Neuron ESB Training: Workflows

Overview

This training will provide you with the knowledge necessary to begin using Workflows in Neuron ESB. After this training you will be able to:

- Describe the three different types of Neuron ESB Workflows
- Create Workflow Definitions and Endpoints
- Deploy Workflows to the Workflow Execution Engine
- Develop a Long-Running Transaction with Neuron ESB Workflows
- View the status of workflow instances using Workflow Tracking

You should complete the exercises provided at the end of the training to confirm your understanding of the material presented.

Prerequisites

- .NET 4.7.2
- SQL Server 2008 SP1 or later
- Neuron ESB Installed

  NOTE: When installing Neuron ESB, you do not need to install the Erlang and RabbitMQ Server software packages. However, you *must* install the ESB Service Management Objects on the "Select Features to Install" page.

Neuron ESB Workflow Overview

Long Running Workflow Capabilities vs Business Processes

Neuron ESB ships with Workflow capabilities that allow companies to design fault tolerant, business resilient workflows to automate critical processes that may span hours, days, weeks or months and cross inter- or intra-company domains. Neuron ESB's Workflow offering is built upon Microsoft .NET Workflow Foundation 4.5, overlaying it with tools, infrastructure, a hosting environment, activity tracking and services necessary to deliver enterprise-level performance and scalability on the Microsoft .NET platform.

In contrast to Workflow, Neuron ESB also provides business process capabilities through a graphical, user-friendly Business Process designer and runtime environment. The Business Process engine targeted real-time requirements where performance, agility and time to market were driving factors. This was often used in low latency environments such as request/response type of messaging to provide either simple VETO or, more complex Scatter–Gather and Service Composition/Orchestration Patterns. Service Composition and Orchestration is commonly used to expose a discrete set of services within an organization as higher-level business services.

Neuron ESB Business Process Documentation can be found here:
https://www.neuronesb.com/article/kbtopic/doc-development-business-processes/

For example, a Business Process used to execute purchase orders may "orchestrate" the execution of several existing services in the organization and/or cloud to retrieve the information needed or update necessary systems. The results of which may need to be evaluated, enriched and/or aggregated and returned as the final response. Some of these activities may be executed asynchronously or even in parallel. Collectively, these activities and services represent a higher-level Order Processing Service, where innovation is created by "composing" existing services into new business capabilities.

In addition to Service Composition/Orchestration, many organizations use the Neuron ESB Business Process engine to build fairly complex business processing scenarios. However, where the Business Process engine excelled in the areas of performance, functionality and ease of use, it lacked certain features such as real time activity tracking, fault tolerance, correlation of long running messages as well as "out of the box" compensation (commonly referred to as "saga" or "long running transactions").

Neuron ESB Workflow adds all these features and more, allowing businesses to automate and manage processes that span cloud, partner, system and organizational boundaries. When critical failures occur in the process or the underlying hardware, workflows can resume where they left off in the Neuron ESB hosting environment. Neuron ESB provides a clustered hosting, that load balances the execution of workflows across multiple servers in dedicated/isolated host processes. This same clustered hosting environment allows failed workflows to automatically rollover onto available servers and start where they left off, providing both resiliency and reliability for mission critical functions.

Neuron ESB Workflow Environment

Using Neuron ESB Workflow, it is possible to build business processes that can span days, weeks, or even months coordinating business activities, responding to business inputs, and integrating business systems. Neuron ESB provides a complete Workflow hosting environment for running workflows as part of, or independent of, your ESB messaging solution.

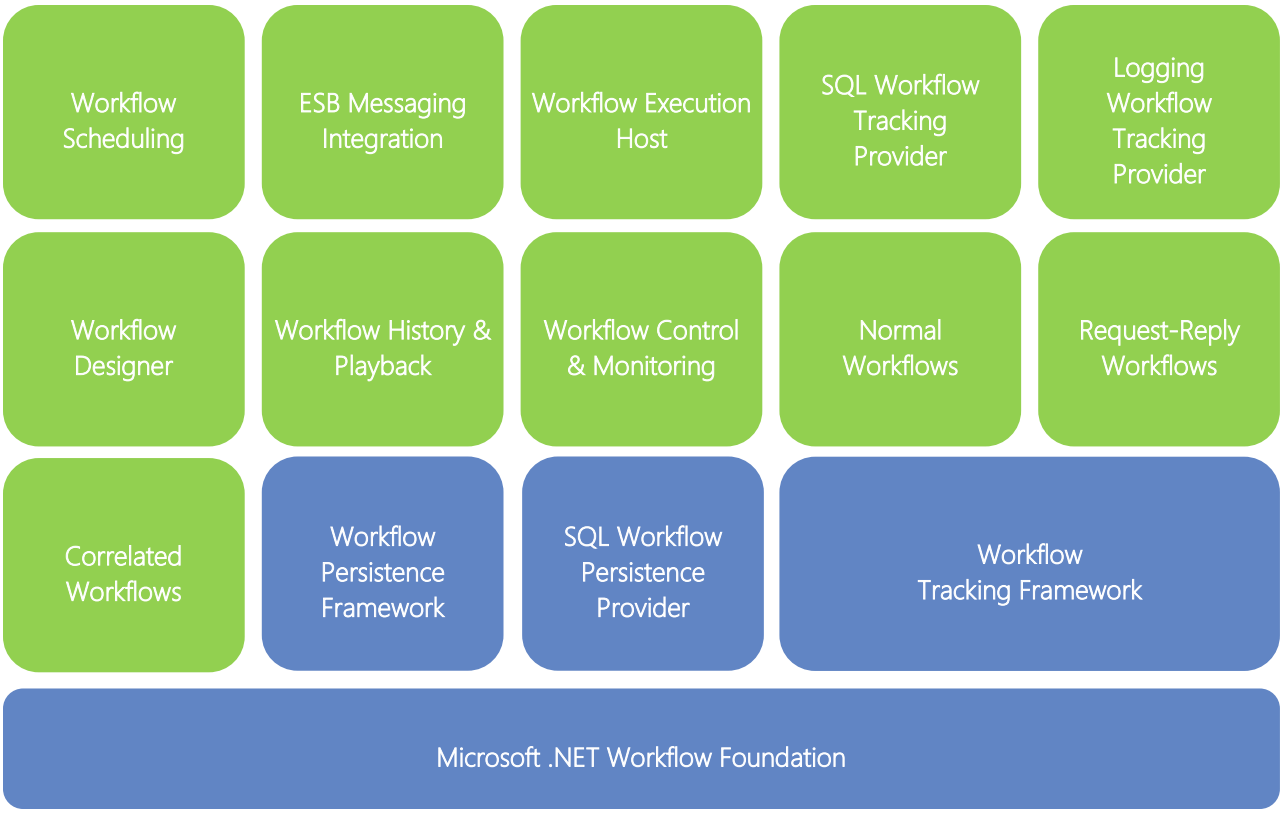


Figure 1 - Neuron ESB Workflow – Green boxes are Neuron ESB provided Infrastructure, tools and runtime services

Features

Figure 1 illustrates the features of Neuron ESB Workflow. The blue highlighted elements are the core features of the Microsoft .NET Workflow Foundation (WF) that come as part of the .NET Framework. The green highlighted elements are the additional features that Neuron ESB provides on top of WF for use in enterprise environments.

Neuron ESB's Workflow is built on WF .NET 4.5. Although Neuron ESB uses WF to manage workflow execution and persistence, significant work was undertaken to make WF manageable, fault tolerant and truly enterprise-ready, including the development of the following:

- Workflow Designer
- Workflow Types
- Workflow Execution Environment
- Neuron ESB Message Integration
- Workflow Tracking and Playback
- Workflow Control and Monitoring
- Workflow Samples

Workflow Designer

Neuron ESB hosts its own Workflow Designer within the Neuron ESB Explorer. The Workflow Designer hosted in the Neuron ESB Explorer is the same designer that developers use inside of Microsoft Visual

Studio to design and build workflows for .NET applications. With Neuron ESB, developers do not need to leave the Neuron ESB Explorer environment or have Visual Studio installed in order to build and edit workflows. Neuron ESB maintains compatibility with Visual Studio workflows allowing workflows to be imported and exported between both environments.

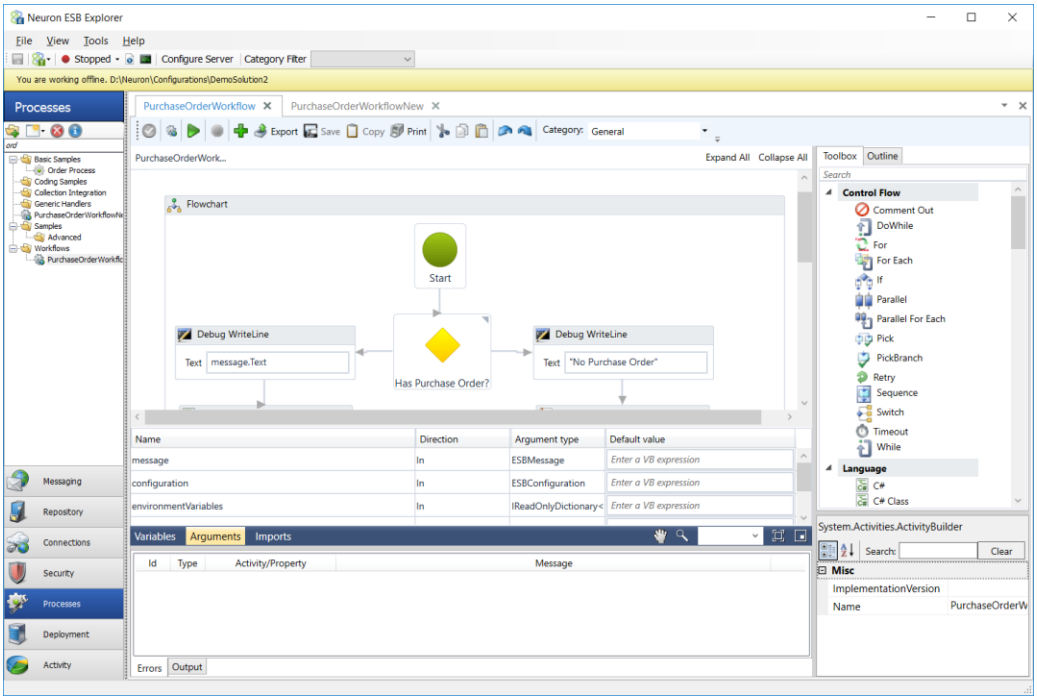


Figure 2 Neuron ESB Workflow Designer – Located within the Neuron ESB Explorer. The toolbox to the right of the designer contains all the out-of-the-box workflow Activities including Neuron specific activities. Over85 activities are included.

Using the Neuron ESB Workflow Designer, users can create new Workflows or import existing Workflows previously created in Visual Studio by selecting either “Create Workflow” or “Import Workflow” from the Processes toolbar.

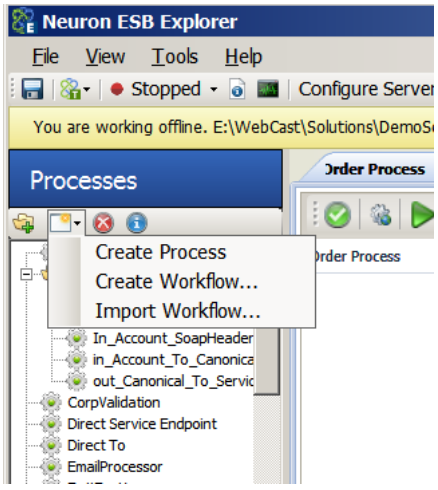


Figure 3 Neuron ESB Processes Menu – Users can create or import existing workflows using the Menu items located on the “New” toolbar button.

Workflow Activity Toolbox

The Workflow Activity Toolbox is located to the right of the Workflow Designer and contains approximately 85 Workflow Activities. Although many are the standard Workflow Activities that ship as part of WF, dozens of others are Neuron ESB specific that enable interaction with Neuron ESB Messaging, Adapters and Service Endpoints or provide some additional level of interaction with the Neuron ESB Messaging system.

Neuron ESB also supports WF custom developed activities as well as external assembly references which can be added through the Neuron ESB Workflow Designer toolbar.

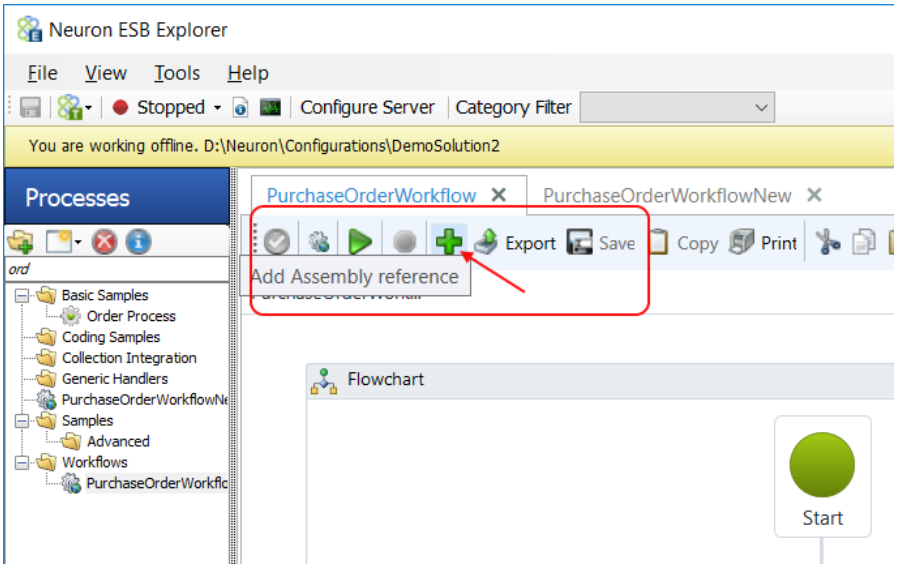


Figure 4 Neuron ESB Add Assembly – Users can add references to external .NET assemblies through the "Add Assembly reference" toolbar button

Custom activities can be added to the Toolbox by copying activity assemblies and dependent assemblies to the Workflows folder under the Neuron ESB instance installation folder and restarting Neuron Explorer (ex: C:\Program Files\Neudesic\Neuron ESB v3\DEFAULT\Workflows). When custom activities are added, they will show up in the Toolbox where they can then be dragged onto the surface of the Workflow Designer.

Workflow Simulation

Neuron ESB's Workflow Designer supports similar testing/simulation features as the Neuron ESB Process Designer. Using the Neuron ESB Workflow Designer, users have the ability to simulate workflow execution for testing at development time. Neuron ESB Explorer's test runner allows developers to pass messages to the workflow and to monitor the workflow's execution in real-time. Several of the workflow activities also support the simulated runtime environment. For example, the *Receive Message* activity will allow developers to test receiving messages during a workflow's execution. There are other Workflow Activities that support a combination of design time and run time testing. Specifically, the Publish Message, Publish Request Message, Audit Message, Adapter Endpoint and Service Endpoint

Workflow Activities can all be tested at design time while interacting directly with the local Neuron ESB runtime solution. To test these, the local Neuron ESB runtime service needs to be loaded and started with the current solution and a valid Source ID and Topic must be provided in the Edit Test Message dialog.

Simulation/Testing is started by pressing the Test Workflow button on the Workflow Designer's toolbar as shown in Figure 5. This will prompt the user for a message via the Edit Test Message dialog (the same dialog used in the Business Process Designer). When the user presses the OK button, the simulation will begin. Each step that executes will be highlighted in Yellow, with any outputs (WriteLine workflow activity or anything that writes to System Console) being written to the Output window located at the bottom of the Workflow Designer. Any errors that occur will be written to the Errors window located at the bottom of the Workflow Designer.

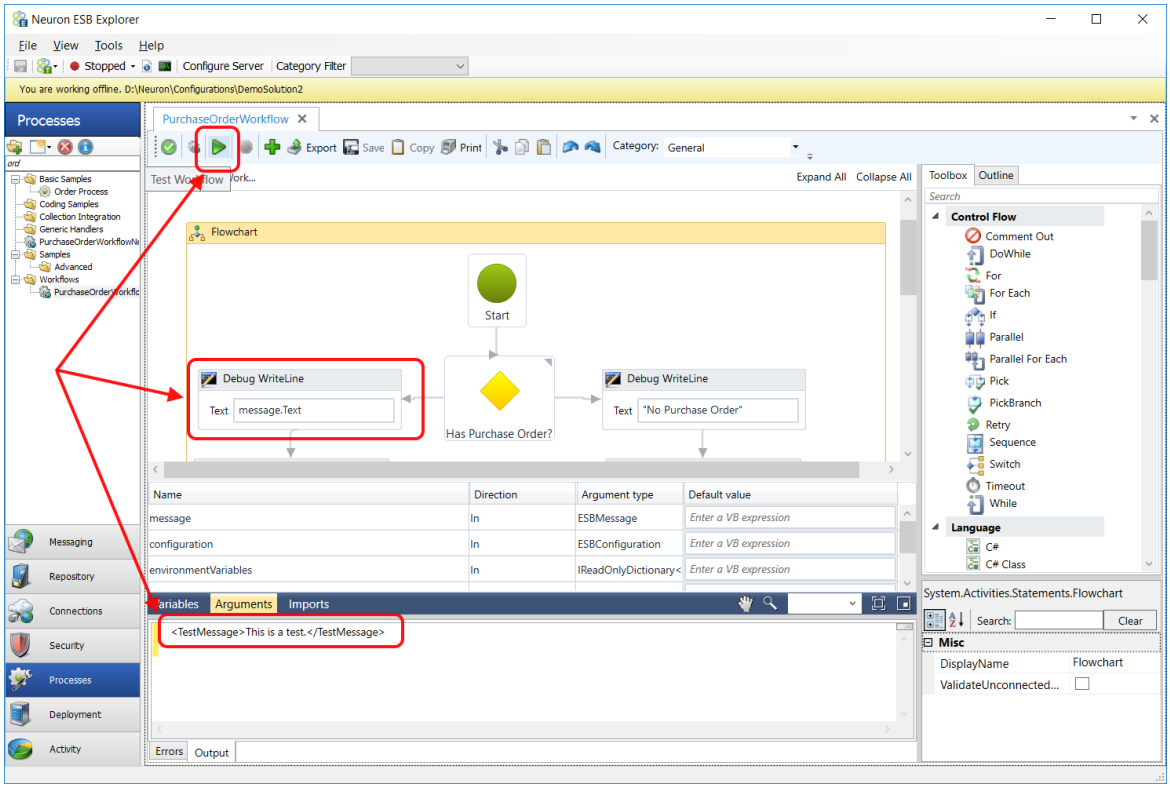


Figure 5 Neuron ESB Workflow Simulation – Workflow Simulation by pressing "Test Workflow" button on toolbar. Each shape executed becomes highlighted in Yellow at time it's executed. Outputs are written to the Output window.

Workflow Types

When users select the "Create Workflow…" option from the Neuron ESB Processes menu, they are presented with a prompt that allows them to choose from 3 basic types of Workflows i.e. Normal, Request/Reply or Correlated Workflow shown in figure 6.

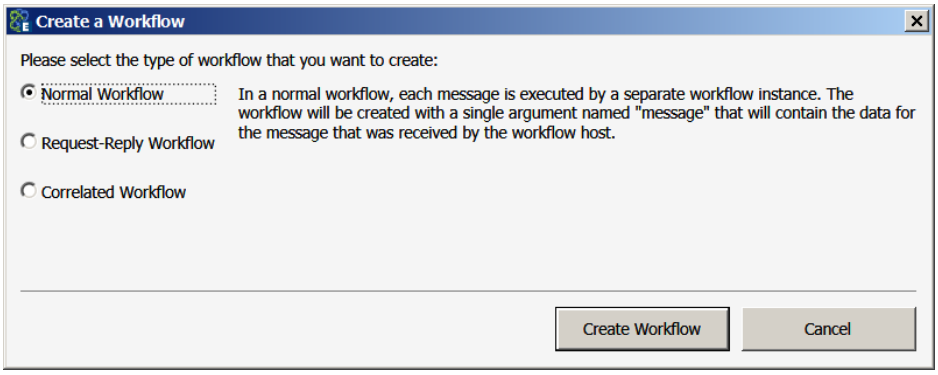
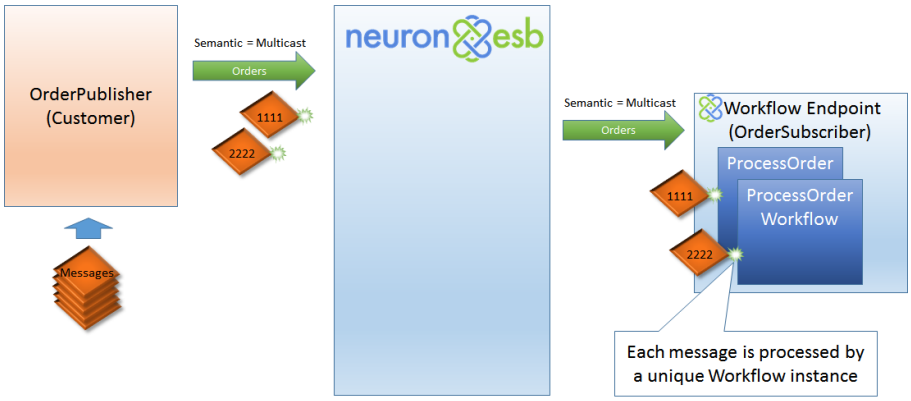


Figure 6 Neuron ESB Workflow Types – Users are prompted to select between Normal, Request-Reply or Correlated Workflow to create.

- **Normal Workflows** are used primarily where a response is not expected from the Workflow.
- **Request – Reply Workflows** are what they imply, where a Request message starts a Workflow and that Workflow instance sends back a response to the client/system that initiated the original Request.
- **Correlated Workflows** are a special type that defines a unique set of messages to be processed by a single instance of a Workflow.

Normal Workflow

Normal Workflow types are most used to create Workflows that subscribe to messages and execute an instance of a Workflow for each message received. When the user creates a *Normal Workflow*, a Workflow definition is created within the Workflow Designer. Once the Workflow is completed, it must be saved and associated with a Workflow Endpoint. A Workflow Endpoint is used to associate the Workflow definition with a Neuron ESB Subscriber, Topic and Endpoint Host. Workflows essentially "subscribe" to messages published to the bus.



When a Normal Workflow is created within the Workflow Designer, the Neuron.Esb namespace as well as 3 arguments specific to Neuron ESB Messaging are added to the Workflow definition, allowing any activity within the Workflow to interact directly with Neuron ESB Messaging or Configuration. The 3 arguments are:

- message

- configuration
- environmentVariables

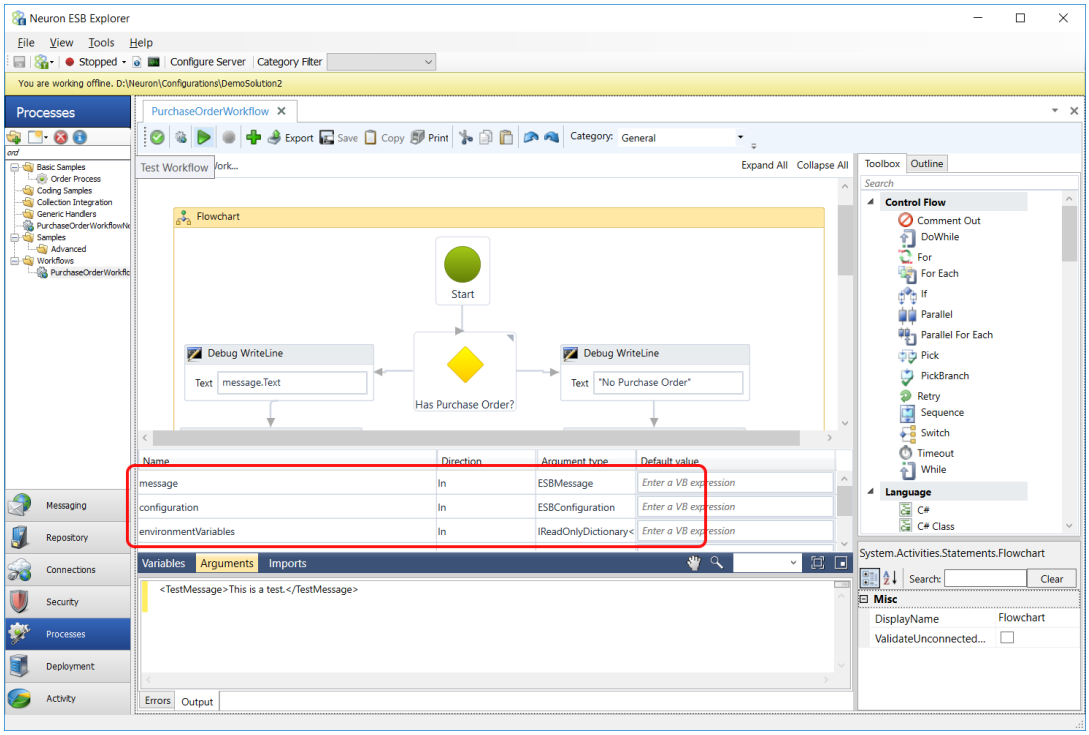


Figure 7 Neuron ESB Arguments Window – Encircled in red, the Arguments window contains the message, configuration and environmentVariables arguments.

As shown in Figure 7, these arguments can be used directly within any activity, including all the Neuron ESB Code Activities such as the C#, C# Class and Visual Basic.NET Activities.

The ***message*** argument represents the original Neuron ESB Message that the Workflow will be initiated by at runtime. All its content and properties are accessible both during design time testing as well as runtime as shown in Figure 8.

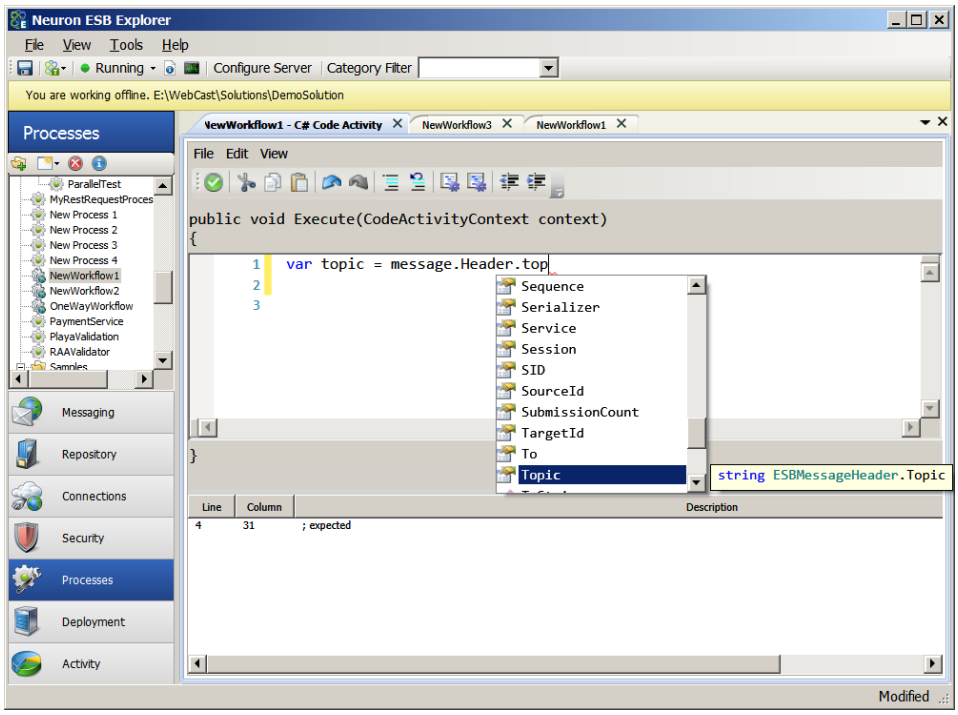


Figure 8 Neuron ESB message Argument – The Neuron ESB Message (message) Argument accessed within a C# Code Workflow Activity. This same argument can be used within any Workflow Activity.

The ***environmentVariables*** argument can be used to retrieve values specific to the runtime environment that the Workflow instance is running in (i.e. Production, Staging, QA, etc.). Neuron ESB Environment Variables are defined and managed within the Neuron ESB Explorer and located under *Deployment->Environments->Environment Variables* section and can be used to configure any Business Process step, Database connection, Adapter and Service Endpoints. Many developers will create application specific Environment Variables, retrieve them at runtime and use their values to drive the business logic within their custom Business Processes or Workflows.

Lastly, the ***configuration*** argument provides access to the entire Neuron ESB solution configuration. Almost all elements of a Neuron ESB solution can be accessed through this object. This can be useful to retrieve XSLT or XML documents, encryption keys, certificates and any other entity contained within the Neuron ESB Solution.

Request-Reply Workflow

Request-Reply workflows are easy to build using the Neuron ESB Workflow Designer by selecting *Request-Response Workflow* in the prompt displayed in figure 9.

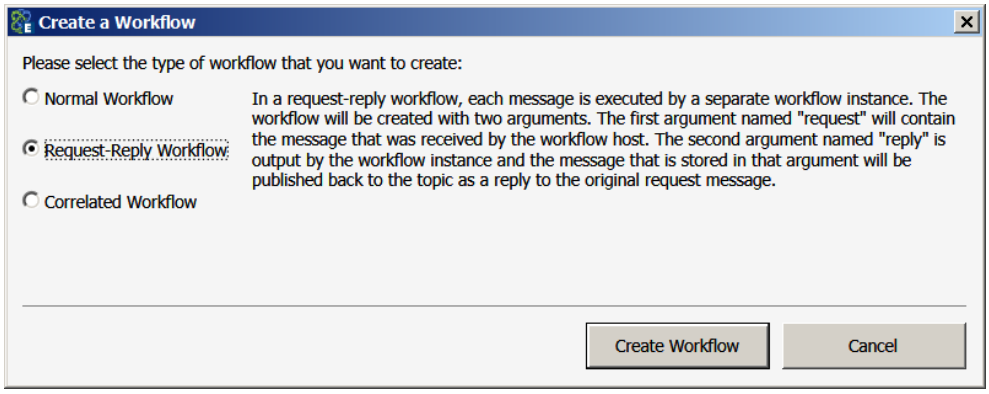
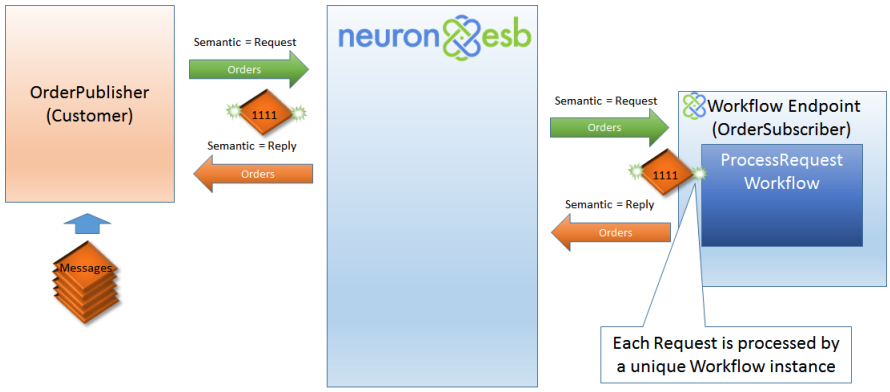


Figure 9 Neuron ESB Workflow Types – Users are prompted to select between Normal, Request-Reply or Correlated Workflow to create. Request-Reply is displayed

By following the pre-defined pattern of accepting a request as an input argument and outputting a message when the workflow terminates, the Neuron ESB workflow engine will automatically return the output message as a reply to the Neuron ESB Party that originally sent the request message. This could be a remotely hosted Neuron ESB Party, an Adapter or even a Neuron ESB Client Connector. The Neuron ESB workflow engine was designed to fit into the messaging patterns employed by Neuron ESB users.



Request-Reply Workflow arguments differ slightly from Normal Workflows. The inbound Neuron ESB Message argument is named *request* (named *message* in Normal Workflows), while the outbound Neuron ESB Message argument is named *reply*. Use the *Create Reply Message* activity to set the *reply* argument and additional code to set its properties. The *reply* argument contains the final Neuron ESB Message that is returned to the original calling Neuron ESB Party as shown in figure 10.

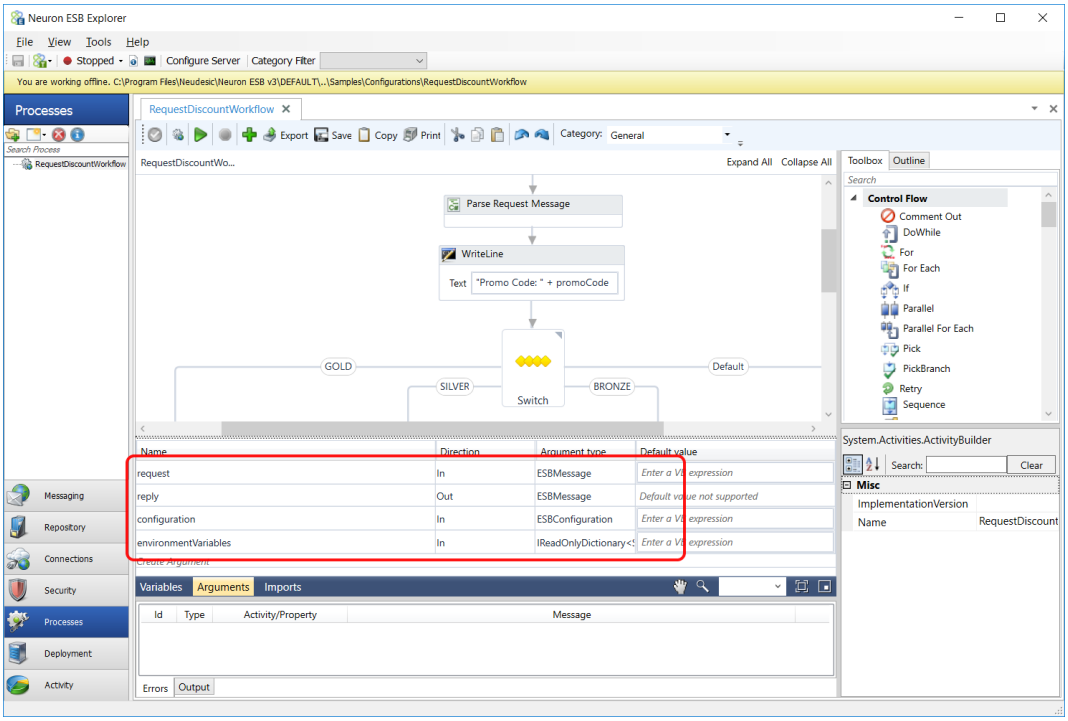


Figure 10 Neuron ESB Arguments Window – Encircled in red, the Arguments window contains the request, reply, configuration and environmentVariables arguments.

Correlated Workflows

The last type of Workflow that can be created is a *Correlated Workflow* as shown in figure 11.

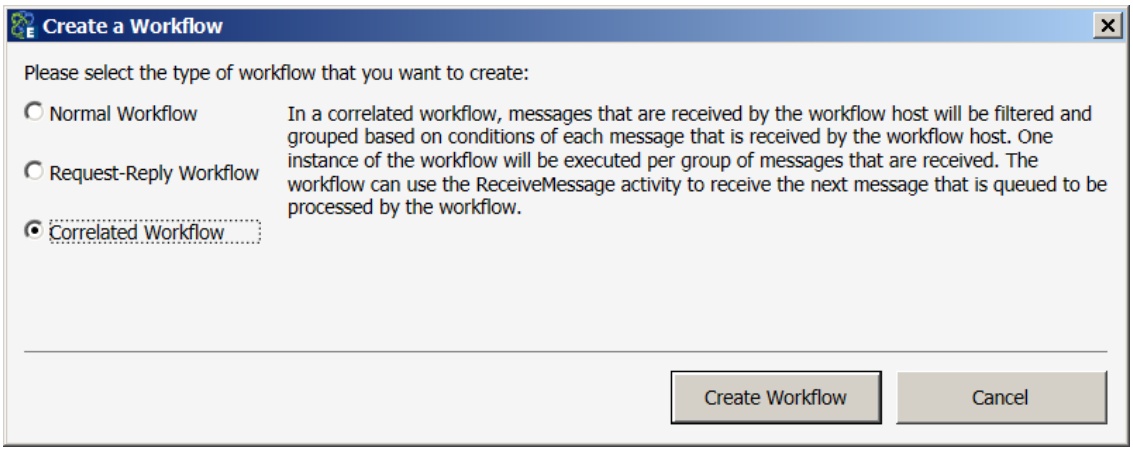
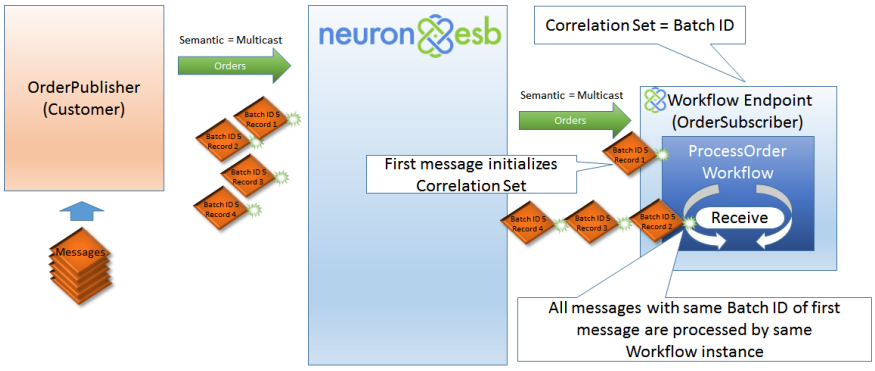


Figure 11 Neuron ESB Workflow Types – Users are prompted to select between Normal, Request-Reply or Correlated Workflow to create. Correlated Workflow is displayed

Correlated Workflows are a special type of Workflow that employs *Custom Correlation* at the workflow endpoint level. Custom Correlation is user-defined criteria that controls what set of messages a single instance of a Workflow will process. This set of messages will be routed to a specific instance of a Workflow (i.e. Singleton pattern), whether that Workflow is running or currently unloaded in the database. Custom Correlation determines the "*uniqueness*" of a set of messages. Although many

instances of a Workflow may still execute, each instance could be processing a set of messages received from the bus, rather than just one, as in the case of Normal Workflows. The first message that launches the instance of the workflow is used to initialize the values of the Correlation Set.



For example, if all messages from a single publisher needed to be processed by the same workflow, the user can choose to correlate messages based off of the session identifier and source identifier. This will result in all messages sent from the same party on the same connection to be processed by the same workflow instance.

Another example would be where a large file was previously split into individual records and published to the bus. A Workflow could be used to aggregate all of the related messages (the individual records) into a new outgoing file. In this case, any combination of message content, custom header, SOAP or HTTP header or even regex and XPATH expressions against the body of the message could be used to define the correlation set for a Workflow instance.

A *Receive Message* Workflow Activity is used within a Correlated Workflow to receive messages that match the Correlation Set definition on the Workflow Endpoint. Usually this is placed within a While or similar loop within the Correlated Workflow since one message is received per execution of the Activity. Hence the *Receive Message* Workflow Activity “follows” the Correlation Set that was initialized when the Workflow Instance was first started (Figure 12).

