



New Features in Neuron ESB 3.0

The release of Neuron ESB 3.0 continues the trend of productivity and innovation for the Microsoft Platform. Neuron ESB 3.0 has been extended to support the Microsoft .NET 4.x Framework and Visual Studio 2010 and 2012. This latest version introduces entirely new configuration, deployment, and management features that will propel large organizations forward more quickly when using Neuron ESB in complex development environments.

Neuron ESB's overriding theme continues to be ease of use and adoption for Microsoft .NET Developers. In this vein, Neuron ESB 3.0 introduces a new Samples Viewer and new samples so that new users can easily learn Neuron ESB. Additionally, significant changes have been made to the Neuron ESB Toolset allowing users to develop Neuron ESB related solutions even faster than before. Lastly, Neuron ESB 3.0 introduces new capabilities expanding custom Business Process development, transports and connectivity options.

Businesses choose Neuron ESB to solve their service and integration problems, while reducing the total cost of ownership of their solutions. Neuron ESB 3.0 delivers, by continually refining and extending its core capabilities in ways that can be effectively leveraged by Microsoft .NET Developers, in agile environments.

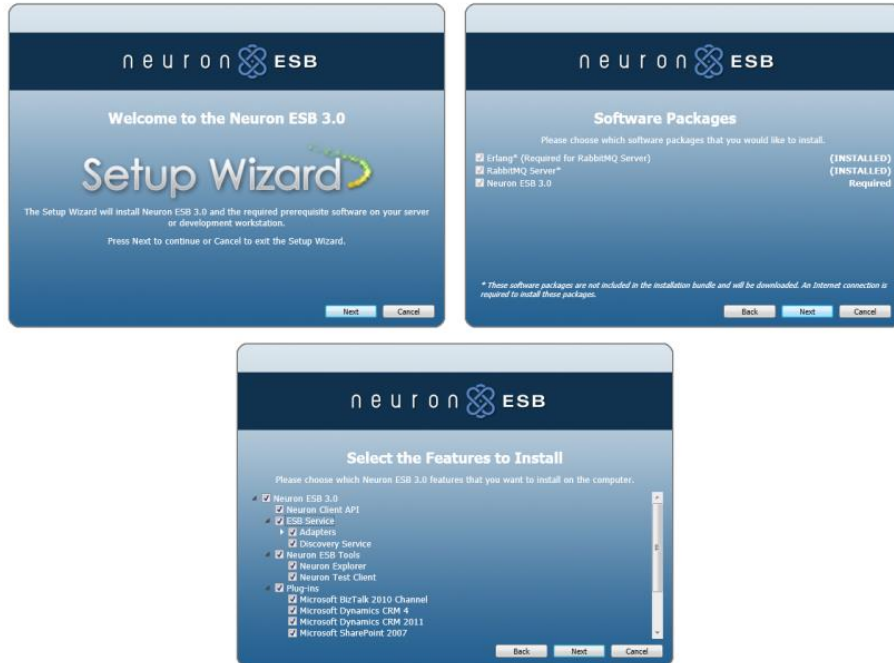
Neuron ESB 3.0 introduces many new features and enhancements, some of which include:

- ⊗ [Management and Administration](#)
 - [New User Interface Experience](#)
 - [Queue Management](#)
 - [Server and Instance Management](#)
 - [Dependency Viewers](#)
 - [Global Category Filters](#)
- ⊗ [Deployment and Configuration Management](#)
 - [New Neuron ESB Configuration storage](#)
 - [Multi Developer support](#)
 - [Incremental Deployment](#)
 - [Command line Deployment](#)
- ⊗ [Microsoft .NET 4/LINQ](#)
 - [Visual Studio 2010 and 2012](#)

- ⊗ **Business Process Designer**
 - [Referencing External Assemblies](#)
 - [Zoom, Cut, Copy and Paste](#)
 - [New Process Steps](#)
 - [Duplicate Message Detection](#)
 - [For Each loop](#)
 - [ODBC](#)
 - [Custom Process Steps](#)
 - [Interface for Controlling UI Properties](#)
 - [Folder hierarchy for UI display](#)
- ⊗ **Neuron Auditing**
 - [Microsoft SQL Azure](#)
 - [Excluding Body and Custom Properties](#)
 - [Failed Message Monitoring](#)
- ⊗ **Adapters and Connectivity**
 - [Topics using Rabbit MQ/AMQP](#)
 - [Topics using MSMQ](#)
 - [POP3 and Microsoft Exchange Adapters](#)
 - [ODBC Adapter enhancements](#)
 - [Azure Service Bus Adapter](#)
- ⊗ **Service Broker**
 - [REST enhancements](#)
 - [REST support for Service Policies](#)
 - [WSDL support for hosted services](#)

Management and Administration

Neuron ESB 3.0 introduces a number of new management features as well as user interface enhancements to facilitate the development and management of solutions, from design through to deployment. The Neuron ESB installation has also been modified to include a bootstrap Setup.exe, which launches the new installation process. This new bootstrap program walks users through optional dependencies to install like RabbitMq and Erlang, in addition to the existing list of Neuron ESB components. This same installation process can also be used to install additional instances of the Neuron ESB runtime service; previously only accomplished by running PowerShell scripts.

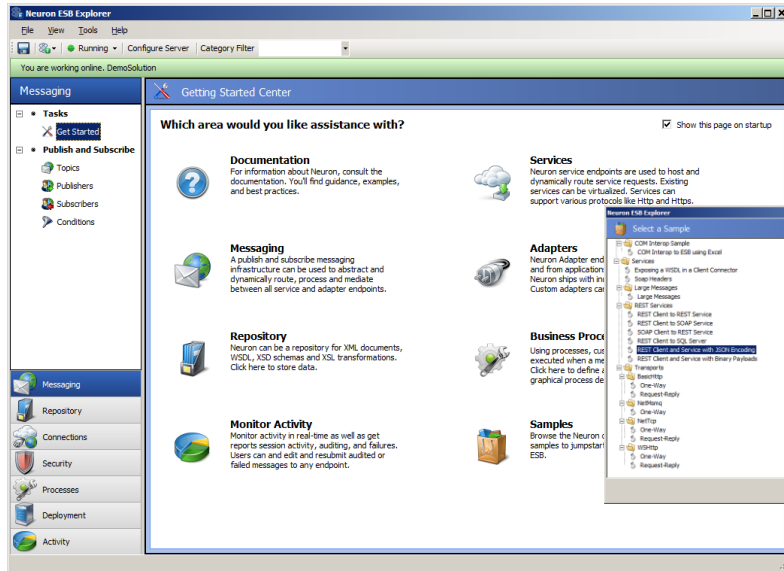


New User Interface Experience

Significant modifications have been made to the Neuron ESB Explorer, both from a functional and usability perspective. When users first open the Neuron ESB Explorer, they are presented with a new “Neuron ESB Connect” screen that provides the ability to connect to various running instances on either local or remote servers, as well open existing configurations or create new ones. If a user chooses to connect, the Neuron ESB runtime service will be automatically started, if stopped. There is no need to manually start services.



Additionally, all long-running operations and processes within the Neuron ESB explorer are now asynchronous. This means operations can be cancelled and navigation to other functional areas are no longer blocked, resulting in a more consistent, usable and reliable experience for users.



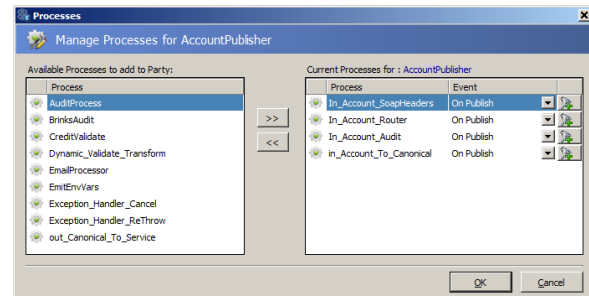
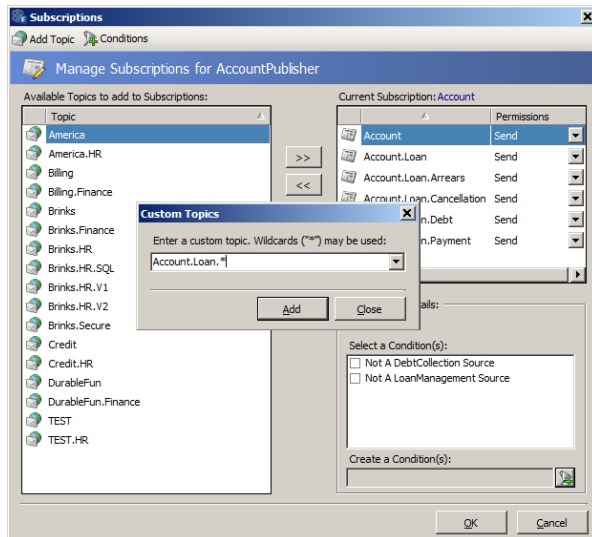
The “Getting Started” screen of the Neuron ESB Explorer, the default starting point for all users, is noticeably different. The toolbar, navigation, taxonomy, icons and

many other functional areas have changed, providing new capabilities and making existing features easier to use.

Prominently placed on the “Getting Started” screen is the “Samples”

launcher. Neuron ESB now ships with an improved “Samples” launcher along with 50 samples, 18 of which are new to Neuron ESB 3.0 and demonstrate various capabilities. For example, users will find samples on how to write custom adapters and Business Process Steps. Samples exist that demonstrate a variety of service broker and Neuron ESB Topic-based publish and subscribe messaging patterns. There are also samples for developing custom Business Processes for mediation and general application integration (EAI).

Managing subscriptions and Business Processes for Publishers and Subscribers has also become easier and more flexible with the new user interfaces depicted below. Users can quickly add topics, wildcards, reusable conditions (i.e. content based routing), and ad hoc conditions to define a subscription and



activate it instantaneously. Consequently, users have the flexibility to add custom Business Processes to either Publishers or Subscribers change the order of those processes, define the conditions on which

they execute, and modify them at runtime.

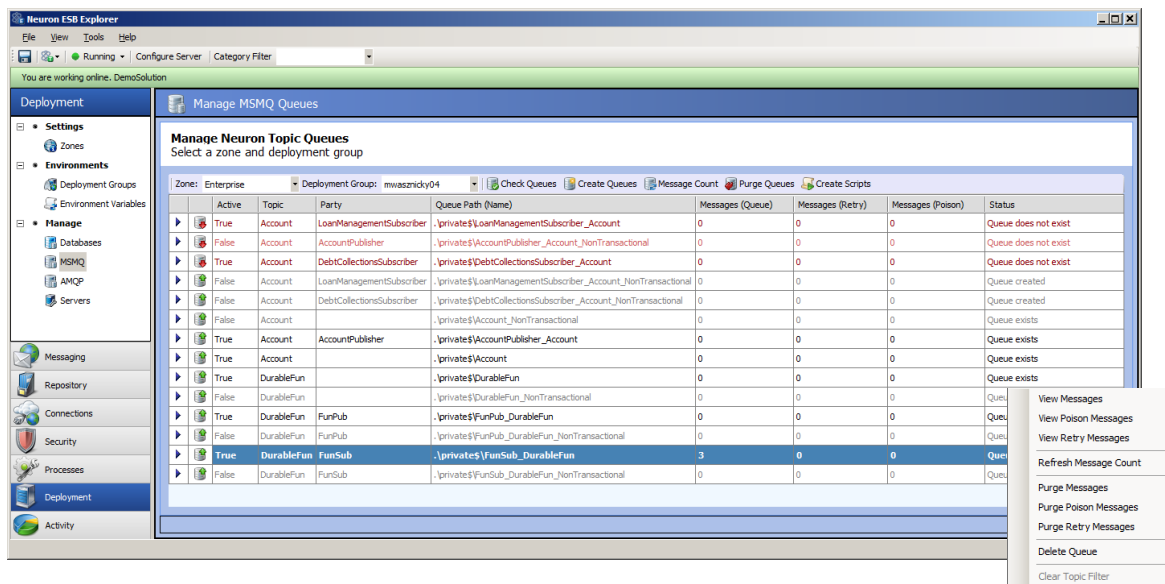
Queue Management

Neuron ESB provides a hierarchical, Topic-based publish and subscribe model to mediate the routing of messages between Parties (Publishers and Subscribers), Adapters, and Service Endpoints. Durable,

guaranteed, and reliable messaging as well as in-memory routing of messages, can be provided by changing the Transport property of a Neuron ESB Topic. Neuron ESB 3.0 provides a number of Transports that users can select for Topics, including those that provide durable, guaranteed, and reliable messaging such as MSMQ and AMQP.

In previous versions of Neuron ESB, there were few administrative features (Create/Purge/Check queues) that provided visibility or allowed users to manage the underlying Queues Neuron ESB used to store messages intended for delivery to subscribing parties. Those features were also restricted to the local server, on which the Neuron ESB Explorer was installed.

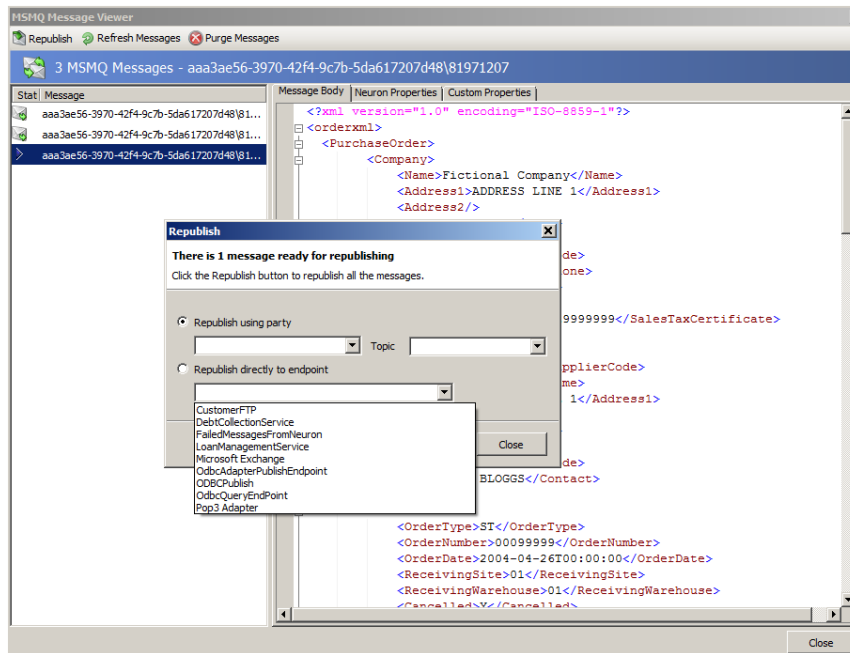
Neuron ESB 3.0 introduces an entirely new administrative and management experience for Neuron Topics configured to use durable, guaranteed, and reliable message Transports.



