -> SCRIPT and Related Schemas

For Integration Types	Schema Needed
All	SCRIPT
Only if you will send nonce key in Headers	Weno Nonce
WOL API EHRs	WENO Online PostRx (Your wrapper for NewRxs prescribers will sign & complete on WOL)

SCRIPT Samples and Guidance. To find samples, & learn what we validate and verify to pass your TEST Plan go here:

Cert dashboard ->Testing Tools -> SCRIPT Body and Guidance

Codes: To find a code for certain fields not otherwise explained, go to: Cert dashboard ->Testing Tools ->Error/Verify and other codes

Unique to WENO's Intermediary Service:

- **WENO ONLINE:** ("WOL") We are the only ePrescribing intermediary service that offers a stand-alone for both prescribers and pharmacies. The stand-alone for each have their required DEA application audits. The stand-alone for pharmacies is the bridge application they can use when their software is not yet connected. The stand-alone for prescribers allows a solo practitioner or a large enterprise/practice the ability to begin ePrescribing within minutes. Smaller EHRs can also integrate with the stand-alone for prescribers, especially when they need to offer their end users EPCS and prefer to dovetail the DEA requirements to another application.

- **ALTERNATIVE PHARMACIES:** WENO marks pharmacies unwilling to use the WENO Online portal to retrieve eRxs with eRx Drug warnings on the WENO Pharmacy directory. Typically, these are chain stores. Patients provide a primary pharmacy, and if the primary pharmacy is not connected to WENO, patients are asked to provide a WENO connected alternative to receive electronic transmissions when their primary pharmacy will not. This is important for the pharmacy's technology vendor to understand this, so they will register all the pharmacies that use their technology so they will show up as connected and ready to receive NewRxs.

- **PRODUCT CODE NDC:** WENO may populate the Product Code field of NewRxs with a patient's specific NDC request when the request does not conflict with the NewRxs. The prescriber's note may bring this to your attention or the field may just be populated. Please be aware of this, to honor a patient's request when possible.

- **DIGITAL SIGNATURES:** WENO does not alter the digitally signed or required electronically signed portions of the NewRx in transit, this of course can be verified by the receiving party. WENO may send the digital signature if provided by the prescriber application, but it will always make sure the flag for the digital signature is shown for EPCS orders.

- **MISCONCEPTION:** WENO's intermediary service does not turn ePrescriptions into faxes. WENO's Intermediary service transfers ePrescriptions to pharmacy applications. In cases, where the pharmacy application happens to be WENO Online's pharmacy application, the WENO Online's pharmacy application service faxes an alert to pharmacies with instructions on how to retrieve the ePrescription. If the pharmacy registers for WENO Online they can control their fax alert settings from their account.

- **PHARMACY OPPORTUNTIES:** WENO provides your pharmacies with the ability to go beyond basic information for routing. When you populate all the attributes available, this can be viewed by patients and prescriber's alike so they have what they need to make well informed decisions.

Step 8: Register IPs

Repeat this step for live environment: YES

QUICK certification method: skip this step

FULL certification method:

To prevent message rejections from unknown IPs, make a list of all IP addresses that will send us messages and register them here: cert dashboard ->Manage Account ->Manage IPs and Endpoints.

Step 9: Understand the Endpoint COMMUNICATIONS FLOW

Repeat this step for live environment: NO

Understand how Intermediary Communication Works:

- You send WENO's endpoint a message you will support receiving
- WENO receives this message from sender going to your endpoint on record.
- WENO will respond in real time (Error or Success) to the sender.
- This first call is ended.
- WENO attempts to deliver the message to the recipient's endpoint on record.
- If it fails, WENO will return its ERROR message to the sender's listening endpoint.
- If it succeeds, WENO will return the VERIFY message to sender's listening endpoint, if flagged for return receipt.
    - **For QUICK Certifications** – WENO handles these VERIFY messages for your when we get the real time STATUS response from your listening endpoint.
      For FULL Certifications – WENO can continue to handle VERIFY for you or you can, but tell your certification manager so your settings can be applied accurately.
- IF your endpoint breaks or is not set up the communication breaks.