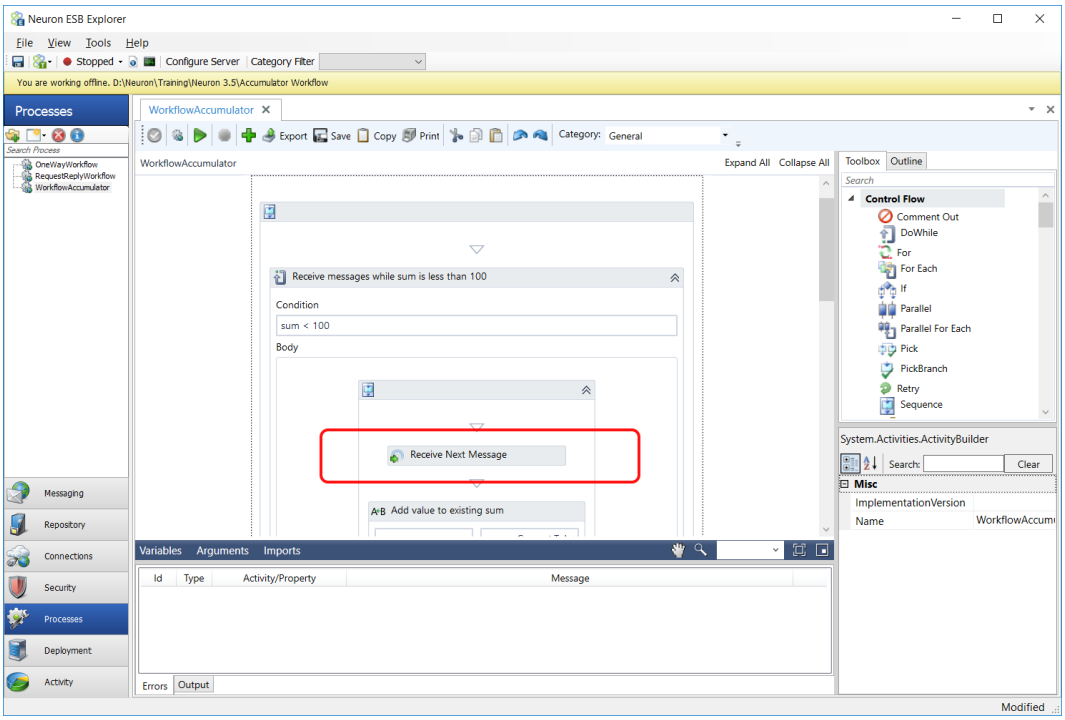


Figure 12 Neuron ESB Sample Correlated Workflow– Encircled in RED is the "Receive Message" Activity. In the sample it is located within a while loop to continually receive messages that match the values of the Correlation Set initialized by the first message that started the Workflow instance. The Receive Message activity will continue to read messages from the Correlated Message Workflow database queue while the sum is less than 100. Once that value is hit, the current instance will complete and a new workflow instance will be created if there are messages pending in the queue that correlate to the completed workflow instance.

When the user creates a Correlated Workflow and assigns it to a Workflow Endpoint, the Correlation Set tab of the Workflow Endpoint is enabled. This allows the user to define the necessary properties that control what messages are processed together by a single instance of a Workflow as shown in the figure below.

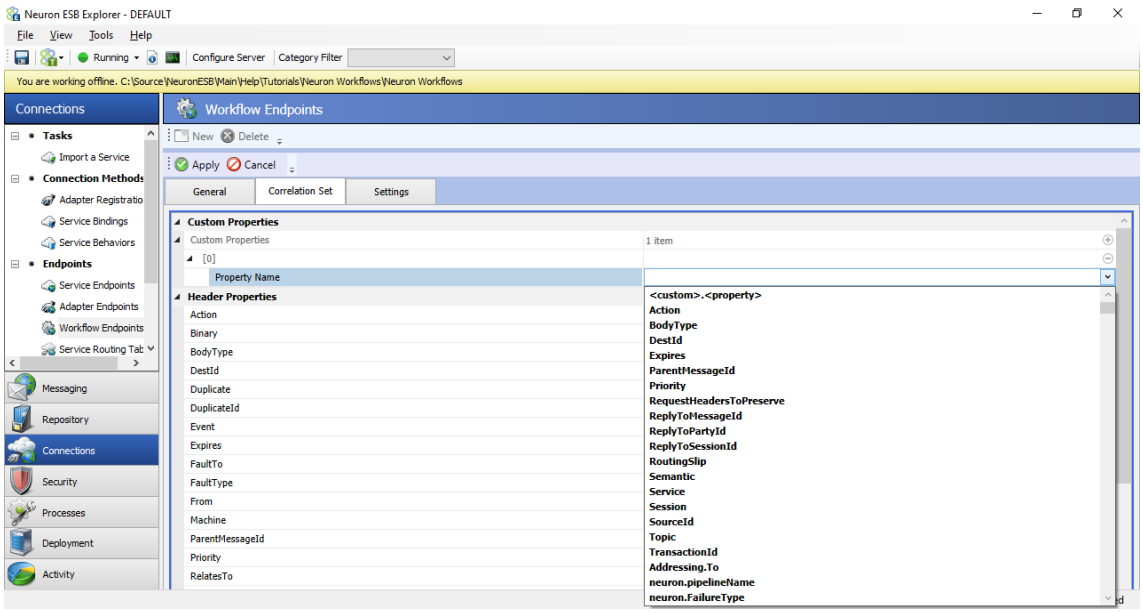


Figure 13 Neuron ESB Workflow Endpoint Correlation Set tab– Users can define the criteria that will determine what set of messages will be processed by each instance of a Workflow. Users can define any combination of either custom properties, Neuron ESB Header properties, SOAP headers, HTTP Headers or Regex or XPATH expressions against the body of each message.

Correlation is not limited to Correlated Workflows. Correlation Sets can also be defined within any type of Workflow using either the *Publish Message* or *Message Correlation Scope* Workflow Activity followed by the *Receive Message* Workflow Activity. Whereas the *Publish Message* Workflow Activity requires that a message be sent to some destination, the *Message Correlation Scope* allows a Correlation Set to be defined within a workflow instance without sending a message. If a running Workflow Instance has any Publish Message or Message Correlation Scope Activities, the message will be placed in the Correlated Message Workflow database queue until the appropriate Receive Message Activity is executed. If there is a Workflow Instance that a message correlates to, but the Workflow Instance has already completed or been canceled, the message will start a new instance of the Workflow. If the Workflow Instance that a message correlates to aborts before the message is received, an error will occur stating that the message has been sent to an aborted workflow and the message will be considered as failed. It will also be audited into the Failed Messages database table.

Workflow Execution Engine

Neuron ESB Workflow Endpoints and Endpoint Hosts are used for hosting workflows. A Workflow Endpoint is a first-class citizen in the Neuron ESB ecosystem, similar to Service Endpoints or Adapter Endpoints. A Workflow Endpoint is associated with a specific Workflow Definition created in the Neuron ESB Workflow Designer. It acts in a subscriber role and is configured to receive messages from a Topic by a specific Subscribing Party. Lastly, it's assigned to a specific Endpoint Host that serves as the runtime host (Neuron ESB workflow engine) for managing the execution of the Workflow Definition associated with the Workflow Endpoint. Users can create any number of Endpoint Hosts that in turn can be assigned to any number of Workflow Endpoints.

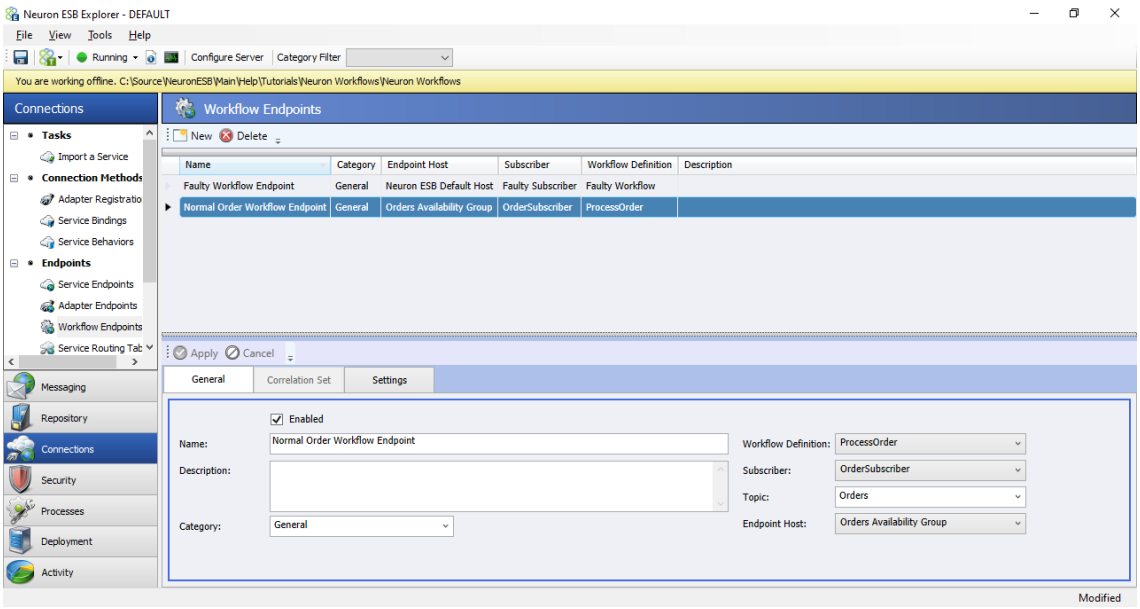


Figure 14 Neuron ESB Workflow Endpoints – The General Tab of Workflow Endpoint allows users to configure the Subscriber ID, Topic, Workflow Definition and Endpoint Host.

Neuron ESB's Workflow Endpoints also employ configurable limits on workflow execution in order to not overload host servers. The Neuron ESB workflow engine utilizes a persistent queue built inside of SQL Server to schedule the execution of workflow instances. The persistent queue allows workflows to be interrupted if it is necessary to restart the ESB Service or the host service, and the workflow engine can restart work processing when the Workflow Endpoint is restarted.

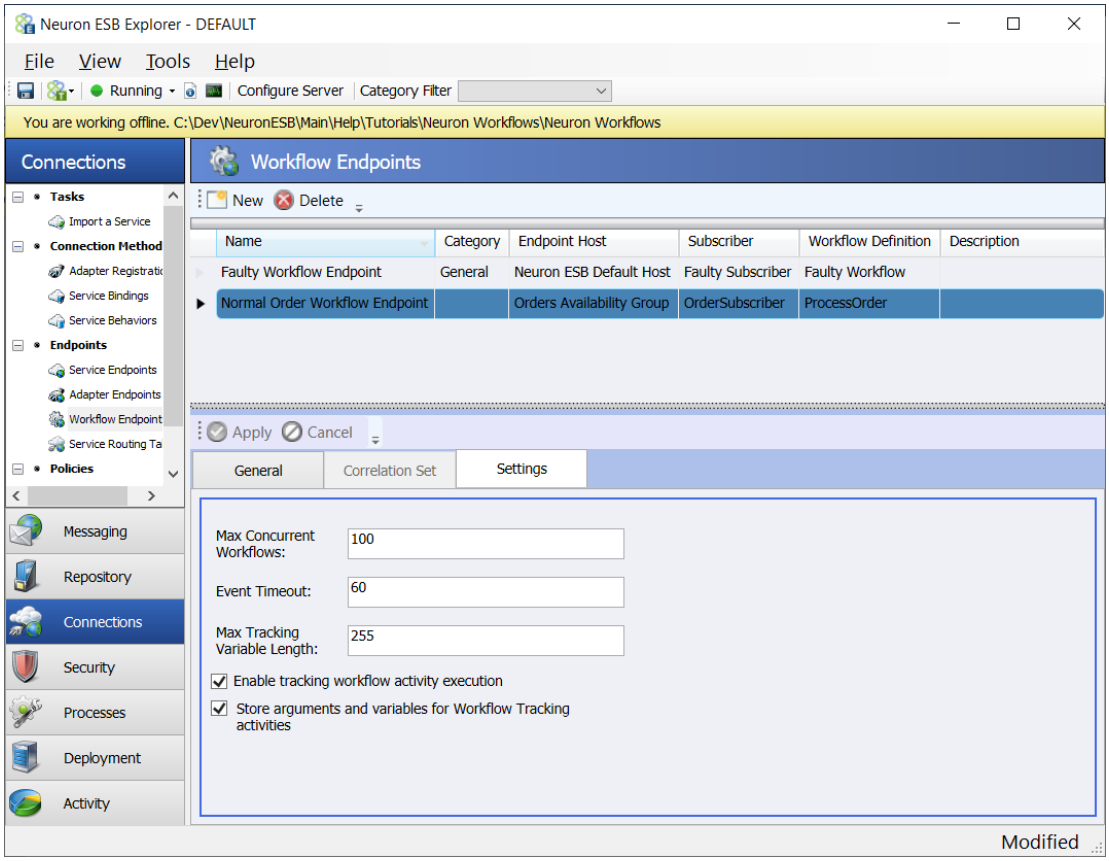


Figure 15 Neuron ESB Workflow Endpoints – The Settings Tab of Workflow Endpoint allows users to configure the Maximum number of concurrent Workflows that should be allowed to run at any one time in the specified hosting environment (i.e. Endpoint Host). It allows users to set limits on the amount of event tracking data that is collected at runtime.

While Workflow Endpoints contain the configuration specific to setting up the environment for executing a Workflow, Endpoint Hosts use the configuration to provide the actual runtime-hosting environment for the Workflow. Endpoint Hosts introduce a new form of isolation and reliability inside Neuron ESB's Runtime Windows service. Endpoint Hosts are isolated into their own process space and execute as child processes of the Neuron ESB Runtime Windows service. If a fatal error occurs that causes the Endpoint Host to fail, the Neuron ESB Server can restart the Endpoint Host on the same server, or on a different server when Neuron ESB is being run in a clustered configuration.

Through the Endpoint Host's Deployment Settings tab, users can configure on what servers, in which specific Deployment Groups they want an Endpoint Host to run on. In a clustered configuration, the Neuron ESB Workflow Engine will schedule Workflow execution to the selected Primary servers within the clustered configuration, monitoring CPU, Memory and the configurable Max Concurrent Workflows property on the Workflow Endpoint's Settings tab. Workflow execution will be evenly spread across all configured servers depending on their resource limits and the type of topic that the Workflow Endpoint's Subscriber receives messages from.

Deploying Workflows

Neuron ESB makes deploying Workflows just as easy as deploying most other entities within Neuron. Once a Workflow Definition has been created and saved (i.e. click the *Apply* button followed by the *Save* button), an Endpoint Host must be created to serve as the runtime host for the Workflow. If one already exists, it may be practical to reuse it. Once the Endpoint Host has either been created or identified, a Topic and Subscriber must be created. Lastly, a Workflow Endpoint must be configured for the Workflow Definition using the Endpoint Host, Subscriber and Topic. Deploying a Workflow involves these 5 entities:

- Workflow Definition
- Endpoint Host
- Topic
- Subscriber
- Workflow Endpoint

In Neuron ESB, each Workflow entity created within the Neuron ESB Explorer (i.e. Endpoint Host, Workflow Endpoint, and Workflow Definition) is saved like all other entities; as individual XML files within dedicated folders on disk. The same deployment methods are used with these as are used with other Neuron ESB entities such as Topics and Endpoints. This also means that when changes are saved, or these entities are deployed to another server, the Neuron ESB Runtime and Workflow host will automatically detect the additions, deletions or changes and load them for execution.

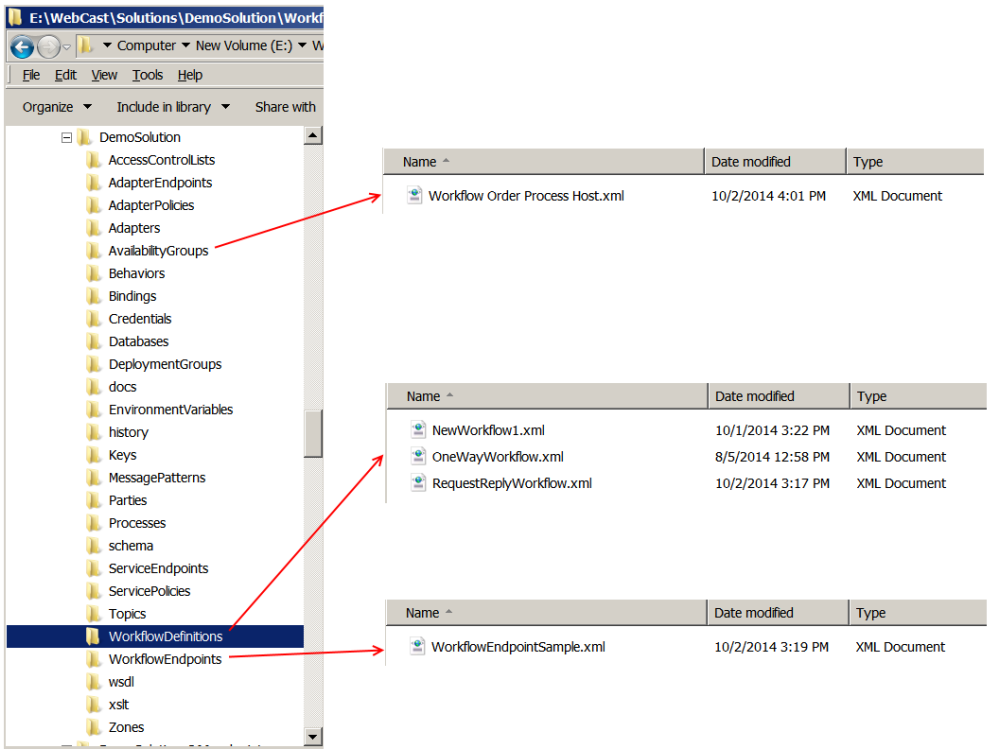


Figure 16 Neuron ESB Workflow Entity Folders – Neuron ESB Workflow Entities are saved as individual XML files in the Neuron ESB Solution directory.

ESB Message Integration

The Neuron Workflow engine and Designer provide a number of integration points into the Neuron ESB Messaging system, some of which are:

- Failed Message Reporting
- Publishing to Topics
- Service Endpoints
- Adapter Endpoints
- Auditing Messages

Failed Message Reporting

Neuron ESB's workflow engine integrates directly with Neuron ESB's auditing service to report on messages that failed while being processed by workflows. This furthers Neuron ESB's goal not to lose messages during transport and processing of messages. Messages that fail during workflow execution can be viewed using Neuron ESB Explorer's Failed Messages report. Additionally, all the messages of a failed or in-flight workflow can be inspected through the new Neuron ESB Workflow Tracking console.

To support integration with Failed Message Reporting, additional properties were added to the Failed Message report:

- Workflow Name
- Workflow Instance ID
- Workflow Endpoint Name.

This allows users to correlate failed messages back against the specific instances of a Workflow that generated them.

Publishing to Topics

Just as in the Neuron ESB Business Process Designer, users can directly publish messages to Neuron ESB Topics using the *Publish Message* and *Publish Request Message* Workflow Activities.

The *Publish Request Message* Workflow Activity allows users to make request/response type calls over the bus by publishing the request message to a Topic (must be Topic other than that configured in the Workflow Endpoint). Neuron ESB will route the message to a subscribing party that could either be an Adapter, Service Connector or a remotely hosted Party (endpoint). That party will process the request and publish the response back to the bus at which point Neuron ESB will return the response back to the Workflow instance that initiated the request. This can be used for bus driven, decoupled, request/response calls.

Alternatively, the *Publish Message* Workflow Activity can be used to publish messages to a Topic using any of the other Neuron ESB message semantics such as Multicast (fire and forget), Reply, Direct or Routed. Once published, the Neuron ESB Messaging system will route the message to all eligible subscribers.

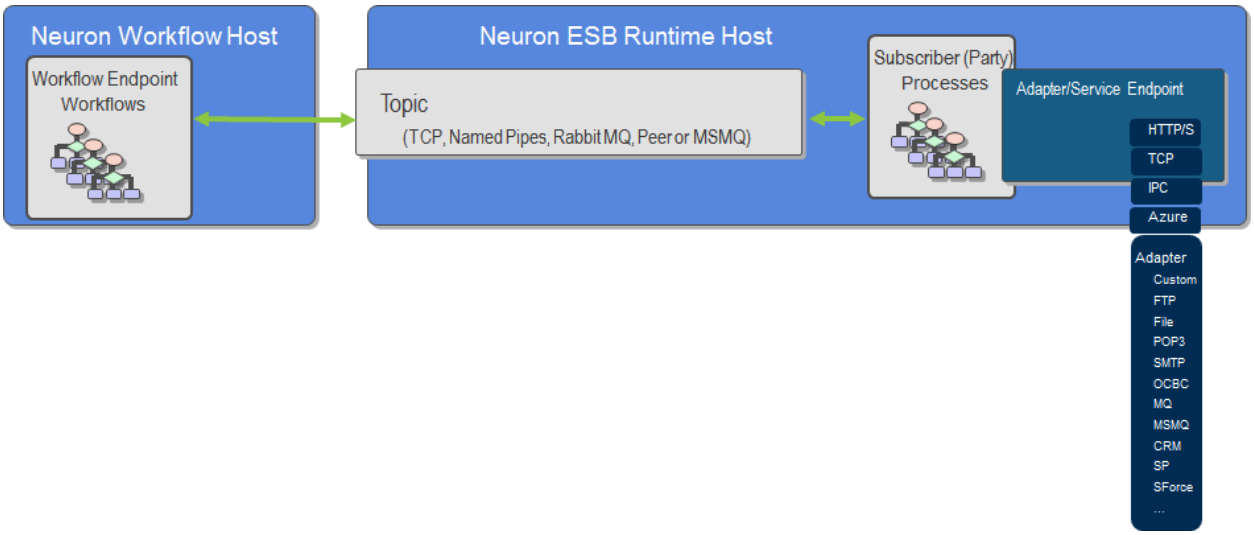


Figure 17 Publish (Request) Message flow– Either the Publish Message or Publish Request Message workflow activity allows for direct integration to the Neuron ESB pub/sub messaging system. These activities can publish directly to any Topic.

The *Publish Message* Workflow Activity can also be used for Correlated request/reply scenarios. Users can define a Correlation Set (a property on the Workflow Activity) and set the semantic to Multicast. Once this is done, the outgoing message that is published to the Topic from the Workflow will be used to initialize the values of the Correlation Set. The initialized values for the Correlation Set are persisted to a user defined correlation variable. To receive the expected response message, a *Receive Message* Workflow Activity must be added to the Workflow following the *Publish Message* Workflow Activity and be configured with the correlation variable (i.e. set the "CorrelationId" property on the Receive Message Activity with the correlation variable). In those cases where it's not practical to publish a message to set the value of a correlation variable, a correlation variable can be set using the *Message Correlation Scope* Workflow Activity just prior to using the *Receive Message* Workflow Activity.

For example, imagine a common partner scenario where a request is sent to a vendor. However, the response may not be received immediately. In fact, it could be hours or days before the response is sent back from the Vendor. In that time, the Workflow would have been unloaded from memory and persisted to the database. The runtime environment may have also been restarted or even moved. Once the response message is received, it's the job of the Neuron ESB runtime environment to route the response back to the correct "instance" of the Workflow, i.e. the Workflow instance that sent out the original request. In many scenarios like this, there will not be a unique identifier within the response message to correlate. In those cases, users can define a Correlation Set on the outgoing request message. The *Correlation Set* is a property exposed by the *Publish Message* Workflow Activity. When activated, it will display the Correlation Set dialog as shown in the figure below. This is identical to the Correlation Set tab's user interface of the Workflow Endpoint.

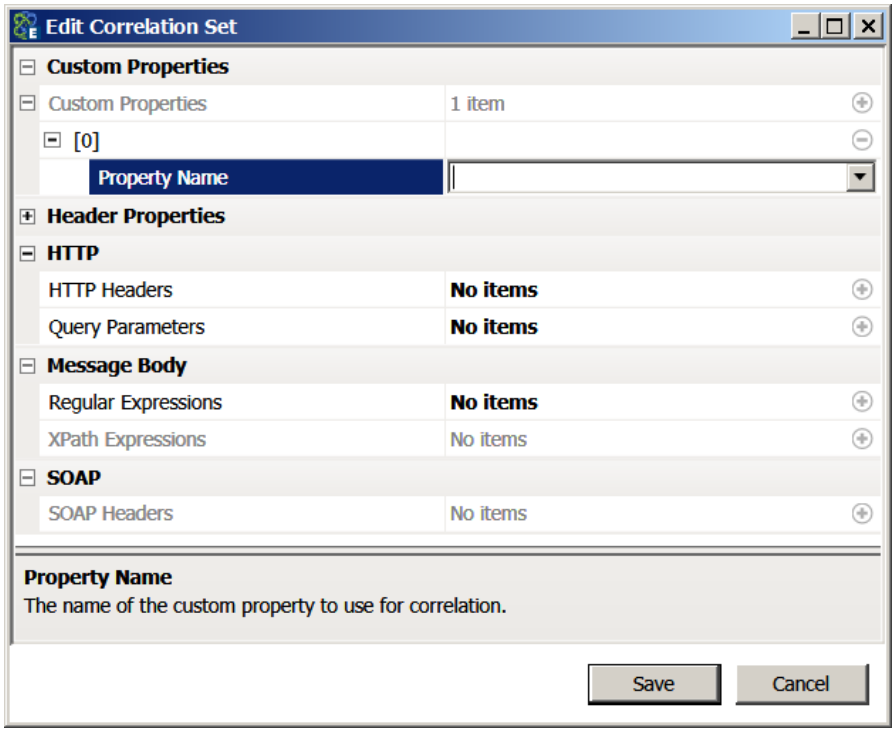


Figure 18 Neuron ESB Publish Message Correlation Set dialog– Users can define the criteria that will determine what set of messages will be processed by each instance of a Workflow. Users can define any combination of either custom properties, Neuron ESB Header properties, SOAP headers, HTTP Headers or Regex or XPATH expressions against the body of each message.

In the case of a purchase order, the user may define a Correlation Set as a combination of customer number, vendor id and purchase order number. When a response message is published to the bus to the expected topic, those property values will be retrieved and evaluated and if a match is found, the response message will be routed to the *Receive Message* Workflow Activity that directly follows the *Publish Message* Workflow Activity on the correct instance of the Workflow. The *Receive Message* Workflow Activity must be used either with a Correlated Workflow, or it must follow a *Publish Message* Workflow Activity. In the latter case, the Receive Message Workflow Activity needs to have the "CorrelationId" property set with a Workflow correlation variable that contains the Correlation ID generated from the Publish Message Workflow Activity.

Service Endpoints

Neuron ESB includes a Service Broker that enables organizations to deploy Neuron ESB as a Service Gateway, providing mediation, security, hosting and a number of other services. Service Connectors are registrations within Neuron ESB that point to existing services hosted within an organization, partner or cloud domain. In previous versions of Neuron ESB, the only way to route a request message to a Service Connector (externally hosted service registered with Neuron ESB) was to publish the request to a Topic, whose underlying publishing service would then route to the Service Connector. This meant that every web service request received by Neuron had to go through the pub/sub infrastructure if an externally hosted service had to be called. Although the pub/sub Topic Transport of choice would usually be an in-memory option, this still led to additional overhead and configuration at runtime.

Neuron ESB provides both a *Service Endpoint* Workflow Activity and Process Step that can be used with either the new Neuron ESB Workflow Designer or the existing Business Process Designer. Either one allows the user to directly call any Service Endpoint registered with Neuron, without the need to publish a request to a Topic, eliminating all pub/sub overhead.

This allows users to create either a Workflow or Business Process that defines an end-to-end solution without the pub/sub abstraction in the middle.

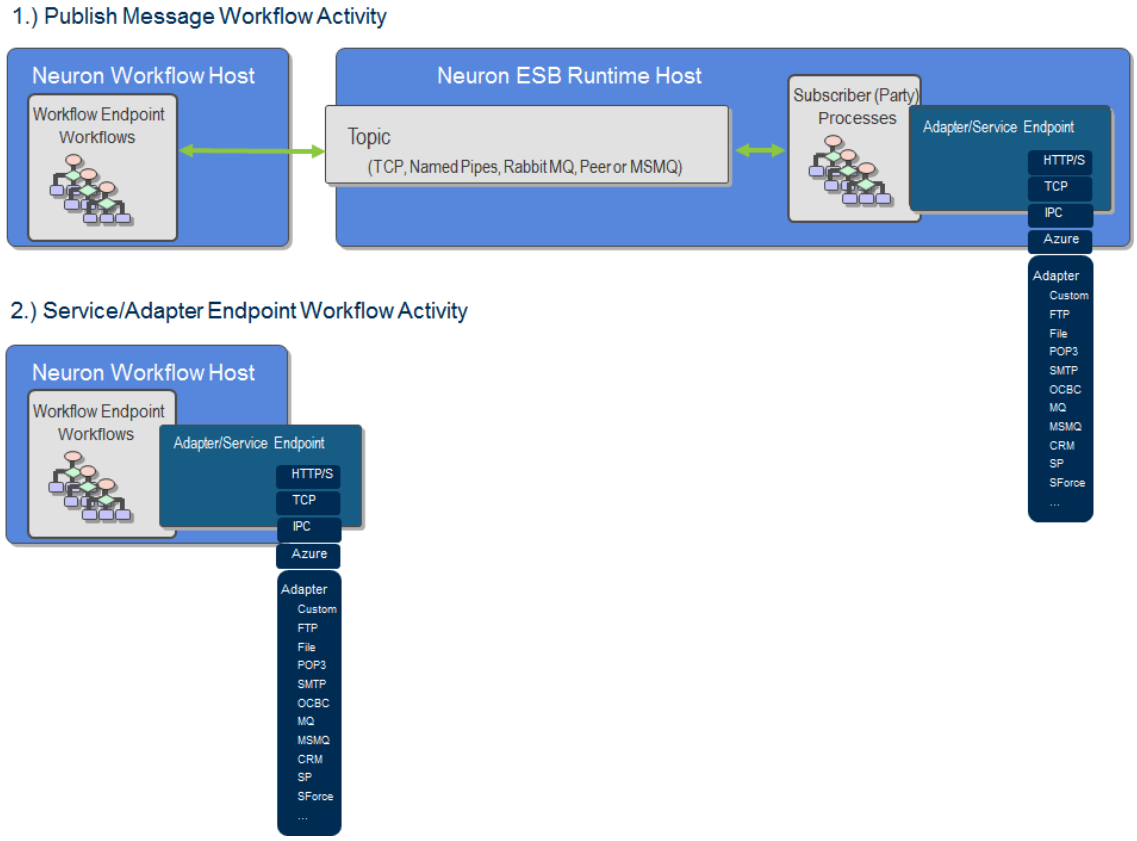


Figure 19 Service Endpoint/Adapter Endpoint Workflow Activities compared with Publish Workflow Activities – When using the Service or Adapter Endpoint Workflow Activities, the entire Neuron ESB Messaging infrastructure is circumvented, allowing registered endpoints to be called directly.

Adapter Endpoints

A core feature of Application and Data Integration servers is their ability to bridge 3rd party applications, databases, technologies, protocols or transports. Neuron ESB provides this through either its library of built-in adapters and by allowing users to build their own adapters using the Neuron ESB Adapter Framework. In many ways, Adapters provide capabilities like those found in Neuron ESB's Service Broker specifically:

- Bridging external endpoints
- Functioning as subscribers

Just as with Service Connectors, Adapter Endpoints would normally need to be called through the Neuron ESB Messaging system where a message is published to a Topic and then routed to eligible subscribers, one of which could be an Adapter Endpoint.

Neuron ESB provides both an *Adapter Endpoint* Workflow Activity and a Process Step that can be used with either the new Neuron ESB Workflow Designer or the existing Business Process Designer. Either one allows the user to directly call any Adapter Endpoint registered with Neuron, without the need to publish a message to a Topic, eliminating all pub/sub overhead (Figure 19).

This allows users to create either a Workflow or Business Process that defines an end-to-end solution without the pub/sub abstraction in the middle. These activities and process steps should always be used with any request/response type of messaging since there will never be more than one subscribing endpoint fulfilling the request. For fire-and-forget messaging (i.e. multicast/datagram), unless there is a need to decouple using the topic-based pub/sub engine as in the case where the publishing process should not know who the subscribing endpoints/parties are or, in circumstances where the process doesn't need to wait until the Adapter Endpoint finishes its processing, using these activities and process steps would be a preferred approach.

Auditing Messages

Neuron ESB provides message-tracking capabilities at several levels, commonly referred to as Message Auditing. When used, messages published or subscribed through the Neuron ESB Messaging system are placed in the Neuron Audit tables (database) where they can be viewed, searched for, edited and resubmitted back to the bus within the Message History report. The first level most commonly used is configured at the Topic level as shown in Figure 20 below.

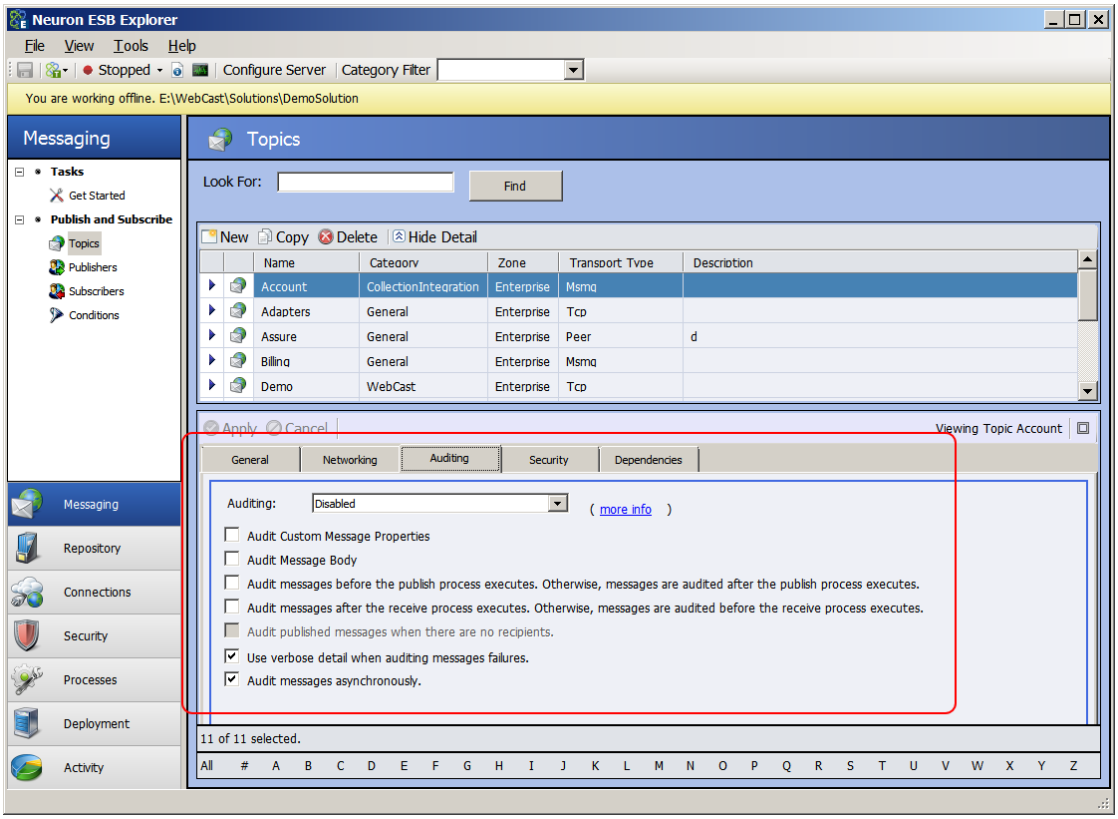


Figure 20 Neuron ESB Auditing Configuration – Global auditing for the Neuron ESB Messaging system can be configured at the Topic level through the Neuron ESB Explorer.

By default, this provides blanket auditing. In other words, all messages that are published to or subscribed to from the Topic will be stored in the Neuron Audit tables. The properties on the Topic's Auditing tab control the time and granularity of the messaging auditing feature. For example, users can choose to "audit" either before or after a process executes or determine whether or not either the message data or custom properties will get tracked, along with the exception stack. Neuron ESB calls this level of Auditing "Global" since it tracks everything that flows through a Topic. In many cases though, users require a more granular level of control over message Auditing. Specifically, the ability to Audit a message at a point in a process where it matters most. In some cases, it may be desirable to audit just a fragment of the message rather than the entire message. For those more common scenarios Neuron ESB provides a configurable Neuron Audit Process Step that users can add to an existing Neuron ESB Business Process to strategically Audit a message as part of a defined process.

The Neuron ESB Audit Process Step can also be used to capture an exception in a Business Process and store the affected message, its context and the exception information are placed in the Neuron Audit tables (database) where they can be viewed, searched for, edited and resubmitted back to the bus within the Failed Message report. This similar feature is also exposed by Service and Adapter endpoint policies, whereas if a delivery failure occurs the policy can be configured to capture the message, context and exception information and place all into the Neuron Audit tables, managed through the Failed Message report.

In Neuron ESB, the Audit Message Workflow Activity is provided, where it can be used in exactly the same way as its counterpart is used in a Neuron ESB Business Process Designer. Users can drag the *Audit Message* Workflow Activity onto the Workflow designer; configure it to capture a specific message and set the properties. Now that message is viewable, searchable and editable within either the Message History or Failed Message reports (depending on context) as shown in the figure below.

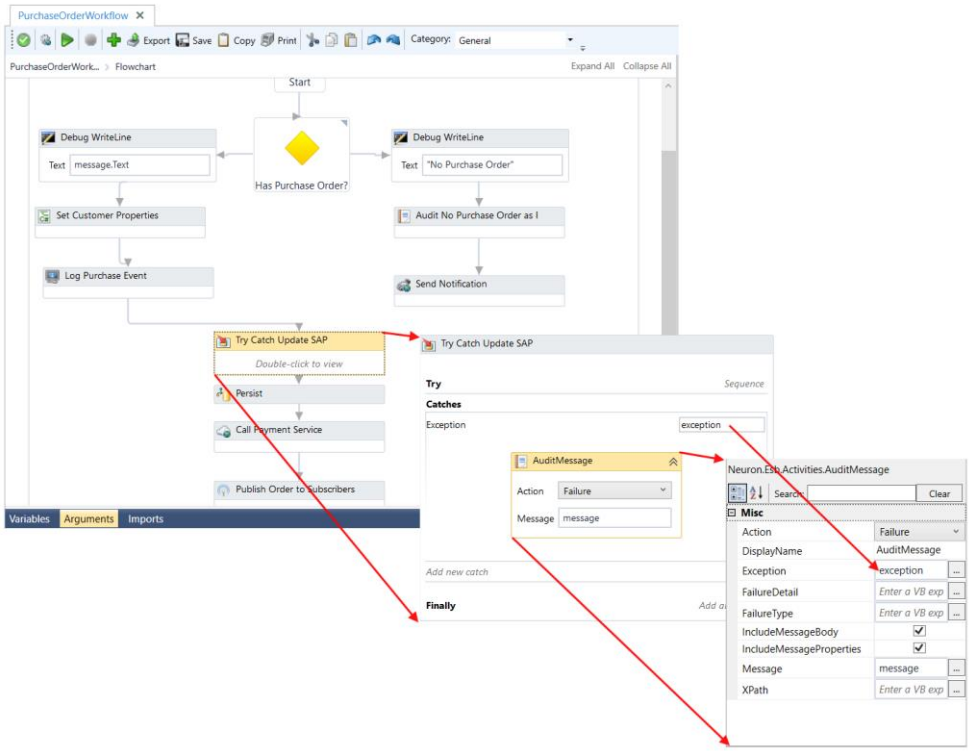


Figure 21 Neuron ESB Auditing within a Workflow – Audit Workflow Activity can be used anywhere within a Workflow to capture the state of the message. It can also be used within a Try/Catch Activity to capture the exception, context and message, storing this information in the Neuron Audit tables where it can be viewed in the Failed Message report.