Neuron ESB 3.0 provides MSMQ and AMQP Management consoles within the Neuron ESB Explorer which can be used to execute the following functions, accessible through the toolbar or through context menus:

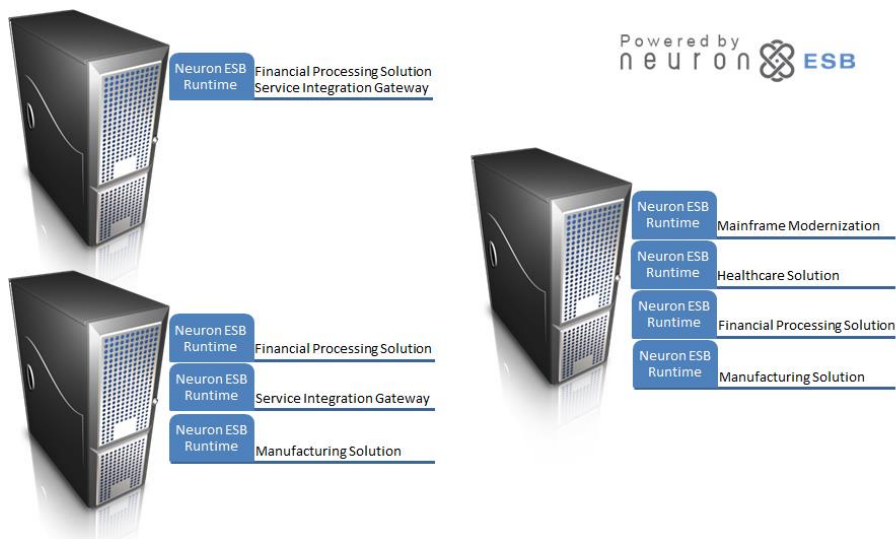
- ⊗ Create Queues
- ⊗ Check Queue Existence
- ⊗ Purge Queues
- ⊗ Delete Queues
- ⊗ Delete Messages
- ⊗ View Messages
- ⊗ Edit/Resubmit Messages
- ⊗ Message Counts

These new features (except for Create and Delete Queues) can be used to manage Queues for either local or remote Neuron ESB servers. Editing pending queued messages and resubmitting them for redelivery is also available from within the Neuron ESB Explorer's Queue Management console.



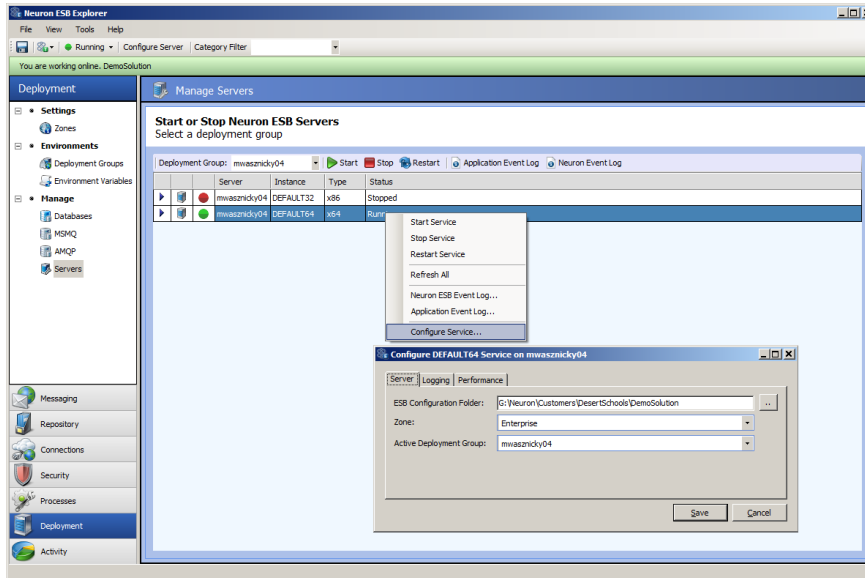
Server and Instance Management

Neuron ESB offers powerful runtime and scalability features that allow businesses to quickly deploy solutions. Solutions can scale both vertically and/or horizontally using the Neuron ESB Runtime Service across one or more servers. The Neuron ESB Runtime Service loads a Neuron ESB Configuration store, which can contain one or more business solutions. Neuron ESB 2.6 as well as 3.0 allows users to install multiple instances of the Neuron ESB runtime service on a single server. Each instance can be configured as either a 32 or 64-bit process, capable of running side by side. Each instance of the runtime can load an independent Neuron ESB Configuration store. This allows organizations to easily partition business solutions to run on a single server and scale across multiple servers.



In previous versions of Neuron ESB, there was no facility to centrally manage and configure all the Neuron ESB runtime service instances installed across the servers in a specific environment. Users could only see that state of the default Instance on the servers and start/stop them.

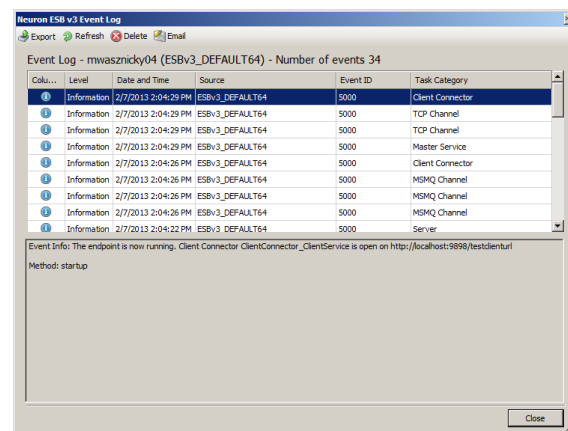
These functions though were synchronous, blocking in nature, and required administrative privileges on all the servers involved. Neuron ESB 3.0 now provides ability to centrally manage and configure all Neuron ESB runtime service instances on all servers within the environment, using the new Server Management Console within the Neuron ESB Explorer.



In Neuron ESB 3.0, all the server management functions have been refactored to be asynchronous in nature, allowing operations against multiple instances simultaneously. Each Neuron ESB runtime service instance, regardless of server location, can be configured, as well as started or stopped, through the Neuron ESB Discovery service. Users can select multiple instances and select

to start or stop them all at once.

The Event Log viewers within the server management console have also been refactored to provide real-time updates as events occur, the ability to export and email the event log by the source of the log (Neuron ESB runtime service name) and the ability to clear the log. Using the Event Log viewers, users can see detailed information, as well as the categories and source of that information.



Dependency Viewers

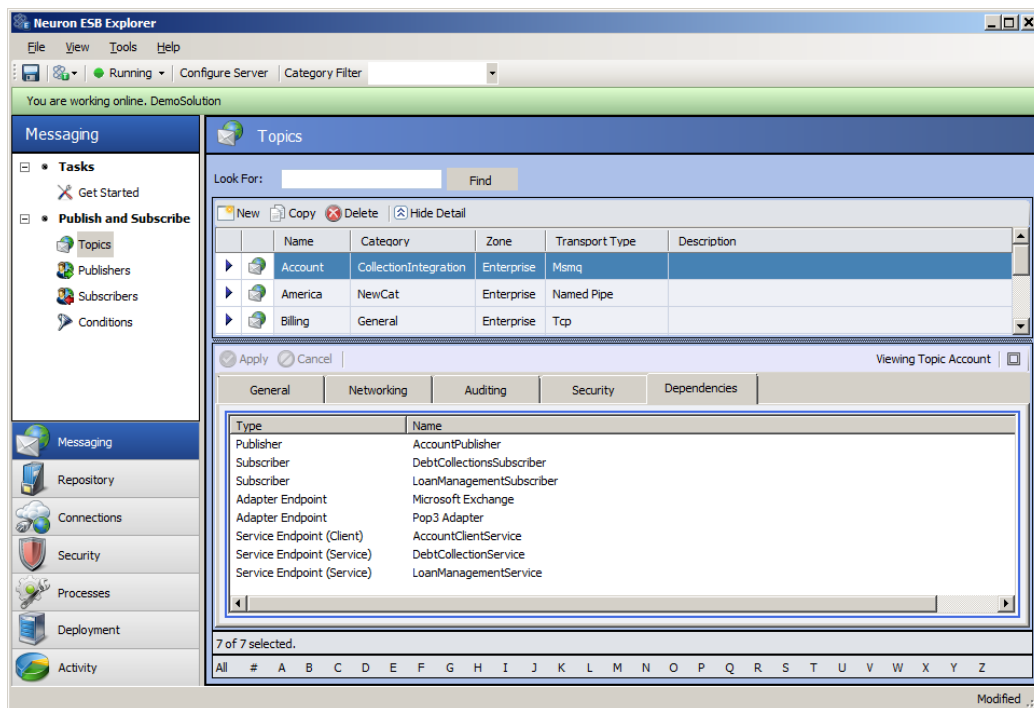
Complex business solutions will often be built with entities that have interdependent relationships with one another. Take for example some of the following use cases:

- ⊗ A Publisher may reference a Topic or Condition through its subscription.
- ⊗ A Publisher may also reference one or more Business Processes.

- ⊗ An adapter endpoint may reference a Publisher and publish messages to a Topic
- ⊗ An adapter endpoint, service endpoint or Business Process Step may reference a Neuron Environmental Variable
- ⊗ A service endpoint may reference a WSDL or SSL Certificate
- ⊗ A database may be referenced by one or more deployment groups

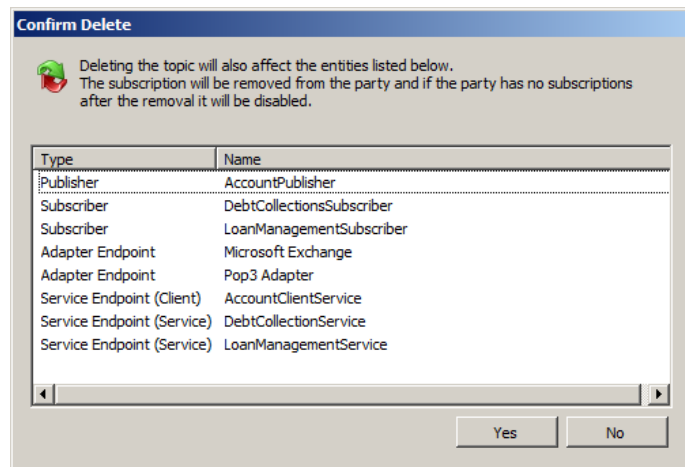
In all of these cases, the ability to quickly see and be alerted to respective dependencies becomes critical when building, modifying, or deploying ESB solutions.

Neuron ESB 3.0 introduces Dependency Viewers and alerts on all Neuron ESB entities managed within the Neuron ESB Explorer. The Dependency Viewers are visible within a dedicated Tab on an entity's configuration screen.



In the screen shot above, the Dependencies tab of the Account Topic is selected, listing all the Neuron ESB Entities by the type and name that reference the Topic. This same Dependencies tab can be found on the following Neuron ESB entity configuration screens:

- ⊗ Topics
- ⊗ Publishers
- ⊗ Subscribers
- ⊗ Conditions
- ⊗ Databases
- ⊗ Environmental Variables
- ⊗ WSDL Documents
- ⊗ Adapter Registrations



- ⊗ Service Bindings
- ⊗ Service Behaviors
- ⊗ Credentials
- ⊗ Access Control Lists
- ⊗ Encryption Keys

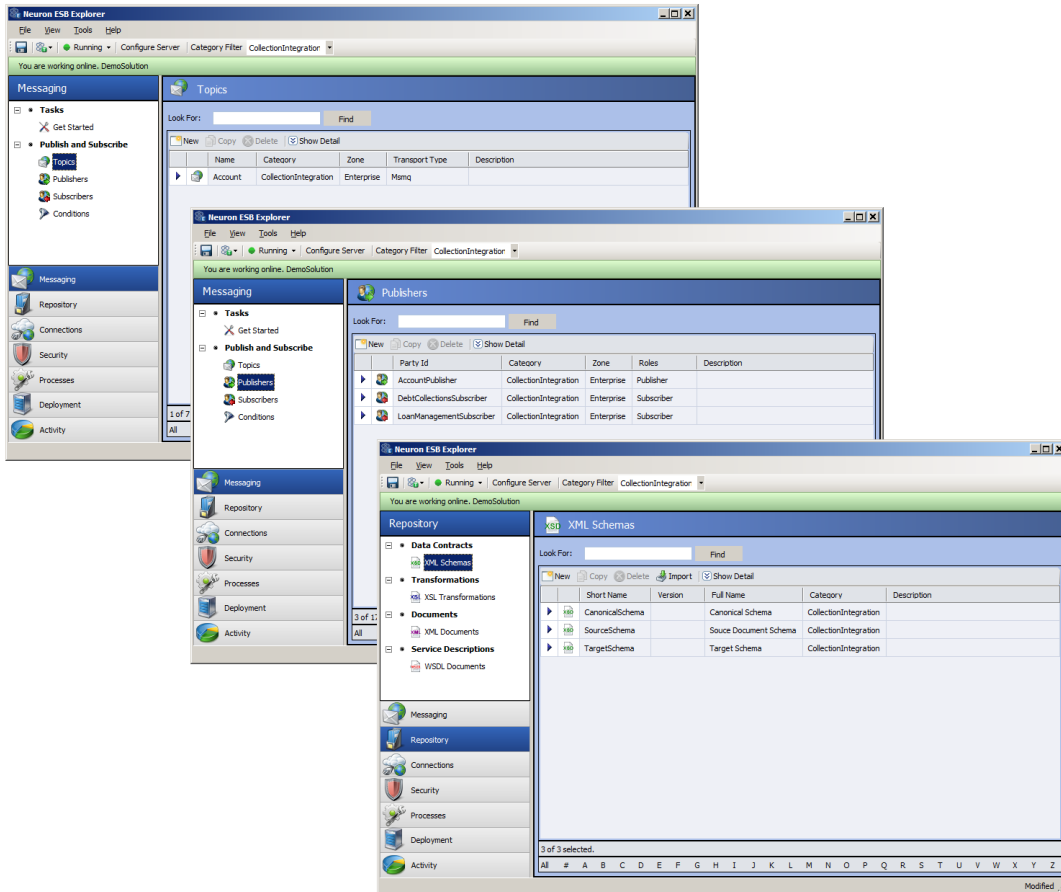
Dependencies can also be checked on specific Business Processes, by clicking on the dependency button located above the Process listing within the Business Process designer.

By tracking dependencies, Neuron ESB 3.0 allows the user to see the impact of their changes and current configuration. When a user attempts to delete an entity that has dependencies, they are prompted with the list of dependencies so that they can make informed decisions.

