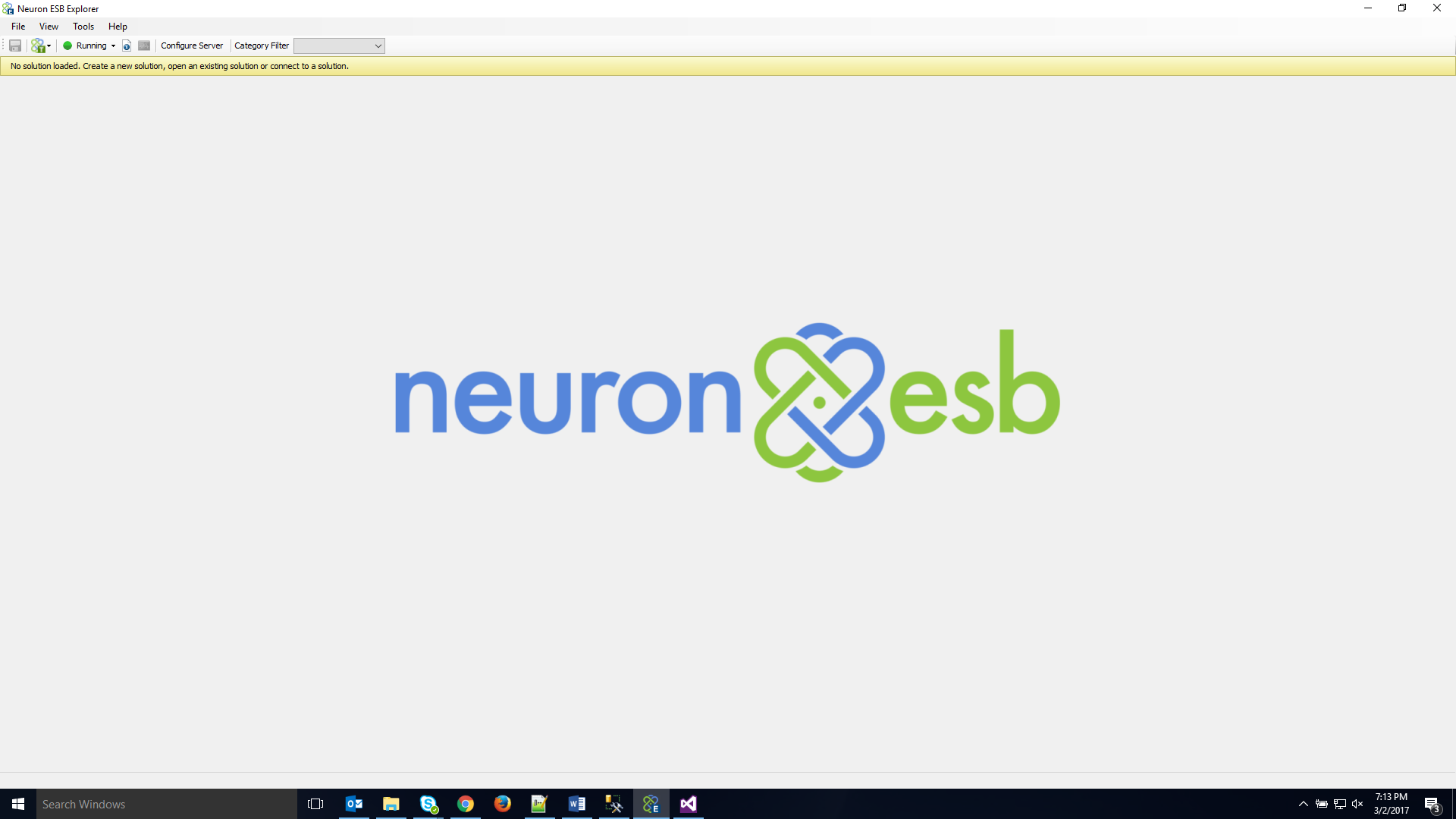
Neuron EDI/X12 Processing

*Using OopFactory X12 Parser*

Introduction

The purpose of this document is to provide guidance on processing EDI/X12 data in Neuron ESB, using the OopFactory open-source parser. In addition, a Windows Forms application has been built to manage Trading Partner Agreements, with the Agreement information stored in a new database, NeuronEDI.