Persistence

During normal execution, a workflow will write its current state to the database. By default, this occurs at the start of the workflow's execution. While this may be acceptable in some scenarios, there are times where you want the workflow to record its current state more often, such as when the workflow has just completed executing a resource intensive operation, has added new data to a data source or has made a call to a service and received a response back. Because it would be best for the workflow not to have to go through these steps again, Neuron ESB offers the persist activity.

The persist activity instructs Neuron ESB (more specifically, Workflow Foundation) to write the current state of the workflow to the database, outside of the default times when it would do so (e.g. when a Workflow Instance unloads). However, it is important to note that the persist activity overwrites the workflow tracking record in the database with the workflow's current state. This means that the workflow will start off from the last persist activity that it encountered, and you cannot tell it to go back to a persist activity prior to that.

The persist activity is not alone in providing this functionality. The Delay Workflow Activity, used to delay the execution of a workflow for a specified period of time, will also write the workflows current state to the database if the configured delay time is greater than 60 seconds. It does this by unloading the entire workflow from memory and placing it in an Unloaded state (as opposed to the Persist Activity which does not Unload the Workflow Instance). The Receive Workflow Activity works similarly to the Delay Workflow Activity in that it will unload the Workflow Instance after the time span set in the Receive Workflow Activity's "Timeout" property elapses.

Workflow Tracking and Playback

Neuron ESB's workflow engine implements proprietary tracking providers that are used to collect and report execution history for the workflow. The primary tracking provider stores workflow execution tracking events and data into a Microsoft SQL Server database. Once collected in the database, Neuron ESB Explorer provides the monitoring interface that developers or administrators can use to observe the status of each executing workflow, and if desired, play back the workflow's execution. Using Neuron ESB Workflow Tracking, users can observe the execution history and state transitions of the workflow as well as pending messages for workflows. Users can also step through the execution of the activities for the workflow, viewing both the input arguments and the values that are output by each workflow activity. Users are also able to explore error conditions and view exception messages for errors that occurred during the workflow.

Neuron ESB Workflow Tracking (seen in the Figure below) is the central user interface for querying and viewing the tracking information for all workflow instances; those that are in process as well as those that have completed. Workflow instances that are in process, completed or aborted can be queried and viewed on the *Workflows* tab. Queued messages awaiting workflow activation or messages to be correlated into existing workflow instances can be queried and managed on the *Messages* tab.

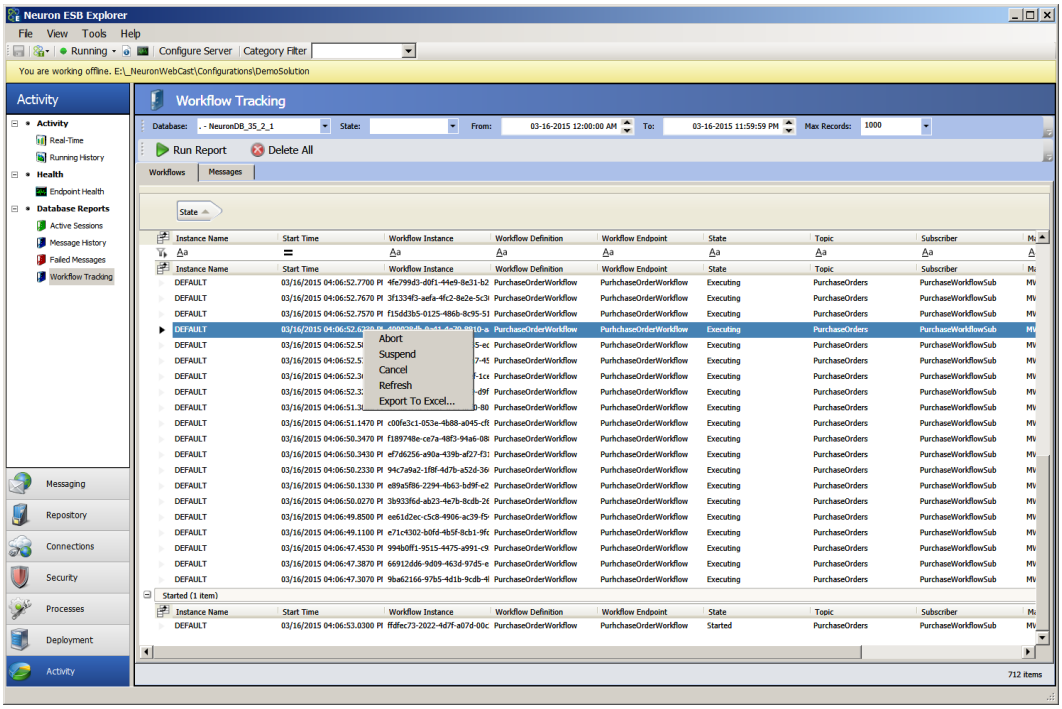


Figure 22 Neuron ESB Workflow Tracking – Workflows Tab - Provides users the ability to see the state of any Workflow Instance during execution or when completed. Users can group, sort, filter, delete, search and choose which columns to include in the main view.

The Workflow Tracking interface allows users to group, sort, filter, search and delete all information related to the execution of Workflows. Users can also choose to Abort, Suspend, and Resume or Cancel a Workflow instance depending on the current state of that instance. Each row in the table represents a unique instance of a Workflow. Double-clicking on any row will bring up the detailed tracking profile of the Workflow Instance. Users can examine the state of the messages, variables and arguments at each stage of the Workflow Instance by navigating through the Tracking events (Tracking Tab) or highlighting a Workflow Activity in the Workflow instance diagram displayed in the main pane. Users can open any ESB Message variable into the Edit Message dialog where it can be edited and resubmitted back to the Neuron ESB Messaging system.

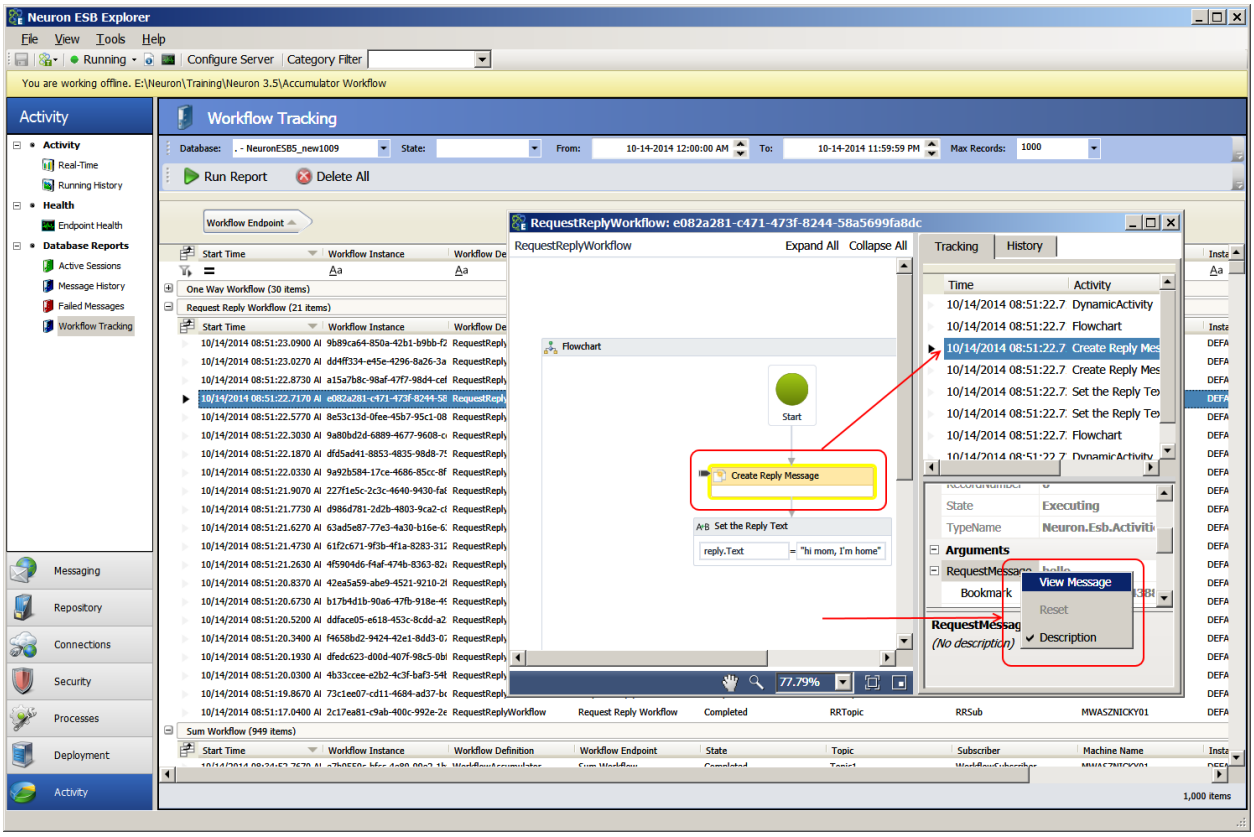


Figure 23 Neuron ESB Workflow Tracking Detail – Users can view the detailed record of any Workflow Instance. This will provide a graphical view of the Workflow Instance, its Tracking information including all state information. Users can navigate the Tracking events to replay the execution of the Workflow Instance.

All the rows in the current view can be exported to Excel by right clicking the grid and selecting “Export to Excel...” from the context menu.

The Messages tab of the Workflow Tracking interface allows users to manage messages queued and awaiting Workflow instance activation. Messages for correlated Workflow instances can also be viewed and managed here.

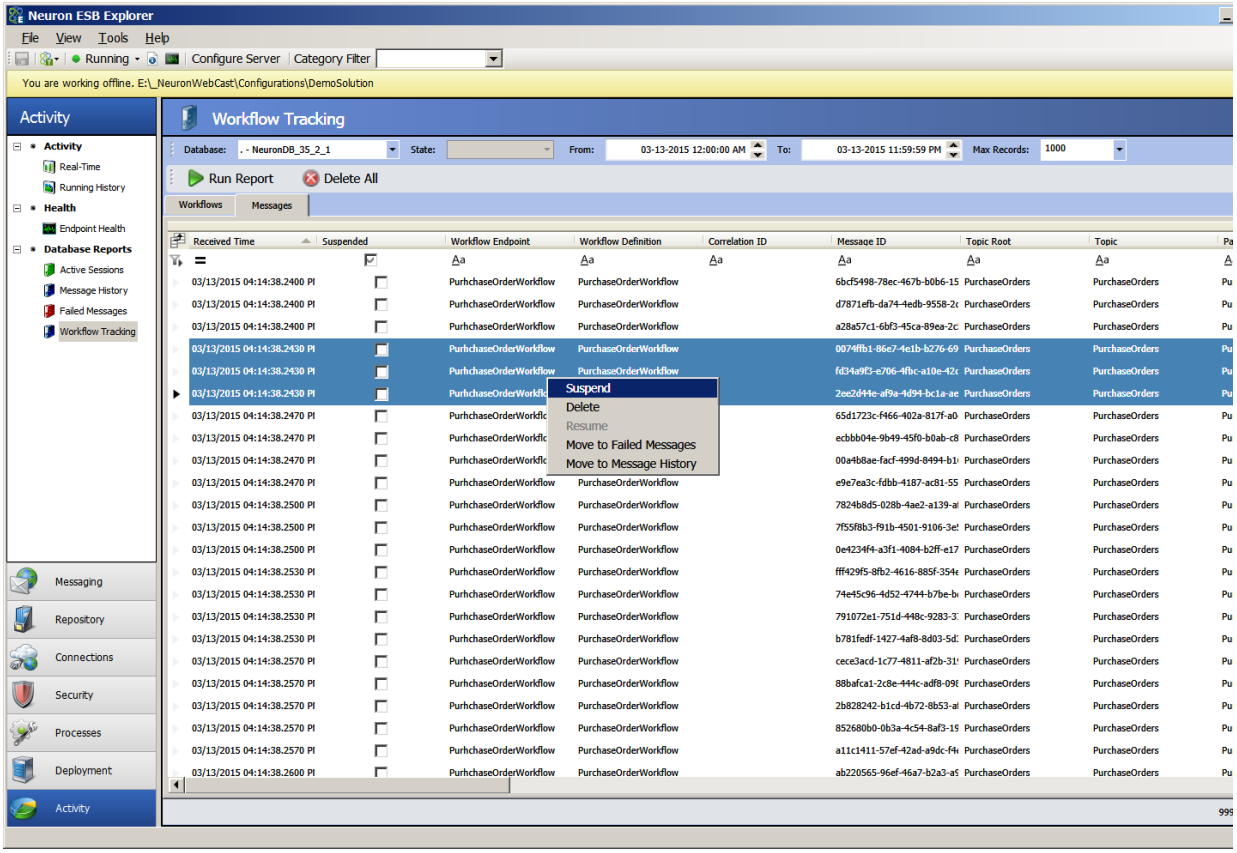


Figure 24 Neuron ESB Workflow Tracking – Messages Tab - Provides users the ability to see the state of any queued messages awaiting Workflow instance activation. Users can group, sort, filter, delete, search and choose which columns to include in the main view.

Users can choose to Delete, Suspend, and Resume pending message awaiting Workflow activation. Additionally, users can choose to select one or more messages and permanently move them to either the Neuron ESB Failed Message or Message History table. Each row represents a Neuron ESB Message which can be opened by double clicking on the row. This will launch the Edit Message dialog where it can be edited and resubmitted back to the Neuron ESB Messaging system.

Restarting a Workflow

When a workflow encounters an error that is not handled by the functionality of the workflow, the workflow will enter into an aborted state and the workflow tracking record will be updated to reflect the situation. This can be caused by an unhandled error in the data processing, the server going offline or some other application error that interferes with the proper operations of the Neuron ESB instance. A workflow in this state can be restarted, either by the Neuron ESB instance that was originally handling the workflow or by another Neuron ESB instance that is running the same configuration (in the case of multiple Primary Endpoint Hosts).

To restart a workflow in an aborted state, workflows in any other state cannot be restarted, you will first need to navigate to workflow tracking and run a report. Running the report and filtering the results by state will allow you to more quickly find the correct workflow instance. Right click on the workflow tracking record for the instance that you would like to restart and select the restart option. This will restart the workflow from the beginning or from the last persistence point that the workflow successfully encountered.

Workflow Control and Monitoring

Neuron ESB provides Workflow control and monitoring through the new Workflow Tracking, Neuron ESB Endpoint Health and WMI Performance Counters. Workflow Tracking provides users with control commands such as Start, Suspend, Cancel, and Abort or Delete that can be used against any selected Workflow instance or group of Workflow Instances. Control commands for Workflow instances are context sensitive, depending on their current state. For example, a Workflow Instance that is in an unloaded or suspended state can be Started, Cancelled or Aborted. An Aborted, Completed or Cancelled Workflow Instance can be deleted while a Started one can be suspended. Control commands are accessed through the right click context menu of the Workflow Tracking grid (as shown in the figure below).

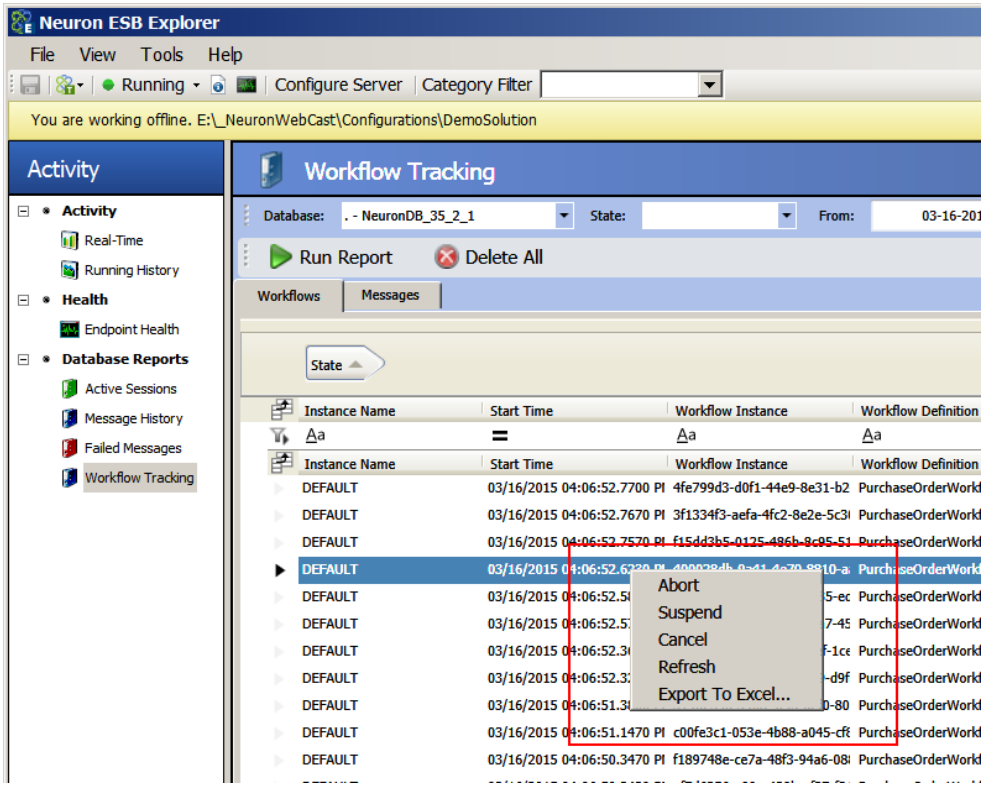


Figure 25 Neuron ESB Workflow Tracking Control Commands – Accessible through the right click context menu. Users can control the execution state of any Workflow instance.

While Workflow Tracking provides detailed information for each Workflow instance, both executing and completed, a summary view of real time Workflow execution activity can be viewed through the Neuron ESB Endpoint Health interface.

Neuron ESB Endpoint Health accommodates real time activity monitoring for the Workflow hosting environments (Endpoint Hosts) as well as the Workflow Endpoints that they host. The interface provides users the ability to group and sort and lists all machines in the selected deployment group.

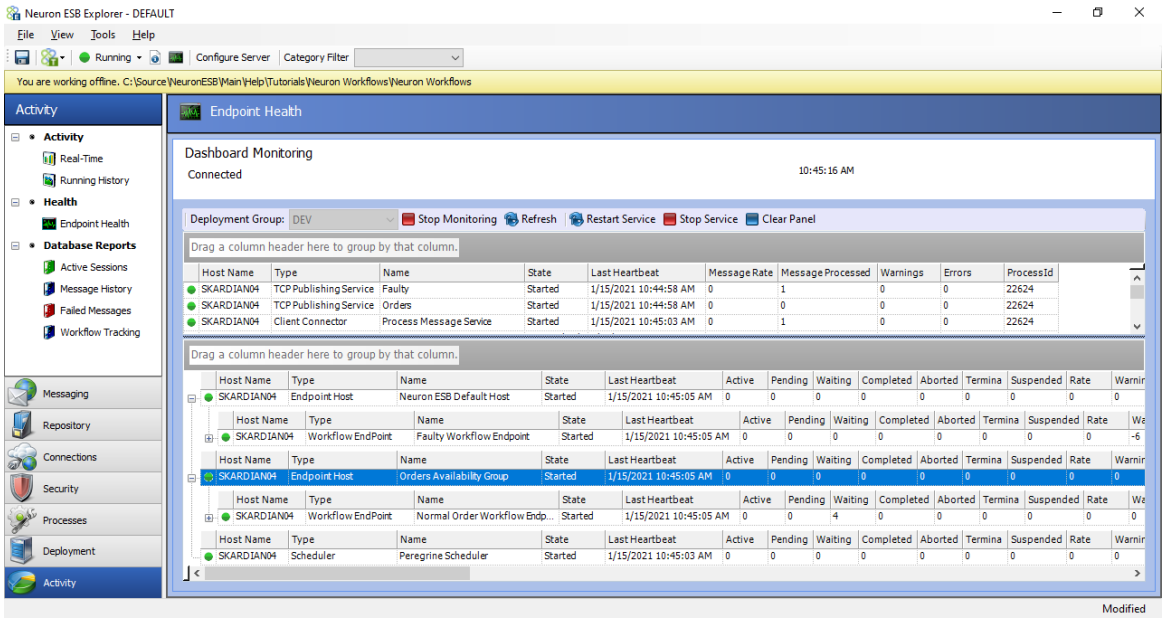


Figure 26 Neuron ESB Endpoint Health – Real time monitoring of both Neuron ESB Messaging and Workflow. Endpoint Hosts (child process) and their Workflow Endpoints can be stopped and started from here. Real time activity of Workflows can be monitored.

The Neuron ESB Endpoint Health interface has a scalable horizontal divider, separating Neuron ESB Messaging entities such as Topics, Adapter and Service Endpoints from Workflow entities such as Endpoint Hosts and their associated Workflow Endpoints. A context menu is exposed at the entity level that allows users to restart any selected entity.

Lastly, two WMI Performance Counter groups represent Workflow and “Neuron Workflow Endpoints”. “Neuron Workflow Endpoints” exposes several counters, including:

- Aborted
- Active
- Cancelled
- Completed
- CompletionRate
- Errors
- Idle
- PendingEvents

- PendingTime
- Persisted
- Terminated
- WaitTime
- Warnings

These WMI counters can used by third party tools for remote monitoring solutions as well as used within Microsoft Performance Monitor.

Neuron ESB WMI Performance Counters Installation

The WMI Counters for Workflow are created by the Setup.exe installer. This feature is represented on the Feature Install page of the installation wizard by the "ESB Service Management Objects" and is disabled (unchecked) by default. Neuron ESB Workflow, specifically the ability to monitor Workflow activity within Endpoint Health, requires that this feature be selected and installed. If this feature is not installed, it can be installed at a later time using the Neuron ESB provided PowerShell scripts.

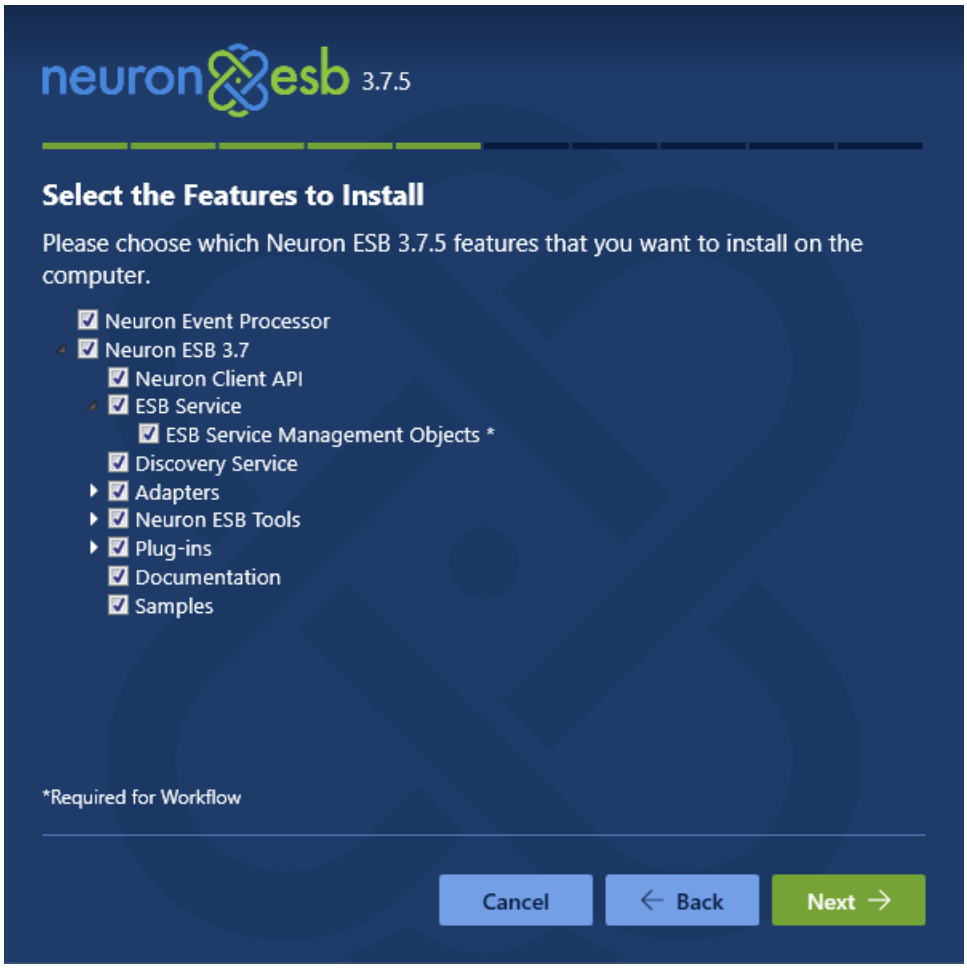


Figure 27 Neuron ESB Setup.exe – Feature selection page of the Neuron ESB installation program. ESB Service Management Objects feature controls the installation of all Neuron ESB WMI Performance counters.

Workflow Samples

Neuron ESB ships several Workflow samples accessible through the Neuron ESB Explorer's Sample Browser, shown in the figure below:

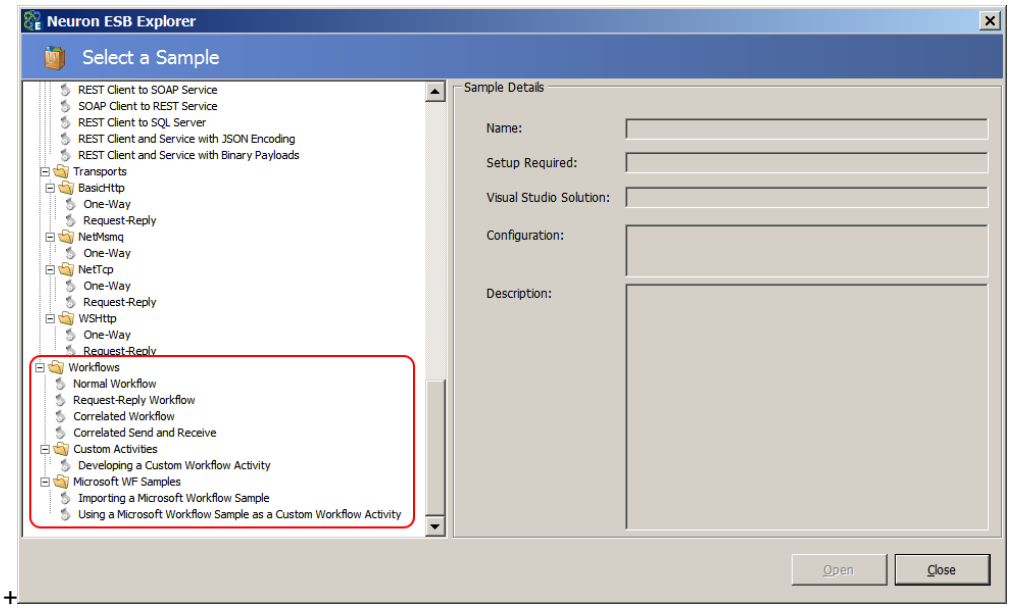


Figure 28 Neuron ESB Sample Browser – Accessible through the Neuron ESB Explorer's View->Samples menu. All Workflow samples are encircled in red.

There are many Workflow Foundation (WF) samples available on the Microsoft Developer Network. Since Neuron ESB hosts its own Workflow Foundation runtime, many samples from Microsoft can be used in Neuron ESB with little or no modifications.

Sample workflows can be imported into Neuron ESB Explorer. This has the advantage of being able to reuse existing workflows and allows users to modify the workflow in the Neuron ESB Workflow Designer.

> *The Microsoft Windows Workflow (WF) Samples can be downloaded and installed from the download link found on the Windows Workflow Samples page.*
>
> *https://www.microsoft.com/en-us/download/details.aspx?id=21459*

The Neuron ESB Workflow samples included are as follows:

Developing a Custom Workflow Activity

This sample demonstrates how to create a custom workflow activity that generates lorem ipsum text. To test, compile the solution, copy the assembly to the \Program Files\Neuron ESB

v3\DEFAULT\Workflows folder and restart the Neuron Explorer. Includes: Visual Studio solution and documentation.

https://www.neuronesb.com/article/kb/developing-a-custom-workflow-activity/

Importing a Microsoft Workflow Sample

This sample demonstrates how to import the *Switch* activity that ships with the Microsoft samples into Neuron ESB Explorer. This sample contains only documentation and requires that you install the Windows Workflow (WF) Samples from the Microsoft site.

https://www.neuronesb.com/article/kb/importing-a-microsoft-workflow-sample/

Microsoft Workflow Sample as a Custom Workflow Activity

This sample demonstrates how to use the *FlowChartWithFaultHandling* activity that ships with the Microsoft samples into Neuron ESB Explorer. This sample contains only documentation and requires that you install the Windows Workflow (WF) Samples from the Microsoft site.

https://www.neuronesb.com/article/kb/using-a-microsoft-workflow-sample-as-a-custom-workflow-activity/

Normal Workflow

This sample shows a simple Normal workflow that logs the text of the incoming ESBMessage as well as the Workflow Instance ID. A Normal workflow is executed once for each message sent to the topic associated with the Workflow Endpoint.

https://www.neuronesb.com/article/kb/normal-workflow/

Request-Reply Workflow

This sample shows a Request-Reply workflow that accepts a promotion code on the text property of the incoming ESBMessage and replies with a new ESBMessage where the text property contains the discount corresponding to the promotion code. A Request-Reply workflow is executed once for each message sent to the topic associated with the Workflow Endpoint.

https://www.neuronesb.com/article/kb/request-reply-workflow/

Correlated Workflow

This sample shows a Correlated Workflow that joins order records after a Process has split them into separate messages. A Correlated-type workflow is executed once for a unique set of messages determined by the correlation set properties configured on the Workflow Endpoint. This is sometimes referred to as a Singleton pattern. In this sample, orders are assigned to batches and the batch ID is used to initialize the correlation set. All orders in batch 'B001' are handled by one instance of the workflow while all orders in batch 'B002' are handled by another workflow instance.

https://www.neuronesb.com/article/kb/correlated-workflow/

Correlated Send and Receive

This sample shows an order message sent to a normal-type workflow that then publishes it to another topic for further processing. When that processing completes the order is sent back to the existing instance of the workflow that published it. Setting the correlation set on the Publish Message Activity to the order ID routes to the original workflow.

https://www.neuronesb.com/article/kb/correlated-send-receive-workflow/

Business Process to Workflow

The business process to workflow sample will show you how to receive information inside of a business process, in order to return an immediate response to a client, while passing the processing of long running task off to a workflow.

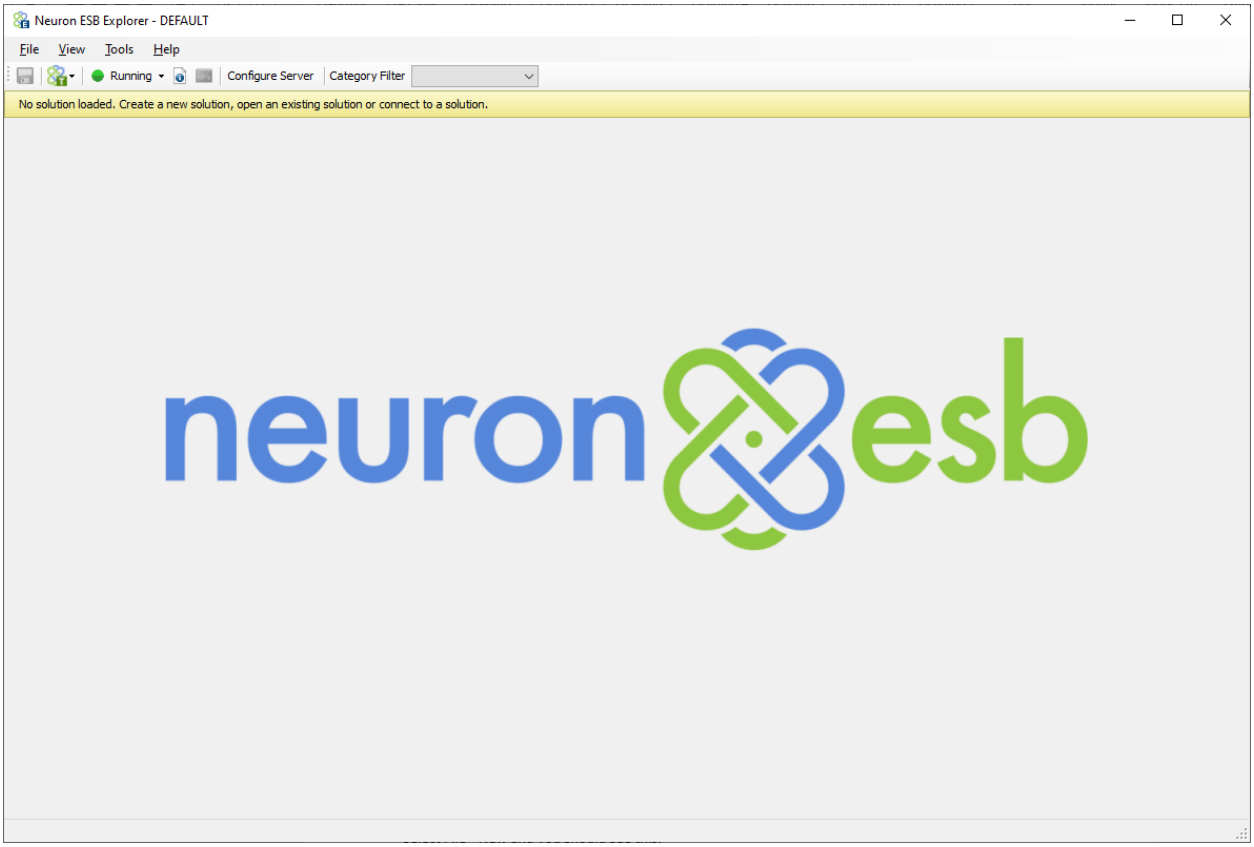
https://www.neuronesb.com/article/kb/business-process-to-workflow-sample/

Exercise – Create a New Configuration

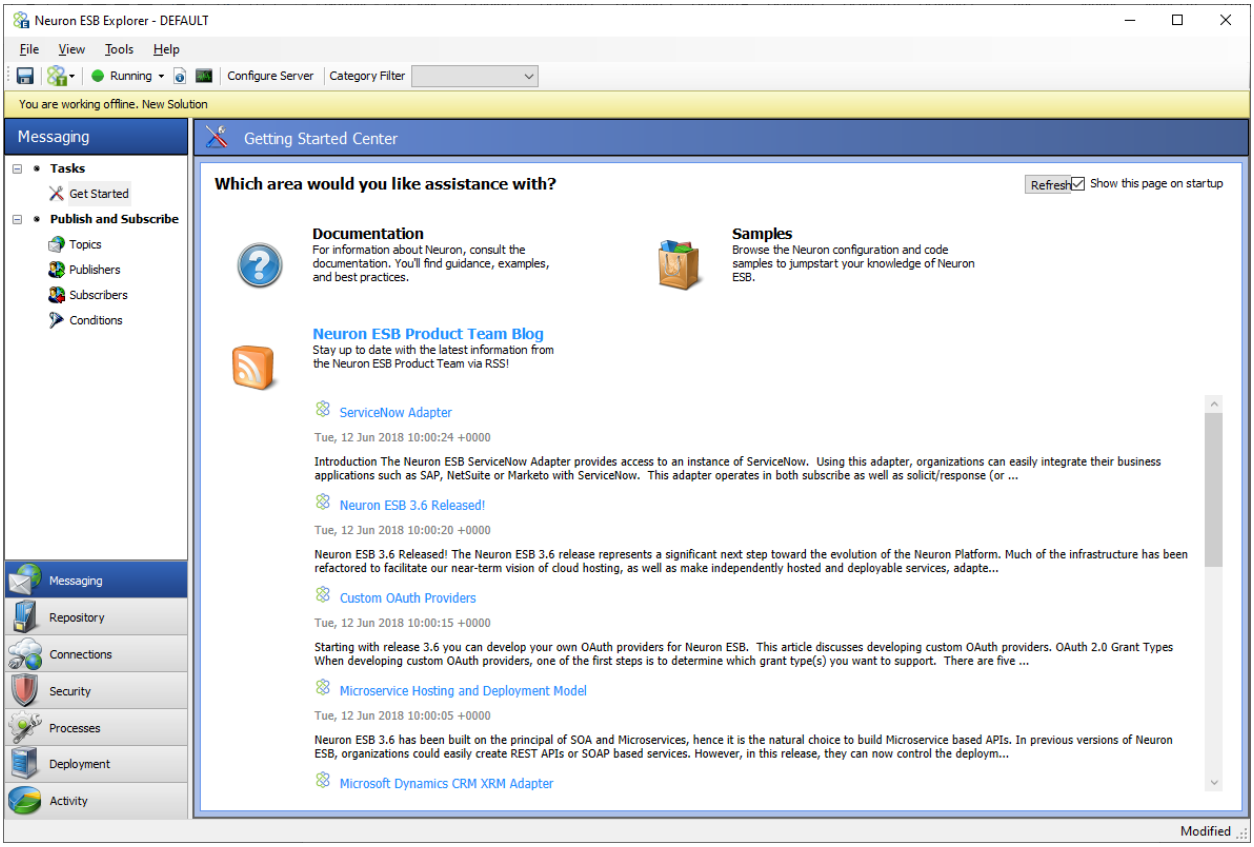
Neuron ESB Configuration

We will now create a Neuron ESB configuration from scratch and configure a Neuron database. If Neuron ESB Explorer is still running, choose the File option on the menu and select Close. If you have closed the Neuron ESB Explorer, then navigate to the shortcut using your Windows Start Menu.