Global Category Filters

Within Neuron ESB, every entity (i.e. Topic, Publisher, Subscriber, Adapter, Adapter Endpoint, etc.) can be configured with a Category. Categories are a way to visibly group related entities within a solution. Adding a Category is as easy as typing in a new name in the category field and clicking “Apply”, followed by “Save”. However, filtering entities by Category used to only be available at the entity level. In other words, if you created a new Category called “Finance” for a topic and then filtered on that, only those topics with the Category “Finance” would be visible within the Neuron ESB Explorer. However, if a user navigated to another entity type, like Publisher, the Category name would not be available to select; nor would the previous filter applied against the Topics be active against the list of Publishers.

In Neuron ESB 3.0, Categories are global in nature with the Category filter drop down now located on the main Neuron ESB Explorer Toolbar. If users enter a new Category within any entity, it will immediately be available to select as an available Category in any other entity type within the Neuron ESB Explorer. If a user selects a Category on the toolbar filter, then only those entities within the Neuron ESB Explorer that are associated with that Category will be visible. This provides users a way to group entities in application related clusters.



Deployment and Configuration Management

Neuron ESB 3.0 introduces an entirely new configuration storage format that provides incremental, command line, and simple XCopy deployment of selected entities. In previous versions of Neuron ESB, the entire ESB configuration was encapsulated in one *.ESB file. Although the *.ESB file based configuration file was convenient for deploying an entire ESB Solution from one environment to another, it had two significant shortcomings including:

- ⊗ Incremental deployment of an individual entity or group of entities (i.e. Topic, Party or Endpoint, etc.) to other environments was not possible.
- ⊗ Multiple users could not work on the same ESB configuration file (*.ESB) at the same time.

The older *.ESB configuration file format prevented users from migrating individual entities to new ESB Solutions.

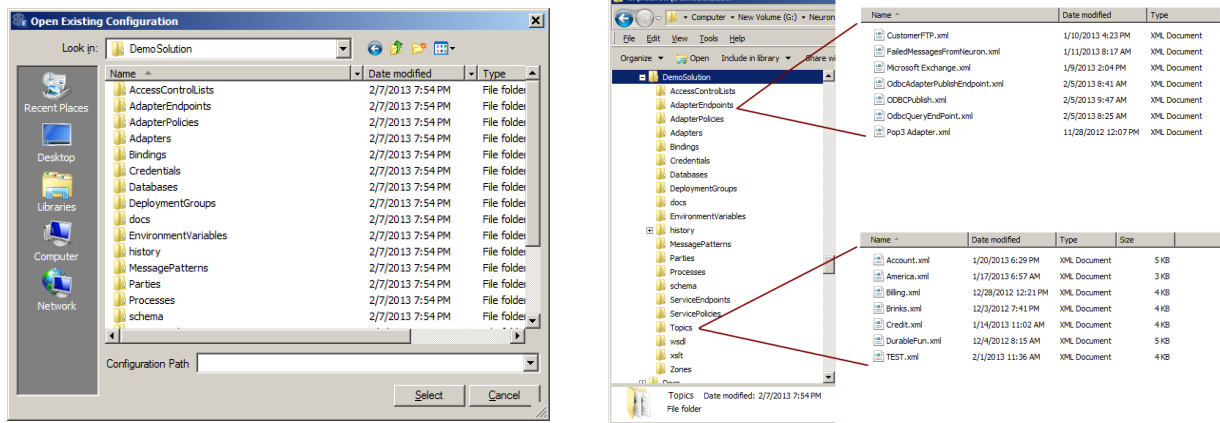
Neuron ESB 3.0's new configuration storage format overcomes these limitations and provides features for automated and command line deployments. For example, in Neuron ESB 3.0, users can select one or

more entities (and their dependencies) from one ESB solution, export them out as a package, and import them into another ESB Solution.

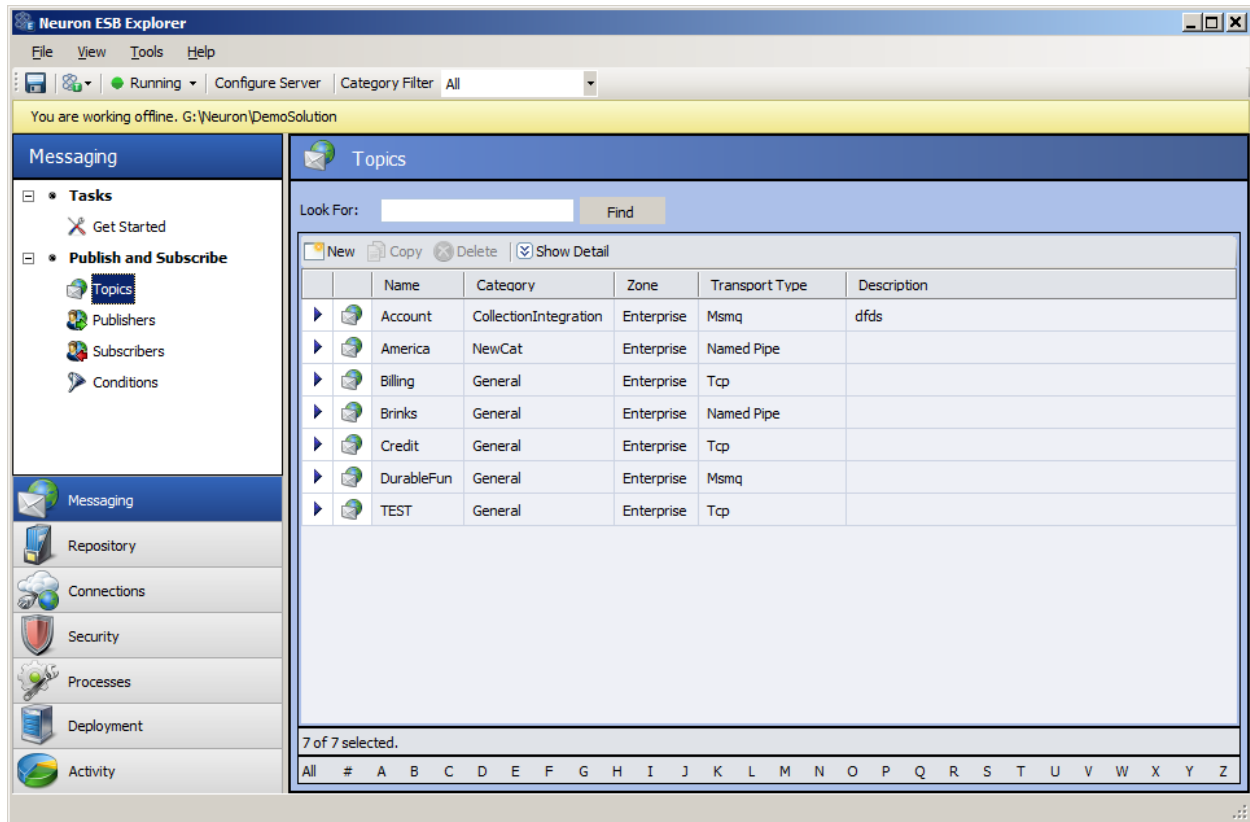
New Neuron ESB Configuration storage

The Neuron ESB 3.0 configuration storage format has evolved from the single file format (i.e. *.ESB file available in Neuron ESB 2.x) to a directory structure consisting of folders representing each entity type. Within each folder, an XML formatted file represents a specific entity such as a Topic, Endpoint, Business Process, etc. The Neuron ESB Explorer is designed to work and store all modifications to the directory structure.

After launching the Neuron ESB Explorer, users can Connect, Open, or Create a new Neuron ESB 3.0 configuration. When Open is selected, users are prompted to select the root folder of an existing Neuron ESB 3.0 configuration directory as depicted below:



Once the Neuron ESB 3.0 configuration is opened, each XML file within the entity folder is loaded and managed within the Neuron ESB Explorer. There is a one-to-one mapping of XML file to managed entity. The figure below displays the Topics represented by XML files within the Topics sub folder of the DemoSolution directory (pictured above).



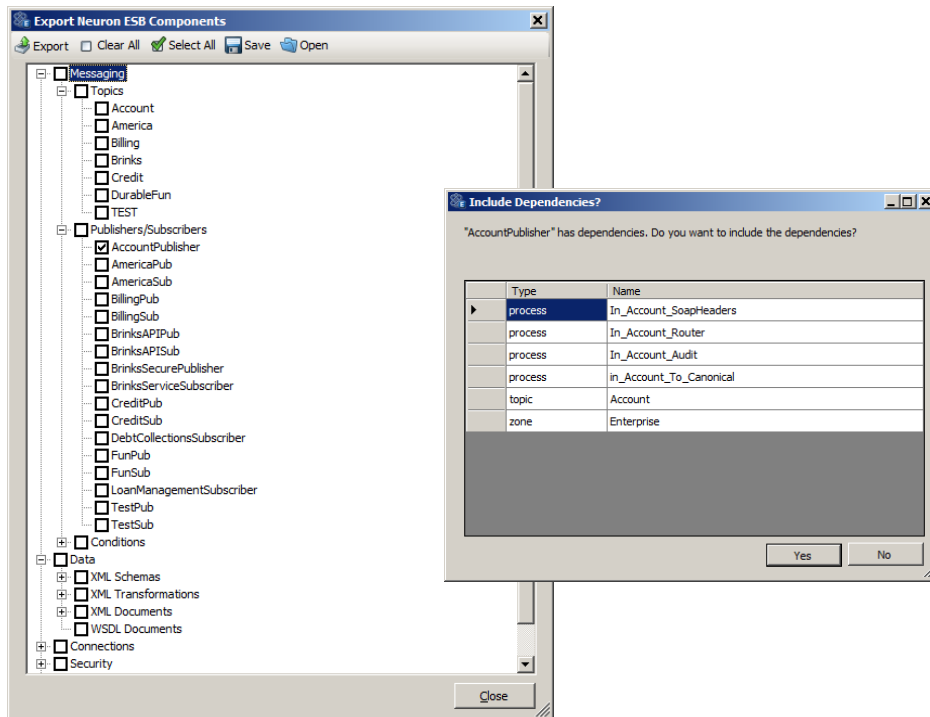
Multi Developer support

The Neuron ESB 3.0 configuration storage format allows multiple Developers to work on individual elements of an ESB Solution. Developers can use source control systems to check in and check out individual entities in a solution. Developers modifying one entity will have no impact on the other entities in a solution. Developers can then merge their changes into a central configuration easily, by checking in their updated entities.

Incremental Deployment

Although Neuron ESB 3.0 still supports deploying the entire solution from one environment to another, users can now deploy only the changes or new entities they want, rather than the entire solution. There are generally three (3) approaches for incremental deployment; 1.) Using the new export/import utility of the Neuron ESB Explorer; 2.) Directly copying the new or modified entities to their respective configuration sub folders on the target server; 3.) Using the new command line interface to automate the deployment of selected entities exported, using the export/import utility.

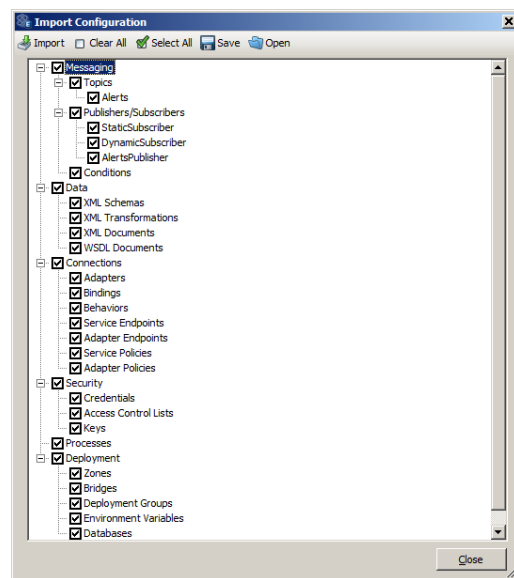
The new Export/Import utility can be accessed from the File menu of the Neuron ESB Explorer. When “Export” is selected, the Export dialog box appears and allows users to select the individual entities to be exported (or all of them). If an entity is selected that has dependencies, the user is prompted with those dependencies and given the option of including them in the export. Once the selection is complete, all the selected entities are exported out to a new *.esb package file.



When “Import” is selected, the user is prompted to select an existing *.esb configuration file. Either a new Neuron 3.0 *.esb package file or an older Neuron ESB 2.x *.ESB configuration file (if migrating a Neuron ESB 2.6 solution to Neuron ESB 3.0) can be selected. The contents of the *.esb file will be displayed in the Import dialog box.

After the import process, the configuration should be saved within the Neuron ESB Explorer by selecting “Save” on the File menu.

If a new configuration was created, the user can save the newly imported entities into a new configuration storage folder.



When either exporting or importing, the entities selected can be saved in an external response file by selecting “Save” on the toolbar. This will save a list of what was selected, not the actual entities. Later, when opening or exporting, the response file can be opened by selecting “Open” on the toolbar. This will effectively automatically reselect the entities represented in the response file. This same response file can be used to automate the export and import of entities using the Neuron ESB command line tool.

Command line Deployment

Neuron ESB 3.0 provides two new command line tools, to automate the import and export of entities from one ESB configuration to another. The command line tools are located in the default instance directory of the Neuron ESB installation folder, and are named ExportConfig.exe and ImportConfig.exe.

To make it easier to re-export the same objects or groups of objects, the ExportConfig.exe program recognizes the presence of response files on the command line. A response file is a simple text file that contains one option per line and can be generated from the Import/Export dialogs. For example, a response file that exports a Topic, Publisher, and Subscriber would look like this:

```
--topic=Topic1
--party=Publisher1
--party=Subscriber1
```

To use the response file on the command line, prefix the path to the response file with an @ symbol. For example:

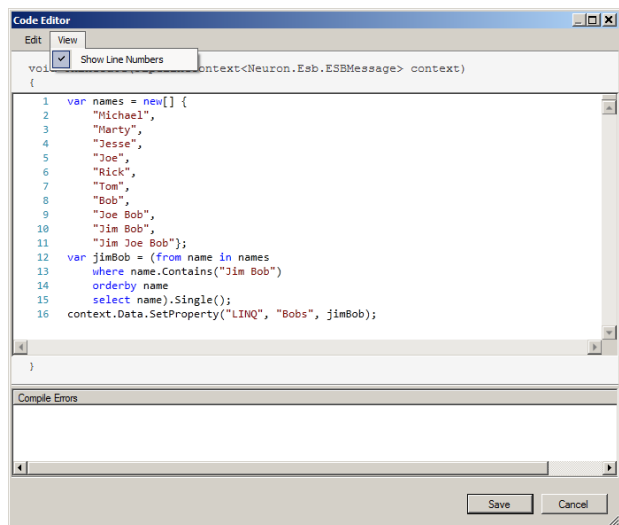
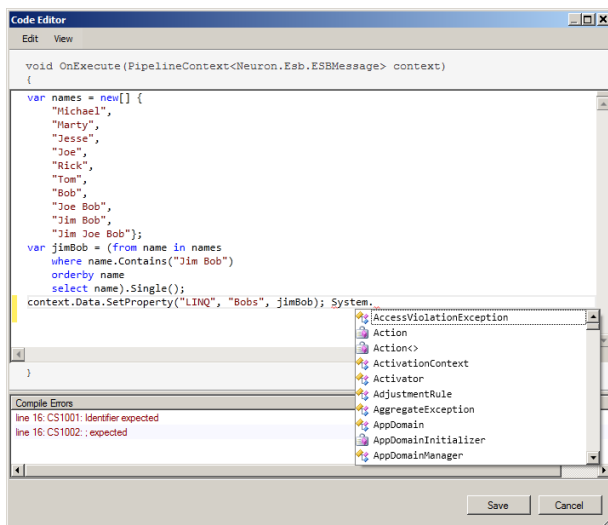
```
ExportConfig @TopicsAndParties.rsp C:\MyConfiguration C:\TopicsAndParties.esb
```

Microsoft .NET 4/LINQ

Neuron ESB 3.0 has been updated to be compatible with the Microsoft .NET 4.0 framework. This allows Neuron, as well as Developers, to take advantage of core features offered in the .NET 4.x framework environment. Developers can now easily interact with projects written in .NET 4, using the Neuron ESB Client API.

