



Switch EHR API Documentation

A Step-by-Step Guide to Going Live

Last update 09-20-2022

CONTENTS

Overview	2
Step 1: Sign Up & Attend Kick Off Meeting.....	2
Repeat this step for live environment: NO	2
Step 2: Set Up Your Certification Dashboard	2
Repeat this step for live environment: YES.....	2
Step 3: Manage Your ePrescribing Participants.....	2
Repeat this step for live environment: YES.....	2
Step 4: Incorporate Drug DB	3
Repeat this step for live environment: YES.....	3
Step 5: Access Directory Data	4
Repeat this step for live environment: YES.....	4
Step 6: Understand the Script Header	4
Repeat this step for live environment: NO	4
Step 7: SCRIPT Body & Samples	4
Repeat this step for live environment: NO	4
Step 8: Register IPs.....	5
Repeat this step for live environment: YES.....	5
Step 9: Understand the Endpoint COMMUNICATIONS FLOW.....	5
Repeat this step for live environment: NO	5
Step 10: Create & Test Your Listening Endpoint	6
Repeat this step for live environment: YES.....	6
Step 11: Prep for Error/Verify Management	13
Repeat this step for live environment: NO	13
Step 12: Practice Sending & Receiving.....	14
Repeat this step for live environment: YES.....	14

Step 13: Access Financial Reports & Get Discount on Transaction Fees	15
Repeat this step for live environment: YES.....	15
<i>To Get a Discount on Transaction Fees</i>	15
Step 14: Complete Your Test Plan.....	15
Repeat this step for live environment: NO	15
Step 15: Going Live.....	15
Version History.....	16

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!

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

1. Go to [Manage Account](#) ->[Manage Clinics](#) and add at least 1 test clinic. Things to know about locations:

Locations come before prescribers because prescribers must be assigned to at least one location.

Each location must show the type of messages it can handle receiving.

All ePrescribing SCRIPT messages sent have corresponding messages from recipients. Location(s) registered will indicate demographics, types of message(s) they can receive back; and a way to assign a specific endpoint to the location, if the company's endpoint will not handle the messages for that location.

2. Go to [Manage Account ->Manage Prescribers](#) and add at least 1 test prescriber. We recommend adding a signature image that can be used for failovers when ePrescribing fails and it is legal to fax the drug rather than fail the transaction.
3. Understand WENO's "Routing ID" convention. Each prescriber has a routing ID made up of 2 parts separated by a colon. The first part identifies the prescriber and the 2nd part identifies the location of the prescriber.

If a prescriber goes to a physical clinic and another location is telehealth or an inpatient facility then the clinic may accept RxRenewalRequest and the urgent care center may not. Pharmacies receive this information in a prescriber directory so they can understand routing information in the same way EHRs receive information to explain a pharmacy's routing information.

Routing ID sample: D1234:C134234 means the D part identifies the prescriber & the C part is the location. Together they make up the routing id.

4. Managing prescribers and clinics can be done manually, if a few, or via our Manage APIs, if many. You are required to keep this information current at all times. To use Manage APIs, you will find the documentation, schema, endpoints, and samples on your cert dashboard here:

[Testing Tools -> Manage Account APIs](#)

Step 4: Incorporate Drug DB

Repeat this step for live environment: YES

1. Purchase the WENO drug database on our website if you have not done so already.
 - a. <https://wenoexchange.com/products-for-sale/>
 - b. The Sample Drug Database is on this page to check it out.
2. Go to [Testing Tools -> Drug Database](#) & download the **Sample Drug Database**, which is a spreadsheet. Go to the sheet called **How to Use the Data** and learn about it and exactly how to use the data for both your UI and the NewRx fields.
3. The Sample Drug Database spreadsheet will also explain how you will call to access and update it about the 25th of each month. Do a full file replace and incorporate it appropriately into your own data.
4. **Changes from time to time.** At our sole discretion, we may add information to the drug database that we think will be helpful. If we do, we will always append it to the right of the last column. Plan accordingly so you won't break if you discover a new column of information added. The sample file will tell you how to use the new information.

Step 5: Access Directory Data

Repeat this step for live environment: YES

WENO's pharmacy directory is available for you to use for your user interface to enter a patient's primary and alternative pharmacy choices as well as to populate the ePrescribing messages going to pharmacies.