Using either path will bring you to the default view in Neuron ESB Explorer. You should see something like below:

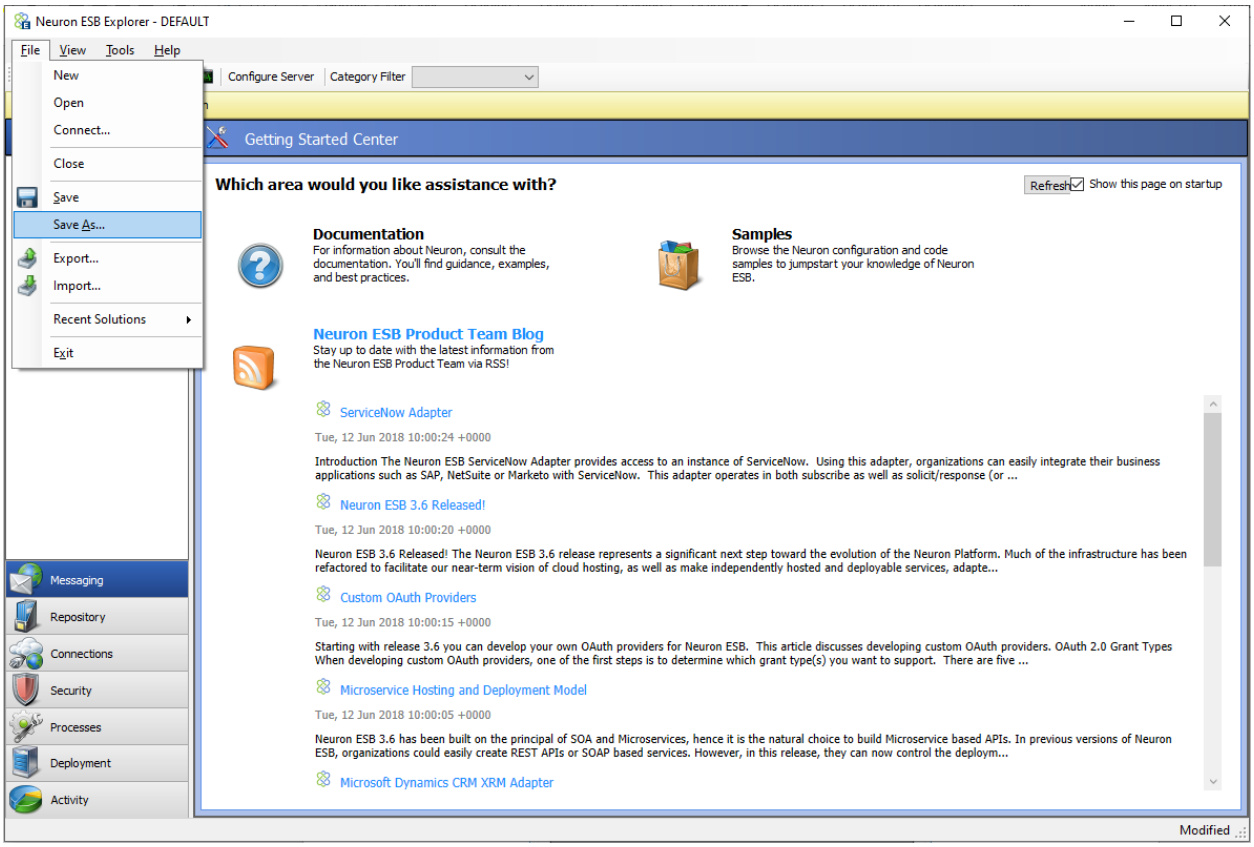


Select File->New and you should see this:

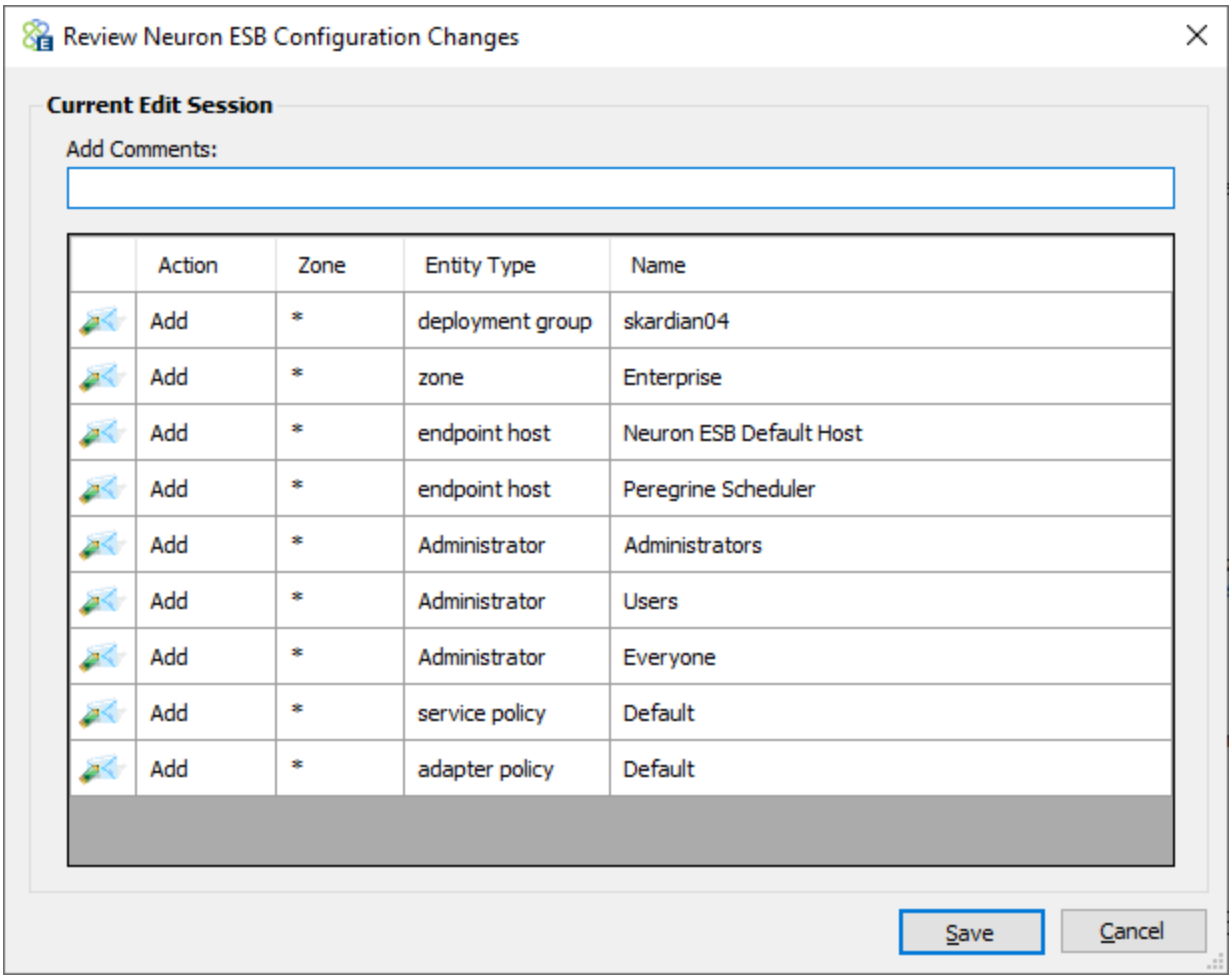


Notice that the Status Bar under the Menu items and above the Navigation Tabs is yellow. Also notice that it says New Solution.

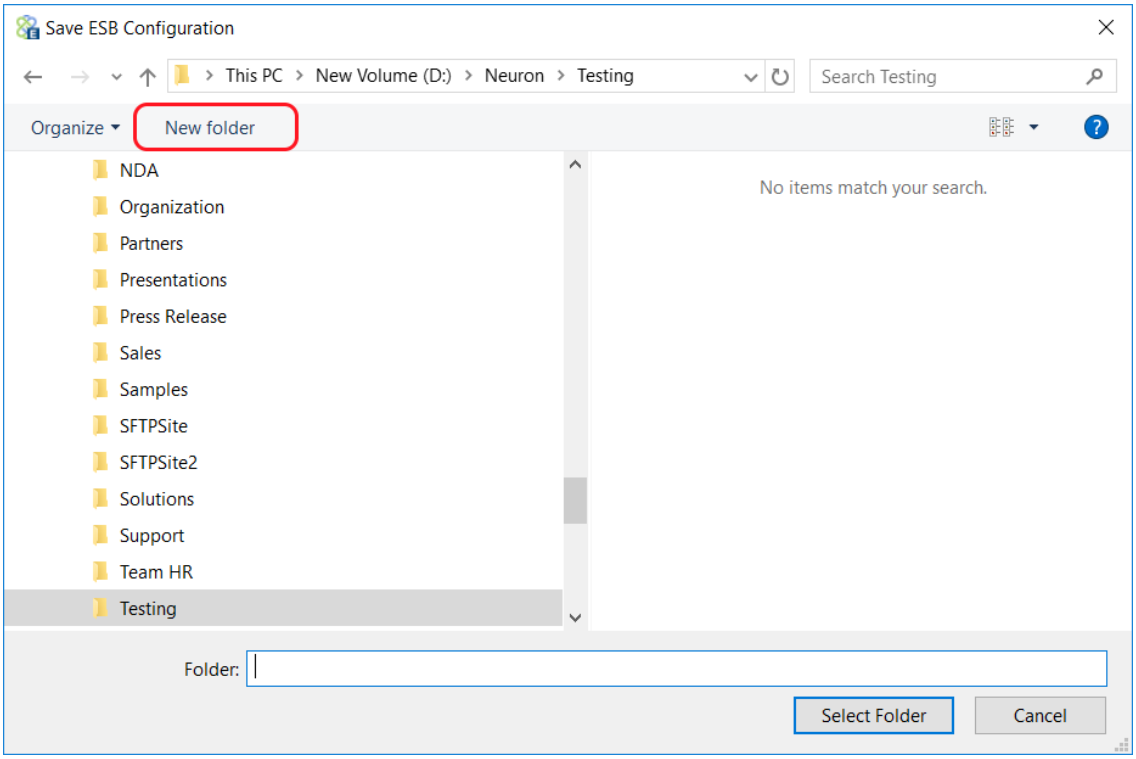
Choose File from the top-level Menu Items and then choose Save As…



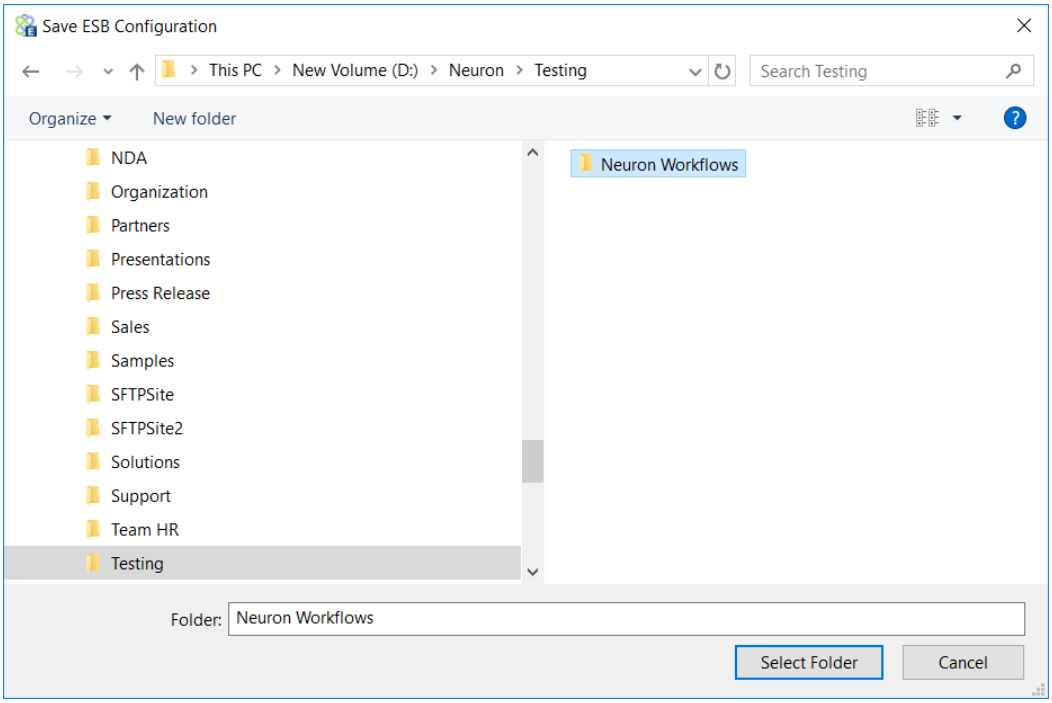
Next you will see a manifest of the changes that were made. Even though you haven't added any entities yet, some are automatically created when you create a new configuration:



Click Save, and a standard Windows File Dialog appears:

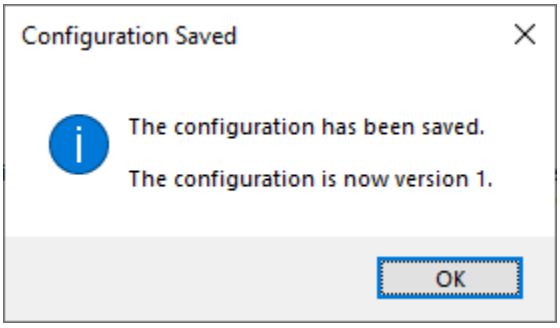


The Neuron ESB configuration is stored in a folder. Prior to saving you need to create the folder that will hold your configuration. Click the New Folder icon in the dialog (outlined in red above) and enter a name for your ESB configuration (i.e. Neuron Workflows).

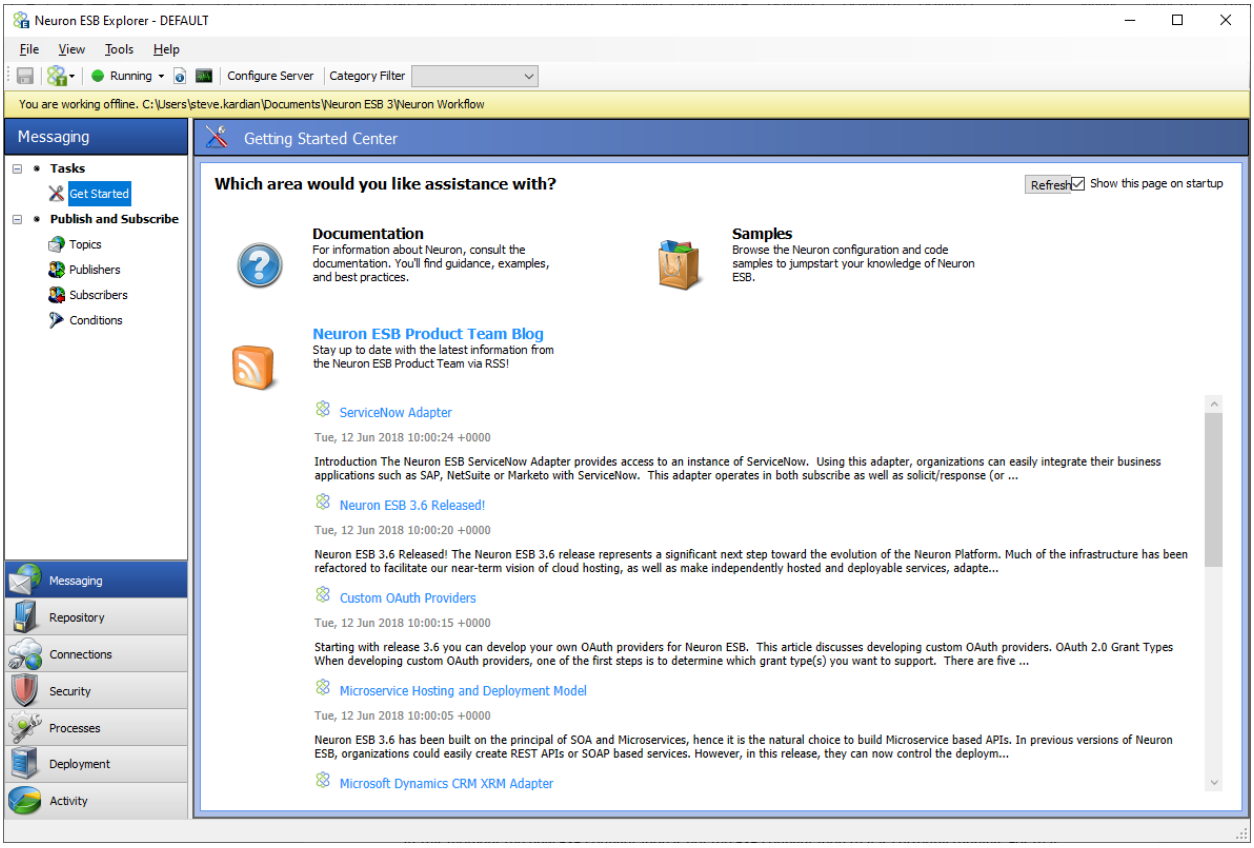


Finally, select the folder and click the Select Folder button.

A prompt is displayed indicating the configuration was saved and which version you are on:



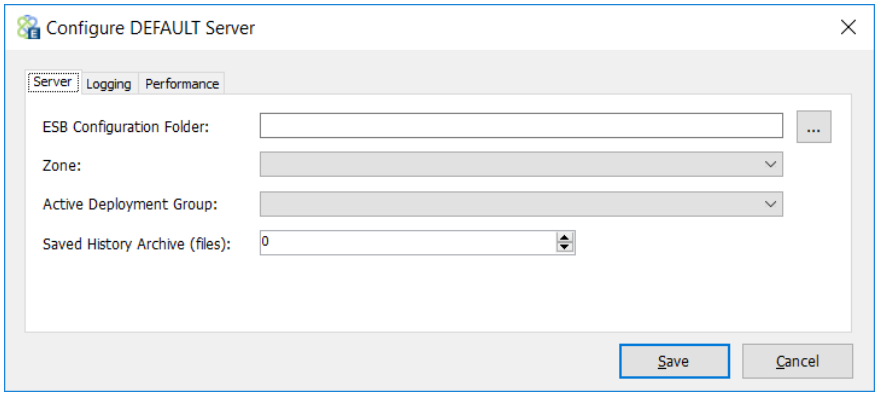
Click OK and you should see something similar to below.



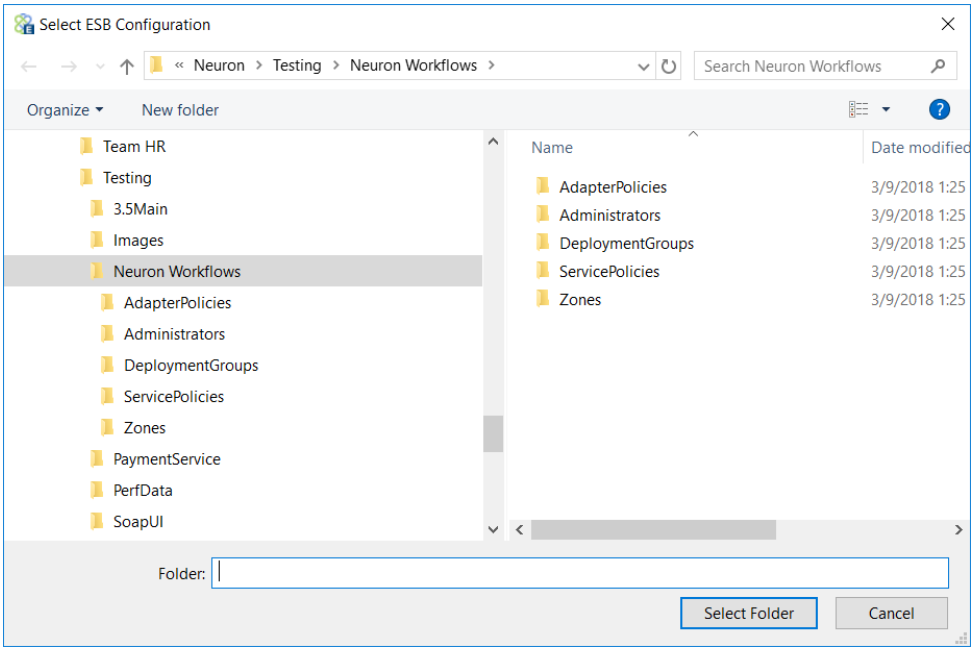
Notice that the path to the ESB configuration now shows in the Status Bar and the color of the bar is still yellow (your path will reflect where you created the “Neuron Workflows” folder).

At this moment the new ESB configuration is not the ESB configuration that is currently running. For that we need to use Configure Server from the Menu Items.

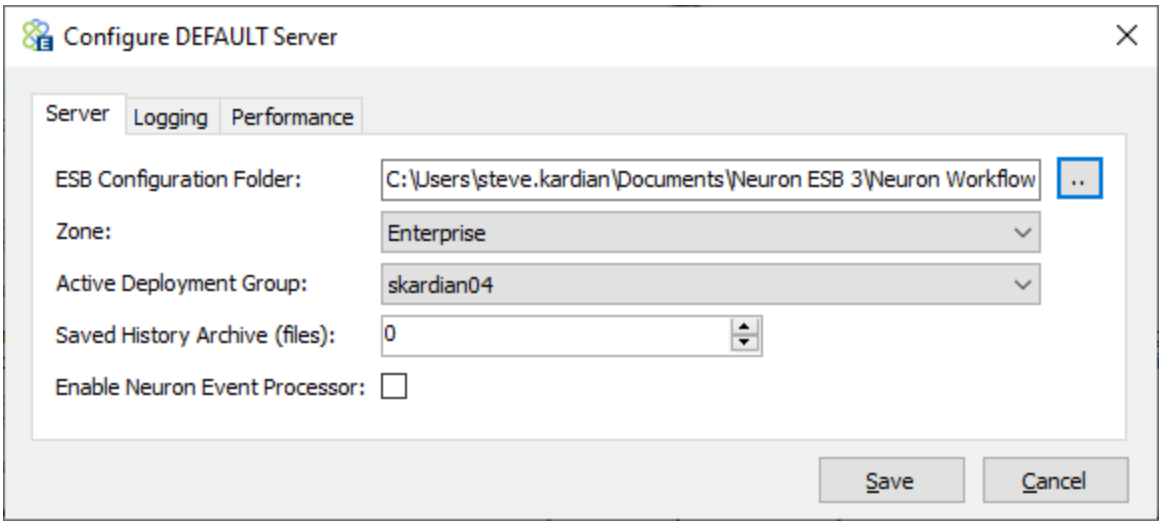
When you click the **Configure Server** button you will see the Dialog like the one depicted below (by default the Neuron ESB Service is not configured to run any configuration):



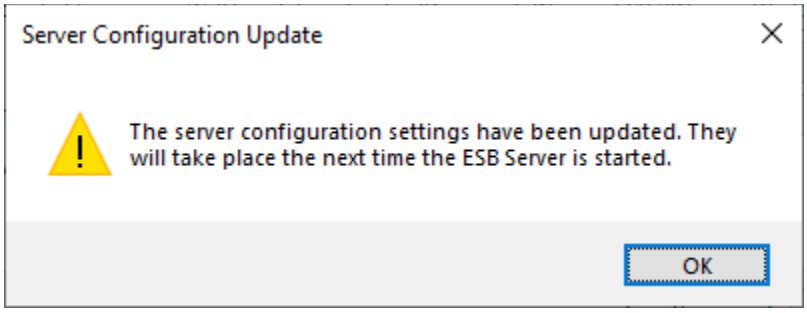
Select the ellipsis next to the ESB Configuration Folder text box to select the folder you previously created and click the Select Folder button:



Note that the Active Deployment Group changes to your machine name. When you create a new configuration, the default active deployment group is the name of your machine:

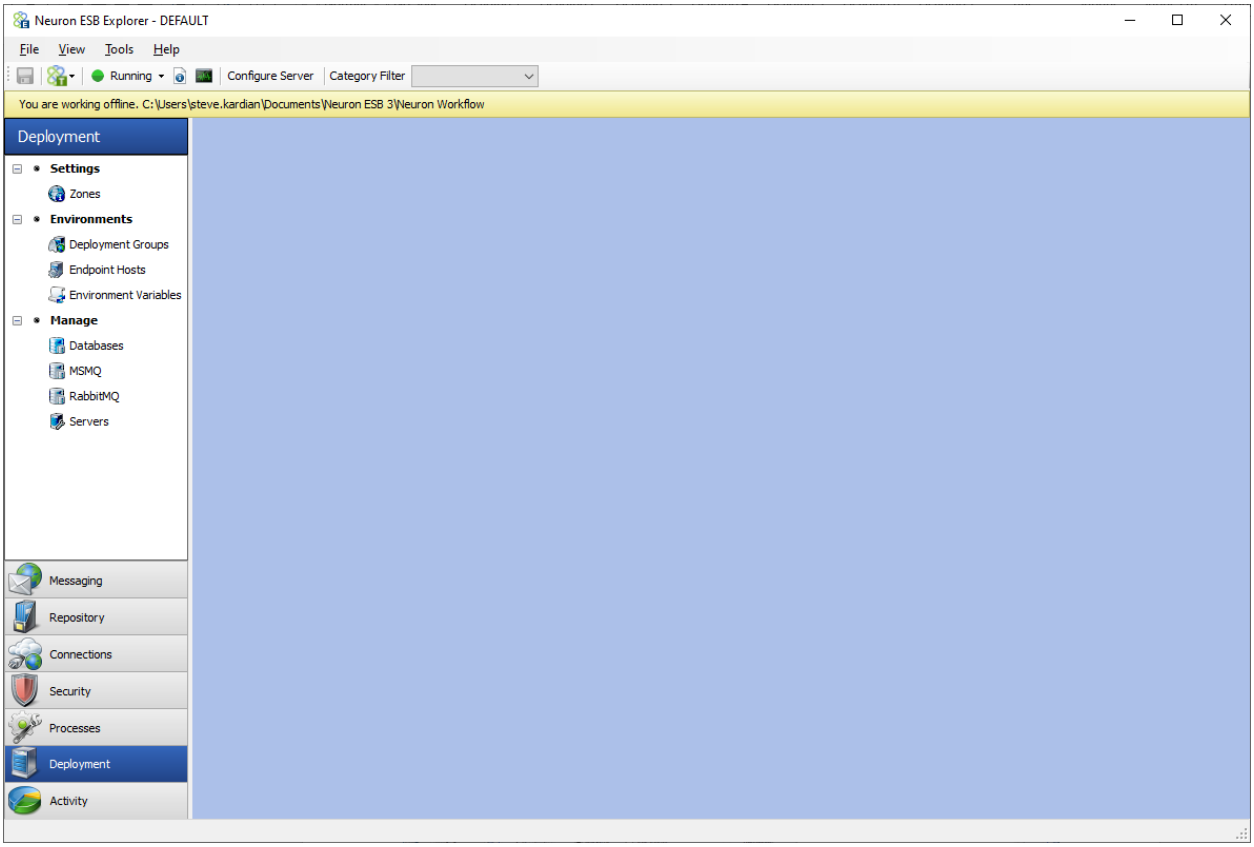


Click Save. When the Server Configuration Update window appears, click OK.

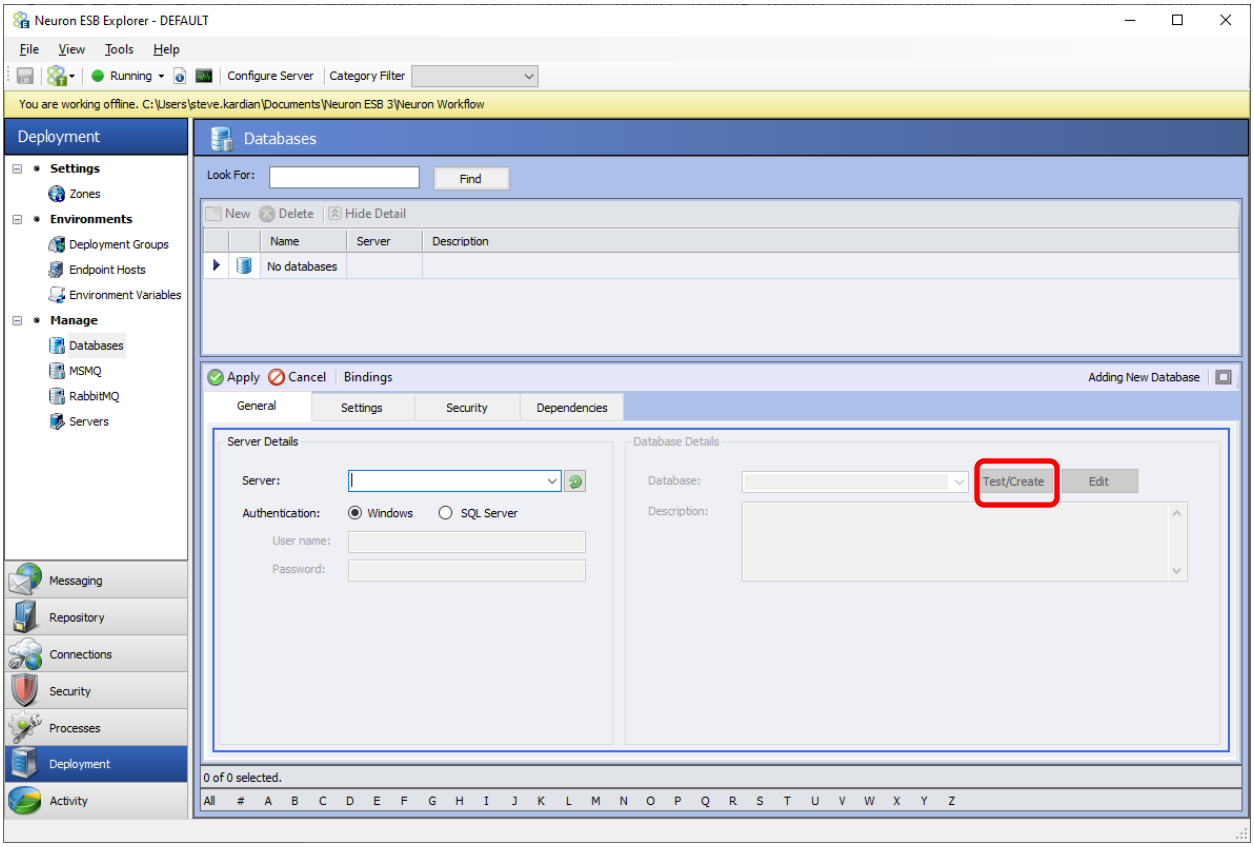


Neuron ESB Database

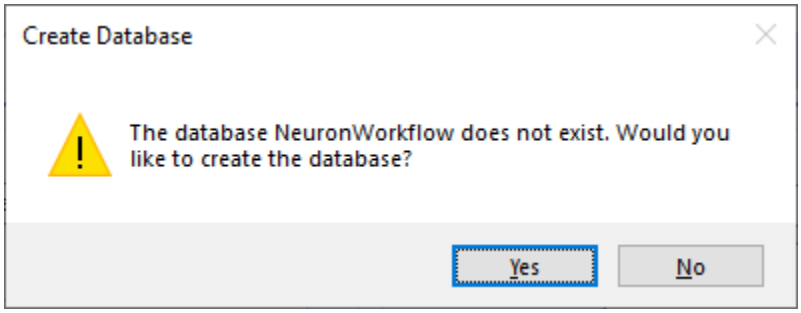
In Neuron ESB Explorer, click on the **Deployment** tab, then click on **Databases**.



Click the **New** button. Set the **Server** property to the location of your SQL Server (use a period or localhost for a local DB). If not using the default instance, make sure you enter the instance name. Set the **Database** property to **NeuronWorkflow** and press the **Test/Create** button.

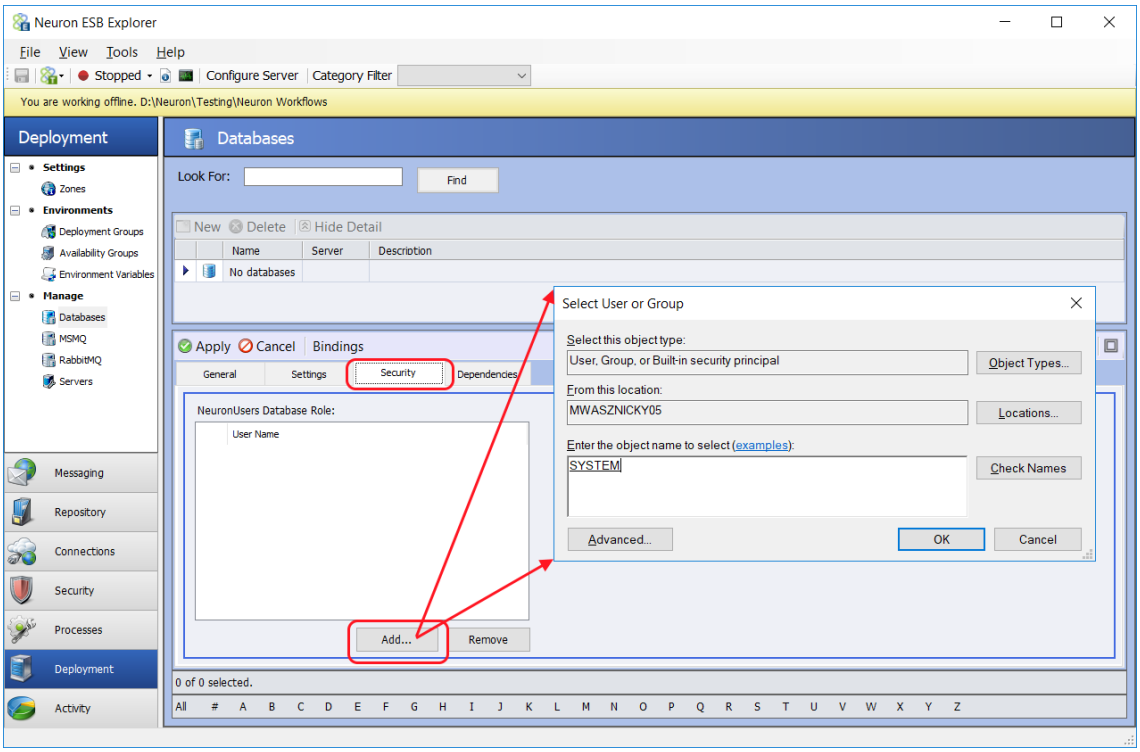


When prompted to create the database, click Yes.



After the database is created, click **OK** on the first prompt, then **Close** on the second prompt.

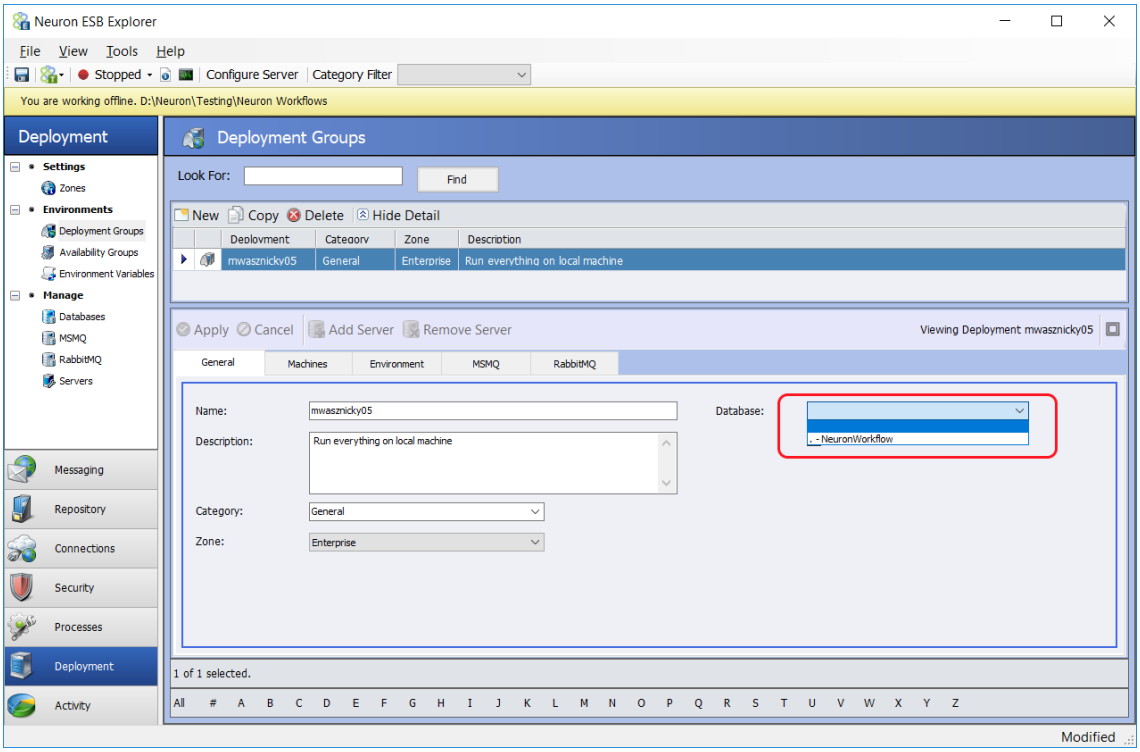
Next, you have to add the Neuron ESB Service account to the **NeuronUsers** Microsoft SQL Server Database Role so that the service has the appropriate permissions to access the Neuron ESB database. To do this, click on the **Security** tab for the database configuration and then click **Add**.



If you used the local system account as the Neuron ESB Service account (when you installed Neuron), then type-in "SYSTEM" in all capitals and click **Check Names**. Otherwise enter the user account that you set the Neuron ESB Service to use. Click **OK**. Click the Apply button, and then click **OK** on the Apply Configuration dialog.

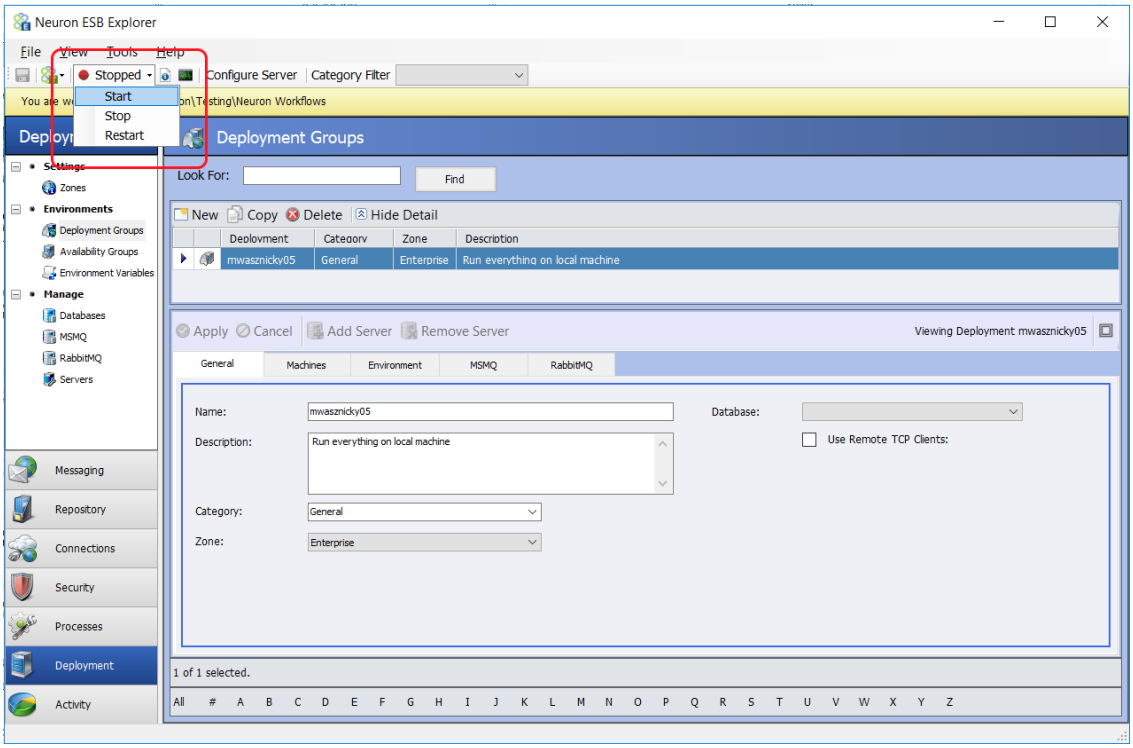
Navigate to **Deployment Groups** and select the only record in the list (the default name is the same as the machine name you created the Neuron ESB configuration on). Select the **NeuronWorkflow** database from the **Database** dropdown list. Click **Apply**.