Developers can develop custom Business Process Steps as well as custom adapters using the Microsoft .NET 4 framework, instead of .NET 3.5. However, previously built Neuron ESB 2.x custom Business Process Steps and custom adapters will still work with Neuron ESB 3.0, without modifications.

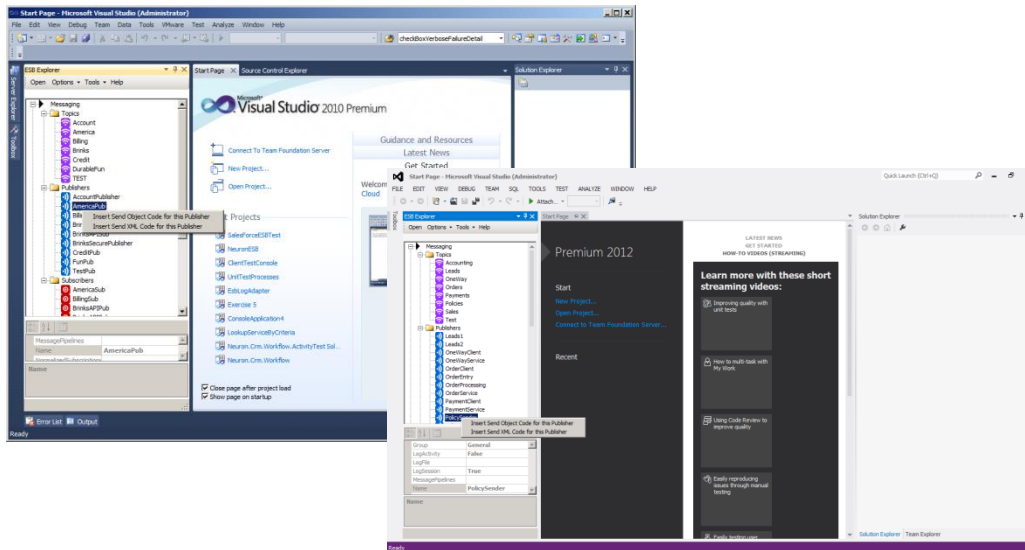
Upgrading Neuron ESB 3.0 to the Microsoft .NET 4 framework resulted in the upgrade of our Business Process Code Step editor so that it now supports the latest .NET language enhancements, including LINQ automated IntelliPrompt and lambdas.



Developers can use the Code Editor in any custom Business Process to write Microsoft .NET C# code, as part of their Business Process. The expandable Code Editor supports adding references to external assemblies (this can also be done at the Business Process level in Neuron ESB 3.0), as well as real-time syntax error reporting and full .NET 4 Intellisense. Line numbers can be toggled on and off within the designer and outputted with compile time errors.

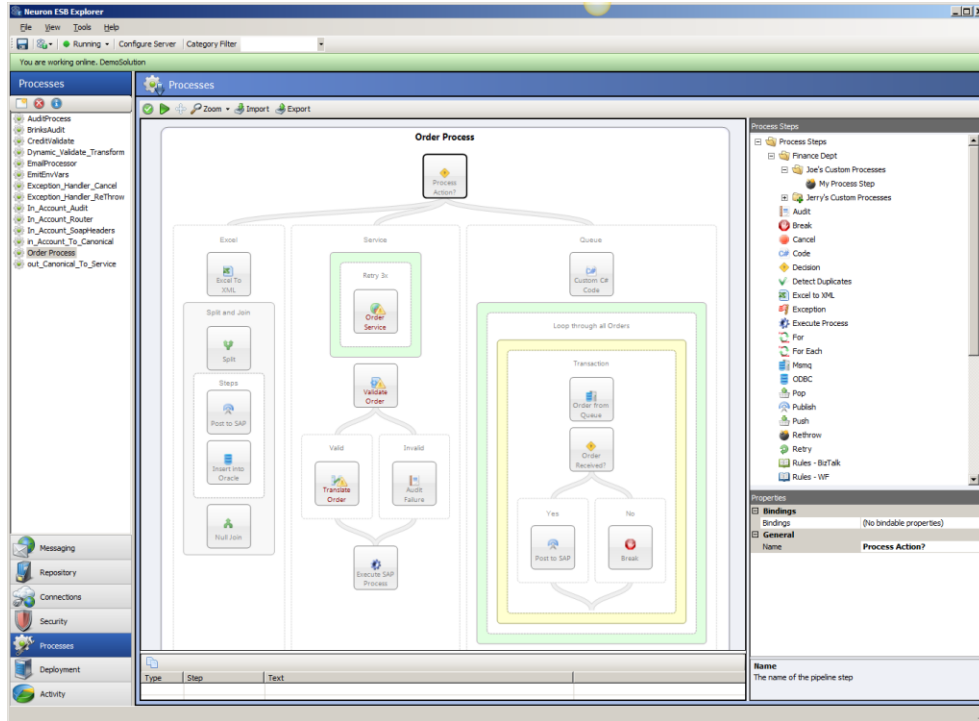
Visual Studio 2010 and 2012

Neuron ESB 3.0 also ships an ESB Explorer plugin for both Microsoft Visual Studio 2010 and Microsoft Visual Studio 2012. These plugins allow Developers to work directly with the entities within a Neuron ESB configuration. They also are used to auto-generate code snippets for the Neuron ESB Client API.



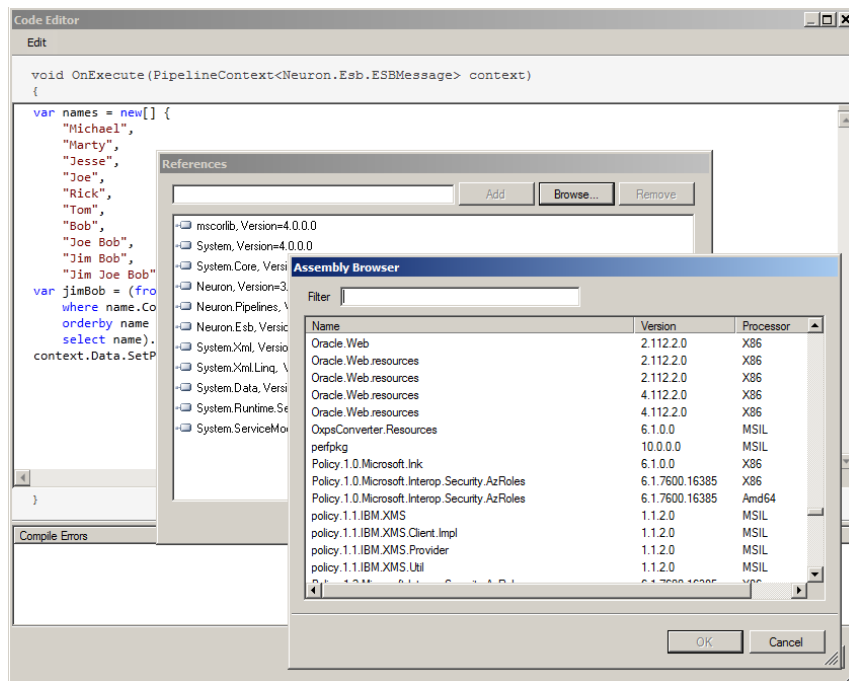
Business Process Designer

The Business Process Designer and Runtime experience continues to evolve in the Neuron ESB 3.0 platform, enabling customers to build robust Business Processing scenarios. More work has been done in this release to increase the usability and development experience, as well as provide more capabilities out of the box. Neuron ESB offers a flexible, easy to use, Business Process Designer with a drag and drop interface. It ships with 34 configurable Process Steps; that do everything from calling a service and updating a database or queue, to parsing an Excel file (as depicted below within the Neuron ESB Explorer). Developers can build custom reusable Process Steps that can be registered in the Process Steps library and added to any custom Business Process.



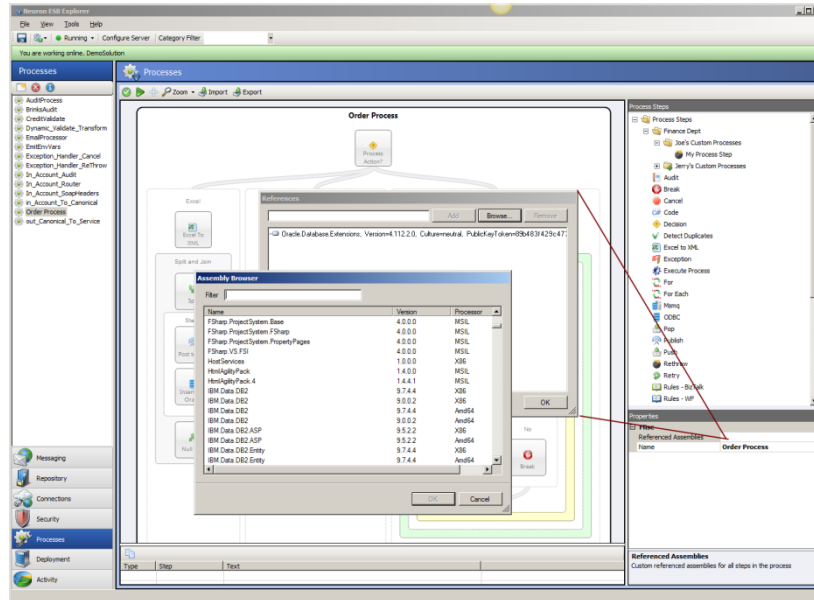
Referencing External Assemblies

In previous versions of Neuron ESB, Developers could create custom .NET projects in Microsoft Visual Studio, compile these into assemblies (*.DLL), and add references to those assemblies (or existing ones) within the Process Code Step editor. This is a common practice when developing or using an existing set of common framework libraries to provide specific capabilities to a Neuron ESB Business Process.



However, if a Business Process used multiple Code Process Steps, it would require that the reference be added to each Code Process Step within the Business Process. For example, if there were 20 Code Steps, the external assemblies would have to be added as references 20 times, provided the external library needed to be referenced in each Step.

In Neuron ESB 3.0, references to external assemblies can now be added at the Business Process level, rather than just the Code Step level. If an assembly is referenced at the Process level, it will automatically be added to each Code Step in the Business Process.

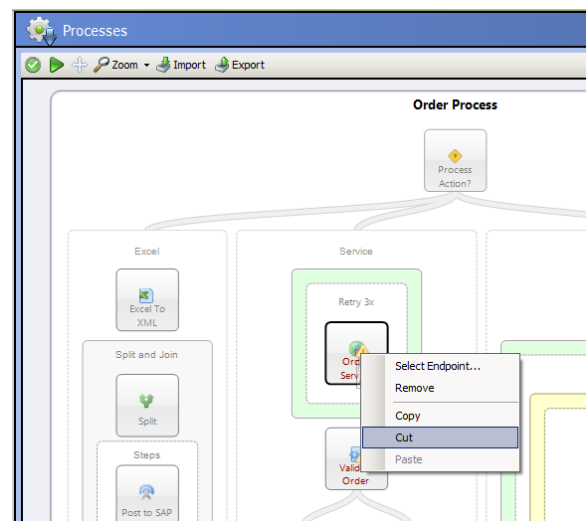


Zoom, Cut, Copy and Paste

Neuron ESB 3.0 introduces a number of usability enhancements to increase the productivity of users developing custom Business Processes within the Neuron ESB Explorer.

By holding down the Control key (Ctrl); users can use their mouse wheel to dynamically zoom in and out of any Business Process, well beyond the static zoom ratios offered on the toolbar.

Additionally, users can now copy or cut any Process Step from one Business Process and paste it in another location or within another Business Process. This eliminates the need to manually duplicate Process Step configuration. Cut, Copy, and Paste can be accessed by simply right clicking on any Process Step to display its context menu.

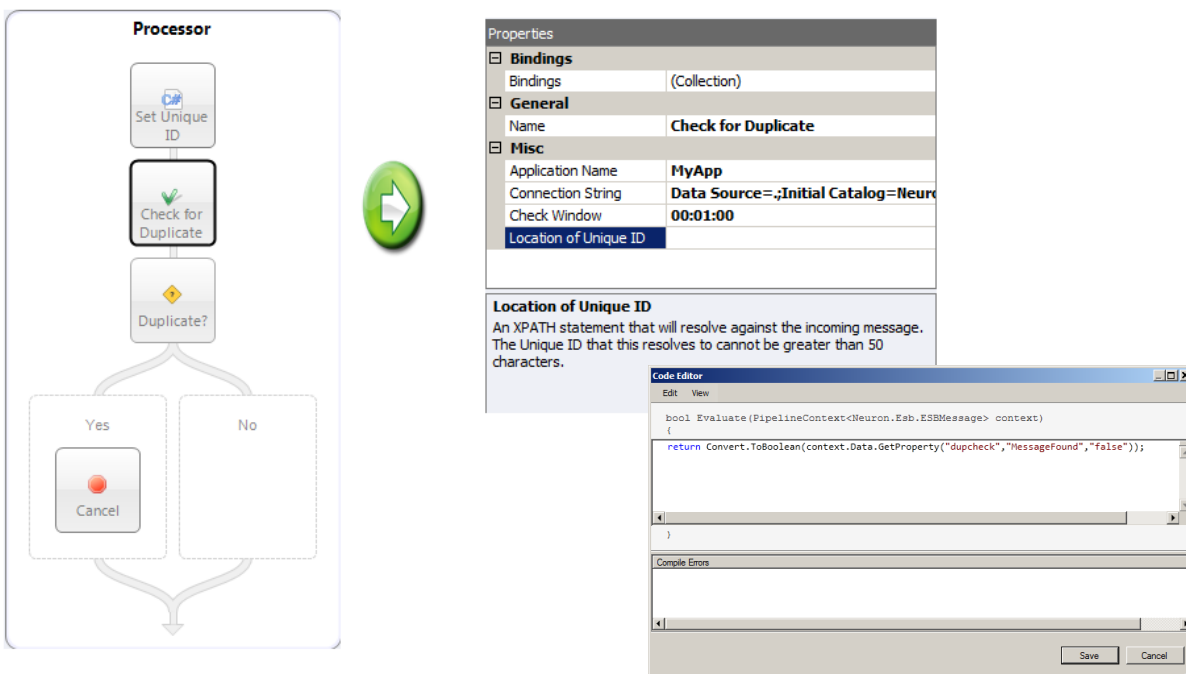


New Process Steps

Neuron ESB 3.0 continues its focus on productivity by introducing several new Process Steps to make building more complex processes faster and easier. By delivering capability in the form of Process Steps that can be graphically composed to build larger more complex Business Processes, Neuron ESB 3.0 significantly reduces the amount of custom code, as well as the time it takes to build and manage Business Processes.