Contents

[**Trading Partner Agreement Management** 3](#_Toc476244710)

[ **Creating Trading Partners** 3](#_Toc476244711)

[ **Updating Trading Partners** 4](#_Toc476244712)

[ **Creating Agreements** 5](#_Toc476244713)

[ **Updating Agreements** 5](#_Toc476244714)

[ **Creating/Updating Trading Partners in Bulk** 6](#_Toc476244715)

[ **Creating/Updating Agreements in Bulk** 6](#_Toc476244716)

[ **Processing Inbound EDI/X12** 7](#_Toc476244717)

[o **EDI/X12 To XML Serialization** 7](#_Toc476244718)

[o **Debatching** 7](#_Toc476244719)

[o **Tracking** 7](#_Toc476244720)

[o **Trading Partner Agreement Validation** 7](#_Toc476244721)

[o **Verifying Inbound Control Numbers** 7](#_Toc476244722)

[o **Dynamic Mapping** 7](#_Toc476244723)

[o **Generating 999 Acknowledgements** 7](#_Toc476244724)

[ **Processing Outbound EDI/X12** 8](#_Toc476244725)

[o **Batching** 8](#_Toc476244726)

[o **XML Input Format** 8](#_Toc476244727)

[o **Debatching** 9](#_Toc476244728)

[o **Trading Partner Agreement Validation** 9](#_Toc476244729)

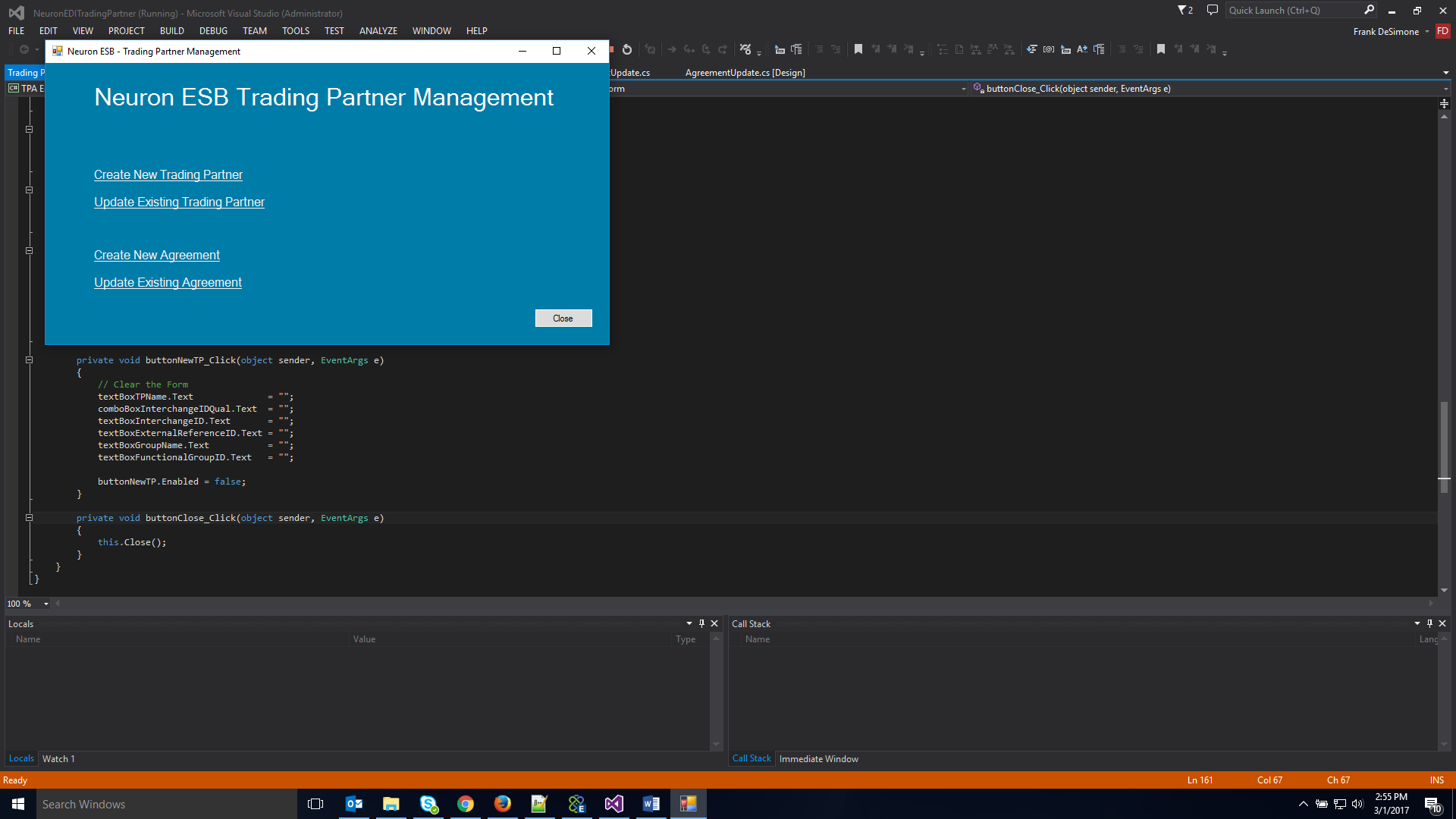
[o **Maintaining Outbound Control Numbers** 9](#_Toc476244730)