Information on setting up a Neuron ESB database can be found at
https://www.neuronesb.com/article/kb/resources/



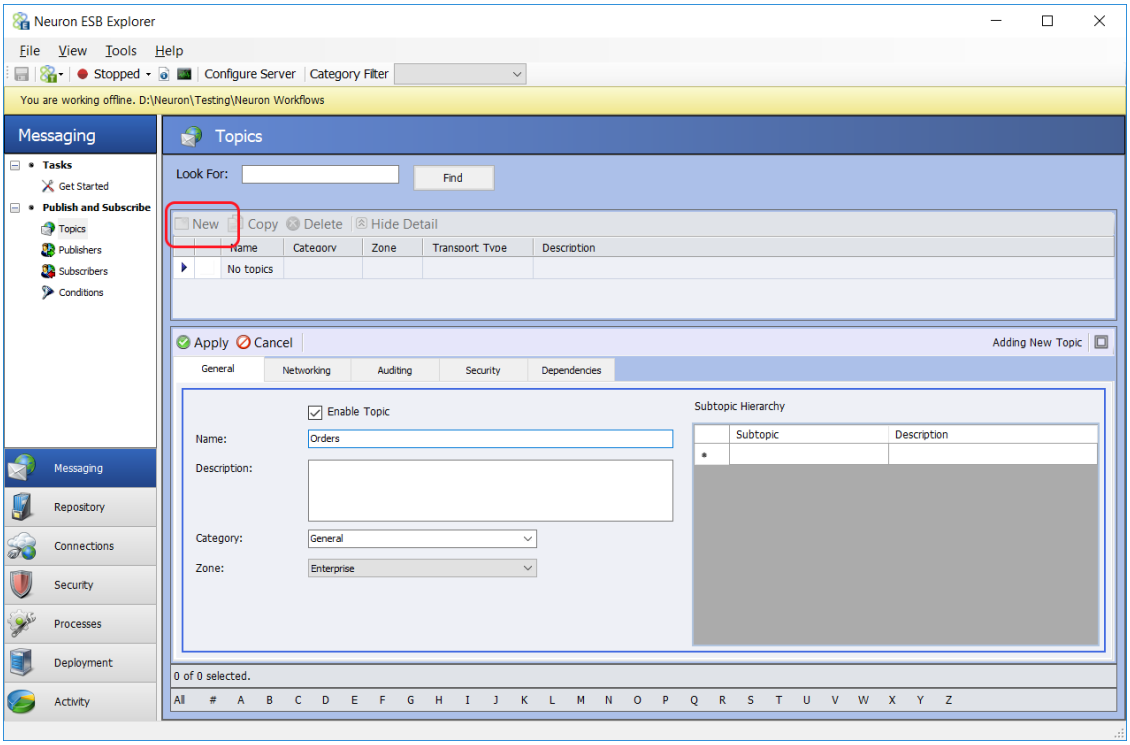
Save the configuration by click File->Save. Click **Save** on review dialog, and then click **OK**.

Use the Neuron ESB Explorer's "Service Control" Toolbar Menu item to Start or Restart the Neuron ESB Windows Service. This will become necessary before we run or test workflows.



Add a Topic, Publisher and Subscriber

Next, we will add a Topic and 2 Parties. Navigate to the **Messaging** tab and choose **Topics**. Click on the **New** button. Rename the topic to **Orders**.



Press the **Apply** button.

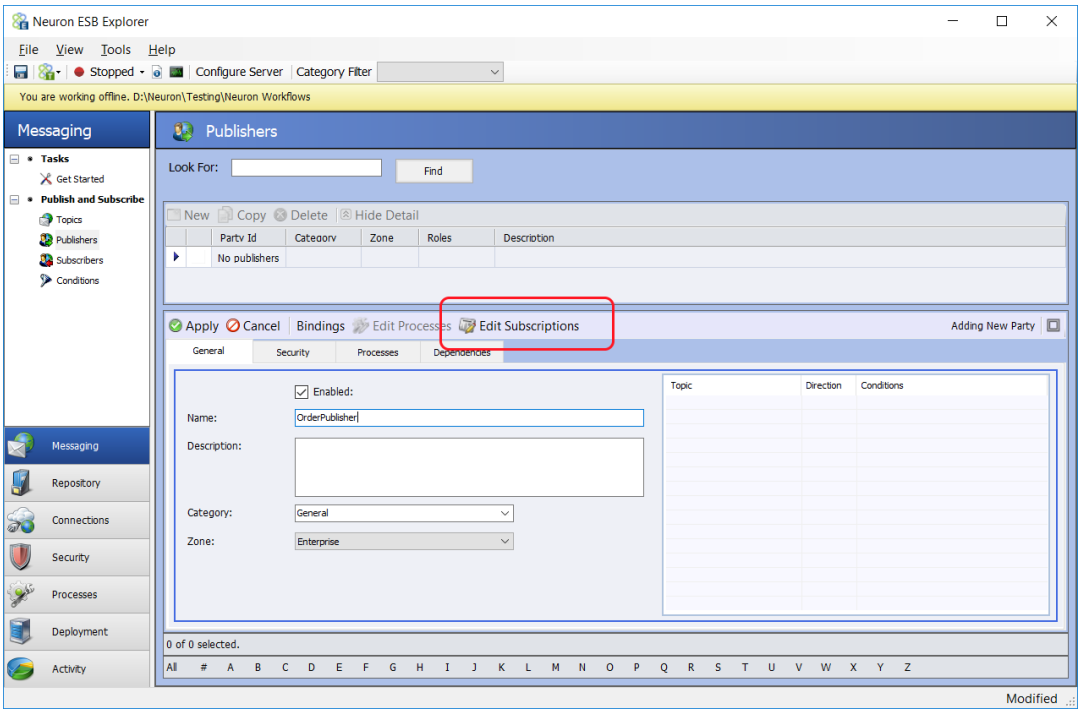
While this tutorial uses TCP as the underlying transport for topics, this is done to ensure that all users, regardless of whether they have installed a message queue on their system, can complete the tutorial. However, it is the recommendation of the Neuron ESB team that whenever you are creating topics for use with workflow you always use a message queue, such as RabbitMQ or MSMQ, as the underlying transport,

Workflow endpoints also allows organizations to set a limit on the number of concurrent instances that are running. By default, the limit is 100 instances. While some environments are capable of handling this or even more, others are not so robust, and this limit might need to be changed

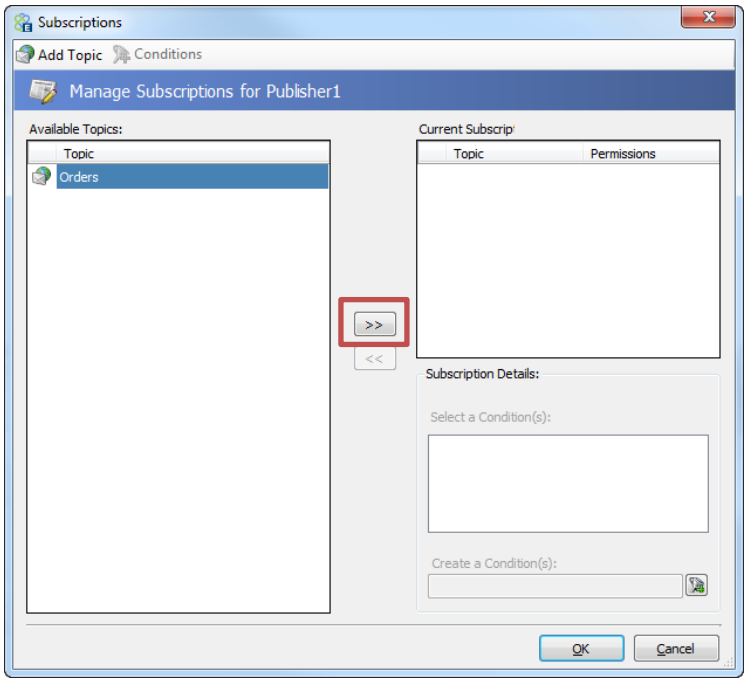
---

Click the **Publishers** option under **Topics**.

Click the **New** button. Change the name to **OrderPublisher** and click the **Edit Subscriptions** button in the editor toolbar:



Select **Orders** in the dialog and click the arrow button that points to the right (outlined in red below):



A Party can be locked down to only Send or Receive. When you create a new Publisher the default permission is Send. For now, leave the default value and press **OK** and then press **Apply**.

Click the **Subscribers** option under **Topics**.

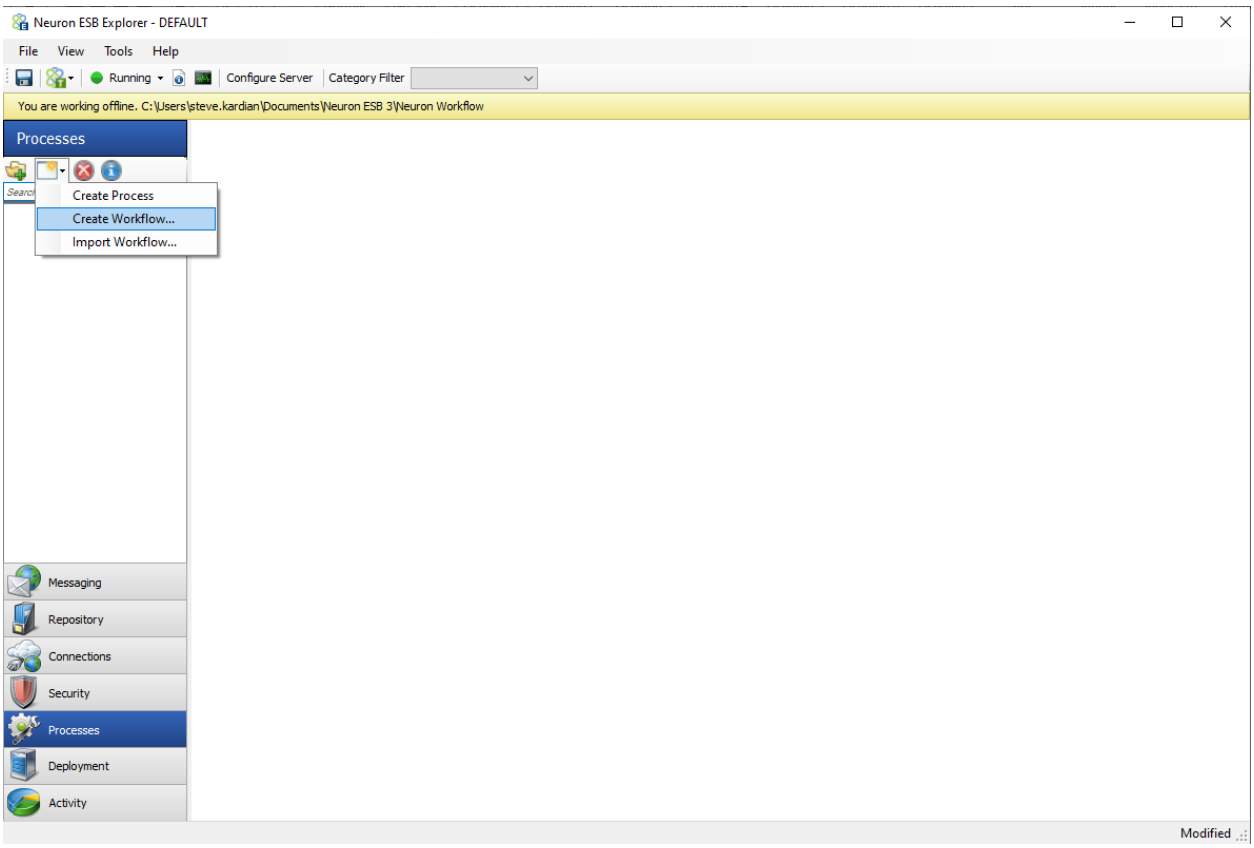
Click the **New** button. Change the name to **OrderSubscriber** and click the **Edit Subscriptions** button in the editor toolbar. Follow the steps above to add a subscription to the **Orders** topic. Notice that when you add the subscription to Orders, the permissions for a subscriber default to “Receive”. Leave the default value and press **OK** and the press **Apply**.

Save the configuration, File->Save.

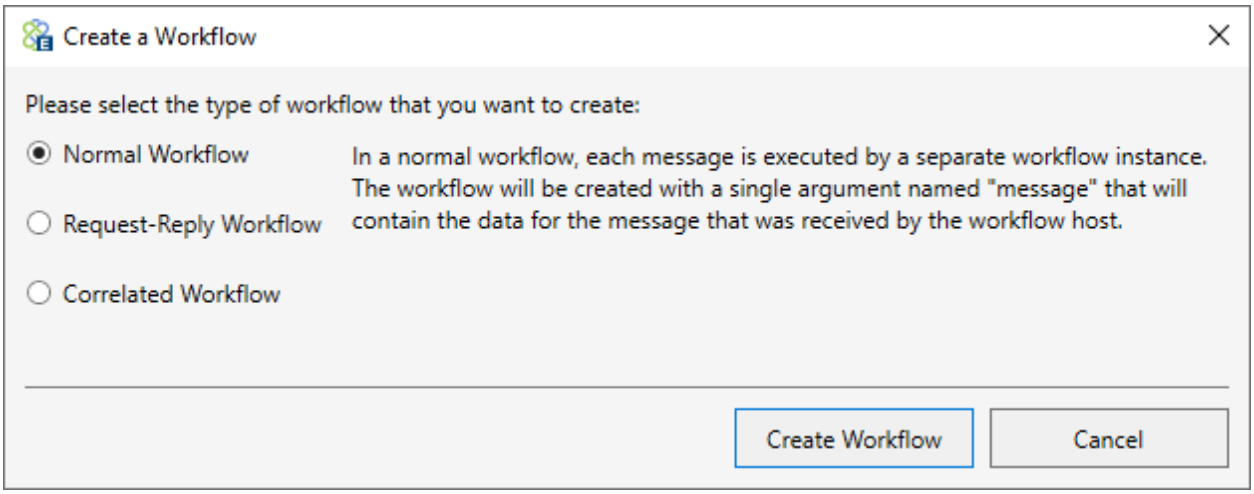
Exercise – Create a Normal Workflow

Workflow Definition

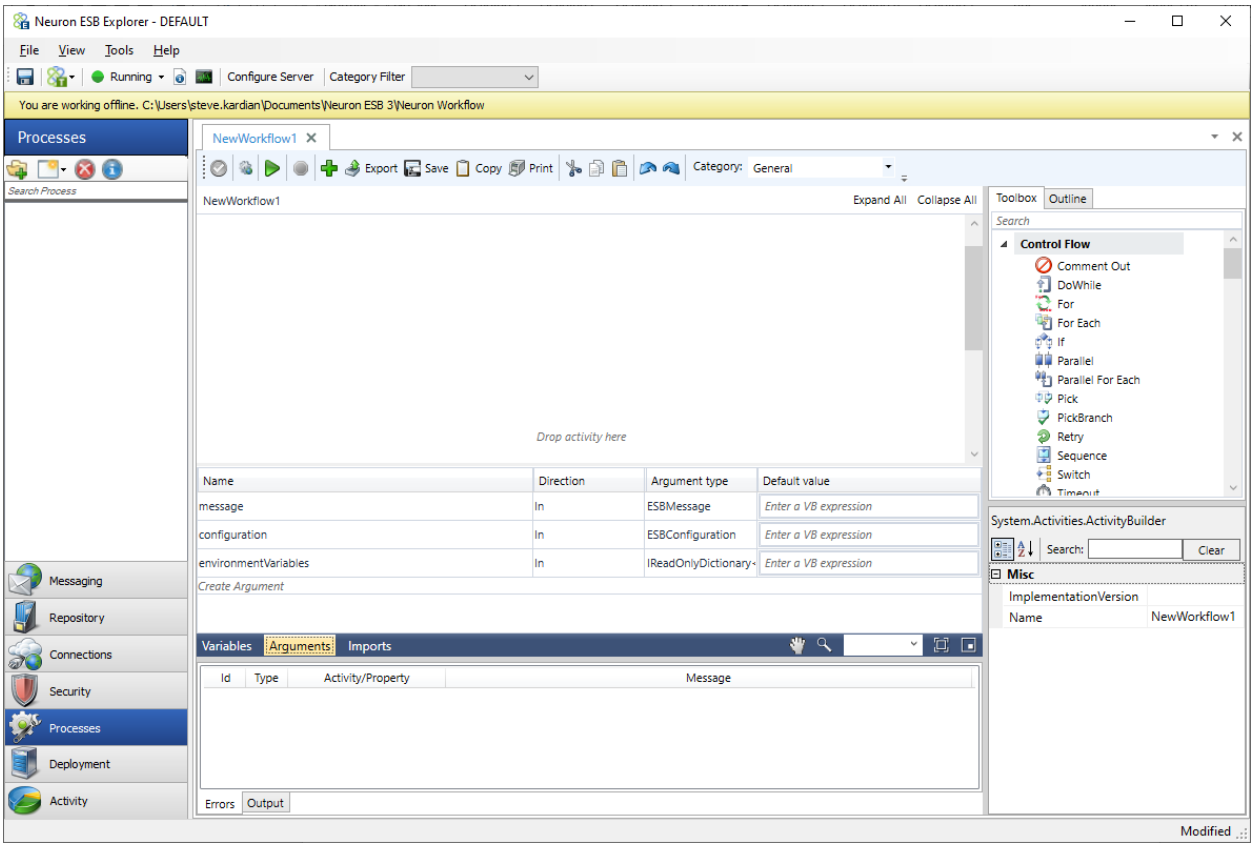
Workflow definitions are created with the workflow designer. To create a workflow, click on the **Processes** tab and then click the **New** button. Select **Create Workflow…**



Select the **Normal Workflow** option in the Create a Workflow dialog.

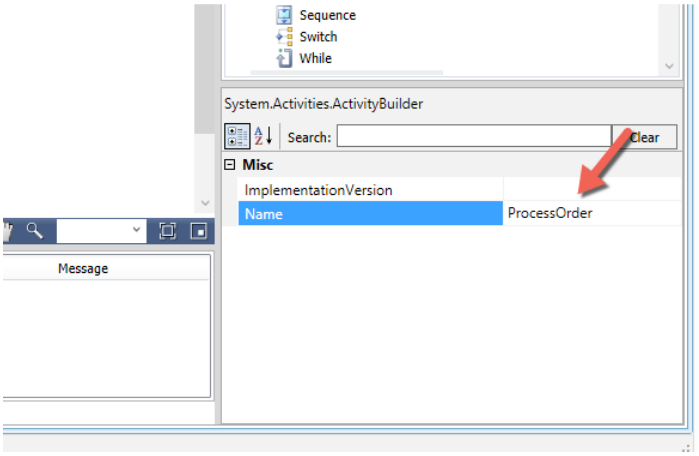


The Workflow Designer with an empty workflow will be displayed. Click on the **Arguments** tab in the Workflow Designer to see the input arguments that are created with a Normal Workflow. The argument *message* will contain the message that caused the workflow to be invoked. The argument *configuration* contains a reference to the entire Neuron ESB Configuration and the argument *environmentVariables* contains a collection of the current values for each environment variable defined in the configuration. There are no output arguments because a Normal Workflow does not return a response message.

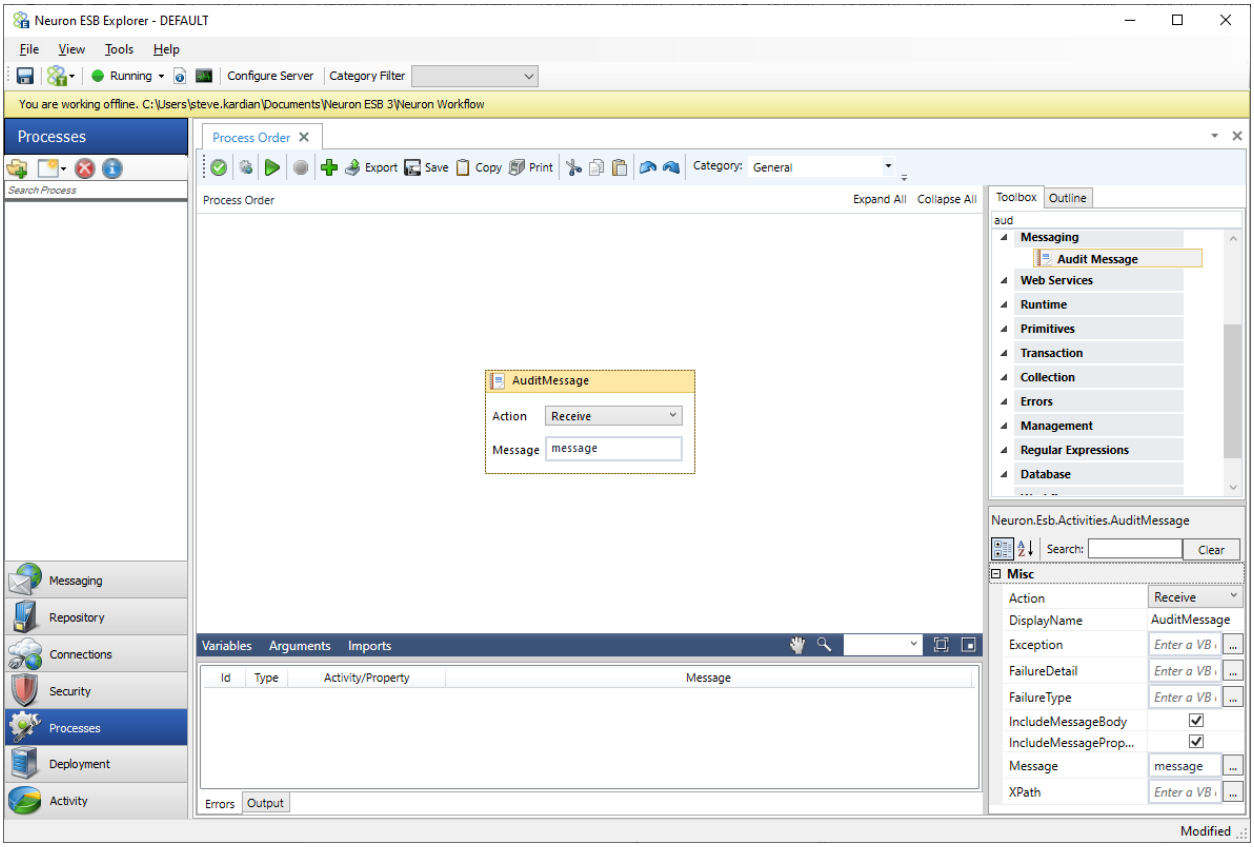


For this exercise you will create a simple workflow that audits the incoming message and then writes the contents of the message to the Neuron ESB log files.

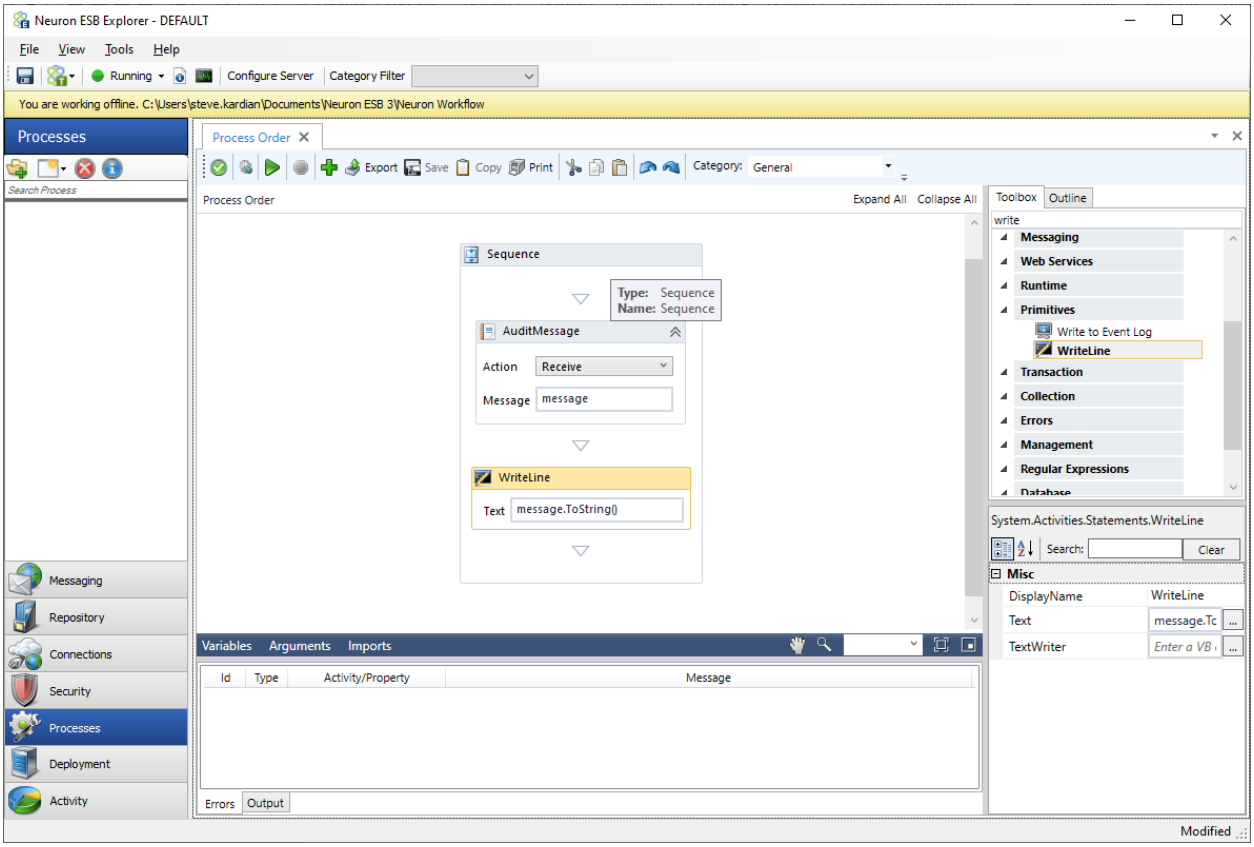
First, click anywhere in the Workflow Designer to show the workflow properties **Implementation Version** and **Name**. Set the **Name** property to *ProcessOrder*.



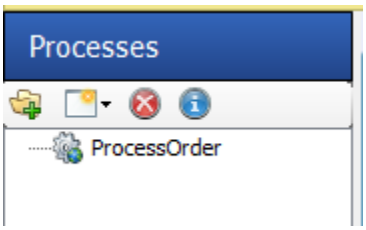
Next, search for the **Audit Message** activity in the toolbox by either scrolling through the list of activities or by using the search box at the top of the toolbox and typing in in “audit”. Drag the **Audit Message** activity to the designer surface. Set the **Action** property to *Receive* and the type in *message* for the **Message** property.



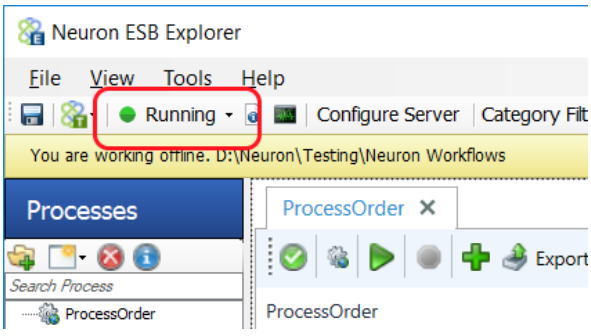
Second, search for the **WriteLine** activity and drag it to the designer just below the **Audit Message** activity. Type in *message.ToString()* for the **Text** property.



Click the **Apply** button, outlined in Red above. The workflow will appear in the list to the left of the designer:



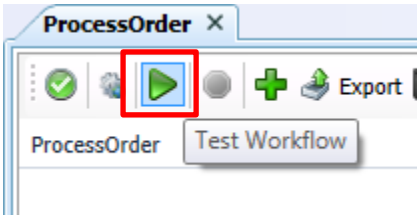
Before moving forward, ensure that the Neuron ESB Windows Service is started by using the Neuron ESB Explorer's "Service Control" Toolbar Menu item to Start the Neuron ESB Windows Service. It should be in a "Running" state with a green indicator as shown below.



Test the Workflow in the Workflow Designer

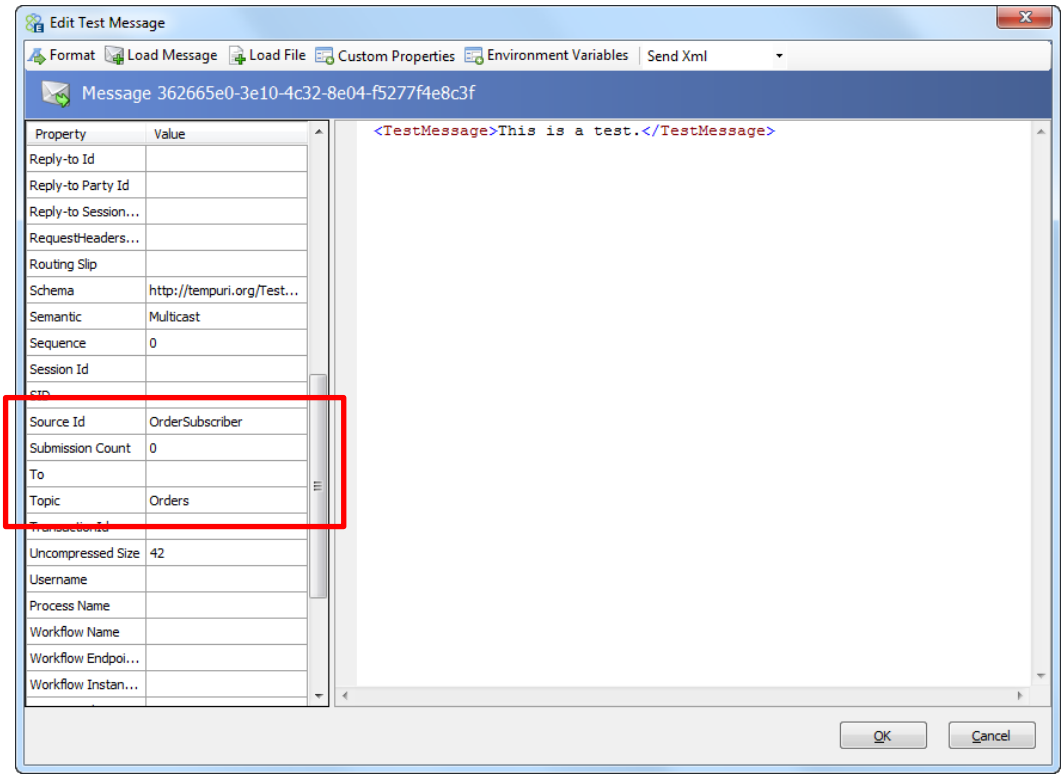
You don't have to deploy a workflow in order to test it. The Workflow Designer includes a testing utility that will step through each activity and display any errors or output generated by your workflow.

To test using the Workflow Designer, click on the **Test Workflow** button on the toolbar:

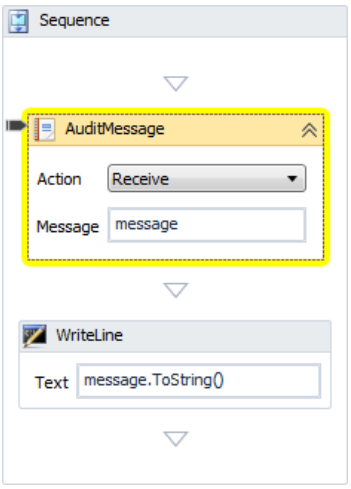


The Edit Test Message dialog will appear. This is where you will set the content of the message that would start an instance of your workflow. You can also set any message header properties, custom message properties, and environment variables that would be use at runtime.

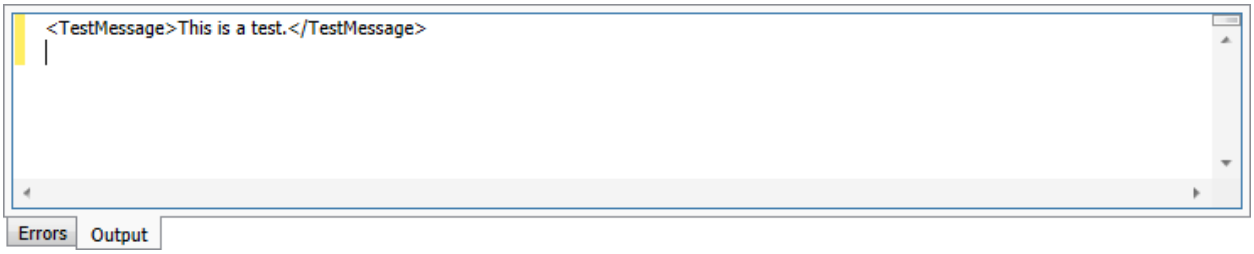
For this test you do not need to change the message body, but you do need to change the header properties **Source Id** and **Topic**. This is because the **AuditMessage** activity requires these values in order to audit the message to the Neuron database. Set the **Source Id** to *OrderSubscriber* and the **Topic** to *Orders*:



Click the **OK** button to start the test. Each activity will be highlighted as it is executed:



At the bottom of the workflow designer is an Output window that displays the output of the **WriteLine** activity. There is also an Errors tab that displays any errors that were encountered while testing the workflow:

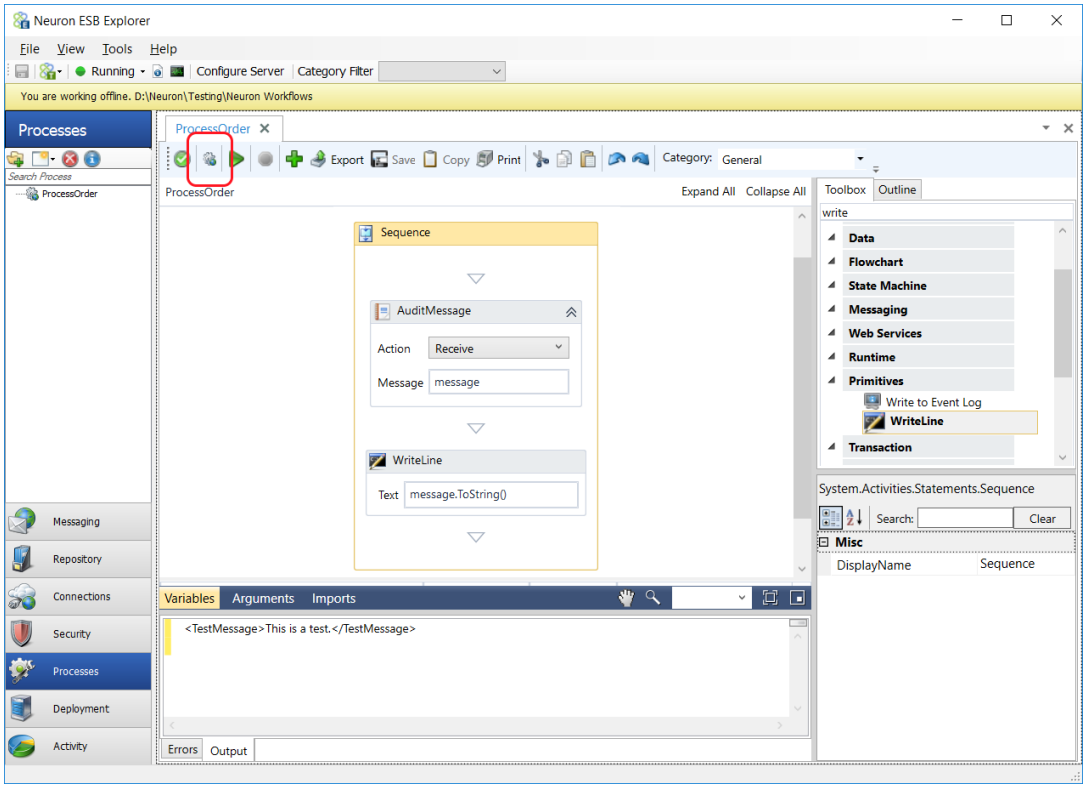


After the test is complete, navigate to *Activity->Database Reports->Message History* within the Neuron ESB Explorer and click the *Run Report* button. The message used to test the workflow should be logged. The message can be opened within the Message Viewer by double-clicking on the row or selecting *view message* from the right click mouse context menu.

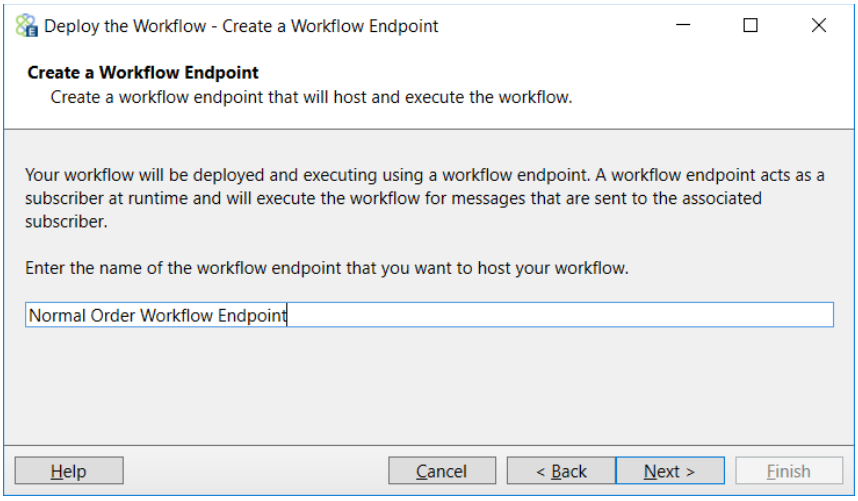
Deploy the Workflow

There is a wizard built into the Workflow Designer that allows you to deploy a workflow definition and endpoint to an Endpoint Host. These steps could be done manually, but we will use the wizard as part of this exercise.

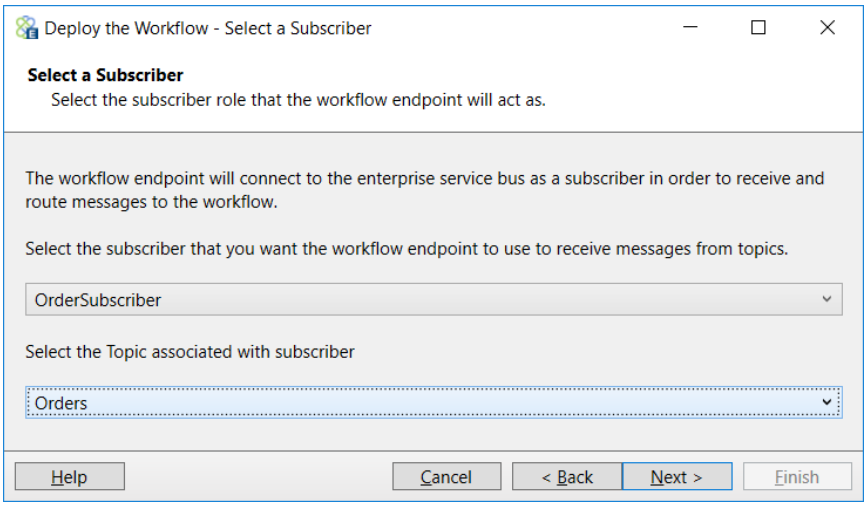
To start the deployment wizard, click on the **Deploy Workflow** button on the workflow designer toolbar.