NOTE: When your endpoint is not set up or is broken, WENO will send an email to all of your dashboard users notifying them that your endpoint is not working. If you set up your listening endpoint before trying to send us any test messages, you will avoid this problem.

Step 10: Create & Test Your Listening Endpoint

Repeat this step for live environment: YES

You must create and register a listening endpoint here from your appropriate dashboard -> Manage Account ->Manage IPs and Endpoints.

The information below tells you why and how to do this. The Samples may not reflect the TO and FROM recipients that makes sense for your environment, so adjust accordingly.

How to Create a Listening Endpoint

1. Create a simple https endpoint. It should not be HTML or a web service. Sample PHP and C# code for this is shown in this section below.
2. It must return either a real time "SCRIPT STATUS or ERROR message to work properly. See samples for each below and also here: Cert dashboard->Testing Tools->SCRIPT Body & Guidance -> SCRIPT Samples
3. Test your listening endpoint here: Your appropriate dashboard->Testing Tools->Endpoint Tester

The Sample SCRIPT STATUS Message in Real-Time

You only need the 001 code as shown for this use real-time success use case.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- STATUS Sample real-time response to WENO when your endpoint successfully receives a message -->
<Message xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
DatatypesVersion="20170715" TransportVersion="20170715" TransactionDomain="SCRIPT"
TransactionVersion="20170715" StructuresVersion="20170715" ECLVersion="20170715">
```

```
    <Header>
        <To Qualifier="M">WENO_Switch</To>
        <From Qualifier="D">EchoBackPrescriberRoutingIDthatReceivedMsg</From>
        <MessageID>whatever</MessageID>
        <RelatesToMessageID>0</RelatesToMessageID>
        <SentTime>2022-05-26T11:48:47.19677Z</SentTime>
        <Security>
            <UsernameToken>
                <Username>PartnerPasswordGoesHere</Username>
                <Password Type="PasswordDigest">YourMD5PartnerPasswordHere</Password>
            </UsernameToken>
        </Security>
        <SenderSoftware>
            <SenderSoftwareDeveloper>NameOrTitle</SenderSoftwareDeveloper>
            <SenderSoftwareProduct>Your software name</SenderSoftwareProduct>
            <SenderSoftwareVersionRelease>V1</SenderSoftwareVersionRelease>
        </SenderSoftware>
    </Header>
    <Body>
        <Status>
            <Code>001</Code>
            <Description>Transaction successful</Description>
        </Status>
    </Body>
</Message>
```

The Sample SCRIPT ERROR Message in Real Time

To understand Error codes to your cert dashboard ->Testing Tools -> Error/Verify and other codes

```
<?xml version="1.0" encoding="utf-8"?>
<!-- ERROR Sample -->
<!-- ERROR messages can be sent to and from any partner.  If ERROR is a real time
response the RelatesToMessageID can be populated with 0 -->
<!-- If ERROR is sent later, the RelatesToMessageID must be the message ID that it
relates to so it can be tied back by the original sender of the message. -->
<Message xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
DatatypesVersion="20170715" TransportVersion="20170715" TransactionDomain="SCRIPT"
TransactionVersion="20170715" StructuresVersion="20170715" ECLVersion="20170715">
    <Header>
        <To Qualifier="M">WENO_Switch</To>
        <From Qualifier="D">EchoBackWhoReceivedMsg</From>
        <MessageID>Unique_Message_ID</MessageID>
        <RelatesToMessageID>0</RelatesToMessageID>
        <SentTime>2022-09-14T04:56:15.45Z</SentTime>
        <Security>
            <UsernameToken>
                <Username>37</Username>
                <Password
Type="PasswordDigest">F7640437662616371E81EDAB24079579</Password>
            </UsernameToken>
        </Security>
        <SenderSoftware>
            <SenderSoftwareDeveloper>Good Guy</SenderSoftwareDeveloper>
            <SenderSoftwareProduct>ACME Software Vendor</SenderSoftwareProduct>
            <SenderSoftwareVersionRelease>V1</SenderSoftwareVersionRelease>
        </SenderSoftware>
    </Header>
    <Body>
        <Error>
            <Code>700</Code>
```