[o **Dynamic Mapping** 9](#_Toc476244731)

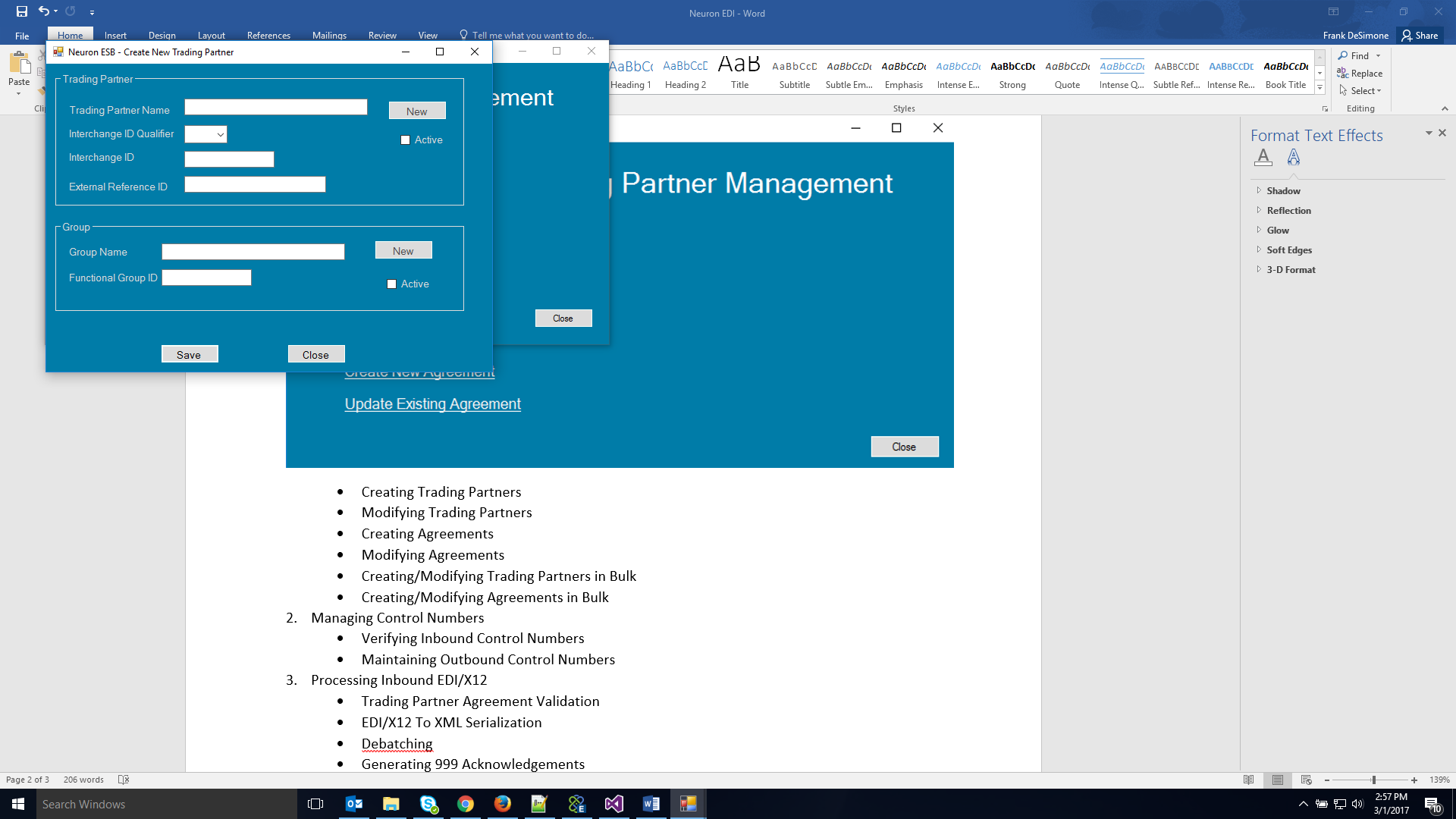
[o **Managing Outbound EDI/X12 Envelopes** 9](#_Toc476244732)