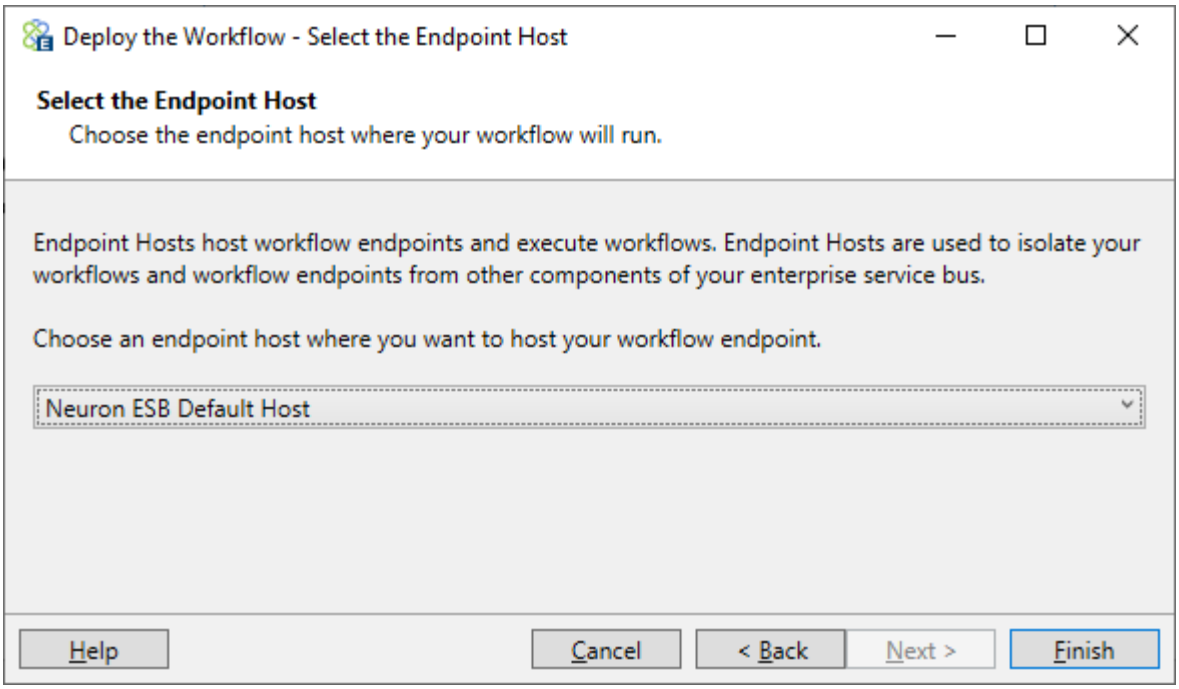
On the **Deploy Your Workflow** page, click **Next**.



On the **Create a Workflow Endpoint** page, enter the name of the Workflow Endpoint you want to create. For this exercise, enter *Normal Order Workflow Endpoint*. Click **Next**.



On the **Select a Subscriber** page, select *OrderSubscriber* for the subscriber and *Orders* as the Topic. Click **Next**.

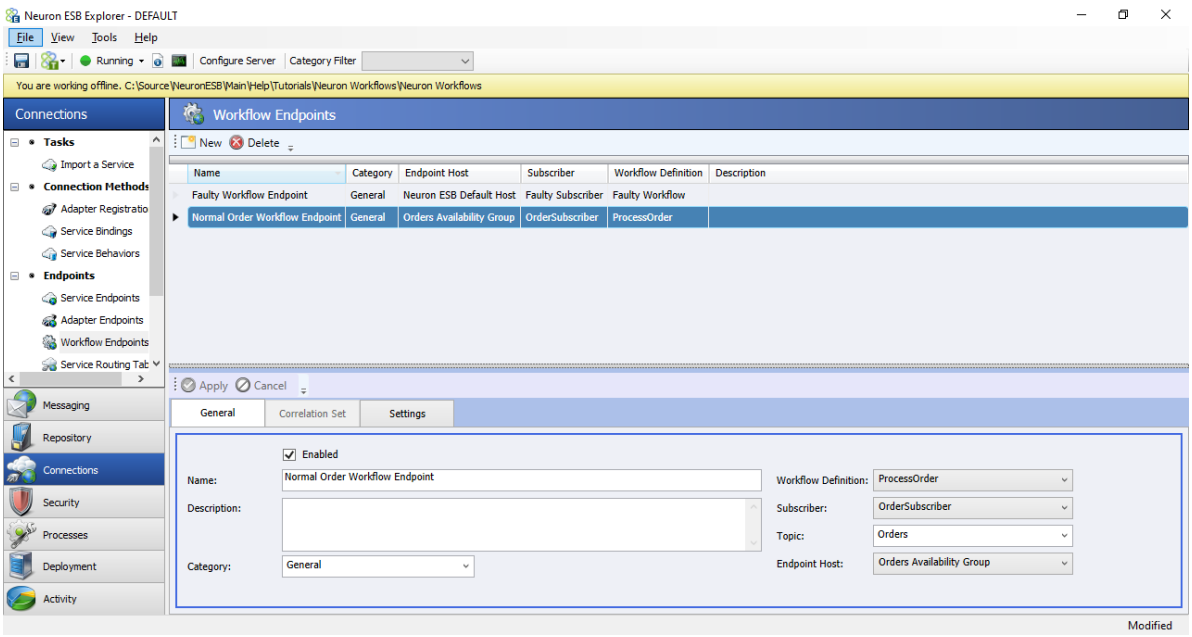


On the **Select the Endpoint Host** page select Neuron ESB Default Host, click the **Finish** button.

Save the configuration by clicking File->Save.

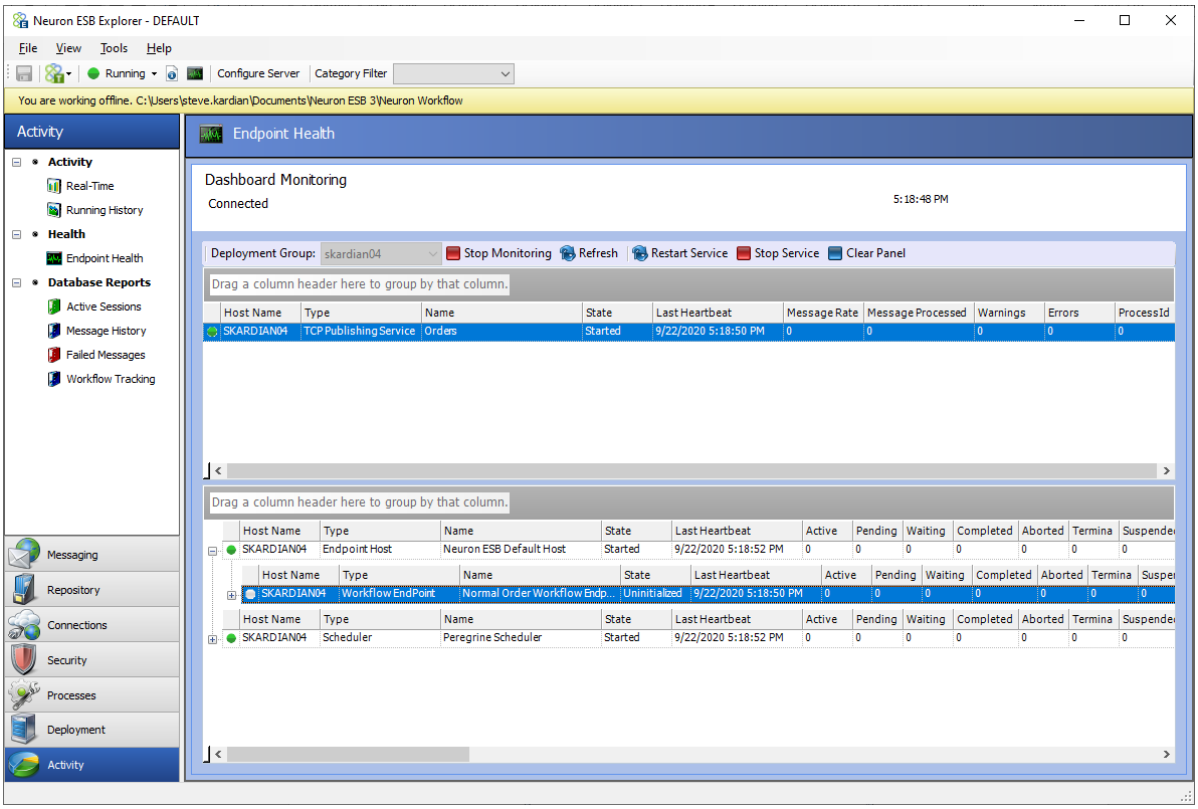
View the Workflow Endpoint

When you deployed the Workflow Definition the Workflow Endpoint was created. To see the Workflow Endpoint, navigate to the **Connections** tab and then select **Workflow Endpoints**. Select the endpoint you just deployed, **Normal Order Workflow Endpoint**. View the selected Workflow Definition, Subscriber, Topic and Endpoint Host. This is the piece of configuration that ties it all together.



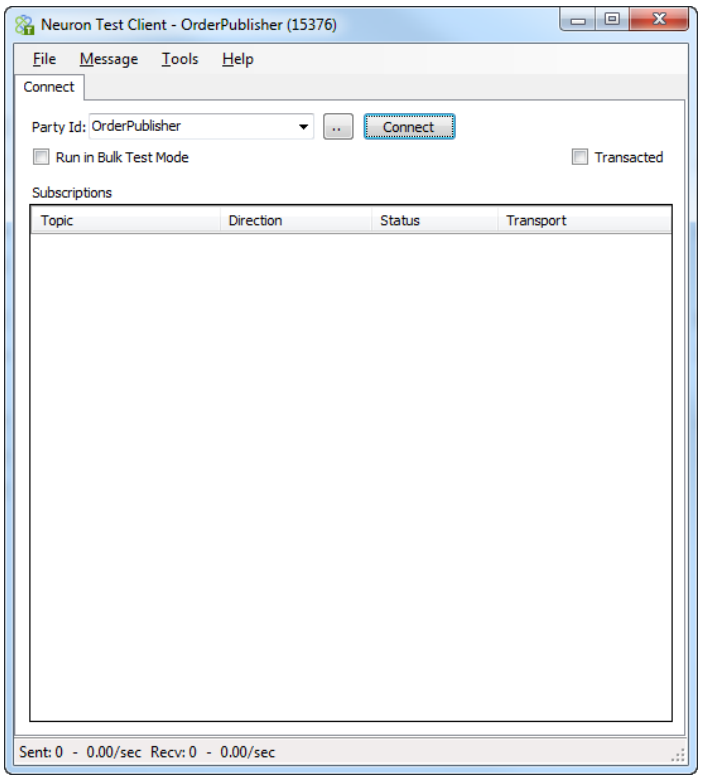
Check the Workflow Status in Endpoint Health

If your Neuron ESB Service is configured to host the solution created in this sample and is currently running, the Workflow Endpoint you just deployed should be in a started state. To verify the Workflow Endpoint is started, navigate to the **Activity** tab and then select **Endpoint Health**. Click the **Start Monitoring** button. The Endpoint Hosts are listed in the lower pane. Expand the Endpoint Host to see the Workflow Endpoints associated with that group. You should see the Workflow Endpoint you deployed with a green status and a state of "Started". If the Workflow Endpoint has a red status or is stopped, look at the Neuron ESB v3 event log for any errors that might indicate why it is stopped. You can also look in the Neuron application logs located at <Program Files>\Neudesic\Neuron ESB v3\logs. There will be individual log files for every endpoint.

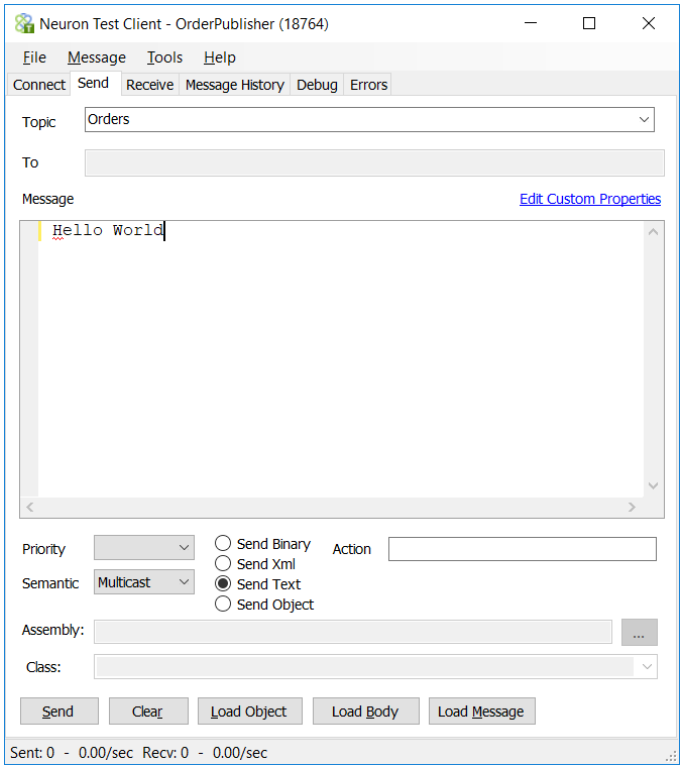


Run the Workflow

To run this workflow, open a Neuron ESB Test Client by clicking Tools->Test Client->1 Test Client from the Neuron ESB Explorer. One instance of the test client will open.

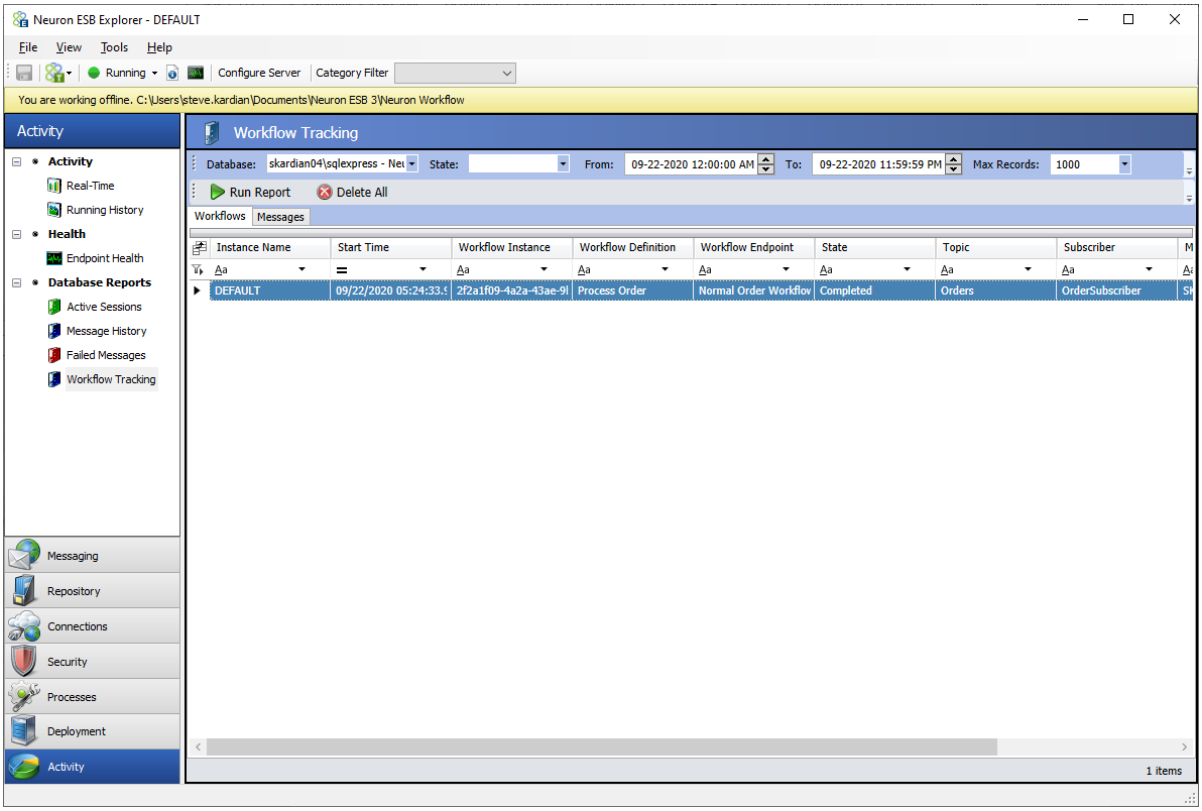


Select **OrderPublisher** as the Party Id and click the **Connect** button.



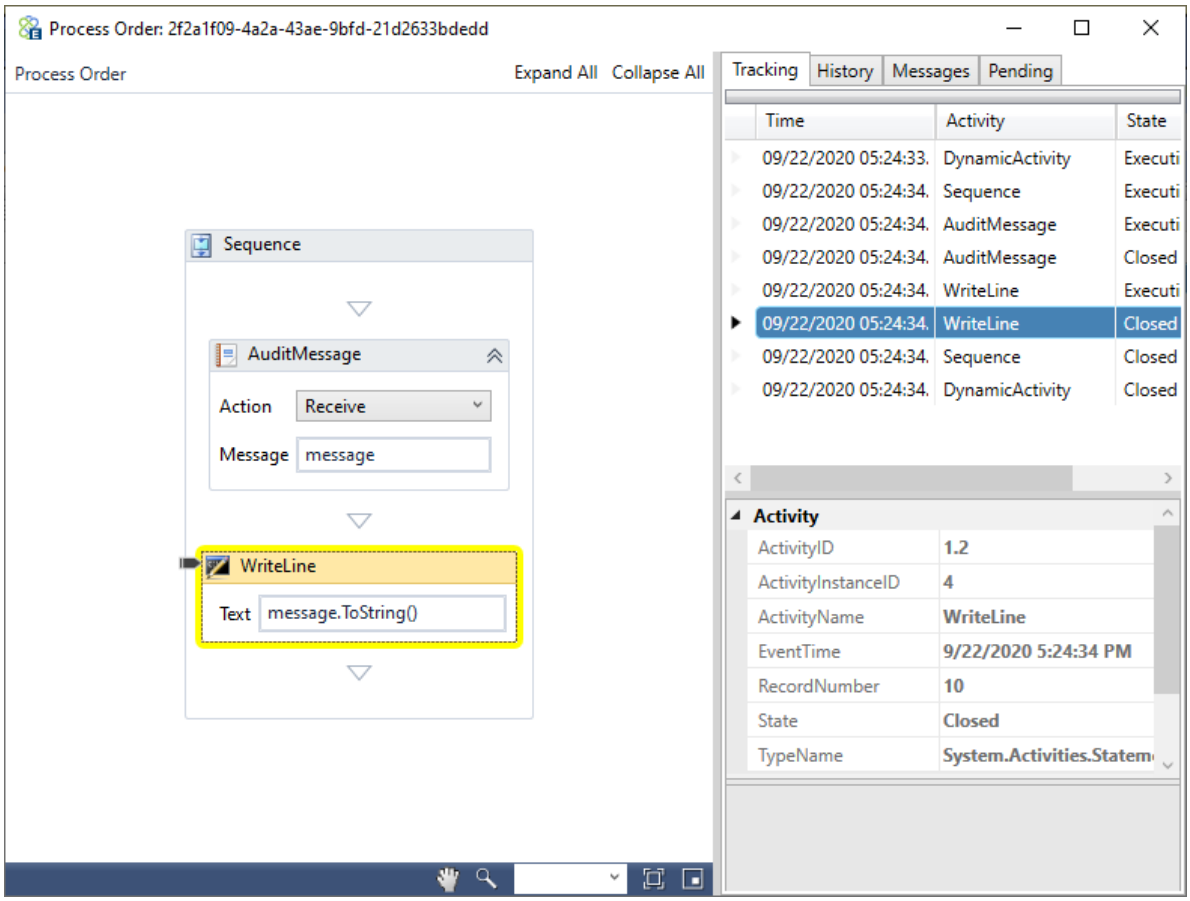
Click the **Send** tab and type a message into the text box. The content of this message doesn't matter for this example – the workflow just audits the incoming message and logs the contents to the Neuron application logs. Click the **Send** button.

To see if the workflow completed, navigate to the **Activity** tab and select **Workflow Tracking**. Click **Run Report** to see a list of the completed workflow instances:



Because we created a Normal Workflow, there will be one instance of the workflow for each message you publish to the Orders topic. View the state of the workflow instance to see if it "Completed" (outlined in red above).

The final workflow instance state diagram and event history can be displayed by using the mouse and double-clicking on the tracking record above to display the window shown below.



Navigating to **Endpoint Health** and clicking the **Start Monitoring** button, you can see how many workflow instances have been executed. The Endpoint Health monitor will display how many instances of the workflow are Active, Pending, Waiting, Completed or Aborted.

Exercise – Create a Long Running Transaction

A long-running transaction is a workflow that can take minutes, hours or even days to complete. Long-running transactions are triggered where there is a lag of time between operations performed in the workflow. For example, the workflow may send a file containing an order to an FTP site where an external partner will process it. Once the partner has processed the order, they will send back a file that contains an order acknowledgement. This exchange may take hours to complete.

For this exercise you are going to modify the *ProcessOrder* workflow created in the previous exercise and turn it into a long-running transaction by adding a correlated send/receive to it. A correlated send/receive requires these items:

- Correlation ID
- Correlation Set
- Publish Message activity
- Receive Message activity

The **Correlation ID** is a variable that contains the value that will be used for routing incoming messages to the correct instance of the workflow. This variable needs to be defined in the workflow and is used in the CorrelationID property on the Publish Message activity. You do not need to set the value of this variable; that will be handled by the Publish Message activity based on its Correlation Set.

The **Correlation Set** is used to initialize the value of the Correlation ID variable. A Correlation Set is composed of any combination of message content, custom header, SOAP or HTTP header or even regex and XPATH expressions against the body of the message that can be AND'd and OR'd together to create an expression when evaluated at runtime, can determine the uniqueness of a message or a set of messages.

For example, the Correlation Set could specify that the value used as the Correlation ID will be found in a custom message property called order.orderId. The developer is responsible for making sure the value exists in the location specified. If the custom message property order.orderId is specified, then that property needs to be created on the message prior to publishing it to the bus. Also note that the Correlation ID value needs to exist in the same location for both the published message as well as the incoming message. In our example above, that would mean that order.orderId would have to be created on both messages.

The **Publish Message** activity will publish the one-way outgoing message, initialize the Correlation Set, and generate a Correlation ID.

The **Receive Message** activity follows the Publish Message activity and participates in the Correlation Set by setting the Correlation ID property to the Correlation ID value returned from the Publish Message activity. Neuron ESB will automatically route messages with the correct Correlation Set value to the workflow instance.

NOTE: *The* ***Message Correlation Scope*** *Workflow Activity can be used in place of the* ***Publish Message*** *Workflow Activity to initialize a Correlation Set without the need to publish a message beforehand.*

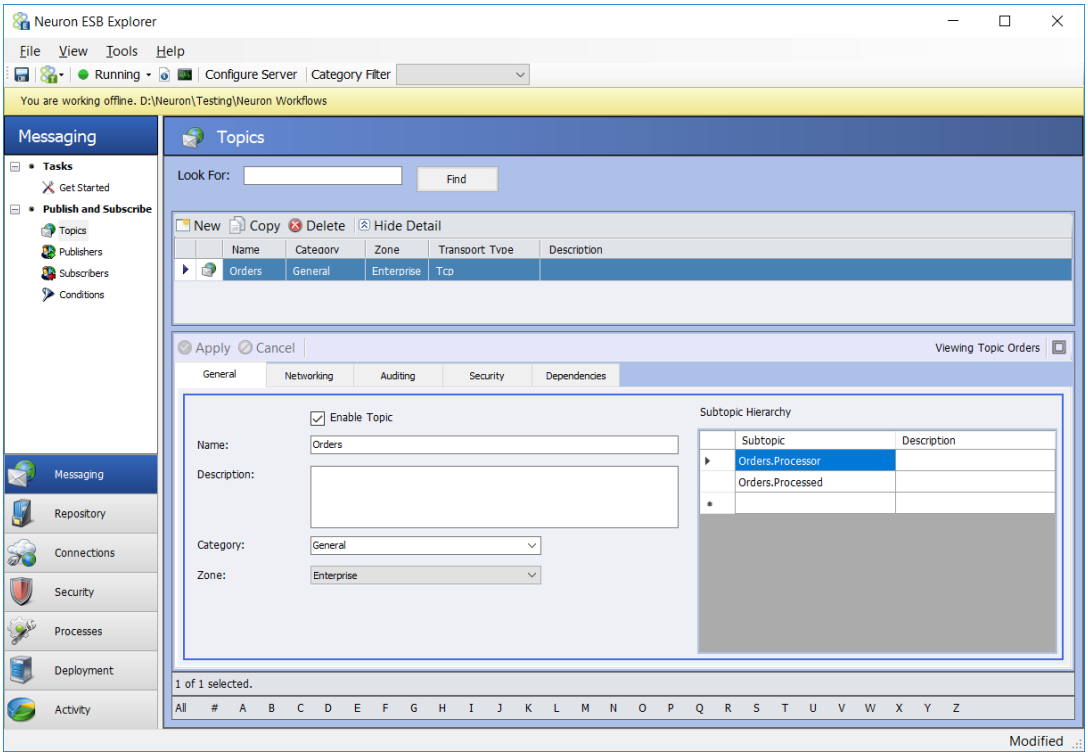
Add new Subtopics

When developing a workflow that includes a correlated send/receive, the workflow's subscriber needs a minimum of three separate topic subscriptions:

1. A receive subscription to a topic that will create a new instance of the workflow
2. A send subscription to a topic where the message creating the correlation set will be published to
3. A receive subscription to a topic where the correlated response will be received from

In the first exercise you created a topic called Orders. For this exercise we are going add two new subtopics – Orders.Processor and Orders.Processed. The first subtopic, Orders.Processor, will be used for routing the order to an external partner. The second subtopic, Orders.Processed, will be for routing the acknowledgement from the external partner back to the correct instance of the workflow.

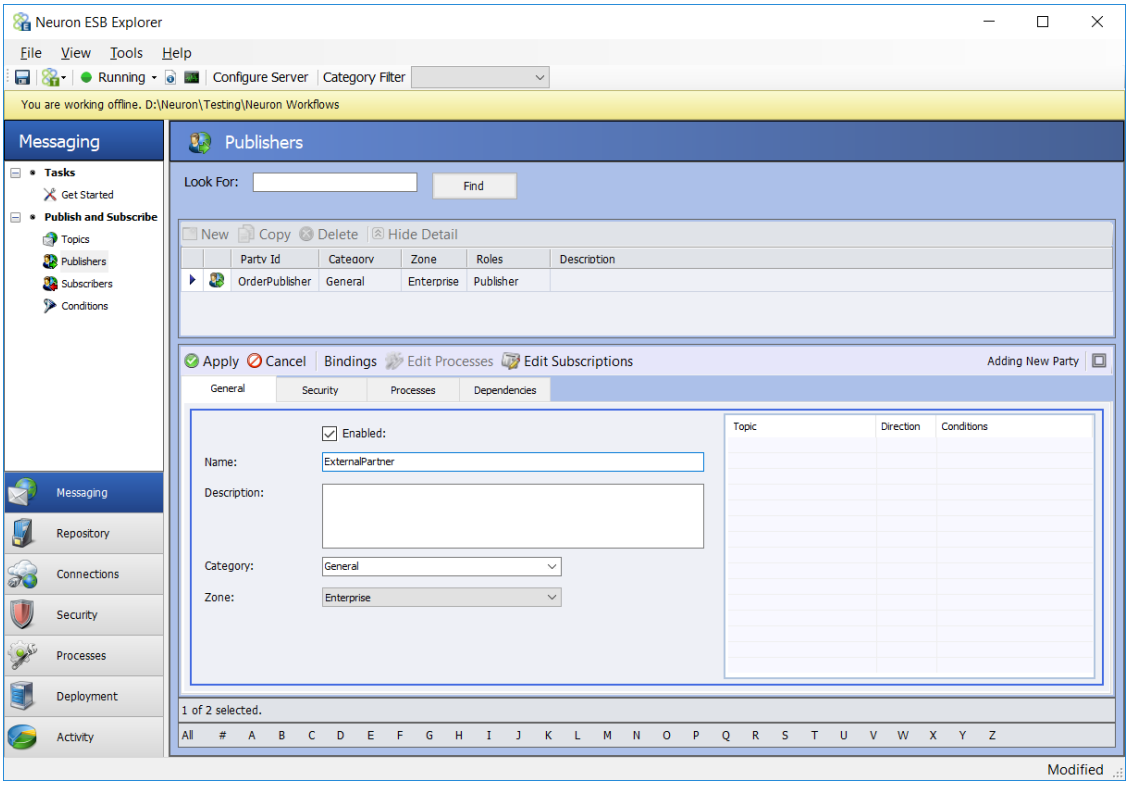
Navigate to the **Messaging** tab and click **Topics**. Select the **Orders** topic and add **Processor** and **Processed** to the list of subtopics and click **Apply**:



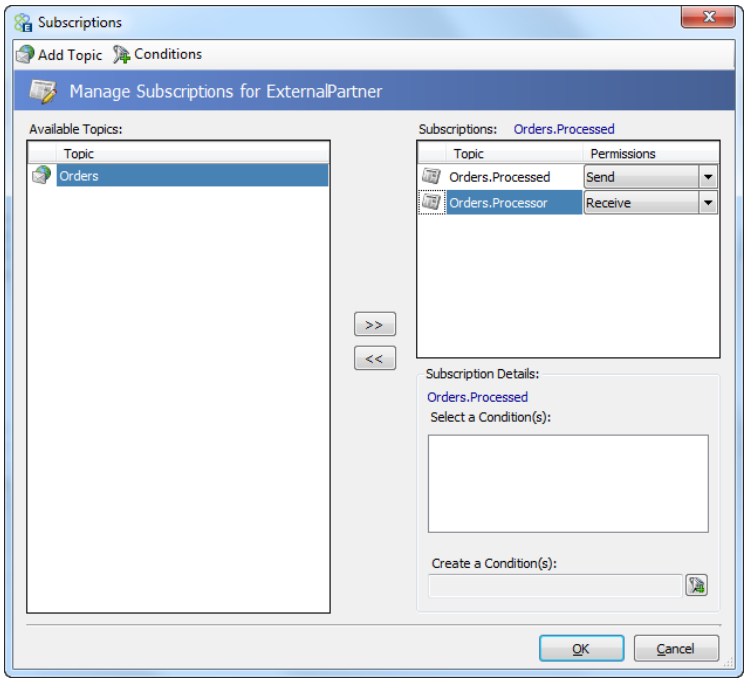
Add a Party

To represent the external partner, you are going to add a new Party to the solution. This party will contain subscriptions to both the subtopics created in the previous step. The subscription to Orders.Processor will have Receive permissions while the subscription to Orders.Processed will have Send permissions.

Navigate to the **Messaging** tab and click **Publishers**. Click the **New** button to create a new publisher, and set the name to **ExternalPartner**:



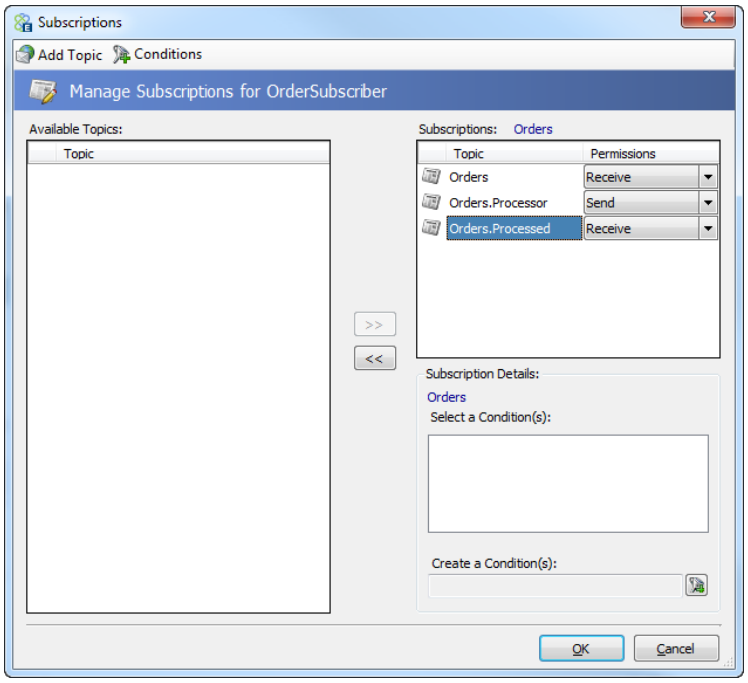
Click the **Edit Subscriptions** button. Add **Orders.Processor** and **Orders.Processed** to the Current Subscriptions list. Change the Permissions for Orders.Processor to *Receive*:



Click the **OK** button and then the **Apply** button for the ExternalPartner party.

Since the workflow will be changed so that it publishes messages on the Orders.Processor topic then the **OrderSubcriber** party associated with the Workflow Endpoint needs to be modified. Add a subscription to the **OrderSubscriber** party so that it is able to **Send** on the **Orders.Processor** topic.

When the **ExternalPartner** sends an acknowledgement message it does so on the Orders.Processed topic. For the workflow to receive it, add a subscription to **OrderSubscriber** so that it can **Receive** on the **Orders.Processed** topic:



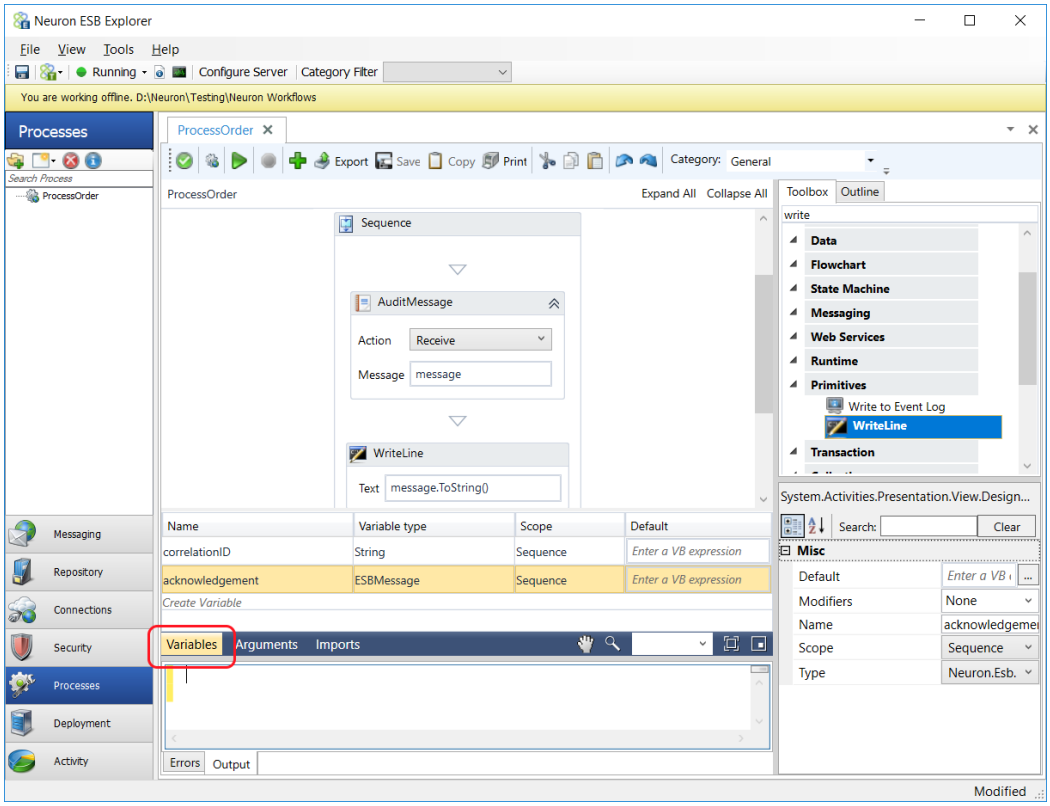
Click the **OK** button and then the **Apply** button for the OrderSubcriber party.

Modify the ProcessOrder Workflow

Navigate to the **Processes** tab and open the *ProcessOrder* workflow (double-click on it).

NOTE: The variable names are case sensitive! Take care when working through the exercises and be sure that you have the correct casing. If you don't, the workflow process will throw an exception.

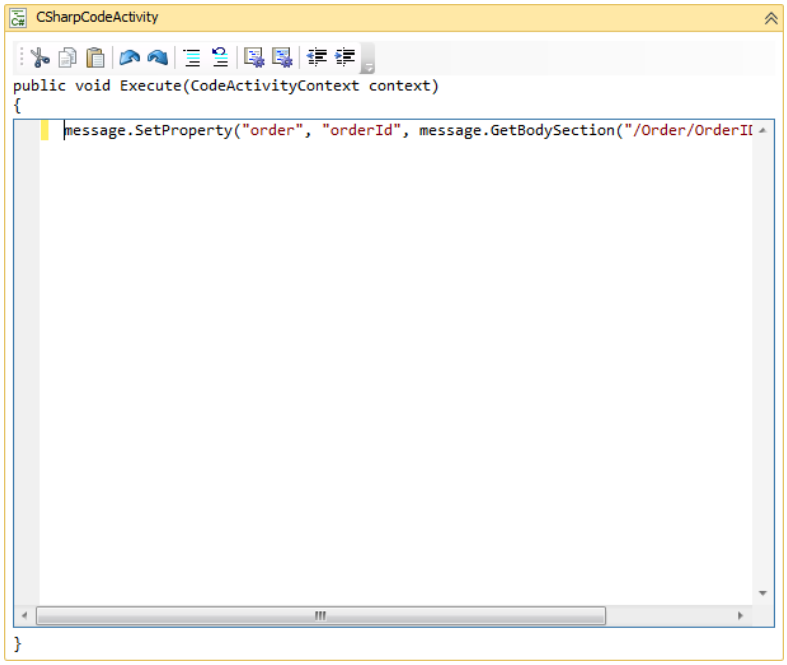
The first step you need to do is create the Correlation ID variable that will be used for correlating the outgoing order with the incoming order acknowledgement and another variable that the incoming response message will be written to. These variables need to be created at the outer scope of the workflow, so select the Sequence activity that contains the **AuditMessage** and **WriteLine** activities. Click on **Variables** (outlined in red below) and then click the line that says **Create Variable**. Set the variable name to *correlationID* and the variable type to String. Click **Create Variable** again and set the variable name to *acknowledgement* and the variable type to Neuron.Esb.ESBMessage:



Second, you need to set the custom message property that will contain the value to correlate the sent and received messages. The incoming message will contain the Order ID we want to correlate on. The location of the Order ID in the body of the message can be found with the XPATH "/Order/OrderID".