Duplicate Message Detection

Many 3rd party applications and transports (i.e. FTP, FILE, Azure Service Bus, etc.) do not support guaranteed “once only” delivery of messages. However, users may need to integrate with these technologies as publication sources to Neuron ESB. In those cases where consistency must be ensured and the chance of duplicate message publication must be eliminated, Neuron ESB 3.0 provides the Detect Duplicates Process Step. This Process Step can be configured to use the current Neuron Audit database. Users can configure the criteria (either at design time or runtime) that define what makes a message unique. Users can also configure the time window in which a duplicate message is to be prevented from being received.



The diagram illustrates the configuration of the Detect Duplicates Process Step. On the left, a flowchart shows a Processor containing three steps: 'Set Unique ID', 'Check for Duplicate', and 'Duplicate?'. The 'Check for Duplicate' step is highlighted with a green checkmark. Below the 'Duplicate?' step, there are two paths: 'Yes' (with a 'Cancel' button) and 'No'. A green arrow points from the flowchart to the configuration panels on the right.

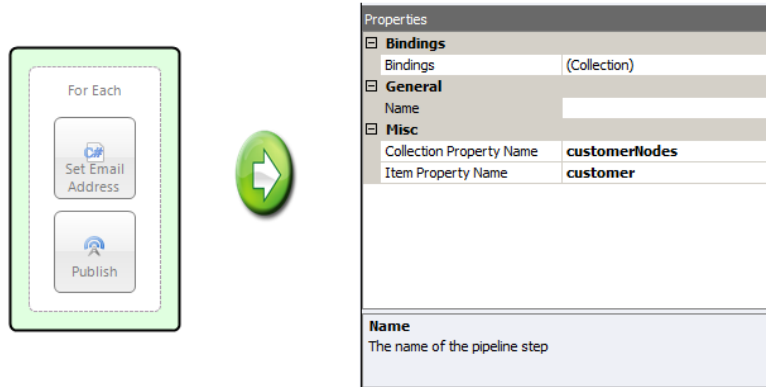
The configuration panels include:

- Properties**
 - Bindings**: Bindings (Collection)
 - General**: Name (Check for Duplicate)
 - Misc**: Application Name (MyApp), Connection String (Data Source=.;Initial Catalog=Neuron), Check Window (00:01:00), Location of Unique ID
- Location of Unique ID**: An XPath statement that will resolve against the incoming message. The Unique ID that this resolves to cannot be greater than 50 characters.
- Code Editor**:

```
bool Evaluate (PipelineContext<Neuron.Esb.ESBMessage> context)
{
    return Convert.ToBoolean(context.Data.GetProperty("dupcheck", "MessageFound", "false"));
}
}
```

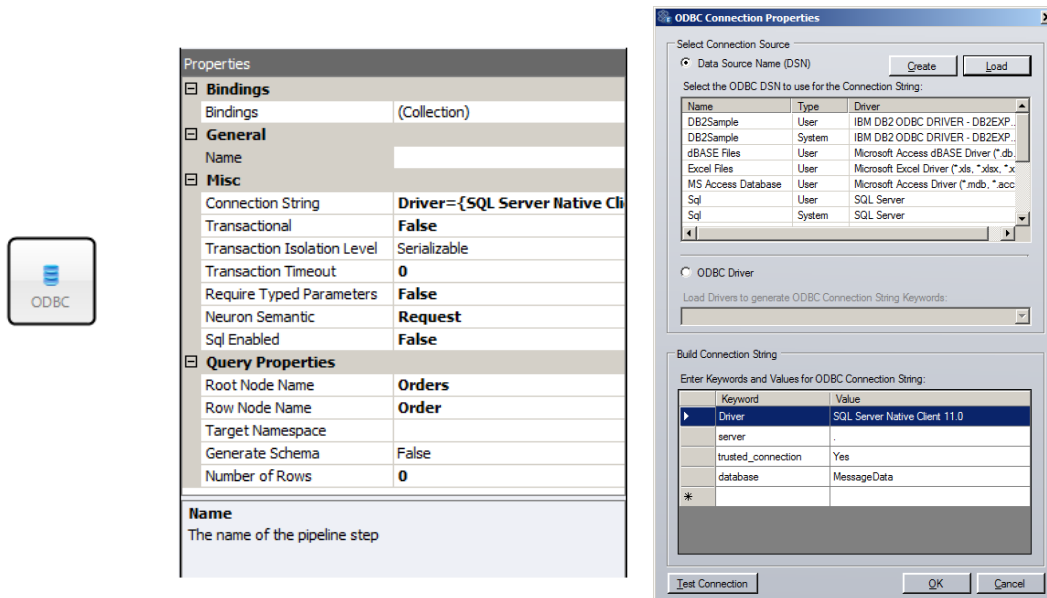
For Each loop

A new Process Step that allows a For Each type looping construct to be configured within a Process has been provided. The existing Break Process Step can be used within the For Loop. Neuron ESB 3.0 ships with a sample that demonstrates how to use the new Process Step. This has advantages over the For Loop Process Step, as any .NET collection that supports enumeration can be used with fewer configurations.



ODBC

The new ODBC Process Step provides powerful capabilities to users building Business Processes that need to integrate directly with ODBC based data sources (i.e. Excel, SQL Server, Oracle, DB2, MySQL, and Lotus Notes, among others). The ODBC Process Step supports one way inserts and updates, as well as query (i.e. Request/Response) types of message patterns. Optionally, the ODBC Process Step can also generate XML Schemas that represent the result sets returned. This Step also leverages the new ODBC connection string builder that the Neuron ESB 3.0 ODBC Adapter uses. Neuron ESB 3.0 ships with a comprehensive sample that demonstrates how to use the new Process Step.



Custom Process Steps

Neuron ESB 3.0 ships with 34 Process Steps that provide a range of functionality and productivity when building complex Business Processes. However, it also provides the ability for users to build their own “reusable” custom Process Steps that can be registered and added to the existing library of Process

Steps. This allows other users to build out custom Business Processes that incorporate those reusable custom Process Steps.

Although the capability to create reusable custom Process Steps was introduced in Neuron ESB version 2.5.13, there was only modest control of the user interface (UI) provided.

Interface for Controlling UI Properties

In Neuron ESB 3.0, the ability to have a more fine-grain control over the UI properties exposed by custom Process Steps has been introduced. By referencing the new `StepUIElement` class, users can control what icon will be displayed when a user drags the custom Process Step onto the Process Designer. In regards to exposing custom properties to be configured at design time, users can now control the order in which the properties will be displayed, as well as dynamically make properties visible/invisible depending on what the user selects within the property grid. Neuron ESB 3.0 ships with a sample (“Complex Custom Process Step”) that demonstrates how to use this new class when building custom Process Steps.

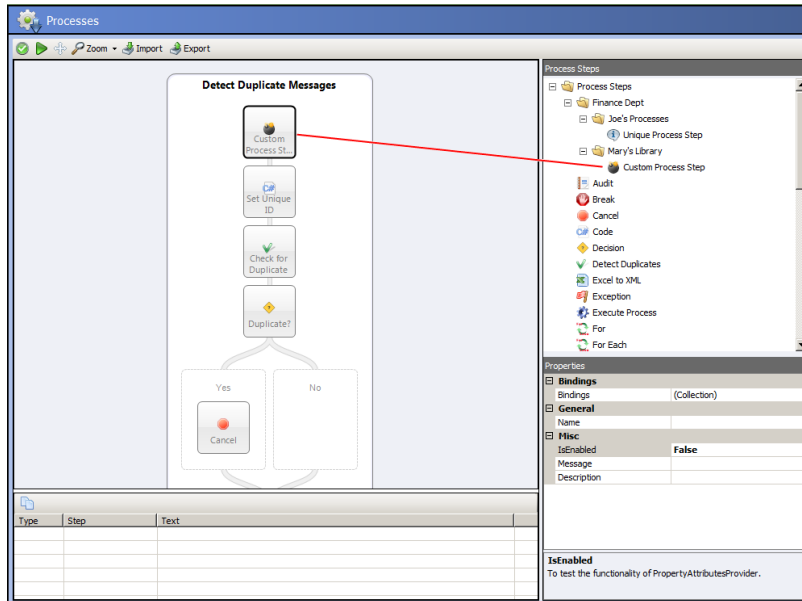
Folder hierarchy for UI display

In previous versions of Neuron ESB, registered custom Process Steps would simply be displayed in-line within the Process Steps Library, located in the Process Designer. There was no visual distinction between custom Process Steps and the 34 Process Steps that ship with Neuron ESB.

In Neuron ESB 3.0, users can now create a folder hierarchy during registration of their custom Process Steps, which will be reflected in the Process Steps library. Users can specify an optional “path” attribute within the “neuronpipelines.config” file as shown below:

```
<neuron.pipelines>
  <StepTypes>
    <add name="UniqueProcessStep" path="Finance Dept\Joe's Processes"
      value="UniqueProcessStep, Version=1.0.0.0, Culture=neutral,
      PublicKeyToken=8924d098b3b48761"/>
    <add name="CustomProcessStep" path="Finance Dept/Mary's Library"
      value="CustomProcessStep, Version=1.0.0.0, Culture=neutral,
      PublicKeyToken=8924d098b3b48761"/>
  </StepTypes>
</neuron.pipelines>
```

When this is done, the commensurate folder structure will be displayed within the Process Designer’s Process Step library, allowing users to easily organize their custom Process Steps.



Neuron Auditing

Neuron ESB provides asynchronous and synchronous message tracking (i.e. Auditing) through its Auditing Service. Auditing can be managed either at the Topic level (Auditing tab of the Topic's properties), or at the Business Process level, through the use of the Audit Process Step. The same Auditing Service is used to provide failed message support. When a message fails during processing or delivery, the message can be persisted with all of its context and exception information, so that users may later view/edit/resubmit the failed message for additional processing.

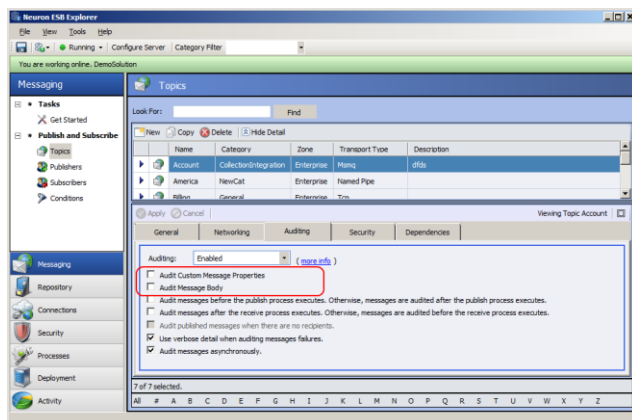
Microsoft SQL Azure

Neuron ESB supports Microsoft SQL Server versions 2005 and above, for the Auditing Service's message store. However, in Neuron ESB 3.0, support for Microsoft SQL Azure has been added. This provides users who host Neuron ESB in an Azure worker or VM role, the ability to configure support for the Neuron ESB Audit Service.

Excluding Body and Custom Properties

The disk space consumed by message tracking is directly related to the size of the message being audited, as well as the number and context of custom properties that the message may have. In Neuron

ESB 3.0, users now have the ability to exclude the persistence of either the message body



| Properties | |
|-----------------|---|
| Bindings | Bindings (Collection) |
| General | Name |
| Misc | Action Receive |
| | Audit Message Body True |
| | Audit Custom Properties True |
| | Failure Type |
| | Failure Detail |
| | XPath |
| Action | The value used for the action column for the audited message. |

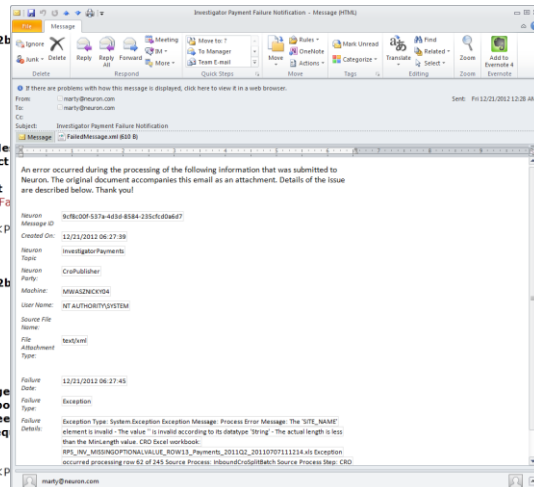
and/or the custom properties that have been added to the message at runtime. This can be configured either at the Topic level or through the use of the Audit Process Step.

Failed Message Monitoring

Neuron ESB offers auditing in the case of failed message delivery or processing. When a failure occurs, the message, its context, and exception information is persisted to the Neuron Auditing Service database. In previous versions of Neuron, users could either run the Neuron Failed Message report within the Neuron ESB Explorer to view all failures, or they could subscribe to the WMI Failed Message event and be alerted the moment a failure occurred. However, in many cases users would also want to have the ability to automate the retrieval, post processing, and routing of the failed message.