[o **XML To EDI/X12 Serialization** 9](#_Toc476244733)

# **Trading Partner Agreement Management**



* **Creating Trading Partners**



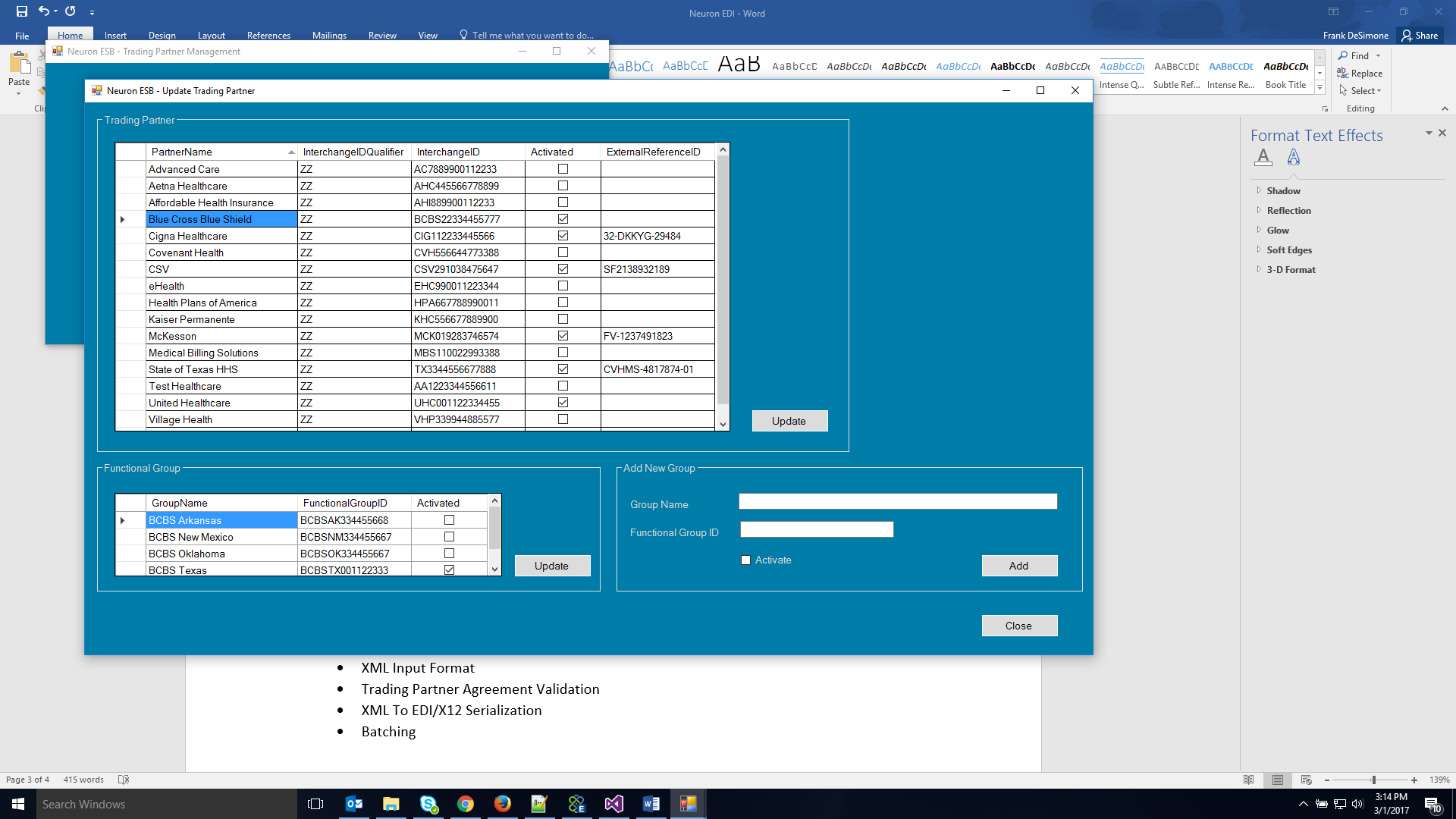
**Trading Partner**

1. Enter the Trading Partner Name
2. Select an Interchange ID Qualifier from the drop down list
3. Enter the Interchange ID
4. Enter an External Reference ID (optional). This can be a value that uniquely identifies the entity in another system, for example, the SalesForce ID. This value can then be used as a cross-reference by other applications to lookup partner/agreement information
5. Checking the ‘Active’ checkbox enables this Trading Partner to send/receive EDI/X12 documents, provided there is an active Agreement as well. Leaving the ‘Active’ checkbox unchecked keeps this Trading Partner disabled. They will exist in the database, but will not be allowed to send/receive EDI/X12 documents
6. Clicking the ‘New’ button clears the screen so that a new Trading Partner can be added