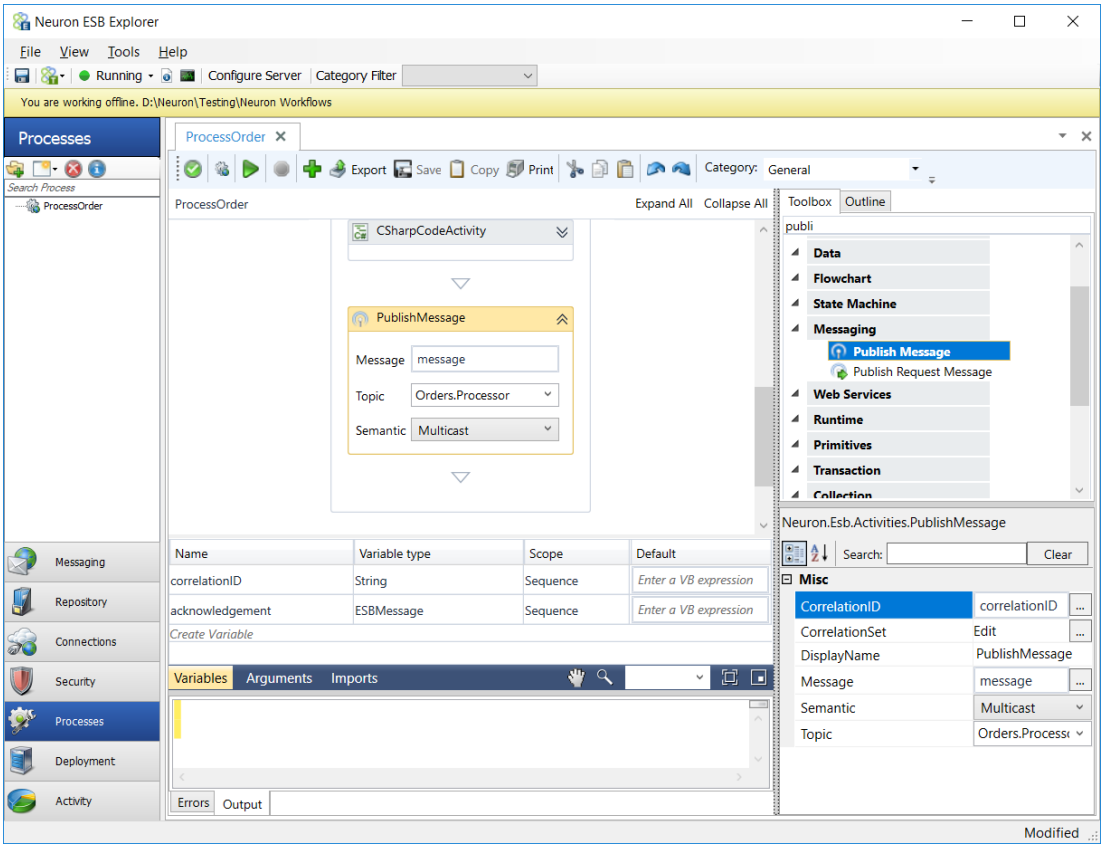
Search for the **C#** activity and drag it onto the designer directly below the WriteLine activity. In the C# activity code window, type:

```
message.SetProperty("order", "orderId", message.GetBodySection("/Order/OrderID"));
```

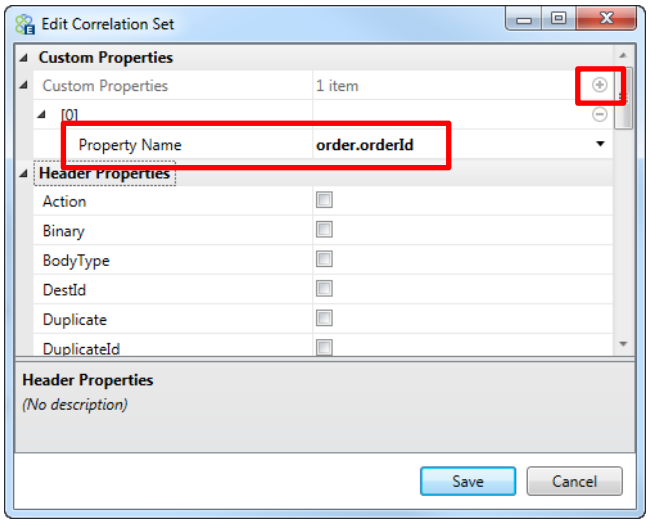


*NOTE: The C#, C# Class and VB.NET code Workflow Activities can be opened in a more robust, dedicated Tabbed Window by selecting **Edit Code** from the right click mouse context menu.*

Third, you need to publish the message to the bus for delivery to the external partner. Search for the **Publish Message** activity in the toolbox and drag it onto the designer surface directly under the **CSharpCodeActivity** activity. Set the **Message** property to *message*, **Topic** to *Orders.Processor* and **Semantic** to *Multicast*:

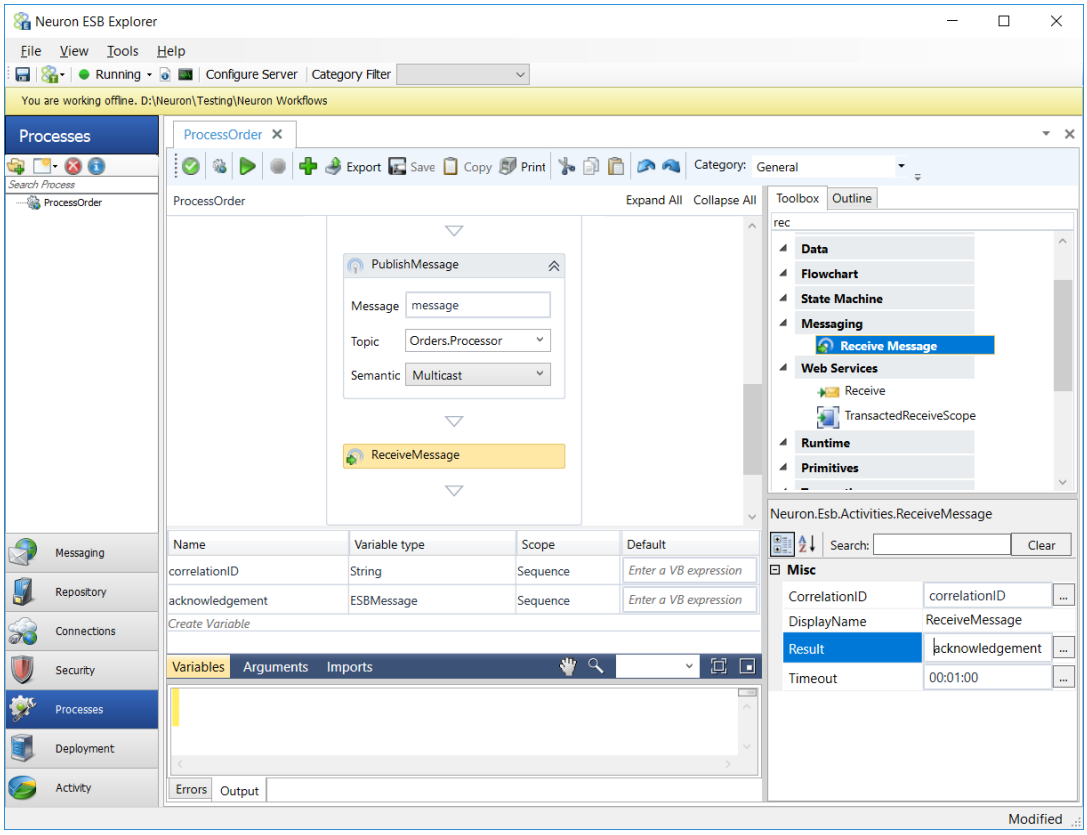


In the property grid for the Publish Message activity, set the **CorrelationID** property to *correlationID*. Click on the ellipsis button for the **CorrelationSet** property. In the Edit Correlation Set dialog click the New icon (outlined in red below) to create a new Custom Property correlation set and set the **Property Name** to *order.orderId*:



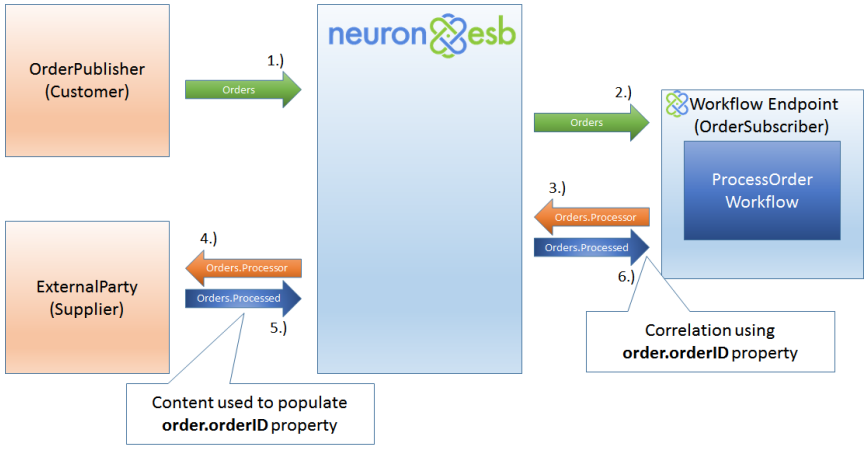
Click the **Save** button.

Finally, you need to add a receive activity for the incoming response to be routed to. Search for the **Receive Message** activity in the toolbox and drag it onto the designer surface directly under the **Publish Message** activity. Set the Result property to *acknowledgement* and the CorrelationID property to *correlationID:*



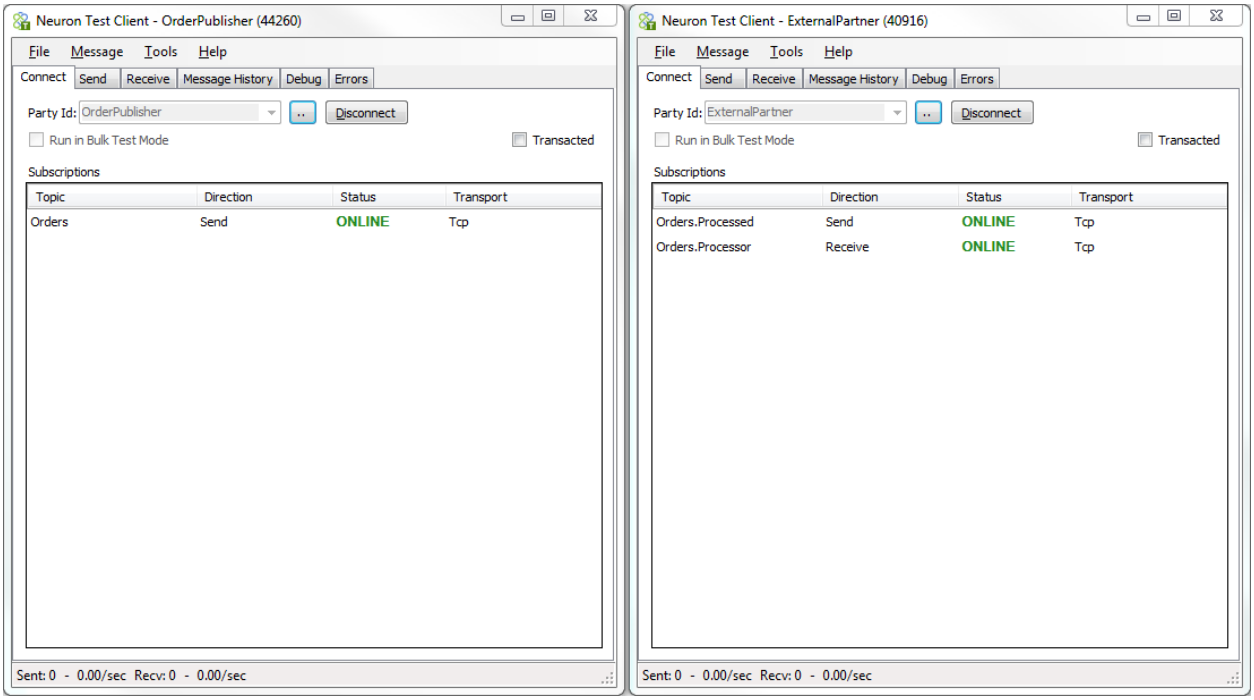
Click the **Apply** button and the save the configuration by clicking **File->Save**.

The process flow for the revised workflow is illustrated as follows:



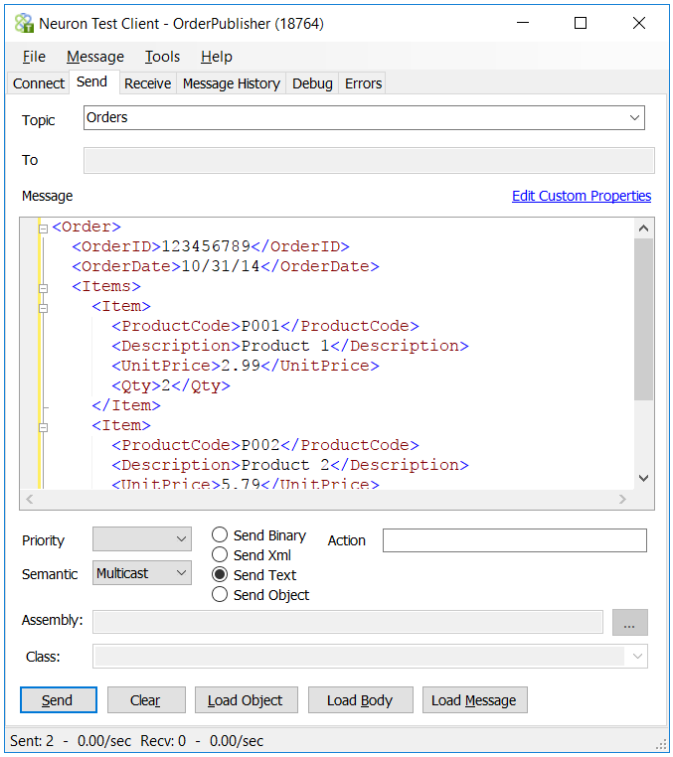
Run the Workflow

Open a second instance of the **Neuron Test Client** (Tools->Test Client->1 Test Client). You should now have two open. The first instance should be connected as the **OrderPublisher** party. Connect the second instance as the **ExternalPartner** party:

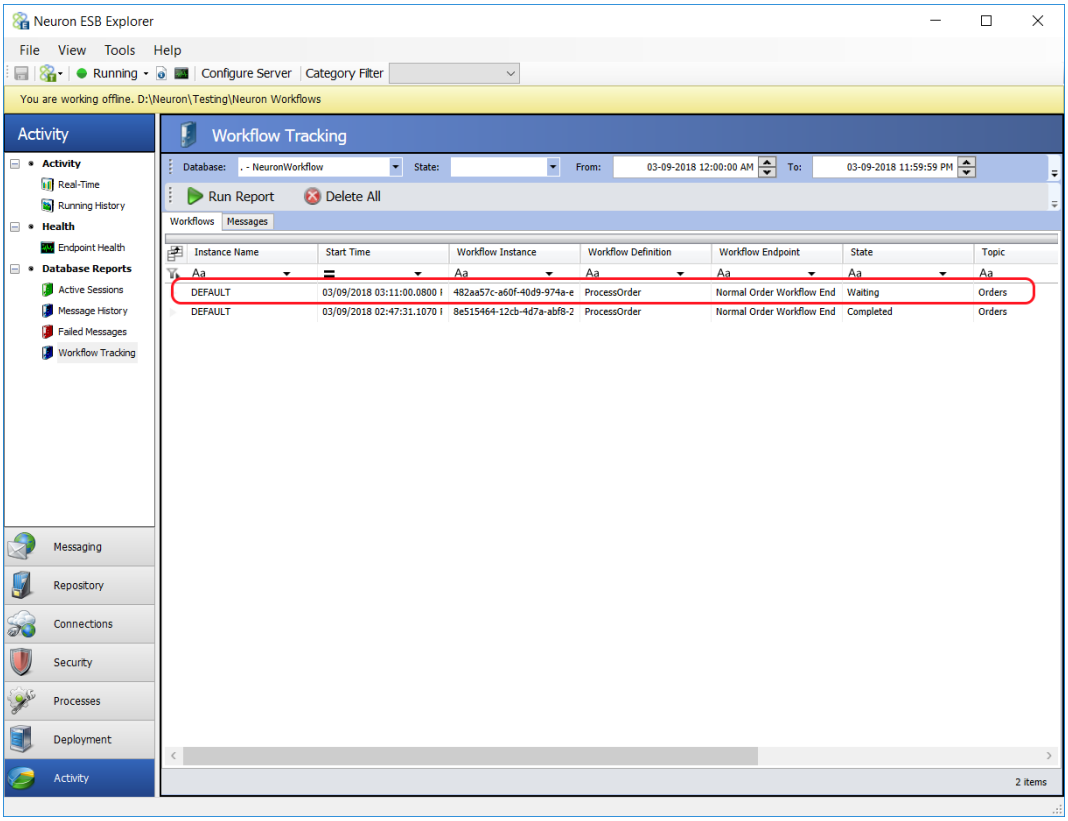


On the **OrderPublisher** test client, click on the **Send** tab and enter this XML:

```
<Order>
  <OrderID>123456789</OrderID>
  <OrderDate>10/31/14</OrderDate>
  <Items>
    <Item>
      <ProductCode>P001</ProductCode>
      <Description>Product 1</Description>
      <UnitPrice>2.99</UnitPrice>
      <Qty>2</Qty>
    </Item>
    <Item>
      <ProductCode>P002</ProductCode>
      <Description>Product 2</Description>
      <UnitPrice>5.79</UnitPrice>
      <Qty>5</Qty>
    </Item>
  </Items>
</Order>
```



Click the **Send** button. This will send the Order XML to Neuron, creating an instance of the ProcessOrder workflow. To see the status of this instance, in Neuron ESB Explorer navigate to the **Activity** tab and select **Workflow Tracking**. Click the **Run Report** button. You should see something similar to what is displayed below:

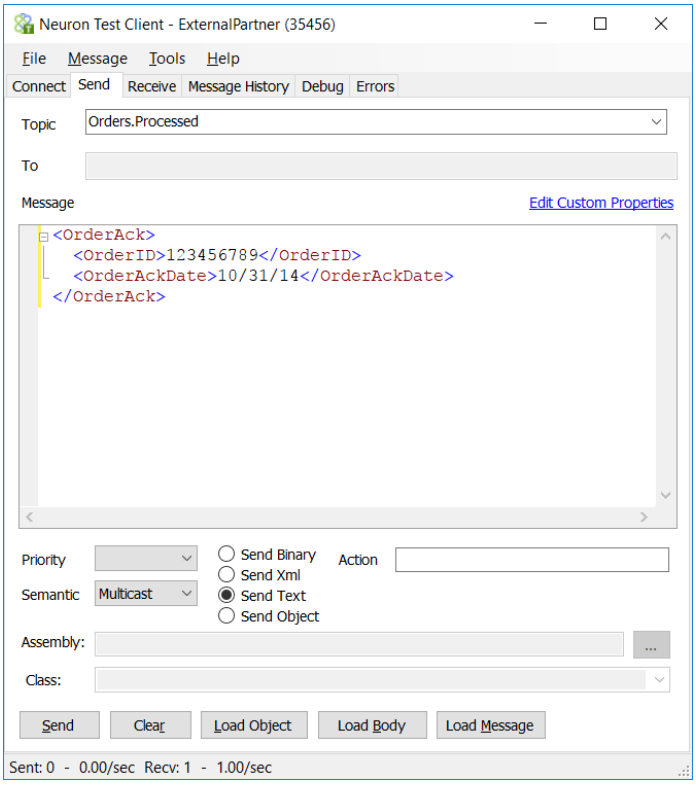


The state of the workflow instance will either be *Waiting* or *Unloaded*. The workflow instance will remain in the *Waiting* state for a short time (the Timeout property on the Receive Message Workflow Activity) before unloading. In either state, a response message with the correct OrderID will be routed back to this instance of the workflow.

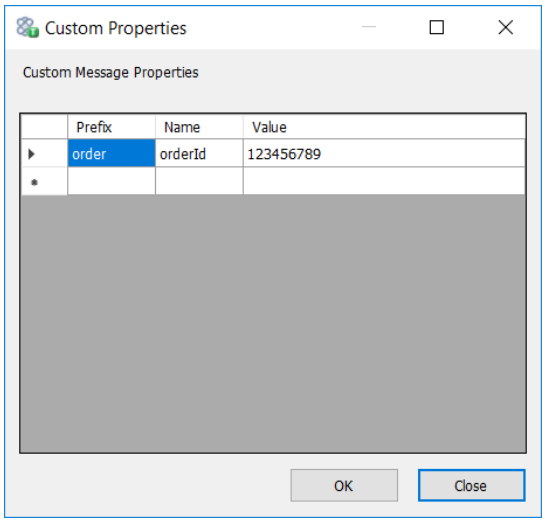
On the **ExternalPartner** test client, click on the **Receive** tab and verify the Order XML was received. If it hasn't been received, check the Neuron ESB vs event log for error messages. The previous step must be completed before continuing on to the next step, otherwise the acknowledgement will not be processed. Once you see the Order XML in the Receive tab, click on the **Send** tab and enter this XML:

```
<OrderAck>
  <OrderID>123456789</OrderID>
  <OrderAckDate>10/31/14</OrderAckDate>
</OrderAck>
```

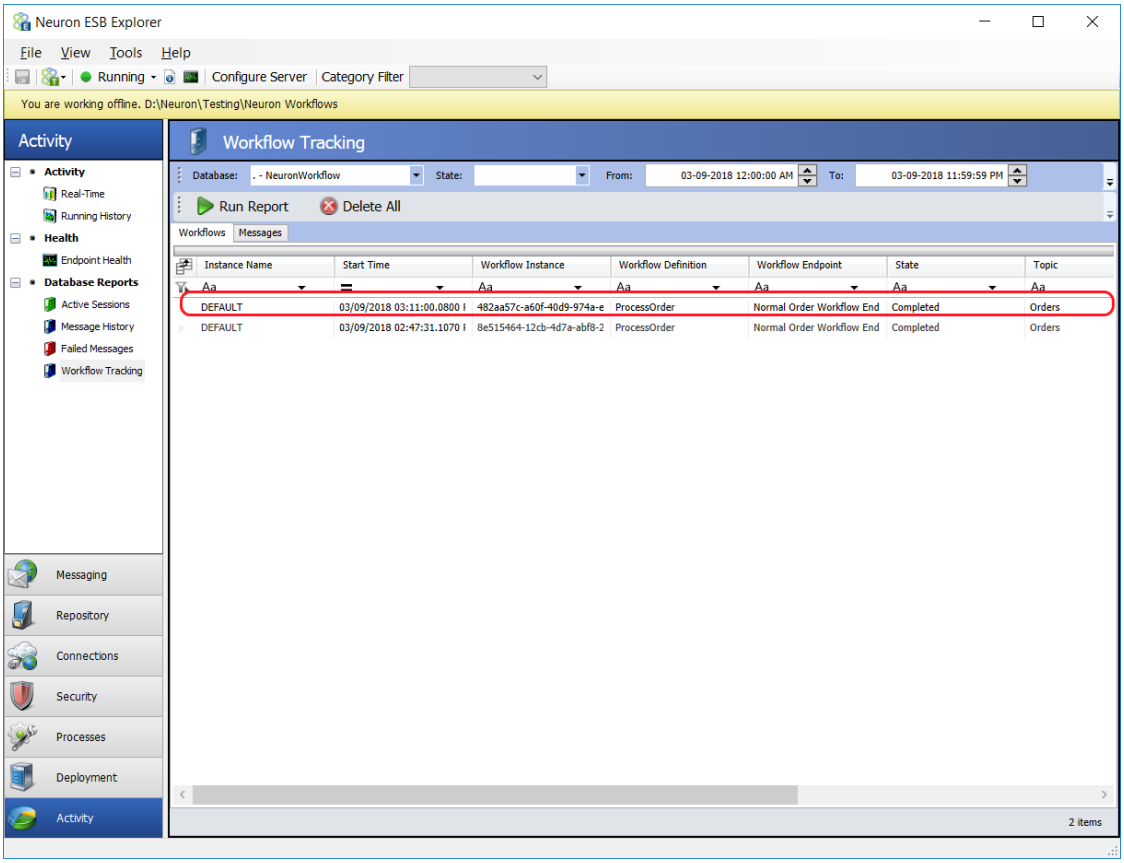
Change the **Topic** to *Orders.Processed*, and make sure the **Semantic** is set to *Multicast*:



Click on the **Edit Custom Properties** link to open the Custom Properties dialog. Add a new line, setting the **Prefix** to "order", the **Name** to "orderId" and the **Value** to "123456789":



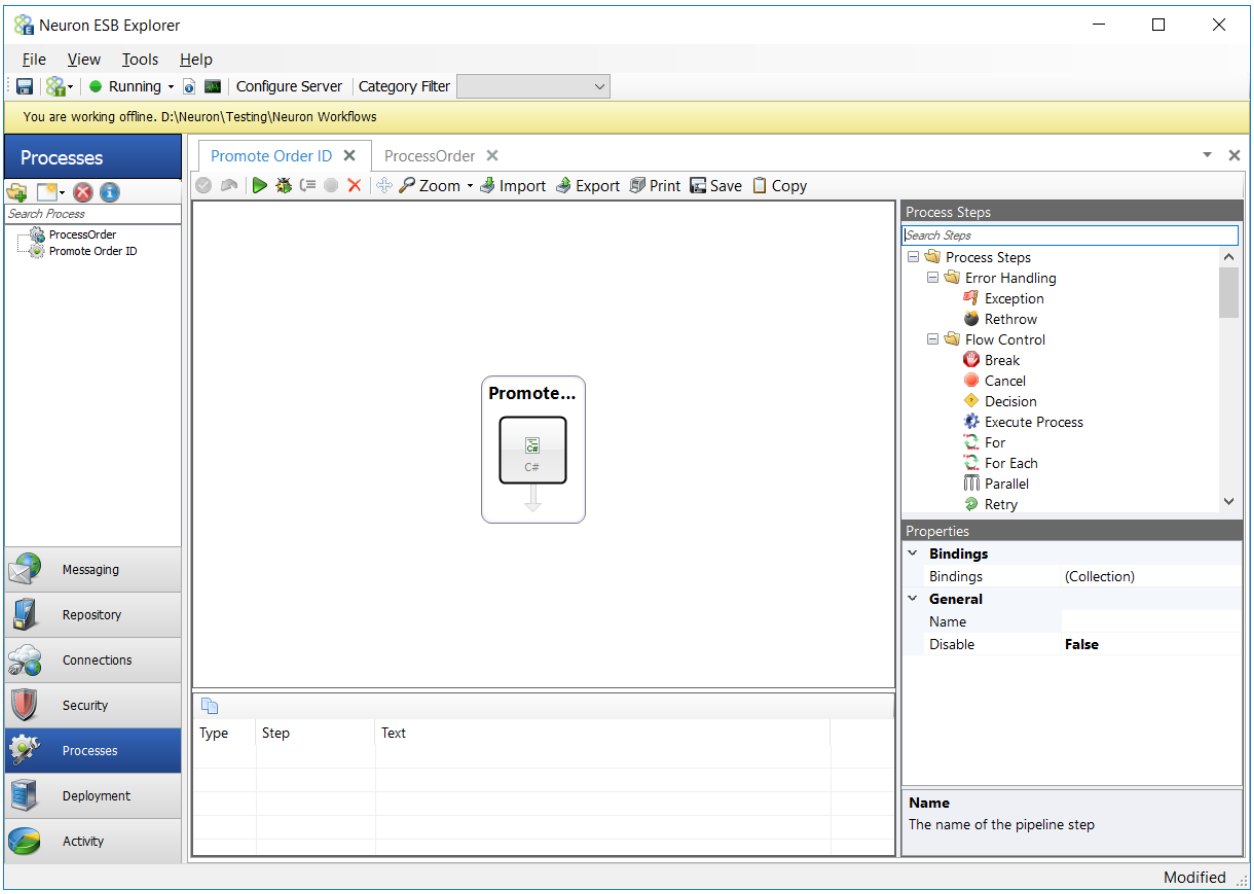
Click **OK**, and then click the **Send** button. This will send the Order Acknowledgement XML to Neuron. Neuron will route the message to the correct instance of the ProcessOrder workflow based on the custom message property you added in the test client. To see that the workflow has completed, navigate to **Workflow Tracking** and click **Run Report**:



Add a Business Process to Automatically Promote the Order ID

It's simple to create a correlated send/receive with Neuron Workflows. There's one more step to add to make this example more realistic. In the last part of the test above you manually added a custom message property to the Order Acknowledgement message. This was to assist in routing the message to the correct instance of the ProcessOrder workflow. However, if that message is coming from an external system, such as an FTP server, there won't be a way to manually set the message property. In this case you need to add a Process that will promote the Order ID to a custom message property for you.

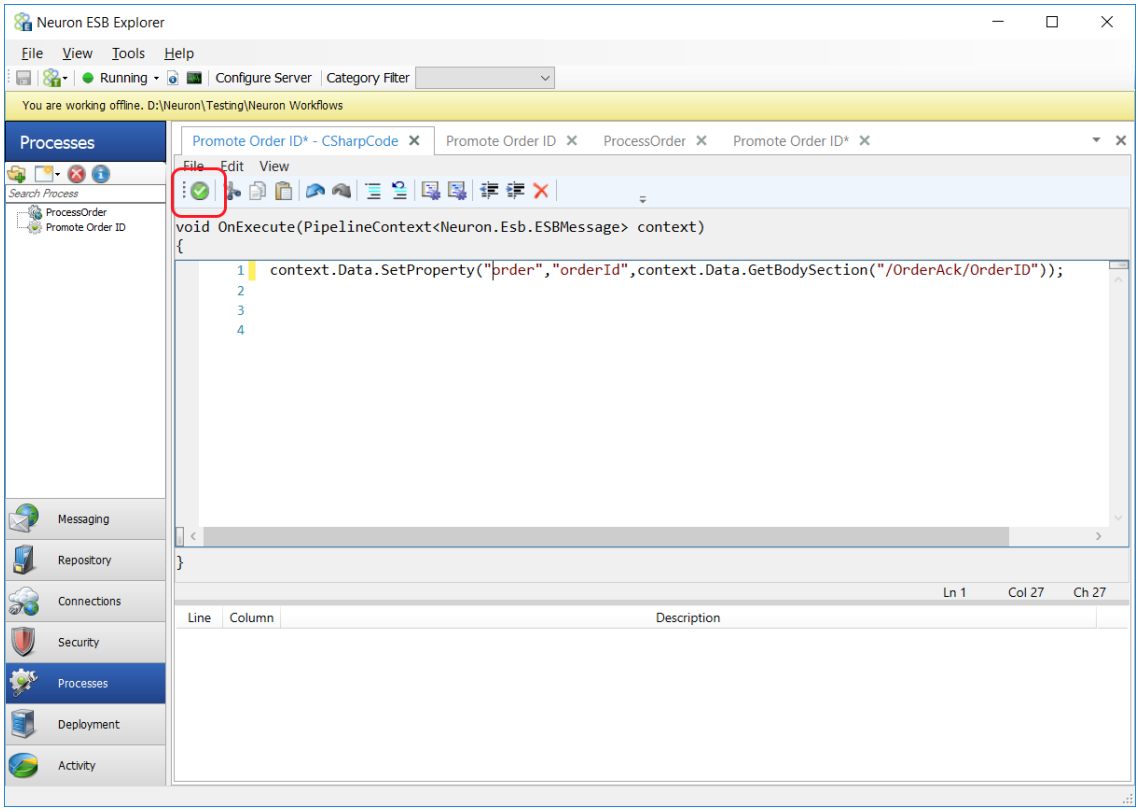
Navigate to the Processes tab and click the **New** button. Select **Create Process**. In the process designer click on the execution block titled **New Process 1**. In the properties window in the lower right change the Name property to **Promote Order ID**. In list of Process Steps find the C# step and drag it into the execution block of the process:



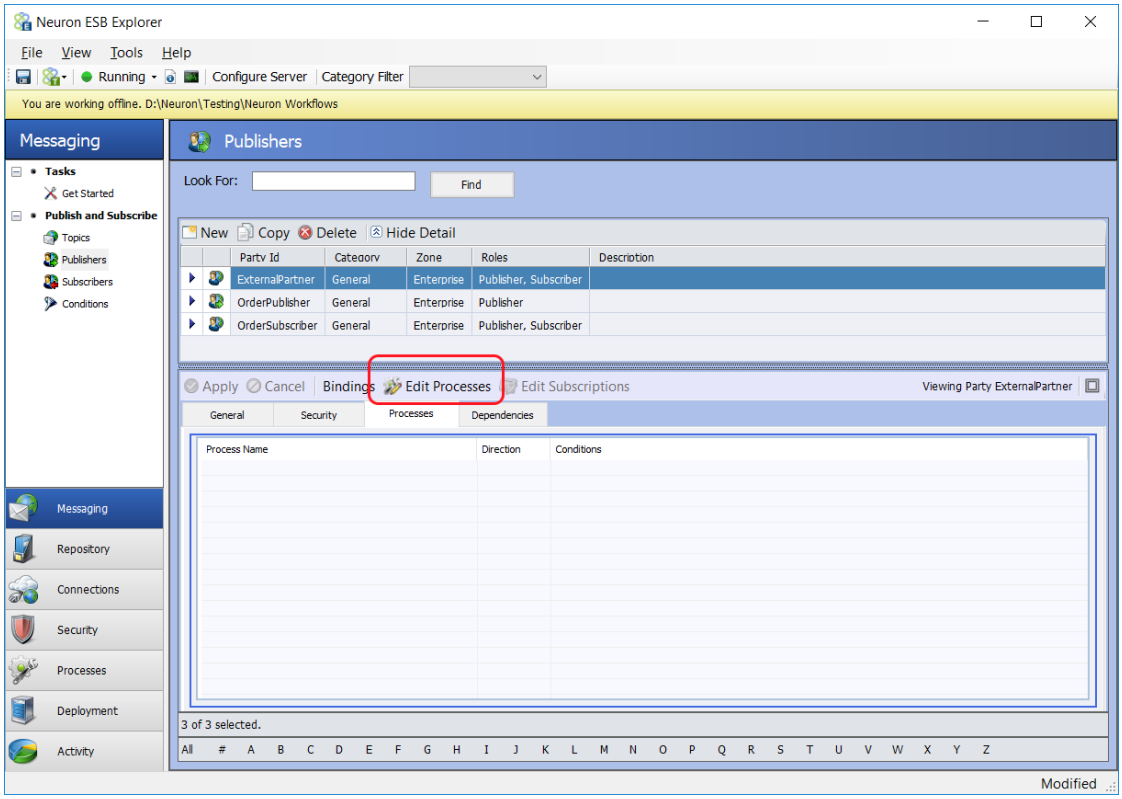
Right click on the C# step and select Edit Code…. In the code editor, enter this code:

```
context.Data.SetProperty("order", "orderId", context.Data.GetBodySection("/OrderAck/OrderID"));
```

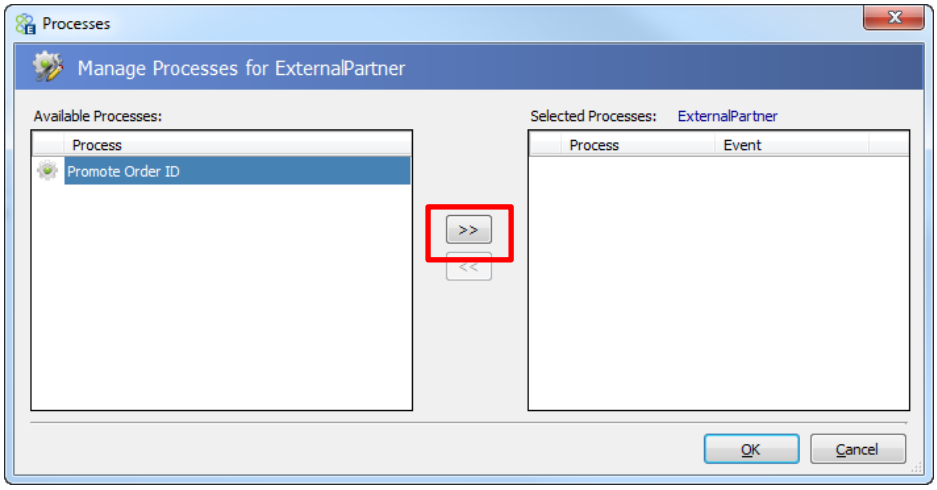
Click the **Apply** button to save the new process.



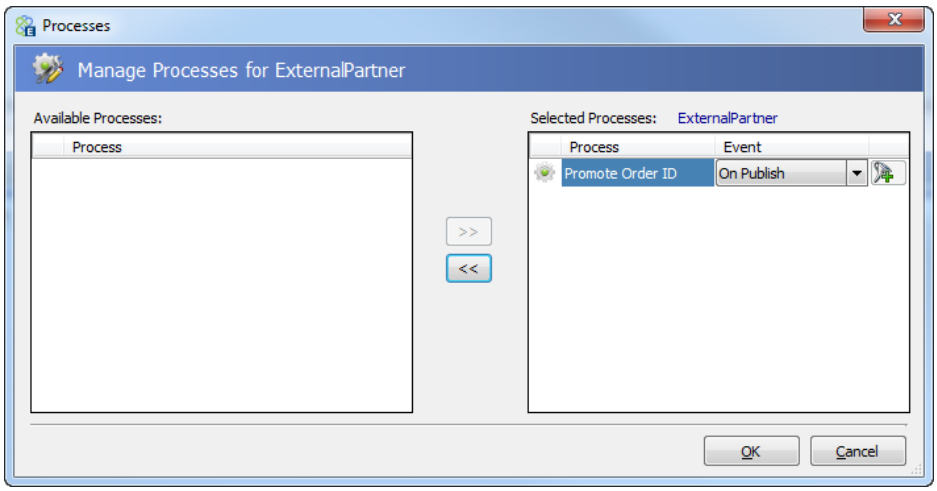
Navigate to the **Messaging** tab and select **Publishers**. Select the **ExternalPartner** publisher and click on the **Processes** tab:



Click the Edit Processes button (outlined in red above) to open the Processes dialog:



Select the **Promote Order ID** process from the list of available processes and add it to the list of selected processes:

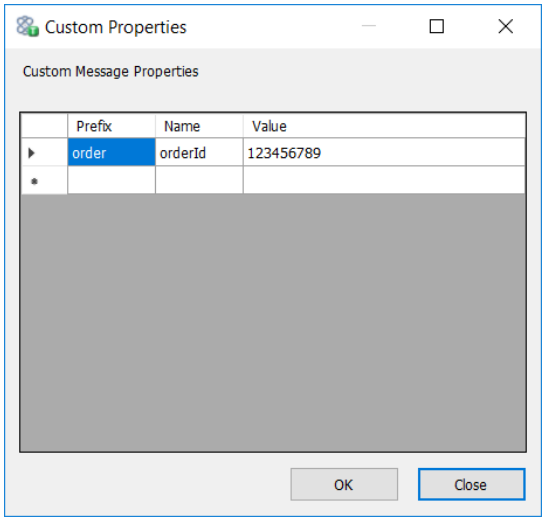


Click the **OK** button and click **Apply** for the publisher. Save the changes by clicking **File->Save**.

Now you can test the workflow again. The **OrderPublisher** test client should still be open and the **Send** tab should still contain the Order XML. Click the **Send** button again to start a new instance of the ProcessOrder workflow.

If you want, navigate to Workflow Tracking in Neuron ESB Explorer and click Run Report. There will be another instance of the ProcessOrder workflow.

In the **ExternalPartner** test client, click on the **Send** tab. Leave the Order Acknowledgement XML but click on the **Edit Custom Properties** link to open the **Custom Properties** dialog:



Select the **order.orderId** property and delete it. We now have a Process that will create this property for us. Click the **OK** button and then click on the **Send** button.

Navigate to Workflow Tracking again to see that the workflow instance has completed.

To illustrate that the custom property was created, return to the ExternalParty Neuron ESB Test Client, navigate to the *Message History* tab and, in the *History* list box, scroll to the last item (which should be the Send Action for the order acknowledgement). In the *Message Header* list box you will see the custom **orderId** property with the prefix **order** value **123456789** under the *Custom Properties* heading.