Neuron ESB 3.0 provides for this by adding a feature to the Neuron Audit Service database. Users can now configure either an ODBC or SQL adapter endpoint to poll the Neuron Audit database and call the **GetFailedMessagesForProcessing** stored procedure. This stored procedure will return a batch of failed messages, including some of the meta-data. A Neuron ESB Business Process can then asynchronously split the batch into individual messages, transform it (using XSL), and email it out to subscribing entities with the original message as an attachment. Once a batch of messages is retrieved, an associated table is updated with their respective message ids, so that the next poll does not return failed messages which have already been retrieved. Depicted below is a sample of the batch file format produced by the stored procedure:

```
<?xml version="1.0"?>
<NEURON_AUDIT_FAILURES>
  - <FAILURE created="2013-01-07T20:36:54.853" messageId="93296f6e-5c24-4172-bcd9-8a580e082b4e">
    <Topic>Neudesic</Topic>
    <Party>NewSubscriber</Party>
    <MessageId>93296f6e-5c24-4172-bcd9-8a580e082b4e</MessageId>
    <Machine>MWASZNICKY04</Machine>
    <User>CORP\marty.wasznicky</User>
    <FailureDateTime>2013-01-07T20:36:57.387</FailureDateTime>
    <FailureType>EndpointNotFoundException</FailureType>
    <FailureDetail>Exception Type: System.ServiceModel.EndpointNotFoundException Exception Message: http://localhost:8742/ that could accept the message. This is often caused by an incorrect if present, for more details. Exception Trace: Server stack trace: at System.ServiceModel.Channels.HttpOutput.WebRequestHttpOutput.GetOutputStream() at System.ServiceModel.Channels.HttpOutput.Send(TimeSpan timeout) at System.Service</FailureDetail>
  - <Document>
    <[[CDATA[<Statement type="Text" sql="select * from PhoneBook where Id = ?"> <Parameters> </Parameters> </Statement>]]>
  </Document>
  - <FAILURE created="2013-01-07T20:36:54.853" messageId="93296f6e-5c24-4172-bcd9-8a580e082b4e">
    <Topic>Neudesic</Topic>
    <Party>NewPublisher</Party>
    <MessageId>93296f6e-5c24-4172-bcd9-8a580e082b4e</MessageId>
    <Machine>MWASZNICKY04</Machine>
    <User>CORP\marty.wasznicky</User>
    <FailureDateTime>2013-01-07T20:36:57.473</FailureDateTime>
    <FailureType>ChannelSendException</FailureType>
    <FailureDetail>Exception Type: Neuron.Esb.Channels.ChannelSendException Exception Message: System.ServiceModel.EndpointNotFoundException Exception Message: There was no endpoint could accept the message. This is often caused by an incorrect address or SOAP action. See Exception Trace: Server stack trace: at System.ServiceModel.Channels.HttpOutput.WebReq</FailureDetail>
  - <Document>
    <[[CDATA[<Statement type="Text" sql="select * from PhoneBook where Id = ?"> <Parameters> </Parameters> </Statement>]]>
  </Document>
</FAILURE>
</NEURON_AUDIT_FAILURES>
```



Neuron ESB 3.0 ships with a sample (“Failed Message Routing”) that demonstrates how to set up and run this specific scenario. Displayed above is a sample email, produced by the “Failed Message Routing” sample. XSL is used within a Business Process to produce an HTML message body for the email.

Adapters and Connectivity

Neuron ESB provides many areas of developer productivity. Neuron exposes extremely simplified interfaces for adapter development as well as real time process development. Adapters function as bridges to external transports, protocols or existing applications that may be in use within an

organization. Neuron ESB provides many “out of the box” adapters to simplify connectivity through messaging and configuration. Additionally, several “event based” publication adapters are also included with Neuron ESB.

Neuron ESB 3.0 introduces several new adapters as well as enhancements to assist organizations connect, compose and expose new capabilities within their environment. Microsoft Exchange and POP3 adapters are now included to simplify email monitoring and the processing of attachments. A Microsoft Azure Service Bus adapter has been included so that organizations can reliably incorporate bridging across the cloud. Lastly, the Neuron ESB ODBC adapter has been enhanced with new capabilities such as supporting Oracle function.

Topics using Rabbit MQ/AMQP

Neuron ESB provides a hierarchical, Topic-based publish and subscribe model to mediate the routing of messages between Parties (Publishers and Subscribers), Adapters, and Service Endpoints. Neuron’s Topic model is composed of a sub topic hierarchy that can more intuitively reflect either an organization’s structure or business requirements. However, Neuron ESB is unique in the industry in that it allows the business to determine the Quality of Service (QoS) attributes at the Topic level, and that many Topics of various QoS attributes can exist side by side. By providing this level of flexibility, organizations do not have to worry about changing their specific business requirements to meet the limitations imposed by other competing products. Some of the critical QoS attributes include:

- ⊗ Throttling
- ⊗ Encryption
- ⊗ Auditing
- ⊗ Transport
- ⊗ Transactions
- ⊗ Durability
- ⊗ Compression

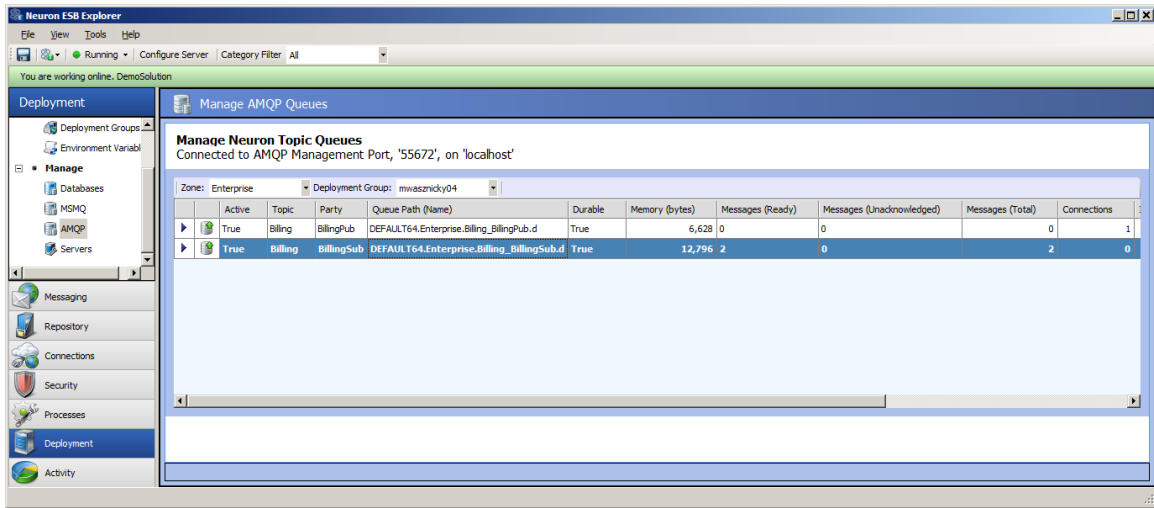
Transport is a critical QoS selection since it affects many aspects that businesses may require including transaction and durability support as well as ordered messaging, guaranteed delivery, once only delivery, scale out, latency and performance. In previous versions of Neuron ESB, several configurable Transports for Topics were supported including:

- ⊗ TCP
- ⊗ MSMQ
- ⊗ PEER
- ⊗ BIZTALK
- ⊗ NAMED PIPES

In previous versions of Neuron ESB, MSMQ played a critical role in providing durable, guaranteed, and reliable messaging for configured Topics.

Neuron ESB 3.0 introduces a sixth Transport; AMQP. AMQP (Advanced Message Queuing Protocol) is an open standard application layer protocol for message-oriented middleware. AMQP can now be used as a Transport property for Topics within the Neuron ESB Explorer. Neuron ESB 3.0's implementation of AMQP is based on Rabbit MQ and can be used (in some scenarios) as an alternative to MSMQ based Topics to provide durable, guaranteed, reliable messaging. When installing Neuron ESB 3.0 users can optionally install Rabbit MQ directly from the Neuron ESB 3.0 setup program.

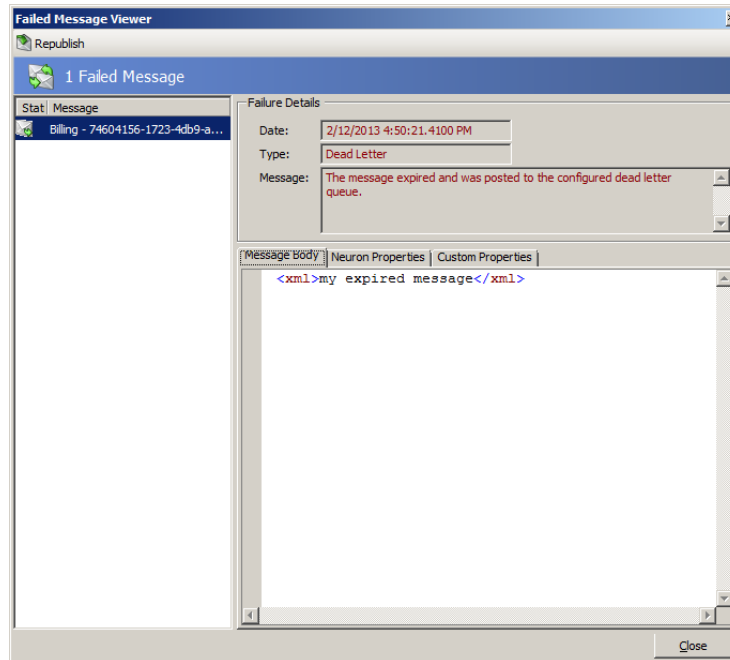
When using AMQP based topics, users can view and manage metrics regarding the underlying Queues that represent the Neuron ESB Publishers and Subscribers as depicted below.



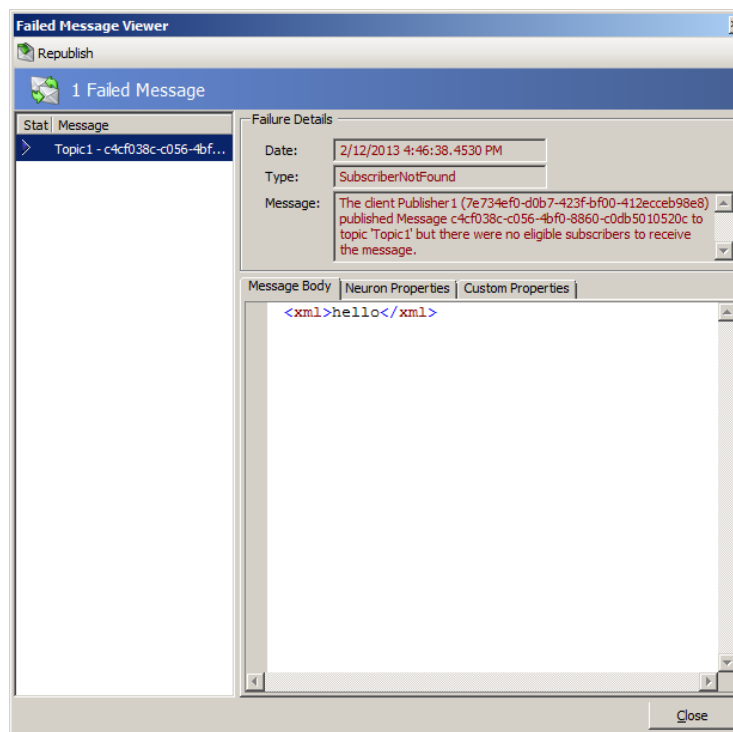
Topics using MSMQ

Neuron ESB 3.0 offers several new enhancements to the MSMQ Transport that can be configured for Topics including new Security, Logging and Dead Letter/Poison Queue handling. First and foremost, automated handling for Poison and Dead Letter Queues. For instance, when a message exceeds its expiration, (default 1 day), and cannot be delivered to its Subscribers (i.e. Subscriber is offline); it is automatically moved into a special queue called a Dead Letter Queue. In other cases where delivery of the message fails and a Neuron ESB Policy is NOT configured, the message is automatically moved to a Poison Queue.

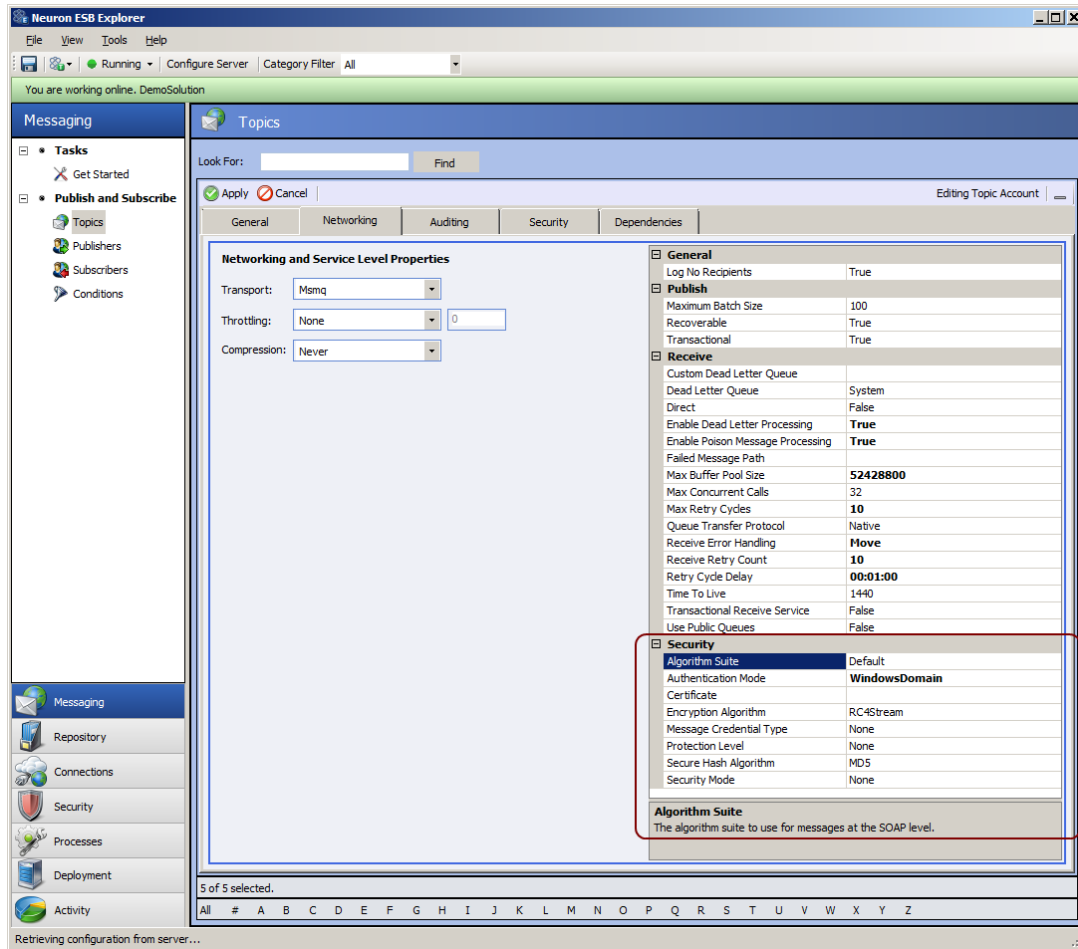
In both cases Neuron ESB MSMQ based Topics can now be configured to monitor and automatically move any messages in Dead Letter or Poison Queue to the Neuron ESB Failed Message Audit Database. This provides a centralized store for all failed messages within the system and allows users to be alerted to any failed message delivery. All failed messages can be viewed through the Failed Message Report within the Neuron ESB Explorer.