**Group**

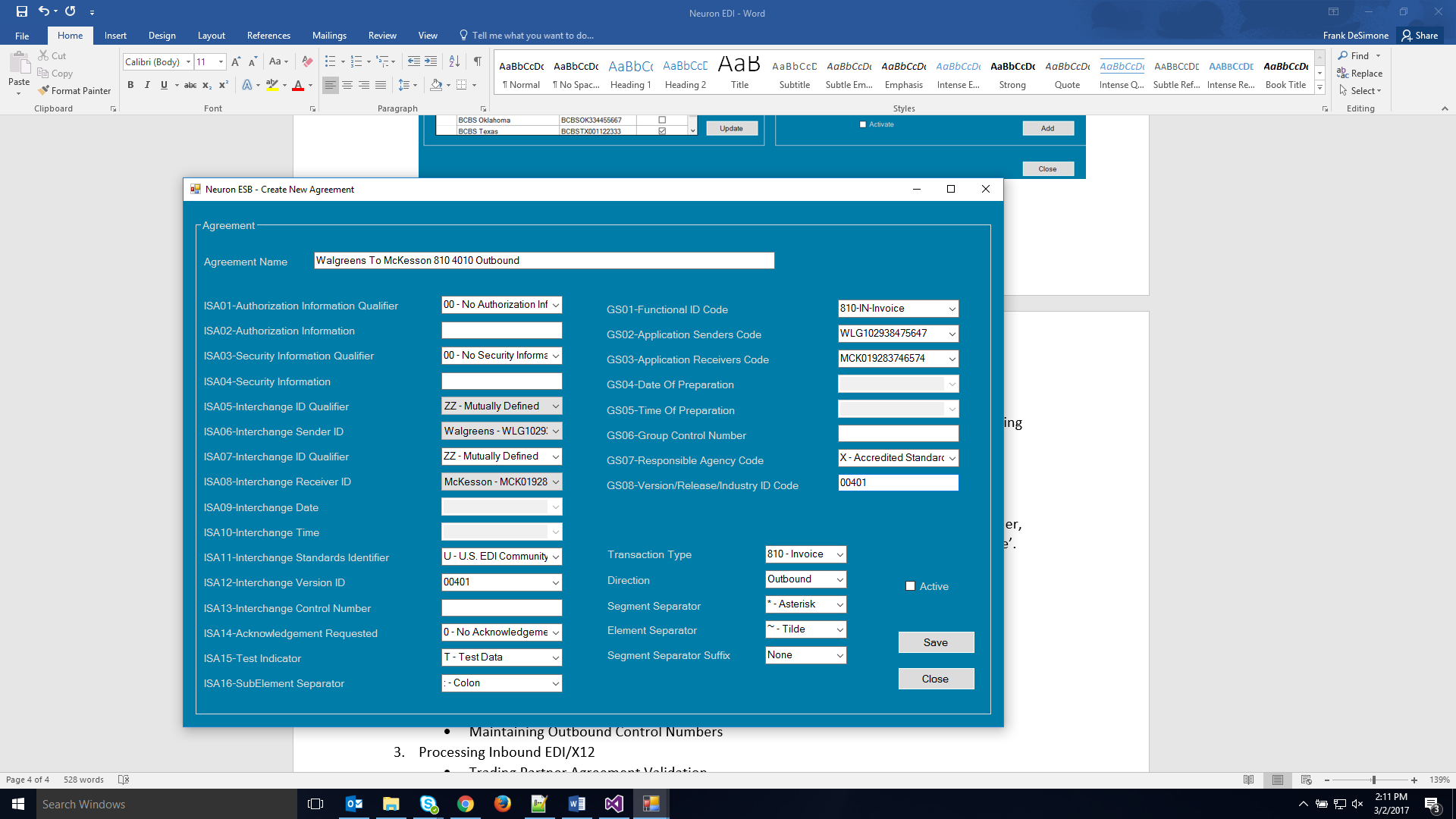
1. Enter a name for the Group associated with the Trading Partner above
2. Enter the Group’s Functional ID
3. Checking the ‘Active’ checkbox enables this Group. Leaving it unchecked keeps the Group disabled
4. To add a new Group to this Trading Partner, first click ‘Save’, then click ‘New’ in the Group container. This clears the Group information. Enter in the new Name and Functional Group ID. Then click ‘Save’ to add the new Group.