```
            <DescriptionCode>144</DescriptionCode>
            <Description>--Description--</Description>
        </Error>
    </Body>
</Message>
```

Sample PHP Code To Set Up Your Listening Endpoint

```
// receive_from_weno.php - receive a message from weno service

function listen()
{
  $messageID = '0';
  $now = date('Y-m-d H:i:s');
  $body = trim(file_get_contents('php://input'));

  $MessageType = ($body[0] == '<') ? 'XML' : 'JSON';

LogMessage("<<< Incoming {$MessageType} at {$now}");
LogMessage($body); // echo incoming message from Switch

  if ($MessageType == 'XML') {
    $xmlBody=simplexml_load_string($body);
LogMessage('PHP representation of message:');
LogMessage(print_r($xmlBody,true));

    $messageID = $xmlBody->Header[0]->MessageID;
    $sFrom = $xmlBody->Header[0]->To;
    $sTo = $xmlBody->Header[0]->From;

    $response = GetStatusXML('000','Transaction Successful', '0', $sToQual, $sFromQual, $sTo, $sFr
om);

    // send response to Switch
print_r($response->asXML());
  }
  else {
    $jsonBody = json_decode($body);
    //LogMessage('PHP representation of message:');
    //LogMessage(print_r($jsonBody));

    $sFrom = $jsonBody->Header->To;
    $sTo = $jsonBody->Header->From;

    $response = GetStatusJSON('000','Transaction OK','0', '','', $sTo, $sFrom);

    echo json_encode($response);
  }

}

function logMessage($line) {
  $filename = 'receive_log'; // log file
  $fp = fopen($filename, "a");
fwrite($fp, $line . PHP_EOL);
fclose($fp);
}
```

```
function GetStatusXML($code, $msg, $messageID, $sToQual, $sFromQual, $sTo, $sFrom)
{
  //
  // be sure you have realtime_status.xml on the specified path
  //
  $xml_path = './';

  $response = simplexml_load_file("{$xml_path}realtime_status.xml");
  $response->Body[0]->Status[0]->Code[0] = $code;
  $response->Body[0]->Status[0]->Description[0] = $msg;
  $response->Header[0]->From[0] = $sFrom;
  $response->Header[0]->From->attributes()->Qualifier = $sFromQual;
  $response->Header[0]->To[0] = $sTo;
  $response->Header[0]->To[0]->attributes()->Qualifier = $sToQual;
  $response->Header[0]->MessageID[0] = $messageID;
  $response->Header[0]->SentTime[0] = date('Y-m-d H:i:s'); //some time
  $response->Header[0]->RelatesToMessageID[0] = $messageID;

  return $response;
}

function GetStatusJSON($code, $msg, $messageID, $sToQual, $sFromQual, $sTo, $sFrom)
{
  $json_path = './';
  $response = json_decode(file_get_contents("{$json_path}realtime_status.json"));
  $response->Message->Body->Status->Code = $code;
  $response->Message->Body->Status->Description = $msg;
  $response->Message->Header->From->{'#text'} = $sFrom;
  $response->Message->Header->To->{'#text'} = $sTo;
  $response->Message->Header->MessageID = $messageID;
  $response->Message->Header->SentTime = date('Y-m-d H:i:s'); //some time
  $response->Message->Header->RelatesToMessageID = $messageID;