In previous versions of Neuron ESB, if a message was published to an MSMQ based Topic when no Subscribers for the message were configured, users had the option to log a warning to the Neuron ESB Event log but the actual message would be discarded. In Neuron ESB 3.0 this has been changed so that the original message is logged within the Neuron ESB Failed Message Audit Database.



Lastly, Neuron ESB 3.0 introduces entirely new security options that can be configured for MSMQ based Topics, including SSL Certificate support.



POP3 and Microsoft Exchange Adapters

Neuron ESB 3.0 introduces 2 new adapters that provide email monitoring and processing capabilities. The new Microsoft Exchange Server and POP3 adapters offer similar functions. Both allow users to monitor email folders on remote servers and publish those emails to Neuron ESB based Topics. Additionally, both are capable of managing and processing any associated attachments and making those attachments available to Neuron ESB for processing within a Business Process. Each one has advantages and limitations which are why both were included in Neuron ESB 3.0.

The Microsoft Exchange Server adapter allows users to connect to either a Microsoft Exchange Account's Inbox, or any folder or sub folder within the Account. For example, a user could specify a mailbox folder path such as "inbox/collateral/test". Some of its features are:

- ❖ Can only be used with Microsoft Exchange Servers
- ❖ Can be used with Microsoft Exchange "Online" (i.e. in the cloud)
- ❖ All communication is SOAP based over port 80
- ❖ Supports integrated Windows security

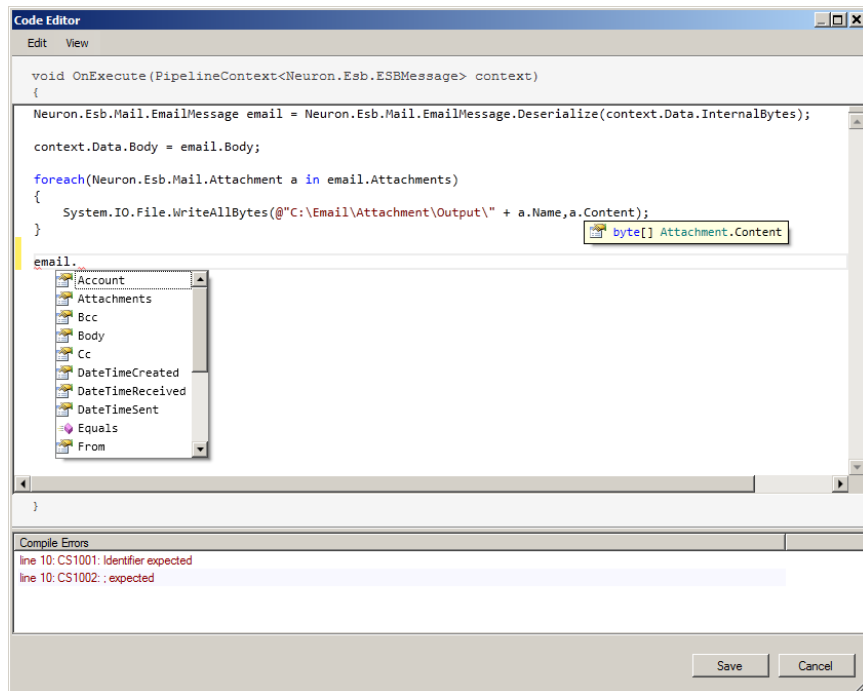
The POP3 adapter uses the standard POP3 protocol and can be used against any Mail server that supports POP3. Some of its features are:

- ⊗ Supports SSL
- ⊗ Can only be used against an account's inbox
- ⊗ Can be used with any Mail server that exposes POP3
- ⊗ Supports secure authentication

Both adapters support an identical email API that users can access within a Business Process Code Step to process all elements of an email (body, message headers, attachments, mime content, etc.). Regardless which adapter is used, the *Neuron.Esb.Mail* namespace is used to process email messages and their attachments. The *Neuron.Esb.Mail* namespace is new to Neuron ESB 3.0 and includes 4 major classes:

- ⊗ Attachment
- ⊗ EmailMessage
- ⊗ Address
- ⊗ Mime

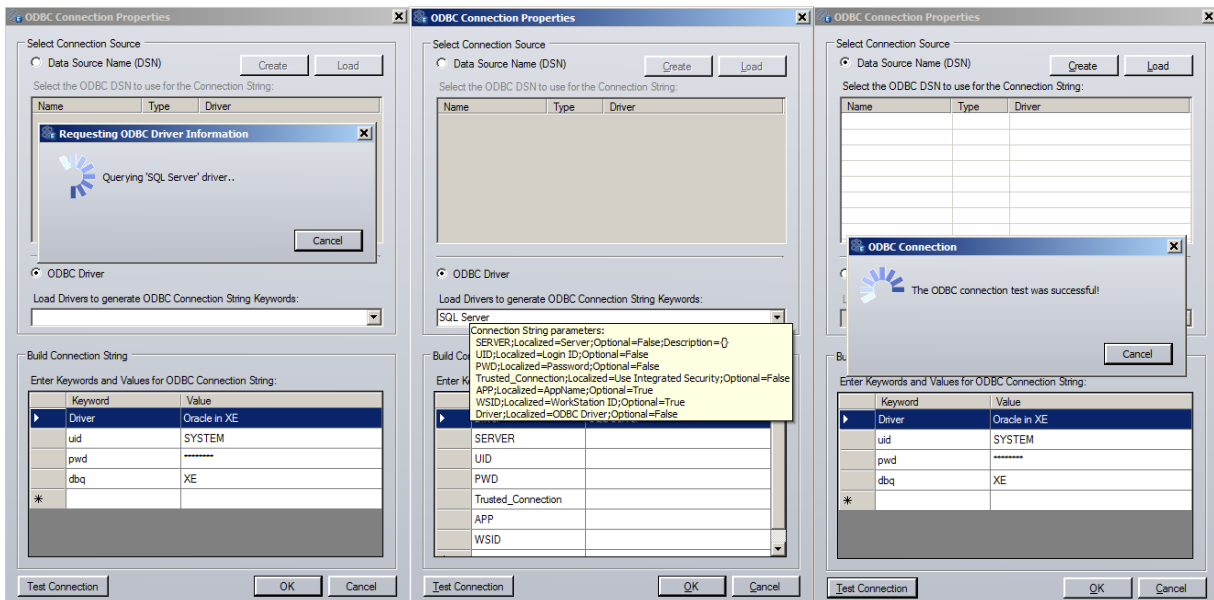
In the figure below, an email is processed and its associated attachments are written out to the file system. These same attachments can also be easily published to a Topic as independent messages. The *Neuron.Esb.Mail.Attachment* class exposes properties such as name, charset, content, length among others. The *Neuron.Esb.Mail.EmailMessage* class can be used to retrieve the collection of email addresses, subject, body as well as a number of other properties.



Neuron ESB 3.0 ships with a comprehensive sample that demonstrates how to use the new POP3 and Microsoft Exchange adapters.

ODBC Adapter enhancements

Neuron ESB 3.0 introduces several new enhancements and capabilities to increase the overall effectiveness and user experience of the ODBC adapter. In Neuron ESB 3.0 the ODBC Adapter's connection builder UI has been re-designed to provide asynchronous support to long running operations such as querying a driver's capabilities and testing connections. Also, once a driver is selected its keywords are queried and displayed for the user as a tooltip, providing guidance for the connection string construction.

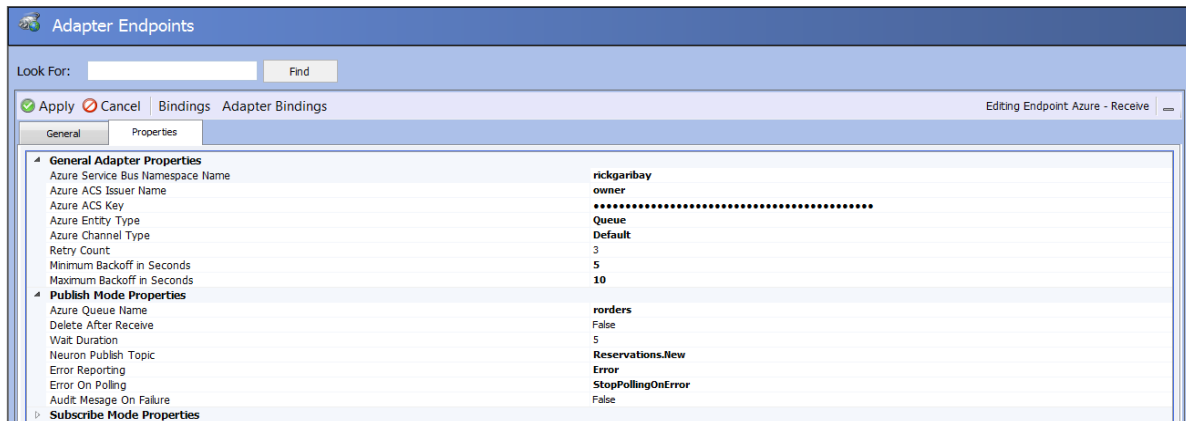


Other enhancements like support for binary parameters, output functions as well as Oracle function calls in Publish mode have also been added.

Azure Service Bus Adapter

Microsoft Azure Service Bus is a Platform as a Service (PaaS) capability provided by Microsoft that provides a messaging fabric hosted by Microsoft Windows Azure. The Azure Service Bus can be used to extend on-premise messaging fabrics such as Neuron ESB by providing pub-sub messaging capable of traversing firewalls.

Neuron ESB 3.0 includes an Azure Service Bus Adapter to provide reliable communication on either endpoint of the Azure Service Bus. The Azure Service Bus Adapter provides full support for the latest capabilities provided by the Windows Azure SDK version 1.7. Once the Neuron ESB 3.0 Azure Service Bus adapter is registered and an Adapter Endpoint is created, configuration is managed through the property grid of the Adapter located on the properties tab of the Adapter Endpoint's Details Pane.



The Azure Service Bus adapter supports the following Azure Service Bus Brokered Messaging features:

- ❏ Send to Azure Service Bus Queue
- ❏ Send to Azure Service Bus Topic
- ❏ Receive from Azure Service Bus Queue
- ❏ Receive from Azure Service Bus Subscription

In addition, the Neuron Azure Service Bus adapter simplifies the development experience by providing additional capabilities required to ensure reliability typical in production scenarios without the need to write custom code including:

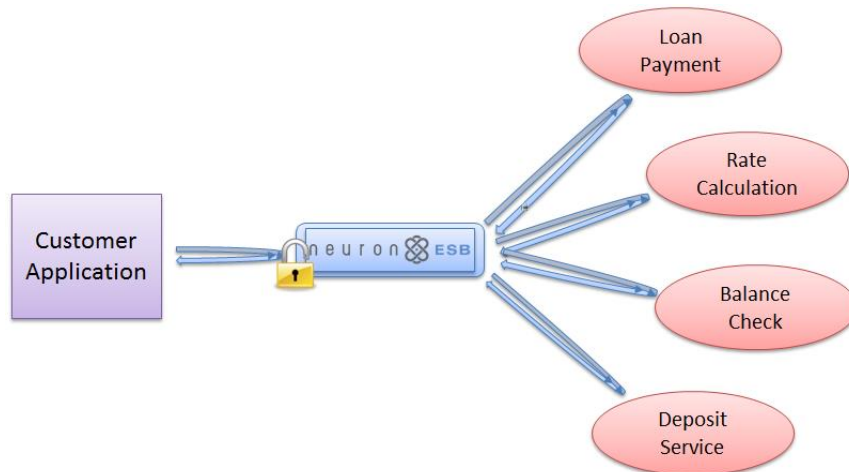
- ❏ Smart Polling
- ❏ Eventual Consistency
- ❏ Transient Error Detection and Retry

Service Broker

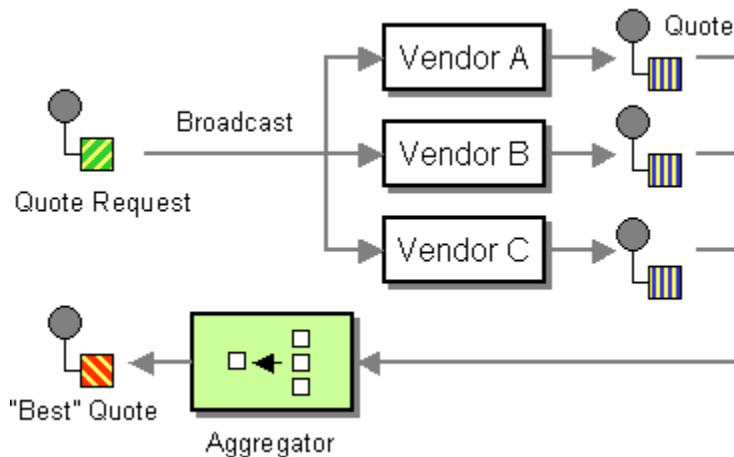
In conjunction with providing a powerful integration platform for the .NET Developer, Neuron ESB 3.0 provides a web service platform facilitating critical functions for organizations interested in successfully adopting a SOA strategy:

- ❏ Service Intermediary
- ❏ Service Router
- ❏ Service Versioning
- ❏ Service Provider
- ❏ Service Patterns

By serving as a broker, Neuron ESB can abstract, process, mediate, and dynamically route incoming service requests to existing services throughout an organization. Neuron ESB can shield customer facing applications from changes that occur in an organization's infrastructure, applications or services.