* **Updating Trading Partners**



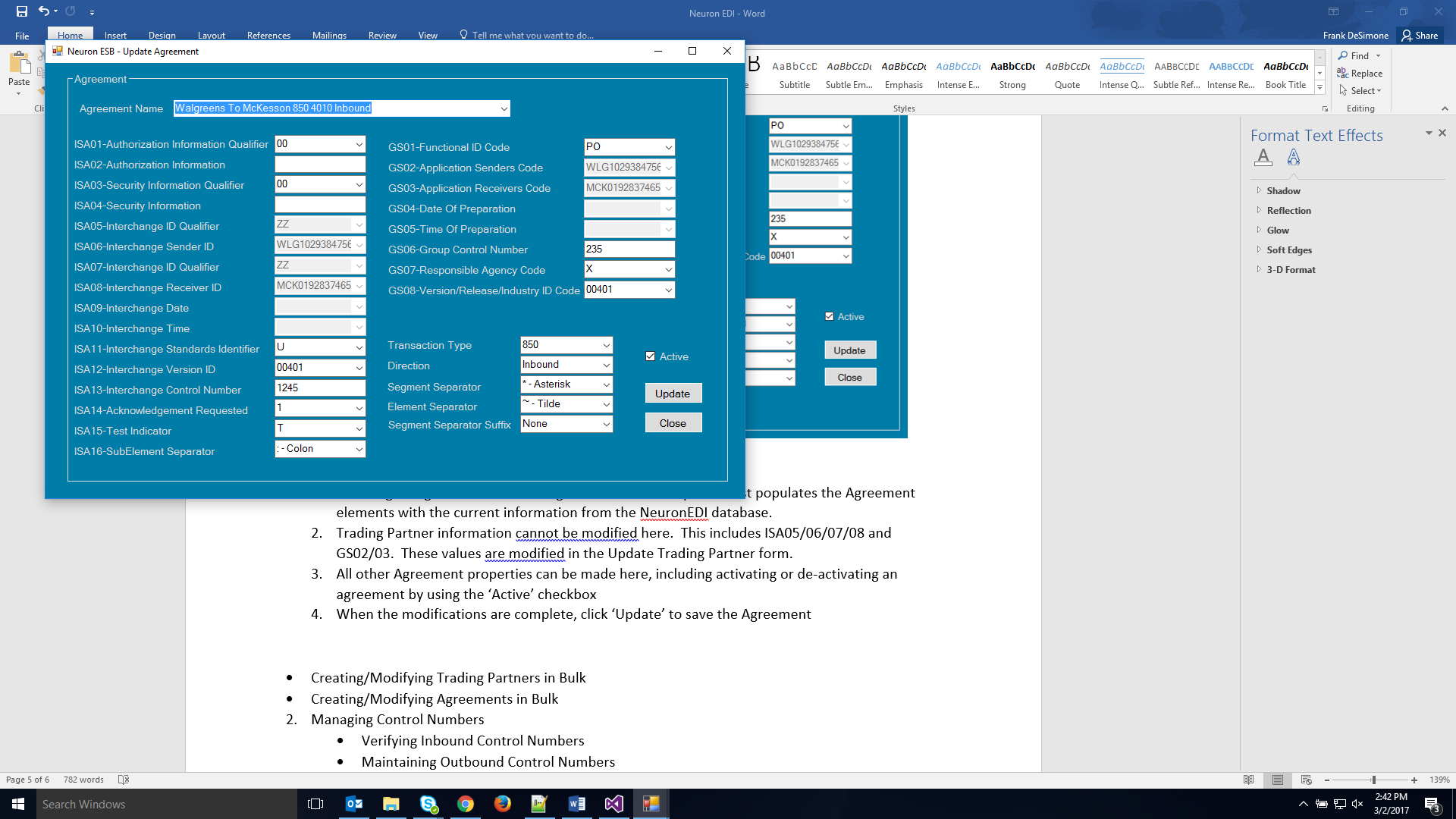
1. The data grid view in the Trading Partner Container allows for fast, simple editing of Trading Partner information.
2. Clicking on any cell in the Trading Partner data grid view also displays associated Group information in the Functional Group data grid view
3. The user can edit any of the cells as needed. Then click ‘Update’ to save the changes.
4. The user can also add a new Group. Enter the information in the Add New Group Container, click ‘Add’. The new group appears in the Functional Group Container. Then click ‘Update’.