  return $response;
}

// listen for incoming messages
listen();
```

Sample C# Code to Set Up Your Listening Endpoint

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Xml;
using System.IO;
using System.Text;
using System.Configuration;

namespace SampleEndPoint
{
    public class SampleEndpointBuiltFromSampleCode : IHttpHandler
    {
```

```
        public void ProcessRequest(HttpContext context)
        {
            context.Response.ContentType = "text/plain";
            string response = WenoMessage();
            context.Response.Write(response);
        }
        public string WenoMessage()
        {
            string response = "";

            try
            {
                // To receive authorization info which is userid:password format
                string authStr = HttpContext.Current.Request.Headers["Authorization"].Replace("Bas
ic ", "");
                authStr = Encoding.UTF8.GetString(Convert.FromBase64String(authStr));
                string messageID = "0";
                MemoryStreamms = new MemoryStream();
                // get message
                HttpContext.Current.Request.InputStream.CopyTo(ms);
                string xmlMessage = System.Text.Encoding.UTF8.GetString(ms.ToArray());
                XmlDocumentxReq = new XmlDocument();
                xReq.LoadXml(xmlMessage);

                // fetch message id
                string sMessagePath = @"/*[local-name() = 'Message']/*[local-name() = 'Header']/*[
local-name() = 'MessageID']";
                XmlNodemessNode = xReq.SelectSingleNode(sMessagePath);
                if (messNode != null)
                {
                    messageID = messNode.InnerText;
                }
                string sFromPath = @"/*[local-name() = 'Message']/*[local-name() = 'Header']/*[loc
al-name() = 'From']";
                string sToPath = @"/*[local-name() = 'Message']/*[local-name() = 'Header']/*[local
-name() = 'To']";

                string sFromQual = "";
                string sFrom = "";
                string sToQual = "";
                string sTo = "";

                XmlNodefromNode = xReq.SelectSingleNode(sFromPath);
                XmlNodetoNode = xReq.SelectSingleNode(sToPath);
                if (fromNode  != null &&toNode != null)
                {
                    sFrom = fromNode.InnerText;
                    sTo = toNode.InnerText;
                    if (fromNode.InnerText.ToUpper().Contains("D"))
                    {
                        sFromQual = "D";
                        sToQual = "P";
                    }
                    else
                    {
                        sFromQual = "P";
                        sToQual = "D";
                    }
                }
```

```
                string messType = "Unknown_Type";
                string sMessageTypePath = @"/*[local-name() = 'Message']/*[local-name() = 'Body']"
;
                if(xReq.SelectSingleNode(sMessageTypePath) != null)
                     messType = xReq.SelectSingleNode(sMessageTypePath).ChildNodes[0].Name;

                // store/process message as per your requirement
                WriteFile(xmlMessage, messType + "_ " + DateTime.Now.ToString("yyyyMMddHHmmssfff")
+ ".xml");

                string[] arrAuth = authStr.Split(new char[] { ':' });
                if (arrAuth.Length == 2)
                {
                    string reqUserID = arrAuth[0];
                    string reqPassword = arrAuth[1];
                    string ClientUserID = ConfigurationManager.AppSettings["UserID"];
                    string ClientPassword = ConfigurationManager.AppSettings["Password"];
                    // send success/failure message in xml format as specified
                    // To note here that in response to becomes from and from becomes to
                    // If you know you are a pharmacy or ehr you can set hardcoded values here
                    //response = GetStatusXmlFromWeno2017071("001", "Transaction Successfull", "",
"M", "WENO");
                    //you can authenticate request as well
                    if (reqUserID == ClientUserID&&reqPassword == ClientPassword)
                    {
                        response = GetStatusXmlFromWeno2017071("001", "Transaction Successfull", "
", "M", "WENO");
                    }
                    else
                    {
                        response = GetErrorXmlFromWeno2017071("900","Authentication Failed","","M"
,"WENO", Guid.NewGuid().ToString().Replace("-", ""));
                    }
                }
                else
                {
                    response = GetErrorXmlFromWeno2017071("900", "Invalid request", "", "M", "WENO
", Guid.NewGuid().ToString().Replace("-", ""));
                }
            }
            catch (System.Exception ex)
            {
                response = GetErrorXmlFromWeno2017071("900", "Unhandled Error", "", "M", "WENO", G
uid.NewGuid().ToString().Replace("-", ""));
            }
            //string sFinalResponse = Convert.ToBase64String(Encoding.UTF8.GetBytes(response));
            return response;
        }
        public static void WriteFile(string strContent, string fileName)
        {
            FileStream fs = null;
            FileInfo fi = new FileInfo(AppDomain.CurrentDomain.BaseDirectory + "\\Messages\\" + fi
leName);
            if (!fi.Directory.Exists)
            {