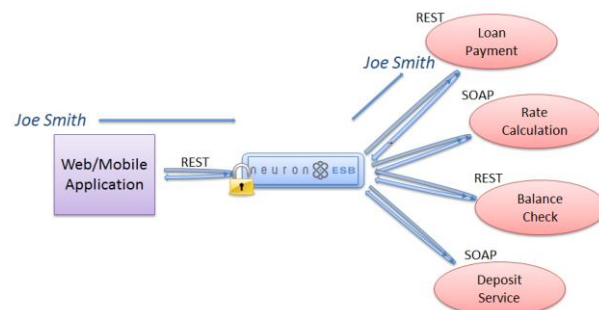
Neuron ESB 3.0 provides a powerful runtime environment for hosting either SOAP or REST based services as well as a real-time Business Process runtime for developing complex service pattern such as Scatter-Gather as depicted below:



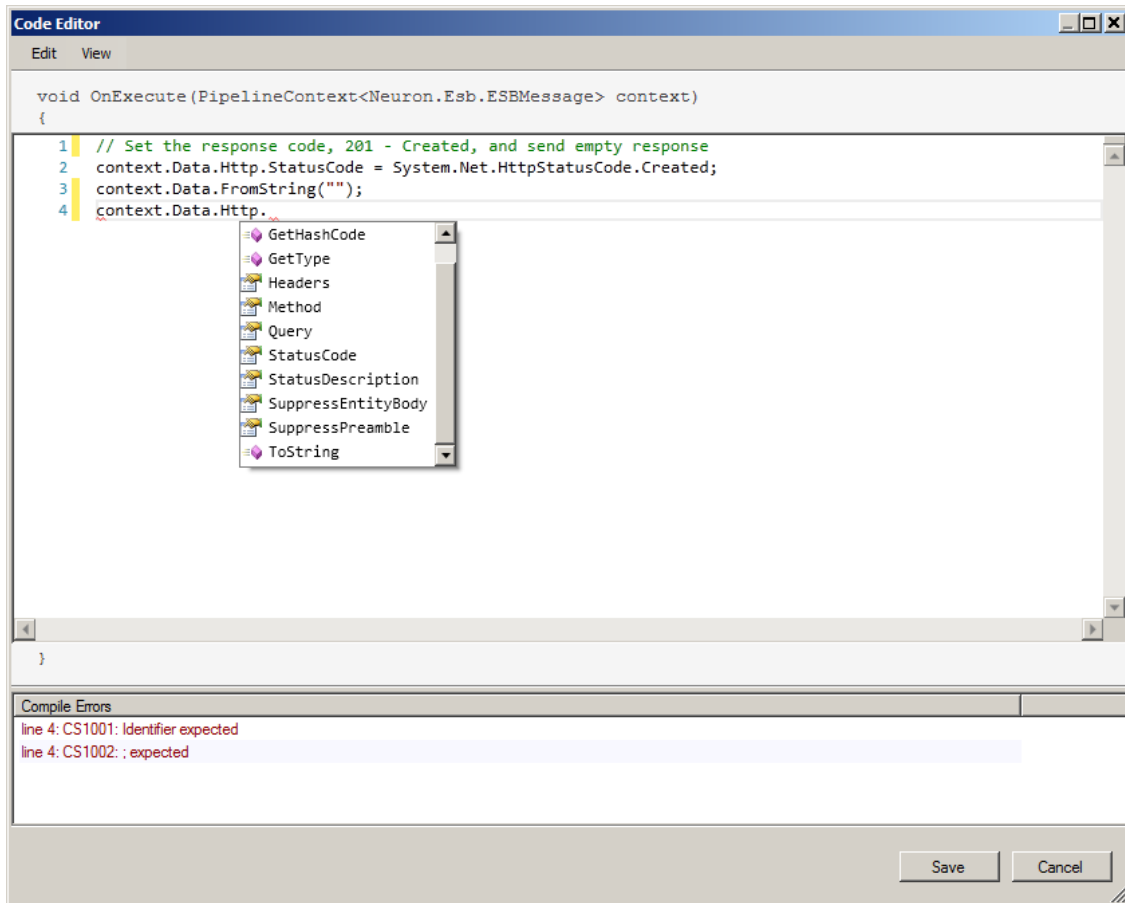
Neuron ESB 3.0 ships with a sample that demonstrates in detail the Scatter Gather Pattern and how to implement it within Neuron ESB.

REST enhancements

Neuron ESB is used by many organizations to broker, route and mediate web service calls for their customer and partners. Although many organizations use SOAP based services, REST based services are an important ingredient for embracing Web 2.0 and mobility scenarios. Many customers today use Neuron ESB to expose existing SOAP based services as REST based services, saving a significant amount of time and development effort in the process.



Neuron ESB 3.0 continues to add support for REST by providing a new object model as well as JSON support. In previous versions of Neuron ESB users had to modify custom property names and text values to manage and manipulate REST communication within Neuron. In Neuron ESB 3.0 introduces a new HTTP class that can be directly referenced within a Business Process Code Step editor as depicted below:

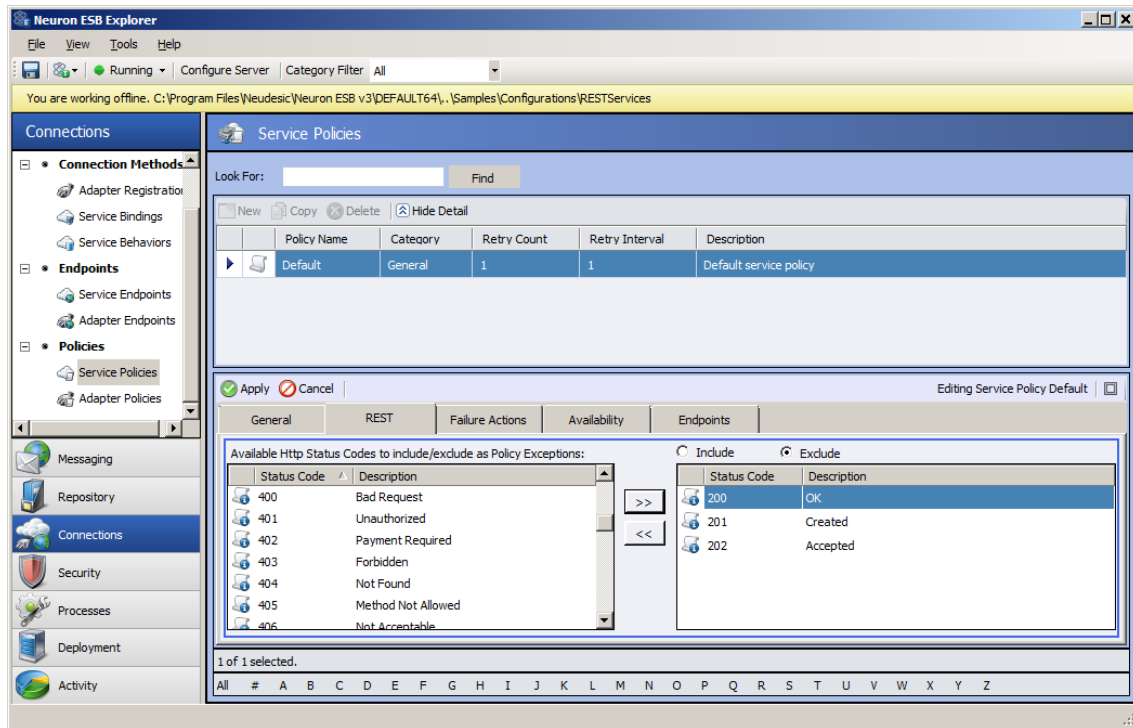


Additionally, Neuron ESB 3.0 ships the Newtonsoft JSON library which can also be referenced directly within the Business Process Code Step editor. Neuron ESB 3.0 ships with 6 different samples demonstrating various REST based routing and mediation scenarios using the new object model and JSON library.

REST support for Service Policies

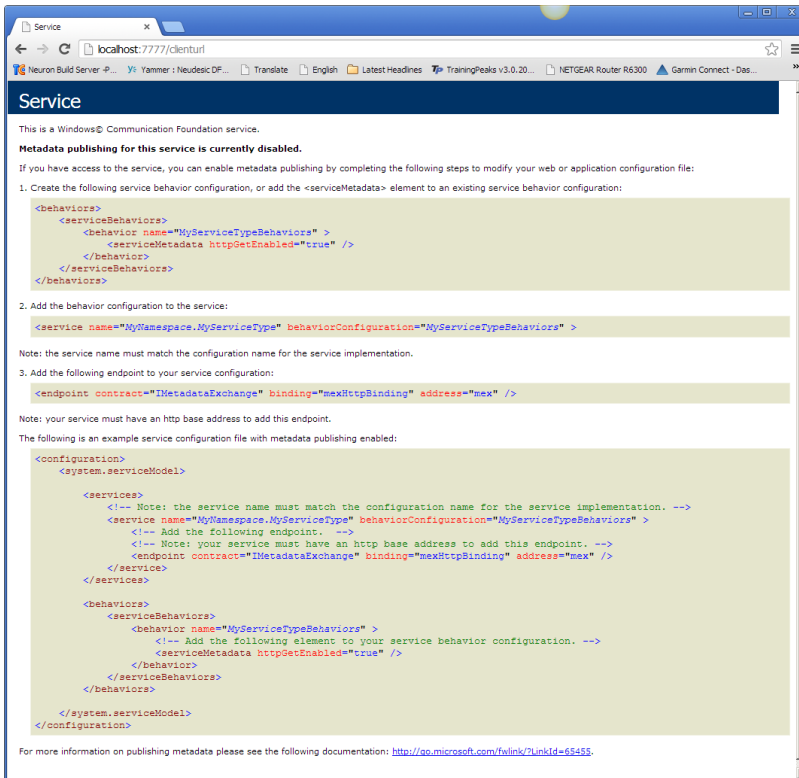
Neuron ESB has both an adapter and service endpoint implementation of Policy that allows customers to manage “how” service and adapter endpoints behave on failure and define what a failure is. Policies can also be used to control endpoint availability and timeouts. Once a policy is defined, it can be assigned to manage either one or more service and adapter endpoints.

With the prevalence of REST based services, Neuron ESB 3.0 introduces REST HTTP Status code support for service policies. Users can now configure a service policy to retry and fail depending on the HTTP Status code returned on a REST service endpoint.



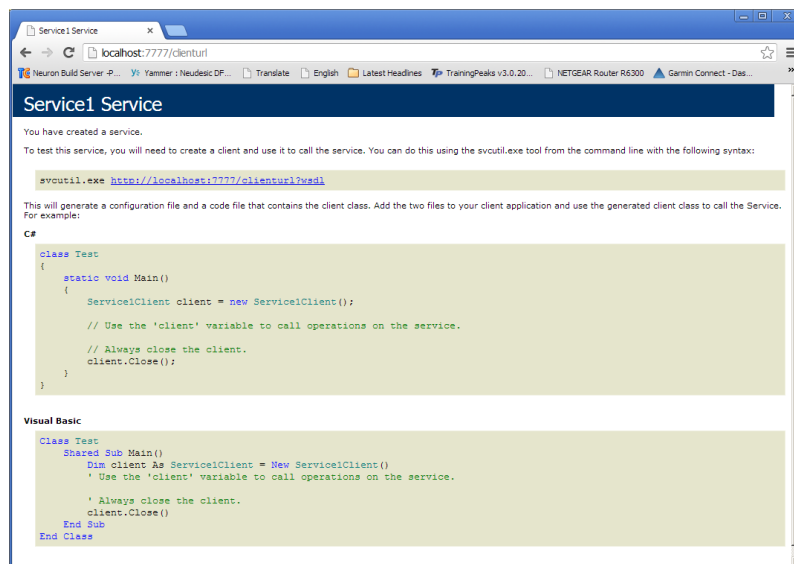
WSDL support for hosted services

Neuron ESB is a powerful web service host. Services hosted by Neuron ESB (i.e. Client Connectors) can be directly configured within the Neuron ESB Explorer. However, Client Connectors are type-less by default. Once you create and enable a Client Connector, if you query the service endpoint (by typing in its URL address into a browser) for its WSDL (Web Service Description Language) you will receive this default response:



Web client developers typically add calls to Web services by importing a WSDL into their development environment. For example, in Visual Studio they would “Add a Service Reference”. However, doing that with a default Client Connector would not produce the desired result since no WSDL is exposed.

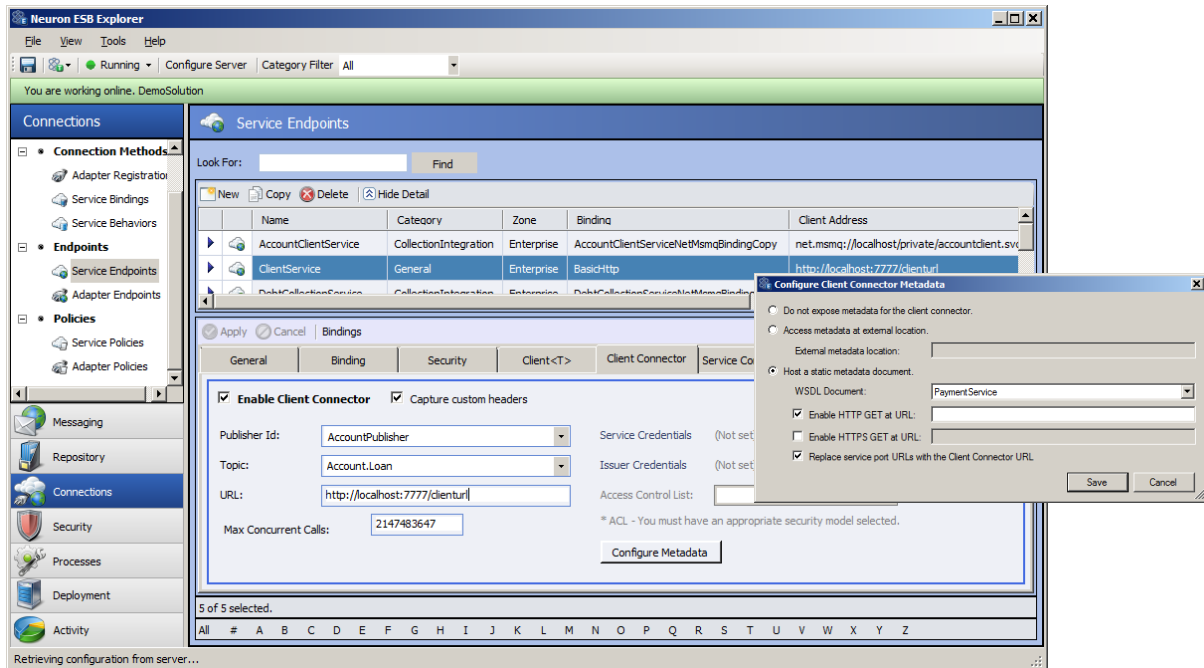
To address this gap in the development of Web clients to Neuron ESB Client Connectors, Neuron ESB 3.0 has added the ability to expose a WSDL. When configured with a proper WSDL, a query of the service endpoint for the WSDL will produce a response similar to this:



Web client developers can then import the WSDL into their development environment as they normally would.

Before configuring a WSDL for a client connector, the WSDL needs to be added to the Neuron ESB Configuration document repository. This can be done either by using the **Import a Service** wizard found in **Connections->Tasks**, or it can be added manually via **Repository->Service Descriptions->WSDL Documents**.

To configure a WSDL for a Client Connector, in **Neuron ESB Explorer** go to **Connections->Endpoints->Service Endpoints** and select the Client Connector. Navigate to the **Client Connector** tab and select the **Configure Metadata** button:



Using Neuron ESB 3.0 users can virtualize existing services within the organization entirely through configuration within the Neuron ESB Explorer.