* **Creating Agreements**



1. To create a new Agreement, first enter a name for the Agreement in the Agreement Name text box. A useful format for the Agreement Name is <sender> To <receiver> <transaction type> <version> <direction>
2. For elements that have an enumeration restriction, a drop down list of valid values will allow the user to select the appropriate value
3. The ISA06(interchange sender) and ISA08(interchange receiver) drop down lists are populated with Trading Partner IDs that have been setup in the system. Selecting a Trading Partner ID from the list also populates the corresponding GS02/GS03 with the Groups that have been setup for that Trading Partner.
4. For the Interchange and Group Control Numbers (ISA13 and GS06), the user may enter the starting control number for the Agreement. If no value is entered, a default value of 0 is used.
5. The Separator values in the drop down lists are restricted to non-alphanumeric Unicode characters (Space character is also excluded) in the C0 set, XML 1.0 standard (<https://www.w3.org/TR/xml-entity-names/000.html>)
6. Checking the ‘Active’ checkbox enables the Agreement

* **Updating Agreements**



1. Selecting an Agreement from the Agreement Name drop down list populates the Agreement elements with the current information from the NeuronEDI database.
2. Trading Partner information cannot be updated here. This includes ISA05/06/07/08 and GS02/03. These values are updated in the Update Trading Partner form.
3. All other Agreement properties can be updated here, including activating or de-activating an agreement by using the ‘Active’ checkbox
4. When the updates are complete, click ‘Update’ to save the Agreement

* **Creating/Updating Trading Partners in Bulk**

1. In scenarios where many Trading Partners must be created or updated, it may not be practical to perform these operations one at a time via the Windows Form application.
2. To create or update Trading Partners in bulk, call Neuron process ‘Process.EDI.Partner.Configure’, passing an XML document that contains the partner information
3. The schema for the Partner XML document is in the Neuron Repository under XML Schemas🡪EDI.TradingPartner

* **Creating/Updating Agreements in Bulk**

1. As with Trading Partners, in scenarios where many Agreements must be created or updated, it may not be practical to perform these operations one at a time via the Windows Form application.
2. To create or update Agreements in bulk, call Neuron process ‘Process.EDI.Agreement.Configure’, passing an XML document that contains the agreement information
3. The schema for the Agreement XML document is in the Neuron Repository under XML Schemas🡪EDI.Agreement

* **Processing Inbound EDI/X12**
  + **EDI/X12 To XML Serialization**
    - An inbound X12 document is serialized to XML by the OopFactory parser. This happens in a code step as follows:

var parser = new OopFactory.X12.Parsing.X12Parser();

OopFactory.X12.Parsing.Model.Interchange interchange = parser.Parse(context.Data.Text);

string xml = interchange.Serialize();

context.Data.Text = xml;

* + **Debatching**
    - After the conversion to XML, the document is debatched by Functional Group, then by Transaction
  + **Tracking**
    - After debatching, there is an Audit step to capture the X12 envelope properties for tracking/reconciliation
  + **Trading Partner Agreement Validation**
    - After the Audit step, the Neuron process calls stored procedure sp\_Agreement\_Validate to verify that an Agreement exists for the X12 Sender, Receiver, Transaction Type, Version, and Direction. The stored procedure also loads
  + **Verifying Inbound Control Numbers**
    - The sp\_Agreement\_Validate stored proc returns the previous control numbers that were received for the Agreement. If the current and previous control numbers are not in sequence, i.e., the current number is not equal to the previous number + 1, then a warning message is logged by Neuron. This may indicate a duplicate transmission(if both current and previous are the same), or may indicate a missing transmission(if current is more than previous + 1).
  + **Dynamic Mapping**
    - If the Agreement is valid, then the Group is debatched by Transaction (ST/SE) and dynamically mapped to internal/canonical format
  + **Generating 999 Acknowledgements**
    - After the mapping is complete, Neuron checks the ISA14-Acknowledgement Requested field. If the value is ‘1’, Neuron calls OopFactory to generate a 999 response.
    - The ISA and GS envelope values are then set, and Neuron calls OopFactory to create the EDI/X12 999 as follows:

//Create memory stream

var service = new OopFactory.X12.Validation.X12AcknowledgmentService();

byte[] byteArray = System.Text.Encoding.ASCII.GetBytes(context.Data.Text);

System.IO.MemoryStream stream = new System.IO.MemoryStream(byteArray);

//execute 999 method

var responses = service.AcknowledgeTransactions(stream);