                fi.Directory.Create();
            }
            fs = new FileStream(fi.FullName, FileMode.Create);
            StreamWritersw = new StreamWriter(fs);
            sw.WriteLine(strContent);
```

```
            sw.Close();
            fs.Close();

        }
        public static string GetStatusXml1(string errorCode, string description, string messageID,
bool IsError,stringfromQual = "",string toQual = "", string from = "", string to = "")
        {
            XmlDocumentxmlTemplate = new XmlDocument();
            if (!IsError)
            {
                xmlTemplate.Load(AppDomain.CurrentDomain.BaseDirectory + "\\xmlTemplate\\Status.xm
l");
            }
            else
            {
                xmlTemplate.Load(AppDomain.CurrentDomain.BaseDirectory + "\\xmlTemplate\\Error.xml
");
            }
            XmlNodeheaderXmlTemplate = xmlTemplate.GetElementsByTagName("Header")[0];
            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Status.FromQualifier
}", fromQual);
            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Status.From}", from)
;
            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Status.ToQualifier}"
, toQual);
            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Status.To}", to);
            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Status.MessageID}",
"0");
            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Status.SentTime}", D
ateTime.Now.ToUniversalTime().ToString("yyyy-MM-ddTHH:mm:ss.fffffK"));
            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Status.RelatesToMess
ageID}", messageID);
            XmlNodebodyXmlTemplate = xmlTemplate.GetElementsByTagName("Body")[0];
            bodyXmlTemplate.InnerXml = bodyXmlTemplate.InnerXml.Replace("{Status.Code}", errorCode
);
            bodyXmlTemplate.InnerXml = bodyXmlTemplate.InnerXml.Replace("{Status.Description}", de
scription);
            return xmlTemplate.InnerXml;
        }
        public static string GetStatusXmlFromWeno2017071(string errorCode, string description, str
ing descriptionCode, string toQualifier, string to)
        {
            XmlDocumentxmlTemplate = new XmlDocument();
            xmlTemplate.Load(AppDomain.CurrentDomain.BaseDirectory + "\\xmlTemplate\\Status2017071
.xml");
            XmlNodeheaderXmlTemplate = xmlTemplate.GetElementsByTagName("Header")[0];
            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Status.FromQualifier
}", "M");
            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Status.From}", "WENO
");
            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Status.ToQualifier}"
, toQualifier);

            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Status.To}", to);

            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Status.MessageID}",
"0");

            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Status.SentTime}", D
ateTime.Now.ToUniversalTime().ToString("yyyy-MM-ddTHH:mm:ss.fffffK"));
```

```
            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Status.RelatesToMess
ageID}", "0");

            XmlNodebodyXmlTemplate = xmlTemplate.GetElementsByTagName("Body")[0];

            bodyXmlTemplate.InnerXml = bodyXmlTemplate.InnerXml.Replace("{Status.Code}", errorCode
);

            bodyXmlTemplate.InnerXml = bodyXmlTemplate.InnerXml.Replace("{Status.DescriptionCode}"
, descriptionCode);

            bodyXmlTemplate.InnerXml = bodyXmlTemplate.InnerXml.Replace("{Status.Description}", de
scription);

            return xmlTemplate.InnerXml;

        }

        public static string GetErrorXmlFromWeno2017071(string errorCode, string description, stri
ng descriptionCode, string toQualifier, string to, string MessageID = "0", string RelatesToMessage
ID = "0")

        {

            XmlDocumentxmlTemplate = new XmlDocument();

            xmlTemplate.Load(AppDomain.CurrentDomain.BaseDirectory + "\\xmlTemplate\\Error2017071.
xml");

            XmlNodeheaderXmlTemplate = xmlTemplate.GetElementsByTagName("Header")[0];

            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Error.FromQualifier}
", "M");

            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Error.From}", "WENO"
);

            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Error.ToQualifier}",
toQualifier);

            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Error.To}", to);

            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Error.MessageID}", M
essageID);

            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Error.SentTime}", Da
teTime.Now.ToUniversalTime().ToString("yyyy-MM-ddTHH:mm:ss.fffffK"));

            headerXmlTemplate.InnerXml = headerXmlTemplate.InnerXml.Replace("{Error.RelatesToMessa
geID}", RelatesToMessageID);

            XmlNodebodyXmlTemplate = xmlTemplate.GetElementsByTagName("Body")[0];

            bodyXmlTemplate.InnerXml = bodyXmlTemplate.InnerXml.Replace("{Error.Code}", errorCode)
;

            bodyXmlTemplate.InnerXml = bodyXmlTemplate.InnerXml.Replace("{Error.DescriptionCode}",
descriptionCode);

            bodyXmlTemplate.InnerXml = bodyXmlTemplate.InnerXml.Replace("{Error.Description}", des
cription);
            return xmlTemplate.InnerXml;
        }

        public bool IsReusable
        {
            get { return false; }
        }
```

```
    }
}
```