1. You are required to call for pharmacy directory updated daily and do a full file replace weekly. If you fail to keep this updated you will experience delivery errors because pharmacies open, move, are bought out, change their name or other attributes, go out of business, and update their WENO connectivity status most every day.
2. From [dashboard -> Testing Tools – Directories](#). Download the first pharmacy directory & open the READ ME file. It will provide what some of the information you need to understand how to use the data and sample code to call for it daily and weekly as required.
3. Learn how to use it for your user interface here: [dashboard ->SCRIPT Guidance and Samples >NewRx Best Practices](#) (in the Pharmacy chapter).
4. **Changes from time to time.** At our sole discretion, we may add information to the pharmacy directory file that we think will be helpful. If we do, we will always append it to the right of the last column. Plan accordingly so you won't break if you discover a new column of information added. The READ ME file will be updated to explain how to use the new information.

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

1. In order to select the SCRIPT message types you plan to support; you will need their schemas. 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 NewRx prescribers will sign & complete on WOL)

- Next find the Samples and Guidance. Review samples of the messages you plan to support. Then understand what we validate and about NewRx Best Practices. It will help you plan your UI and avoid common pitfalls when implementing the fields that make sense for your environment. To do this go to:

Dashboard ->[Testing Tools](#) -> [SCRIPT Body and Guidance](#)

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

Step 8: Register IPs

Repeat this step for live environment: YES

- 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](#).
- Each of your registered locations can have specifically assigned IPs registered as well if this is needed.

Step 9: Understand the Endpoint COMMUNICATIONS FLOW

Repeat this step for live environment: NO

Every ePrescribing partner needs a listening endpoint or WENO cannot communicate to you. In the next step we teach you how. This step explains why you need one and what happens if it breaks:

Why have a “listening endpoint” ?:

Even if your EHR will only support sending NewRx, you must still have a listening endpoint. It will serve, at a minimum, for WENO to communicate transmission failures (ERROR messages) back to you; and, if you flag NewRx for return receipt, you can expect to receive both the VERIFY or ERROR messages.

Anytime now or later your same listening endpoint can be tested and then enabled to receive other SCRIPT message types you plan to support. Like RxRenewals, ChangeRx, etc.

Understand how Intermediary Communication Works:

- You send WENO’s endpoint a NewRx or another message you will support sending.
- WENO will respond in real time (Error or Success). It only means WENO accepts/rejects.
- The call is ended.
- Then WENO attempts to deliver the message to the recipient.
- If it fails, WENO will return its ERROR message to your listening endpoint.
- If it succeeds, your endpoint only gets the VERIFY message if you flagged it for return receipt.
- IF your endpoint breaks or is not set up before you start sending messages, 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, $sFrom);

// 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("Basic ", "");

                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']";
        XmlNode messNode = 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 = "";

        XmlNode fromNode = xReq.SelectSingleNode(sFromPath);
        XmlNode toNode = 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);
    StreamWriter sw = 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.xml
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.ffffffK"));
        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.ffffffK"));
        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.ffffffK"));
        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

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 explain use of test data & give you a roadmap to live.

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

To understand the correct endpoints used to send go to:

Dashboard ->Testing Tool -> Endpoints To Send SCRIPT Messages to others

Test Pharmacies to Use

Here are some Test Pharmacies to send to with will give you results as hinted by in the name or NCPDP:

Pharmacy Name	NCPDP
Test Direct Pharmacy	1234567
Test WOL Registered Pharmacy	7654321
Test WOL TBI Pharmacy (not registered yet)	TestWOLPendingPharmacy
Test WOL Fax Alert Failure Pharmacy	TestWOLFaxAlertFailurePharmacy
Test Transmission Failure	TestTransmissionFailure
Test ERx Rejected by Pharmacy	TestERxRejectedByPharmacy

Find the Sending Endpoint and Details

Go to cert dashboard ->Testing Tool -> Endpoint to Send SCRIPT Messages to Others to get the specific endpoint and details.

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 if you need this sample code.

Step 13: Access Financial Reports & Get Discount on Transaction Fees

Repeat this step for live environment: YES

WENO provides a file for you to be detailed on the transactions you send that are billable. You can always access it on dashboard. It will map to any invoice for transactions. Prior to go live, your certification manager will ask you for a credit card to bill for transaction fees.

To Get a Discount on Transaction Fees

Ask your Certification Manager to send you a very simple way to get your discount. Not all EHRs qualify.

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