// Set interchange envelope values

var interchange = new OopFactory.X12.Parsing.Model.Interchange(DateTime.Now, int.Parse("000000001"), true);

interchange.AuthorInfoQualifier = context.Data.GetProperty("X12", "999-isa01").ToString();

interchange.AuthorInfo = context.Data.GetProperty("X12", "999-isa02").ToString();

interchange.SecurityInfoQualifier = context.Data.GetProperty("X12", "999-isa03").ToString();

interchange.SecurityInfo = context.Data.GetProperty("X12", "999-isa04").ToString();

interchange.InterchangeSenderIdQualifier = context.Data.GetProperty("X12", "999-isa07").ToString();

interchange.InterchangeSenderId = context.Data.GetProperty("X12", "999-isa08").ToString();

interchange.InterchangeReceiverIdQualifier = responses.First().SenderIdQualifier;

interchange.InterchangeReceiverId = responses.First().SenderId;

interchange.SetElement(12, context.Data.GetProperty("X12", "999-isa12").ToString());

// Set Group envelope values

var group = interchange.AddFunctionGroup(context.Data.GetProperty("X12", "999-gs01").ToString(), DateTime.Now, int.Parse(context.Data.GetProperty("X12", "999-gs06").ToString()));

group.ApplicationSendersCode = context.Data.GetProperty("X12", "999-gs02").ToString();

group.ApplicationReceiversCode = context.Data.GetProperty("X12", "999-gs03").ToString();

group.VersionIdentifierCode = context.Data.GetProperty("X12", "999-gs08").ToString();

OopFactory.X12.Validation.X12Extensions.Add999Transaction(group, responses);

context.Data.Text = interchange.SerializeToX12(true);

* **Processing Outbound EDI/X12**
  + **Batching**
    - The Neuron Outbound EDI process assumes that the outbound data is already batched, and that it includes the data necessary to lookup the Agreement. This can take place in a preceeding Neuron process, or by another application.
  + **XML Input Format**
    - The format of the input XML document must include Interchange, Group, and Transaction nodes. The Group node must contain a node called AgreementLookup that contains the information necessary to lookup the outbound Agreement. The Transaction node(s) will contain the actual message(s) that will be mapped to EDI/X12 format.

Example:

<Interchange>

<Group>

<AgreementLookup>

<SenderInterchangeID>TX3344556677888</SenderInterchangeID>

<ReceiverInterchangeID>CIG112233445566</ReceiverInterchangeID>

<InterchangeVersion>00401</InterchangeVersion>

<TransactionType>820</TransactionType>

<SenderGroupID>TX3344556677888</SenderGroupID>

<ReceiverGroupID>CIG112233445577</ReceiverGroupID>

<GroupVersion>00401</GroupVersion>

</AgreementLookup>

<Transaction>

<Header>

<TransactionHandlingCode>C</TransactionHandlingCode>

<TotalPaymentAmount>3281.44</TotalPaymentAmount>

<CreditDebitFlagCode>C</CreditDebitFlagCode>

<PaymentMethodCode>ACH</PaymentMethodCode>

<PaymentFormatCode>CTX</PaymentFormatCode>

<PayerDFINumberQual>01</PayerDFINumberQual>

<PayerDFINumber>3487432</PayerDFINumber>

.

.

.

</Transaction>

</Group>

</Interchange>

* + **Debatching**
    - The XML document is then debatched by Group, then by Transaction.
  + **Trading Partner Agreement Validation**
    - At the Group level, Neuron calls the sp\_Agreement\_Get stored procedure to verify that an Agreement exists, and returns the Agreement properties
  + **Maintaining Outbound Control Numbers**
    - The sp\_Agreement\_Get stored procedure increments the current control numbers in the database, and returns them to Neuron. It also rolls the numbers over to ‘1’ if the upper limit **has been reached (‘999999999’).**
  + **Dynamic Mapping**
    - At the Transaction level, mapping is executed for each Transaction. The output XML must follow the structure of the OopFactory specification for the given Transaction.
  + **Managing Outbound EDI/X12 Envelopes**
    - After the mapping, the ST/SE envelope information is generated. Then, at the Group level, the GS/GE envelope information is generated
  + **XML To EDI/X12 Serialization**
    - At this point, we have an XML representation of the outbound EDI/X12 document that follows the OopFactory specification. Neuron then calls OopFactory to convert the XML to EDI/X12 format as follows:

var parser = new OopFactory.X12.Parsing.X12Parser();

string x12 = parser.TransformToX12(context.Data.Text);

context.Data.Text = x12;