Step 11: Prep for Error/Verify Management

Repeat this step for live environment: NO

In an earlier step you tested for sending WENO the real time Error Message, but in this step, you must understand the ERRORs you will receive when there is a problem identified by a recipient of your message; as well as, the VERIFY message, if you flag messages sent for return receipt.

Review the information shown here, so you can prepare your application for these Error and Verify messages:

Cert dashboard ->Testing Tool -> Error/Verify Management

Step 12: Practice Sending & Receiving

Repeat this step for live environment: YES

You will just practice receiving NewRxs and responding in real time if using the quick certification method.

If you are doing the full certification, then practice sending and receiving the message types you will support.

Remember To Go Beyond Schema Validations

WENO requires your SCRIPT messages to go beyond schema validation, so refresh yourself with the use of these resources found on your Testing Tool page while practicing:

i. Validator Tool – toggle dropdown to validate against a specific schema
ii. SCRIPT Guidance & Samples / What We Validate & NewRx Best Practices
iii. Your Test Plan – will give you the test prescriber ID to use and your test plan.

Dashboard's Transaction Page for Drilling Down

Use the Transaction page on your dashboard to find all the messages you send/receive. You can drill down to the actual messages sent and received for troubleshooting or to get details.

Endpoints Used to SEND if Doing the Full Certification

To understand the correct endpoint, go to: dashboard ->Testing Tool -> Endpoints To Send SCRIPT Messages to others

To Practice Receiving Messages to your Listening Endpoint(s)

go to cert dashboard ->Testing Tool -> Endpoint Tester and follow the instructions on the screen.

Sample PHP Code to Send Messages for Routing

```
try
{
 $message = '';
 echo $message;

 $soapclient = new SoapClient('https://cert.wenoexchange.com/wenox/service.asmx');
 $param=array('inputString'=>$message);
 $response =$soapclient->wenoswitch($param);
 var_dump($response);
 $array = json_decode(json_encode($response), true);
 print_r($array);
```

```
 foreach($array as $item)
 {
   var_dump($item);
 }
}
catch(Exception $e)
{
 echo $e->getMessage();
}
```

Sample C# Code to Send Messages for Routing

Ask your certification manager for this if you need the sample in C#.

Step 13: Access Financial Reports

Repeat this step for live environment: YES

For PTVs and MainPTVs Only

Your dashboards provide a way for us to push your endpoint a report of billable transactions. This report will not only map our invoice to you, it will provide a way for you to do downstream billing. The report is always accessible by manual download as well.

Prior to going live, we must know who to invoice for billable transactions. Let your certification manager know this if this is not already arranged.

Step 14: Complete Your Test Plan

Repeat this step for live environment: NO

To understand how to complete your test plan and go live go to your cert dashboard ->Testing Tools ->Test Plan & Go Live

Step 15: Going Live

When you are cleared to go live:

1. Your certification manager will register you for the live dashboard.
2. These steps to repeat for live preparation are indicated in their subtitles.
3. Contact us to do a press release or announcements.
4. Remember your certification environment will continue with your live account and can be used for future development or testing.
5. We will communicate any updates or changes as needed, so keep your company data current including developers with access to your account.

We look forward to serving your ePrescribing needs and thank you for your business.

Version History

September 20, 2022 – Initial version for website