Extra Credit

You can easily validate that the correct Acknowledgement message reloads and completes the correct workflow instance by adding a slight modification to the Workflow that logs both the original Order ID that started the Workflow instance as well as the Order ID received by that same instance via the Acknowledgement message.

Create variable that will be used for storing the original Order ID number. This variable needs to be created at the outer scope of the workflow, so select the Sequence activity that contains the **AuditMessage** and **WriteLine** activities. Click on **Variables** and then click the line that says **Create Variable**. Set the variable name to *originalOrderID* and the variable type to String.

Next, right click on the C# step and select Edit Code…. In the code editor, directly underneath the existing line of code, enter the following:

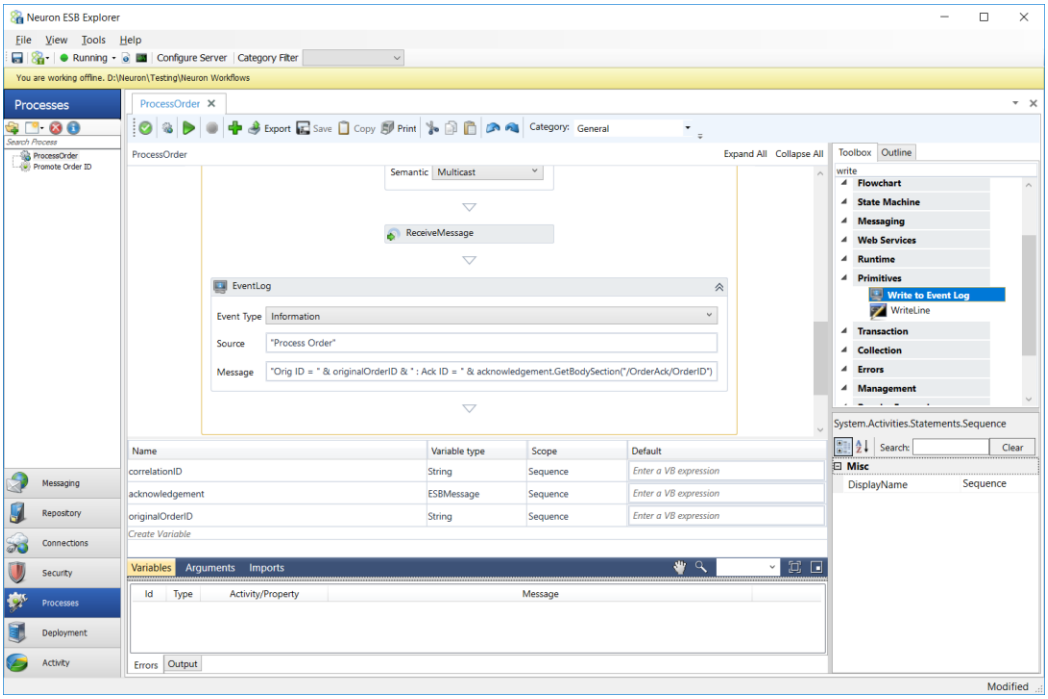
```
originalOrderID = message.GetBodySection("/Order/OrderID");
```

Click the **Apply** button to save the modification to the process in memory:

Lastly, add the **Write to Event Log** Workflow Activity to the ProcessOrder Workflow directly under the existing **Receive Message** Activity. Configure the **Write to Event Log** Workflow Activity properties as follows:

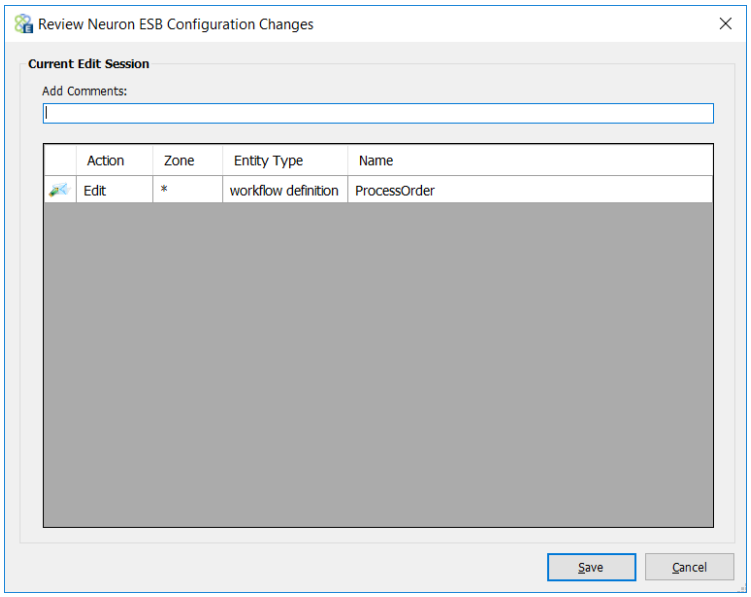
- Type = Information
- Source = “Process Order”
- Message =
```
"Orig ID = " & originalOrderID & " : Ack ID = " & _
                                acknowledgement.GetBodySection("/OrderAck/OrderID")
```

The resultant Workflow should look as follows:



These changes will effectively write the original Order ID as well as the Order ID contained within the received acknowledgement message to the Windows Application Event log.

Save the changes to disk by clicking **File->Save**. The Review Neuron ESB Configuration Changes dialog should appear displaying the changes recorded within the Neuron ESB Explorer since the last time **File->Save** was executed.



Run more Workflows

With your existing **Neuron Test Client** instances open and connected, navigate to the **OrderPublisher** connected **Neuron Test Client**.

On the **OrderPublisher** test client, click on the **Send** tab and enter this XML:

```
<Order>
  <OrderID>123456789</OrderID>
  <OrderDate>10/31/14</OrderDate>
  <Items>
    <Item>
      <ProductCode>P001</ProductCode>
      <Description>Product 1</Description>
      <UnitPrice>2.99</UnitPrice>
      <Qty>2</Qty>
    </Item>
    <Item>
      <ProductCode>P002</ProductCode>
      <Description>Product 2</Description>
      <UnitPrice>5.79</UnitPrice>
      <Qty>5</Qty>
    </Item>
  </Items>
</Order>
```

Modify the value in the OrderID xml element to:

2435

Hit **Send**. Now follow this by publishing 4 more messages as follows:

1. Change the value in the OrderID xml element to:

   4567

   Hit **Send**.

2. Change the value in the OrderID xml element to:

   8765

   Hit **Send**.

3. Change the value in the OrderID xml element to:

   6789

   Hit **Send**.

4. Change the value in the OrderID xml element to:

   9098

   Hit **Send**.

This effectively has sent 5 unique Order XML messages to Neuron, creating an instance of the ProcessOrder Workflow for each of the messages sent.

To see the status of each instance, in the Neuron ESB Explorer navigate to the **Activity** tab and select **Workflow Tracking**. Click the **Run Report** button. You should see 5 new Workflow instance records. The state of the workflow instances will be *Waiting*. However, if you wait longer than a minute, the Workflow instances will unload from memory and serialize into the database, at which point their state will be changed to *Unloaded*.

On the **ExternalPartner** test client, click on the **Messages History** tab and verify the Order XML messages were received. If they were not received, check the Neuron ESB vs event log for error messages. The previous step must be completed before continuing on to the next step, otherwise the acknowledgements will not be processed. Once you have verified that the messages have been received, click on the **Send** tab and enter this XML:

```
<OrderAck>
  <OrderID>123456789</OrderID>
  <OrderAckDate>10/31/14</OrderAckDate>
</OrderAck>
```

Change the **Topic** to *Orders.Processed*, and make sure the **Semantic** is set to *Multicast*:

Since we previously published 5 Order XML messages, each with its own unique Order ID, we need to publish 5 acknowledgements, each one containing an Order ID that was previously published. In this exercise we are going to publish the acknowledgements in a different order.

Modify the value in the OrderID xml element to:

4567

Hit **Send**. Now follow this by publishing 4 more acknowledgment messages as follows:

1. Change the value in the OrderID xml element to:

   6789

   Hit **Send**.

2. Change the value in the OrderID xml element to:

   8765

   Hit **Send**.

3. Change the value in the OrderID xml element to:

   9098

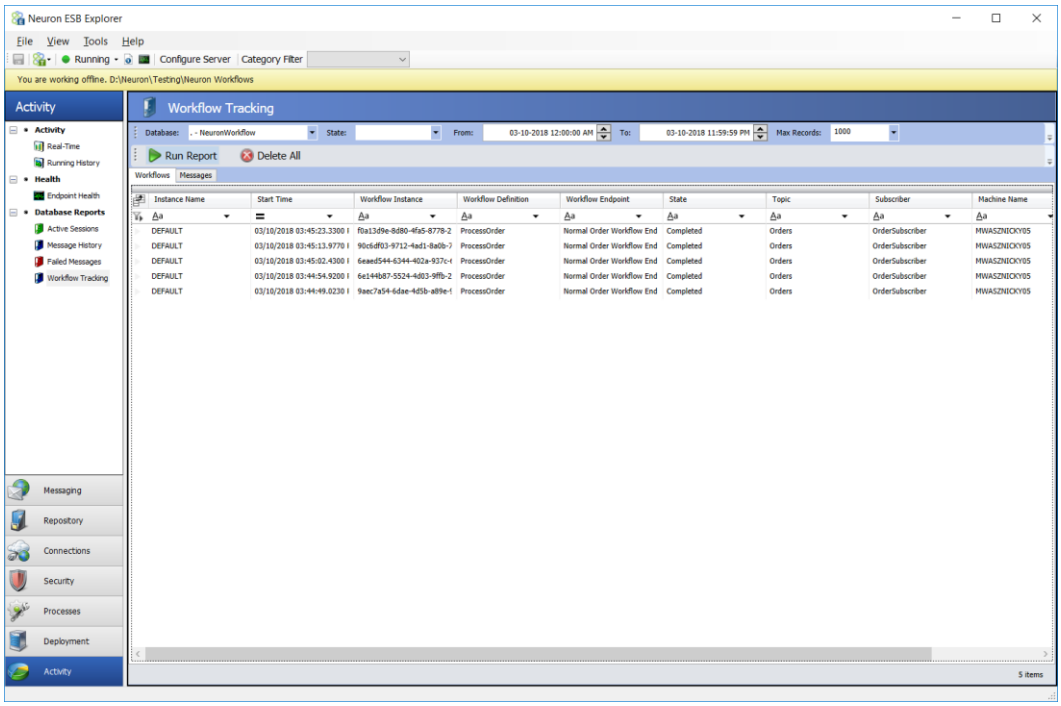
   Hit **Send**.

4. Change the value in the OrderID xml element to:

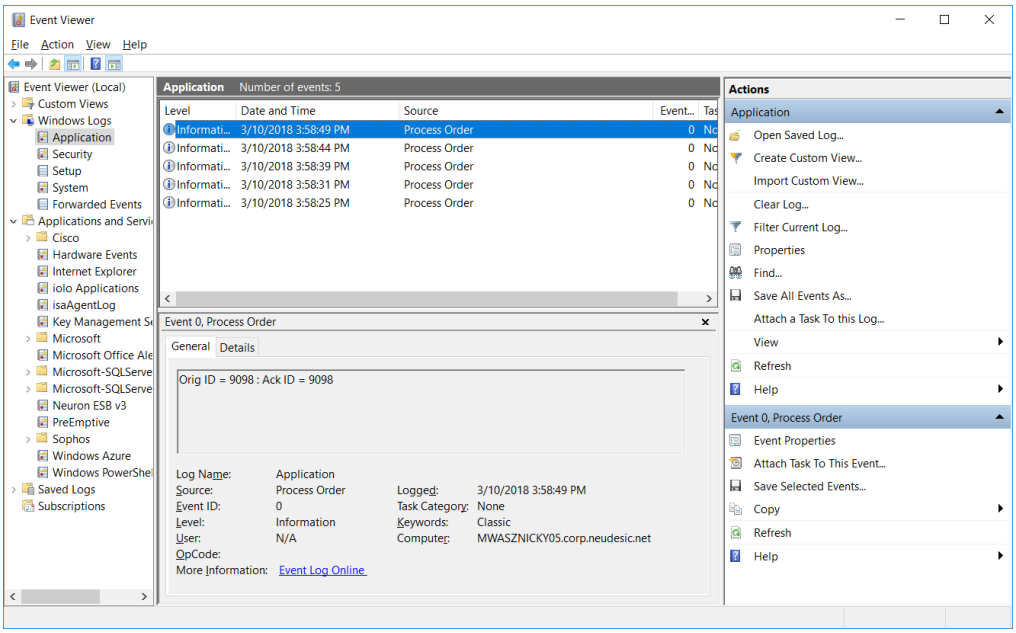
   2435

Hit **Send**.

This will send the Order Acknowledgement XML messages to Neuron. Neuron will route the message to the correct instance of the ProcessOrder workflow based on the OrderID Xml element value. To see that the workflow instances have completed, navigate to **Workflow Tracking** and click **Run Report**:



Lastly, you can verify that the correct Workflow instance processed its expected acknowledgement message by opening the Windows Event Viewer, navigate to the Application log and view the 5 entries that were written, each by one Workflow instance. Each entry will log the original Order ID that started the instance as well as the Order ID that was contained in the received acknowledgement message:



Exercise – Restart an Aborted Workflow

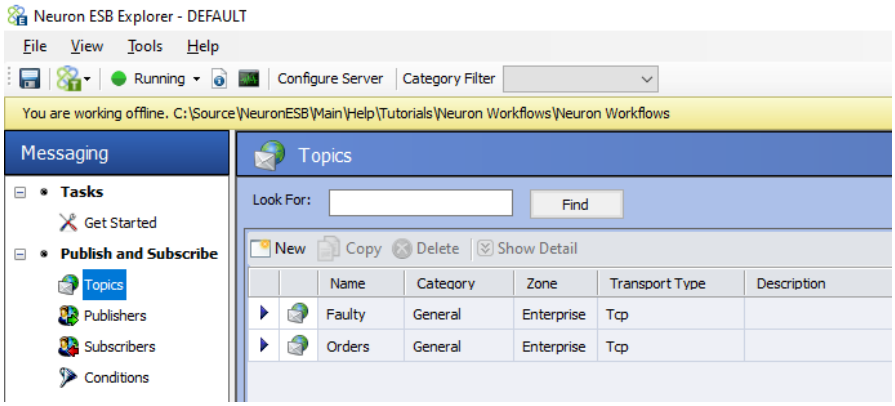
In any application there is always the potential for errors and workflows are no different. Whether this is caused by malformed data being sent to the workflow, services not being available or business logic determining that it is necessary to abort the process, there is always the potential for something to go wrong. The ability to deal with the error quickly and easily, and continue processing data, is the important thing.

In Neuron ESB when a workflow instance encounters an error it will enter into an aborted state. In this state the workflow instance is no longer capable of processing data. However, it is possible to restart a workflow instance which has been aborted, once the error has been corrected. In this exercise we are going to look at how to restart a workflow that has been aborted.

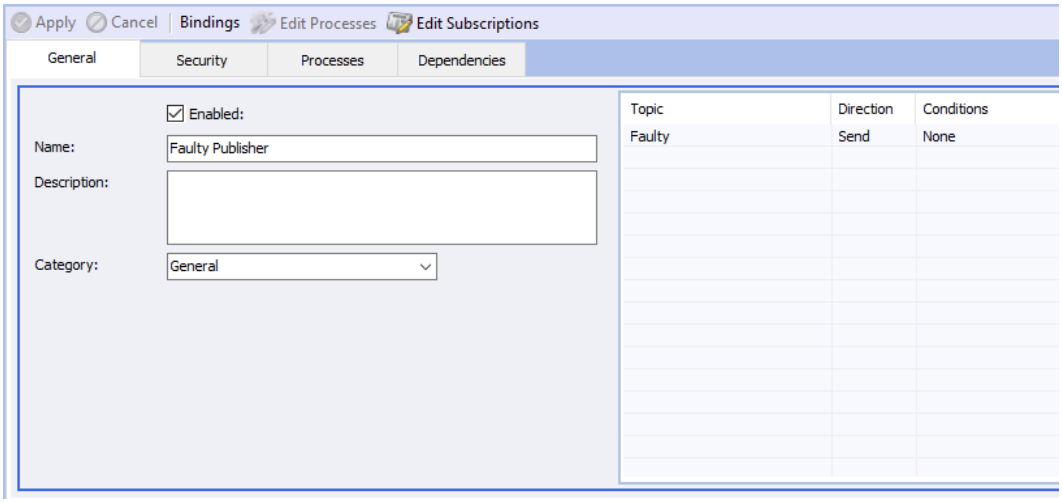
Add Topic, Publisher and Subscriber

We will begin by creating a topic, publisher and subscriber for our exercise. The Workflow Endpoint needs a subscriber to host and a topic on which to receive messages. The publisher is needed in order to push messages to the topic.

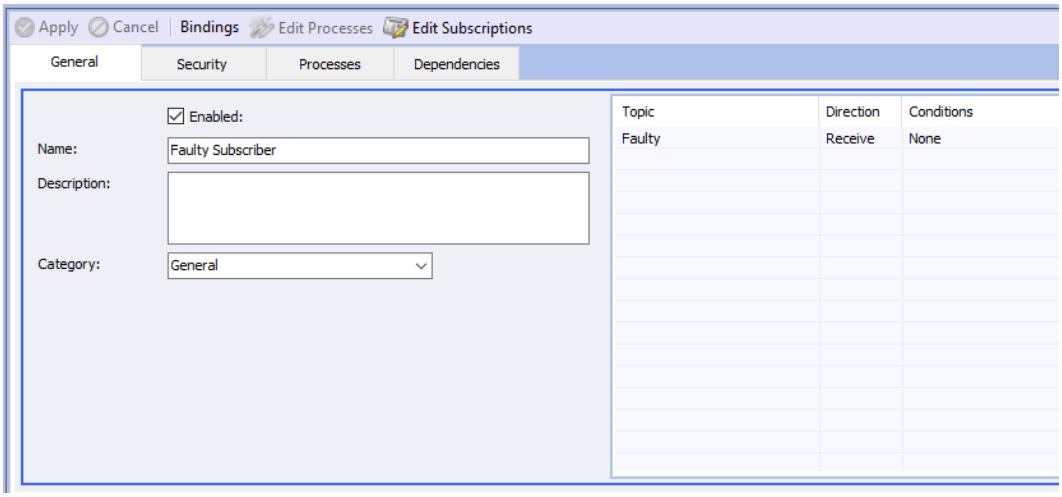
- Navigate to Messaging -> Topics
- Click the New button to create a new Topic named Faulty
- Apply your changes



- Navigate to Messaging -> Publishers
- Click the New button to create a new publisher named Faulty Publisher
- Edit Faulty Publisher's subscriptions and give it a subscription to Faulty with send permissions.



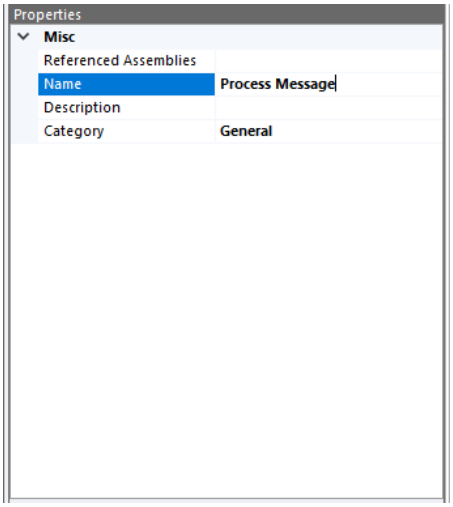
- Navigate to Messaging -> Subscribers
- Click the New button to create a new subscriber named Faulty Subscriber
- Edit Faulty Subscriber's subscriptions and give it a subscription to Faulty with receive permissions



Create Pseudo Service

In our workflow we are going to call out to a service that will initially be down. So, we need to create the service that we are attempting to contact. As this is just an exercise, we are going to use a client connector as the service that we are calling out to. In order to ensure that our client connector sends us back a response we need to create a quick business process for it.

- Navigate to Processes
- Click the New button and select Create Process
- Click New Process 2 and change the name of the process to Process Message in the Property Grid



- Drag a C# step on to the Business Process Designer
- Double click the C# step to open the Language Editor and enter the following code

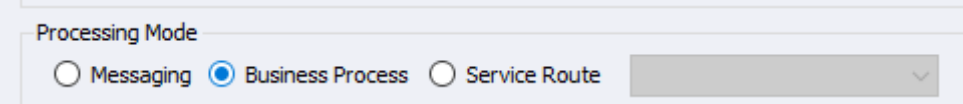
```
context.Data.Text = "<Response>Message has been processed
successfully</Response>";
```

- Apply your changes and close the Language Editor tab.

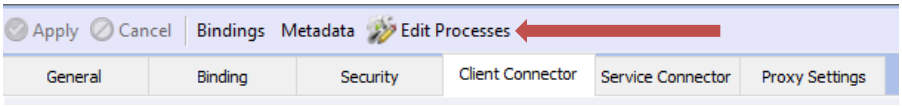
- Close the Process Message tab as we are done with this Business Process.

Now we can move on to creating the client connector that will host our process and serve as a pseudo service for our exercise.

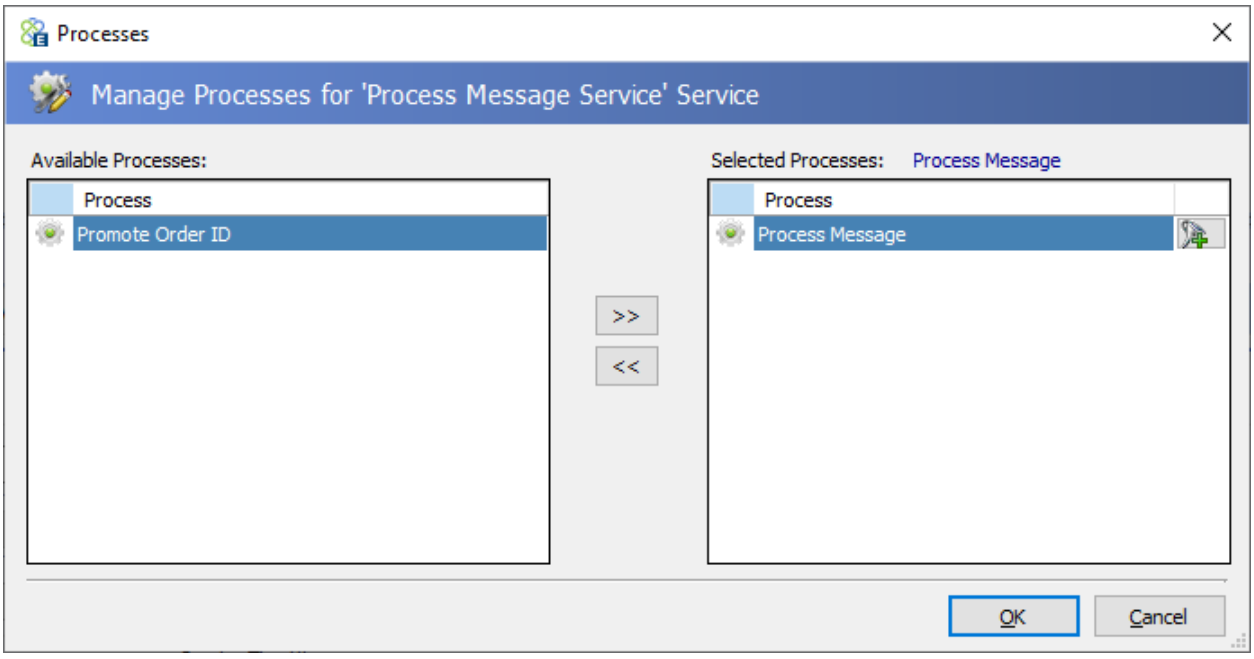
- Navigate to Connections -> Service Endpoints
- Click the New button to create a new service endpoint named Process Message Service
- On the client connector tab
    - Check the Enable box
    - Enter http://localhost:9099 in the URL field
    - Click Business Process under Processing Mode



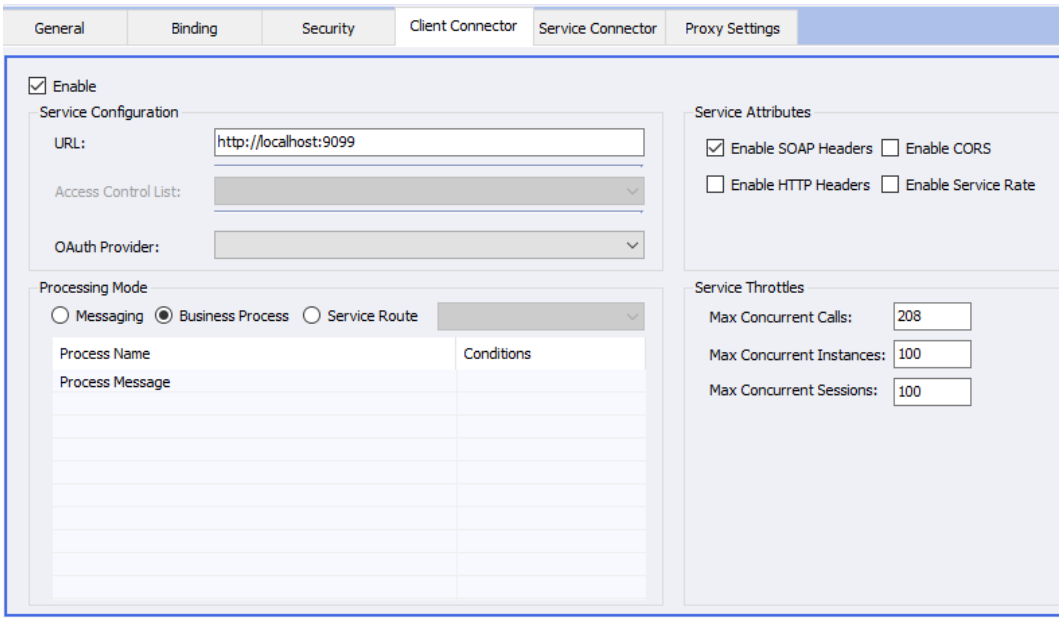
    - Click Edit Processes



    - Move Process Message to the Selected Processes field



    - Click OK to close the Processes dialog

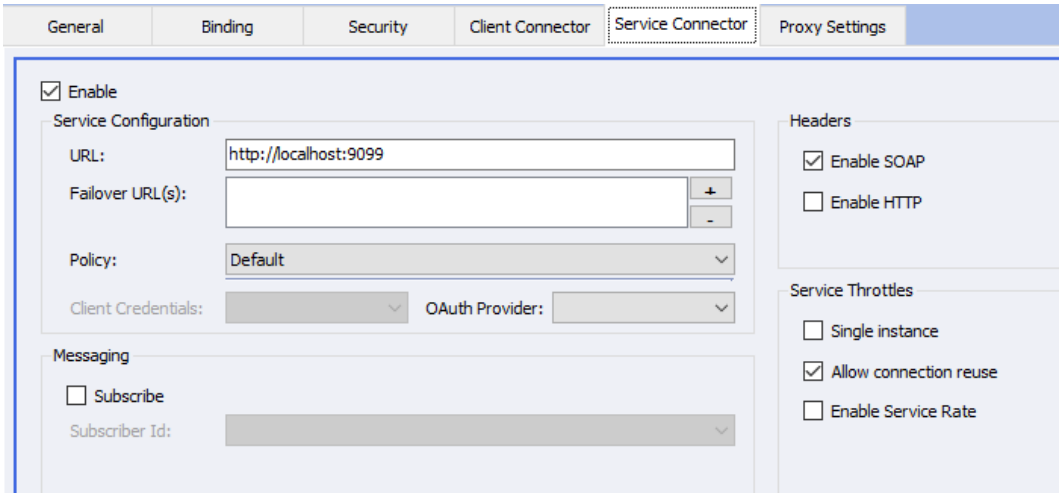


- Apply your changes.

Create a Service Connector

Next, we need to create a service connector that our workflow will use to call the service that we just created.

- Navigate to Connections -> Service Endpoints
- Click the New button to create a new service endpoint named Message Service
- On the service connector tab
  - Check the Enable box
  - Enter http://localhost:9099 in the URL field
  - Uncheck the Subscribe box

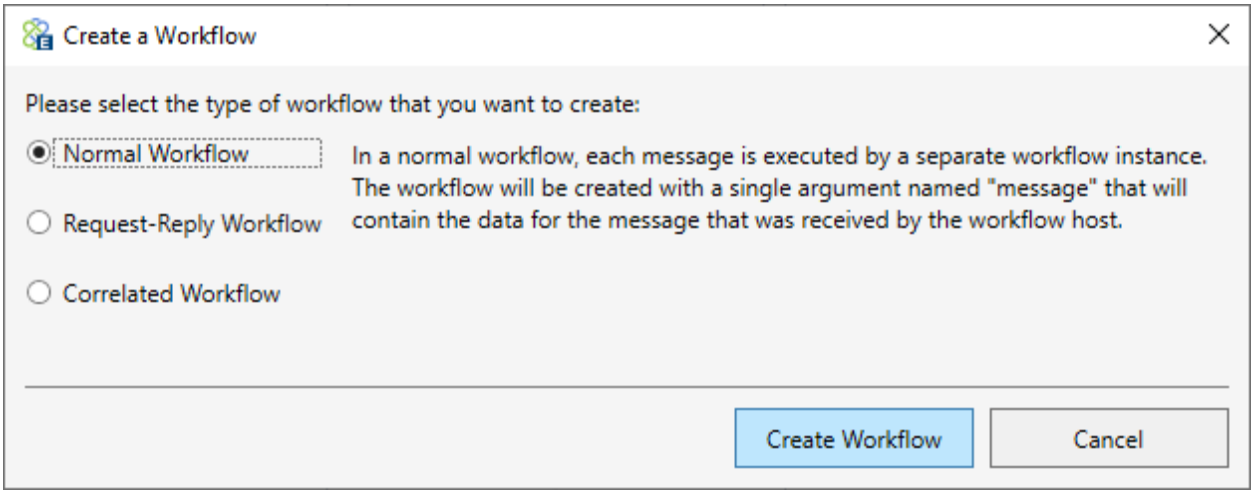


- Apply your changes.

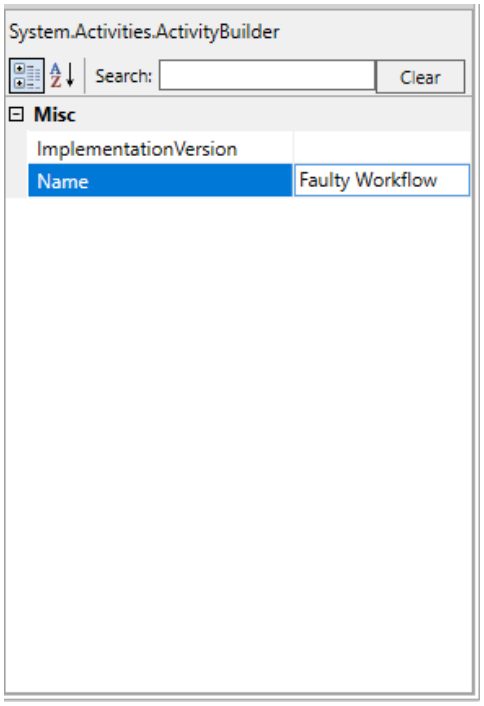
Create the Workflow Definition

We can now move on to creating a workflow definition for our workflow endpoint to host.

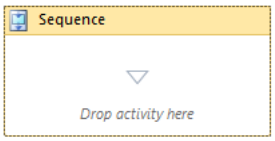
- Navigate to Processes
- Click the New button and select Create Workflow
- In the Create a Workflow dialog that pops up select Normal Workflow and click Create Workflow.



- Click anywhere in the Workflow Designer to show the workflow properties. Set the name property to Faulty Workflow.



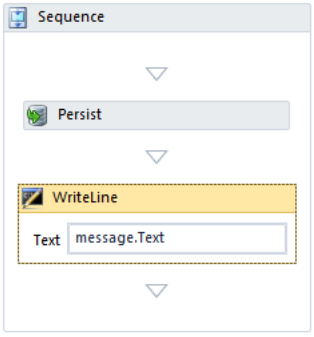
- In the Toolbox use the search bar to search for the Sequence activity and drag that onto the Workflow Designer.



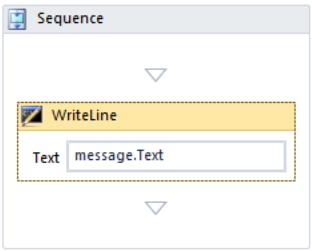
In order for a workflow to be able to restart from an error, you need at least one persist point in the workflow definition. So, we will add one at the very beginning.

The Persist Workflow Activity can be used more than once within a workflow. Generally, the Persist Workflow Activity should be placed before any activity or set of activities, where if a failure were to occur, the user or the system would want to restart the workflow at the last successfully executed Persist point. Restarting a workflow can happen in a few ways, either initiated by the user through the Workflow Tracking interface or, by the system during an outage/failure recovery. For example, if the Neuron Endpoint Host executing the Workflow was deployed to more than 1 server and the current executing server failed, the Neuron Runtime would automatically restart the Workflow Instance at the last successfully executed Persist point on the remaining active server.

- Search for the Persist activity. Drag it on to the Workflow Designer and drop it in the Sequence container.

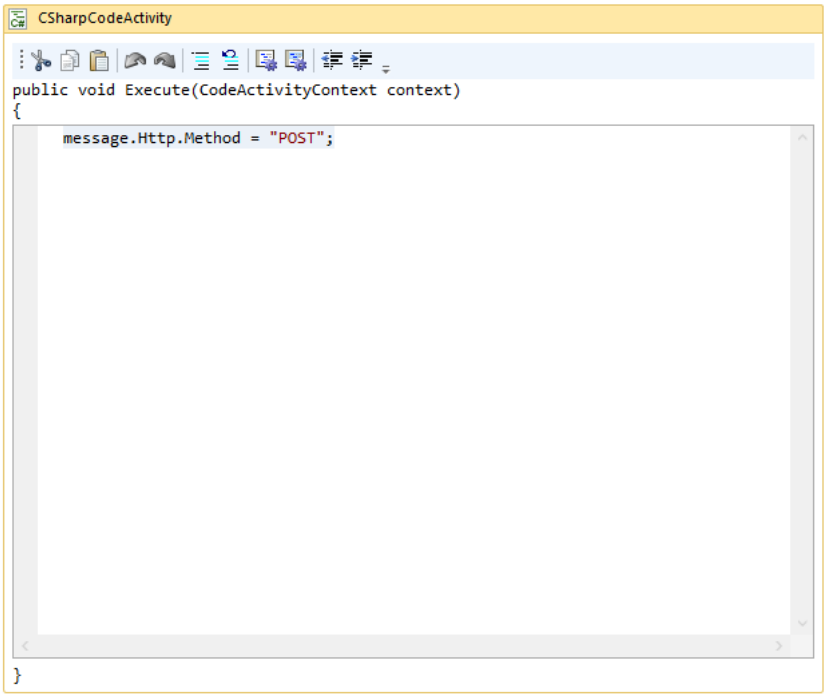


- Next search for the WriteLine activity. Drag it on to the Workflow Designer and drop it in the Sequence container right below the Persist activity.
- In the Text field of the WriteLine activity enter message.Text

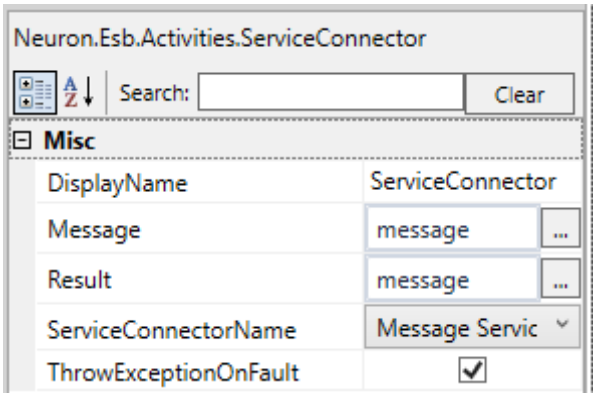


- Now search for the C# activity. Drag it on to the Workflow Designer and drop it in the Sequence container right below the WriteLine activity.
- In the C# activity enter the following code

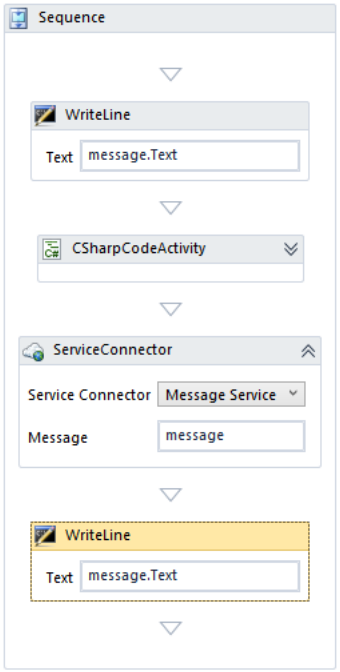
```
message.Http.Method = "POST";
```



- Collapse the C# activity.
- Search for the Service Endpoint activity. Drag it on to the Workflow Designer and drop it in the Sequence container right below the C# activity.
- In the Property Grid
    - Enter message in the Message field
    - Enter message in the Result field
    - Select Message Service from the ServiceConnectorName dropdown list
    - Check ThrowExceptionOnFault



- Search for the WriteLine activity. Drag it on to the Workflow Designer and drop it in the Sequence container.
- In the Text field of the WriteLine activity enter message.Text



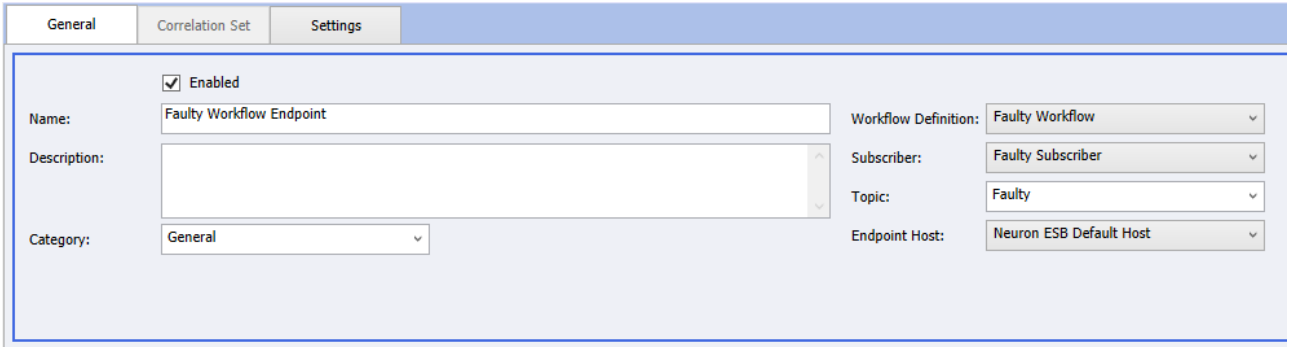
- Apply your changes to the Workflow Definition as we are done creating it.

Create the Workflow Endpoint

With the workflow definition complete we next need to set up the workflow endpoint to host the definition.

- Navigate to Connections -> Workflow Endpoints
- Click the New button to create a new Workflow Endpoint named Faulty Workflow Endpoint
- On the General tab
    - Check the Enabled checkbox
    - Select Faulty Workflow from the Workflow Definition dropdown list

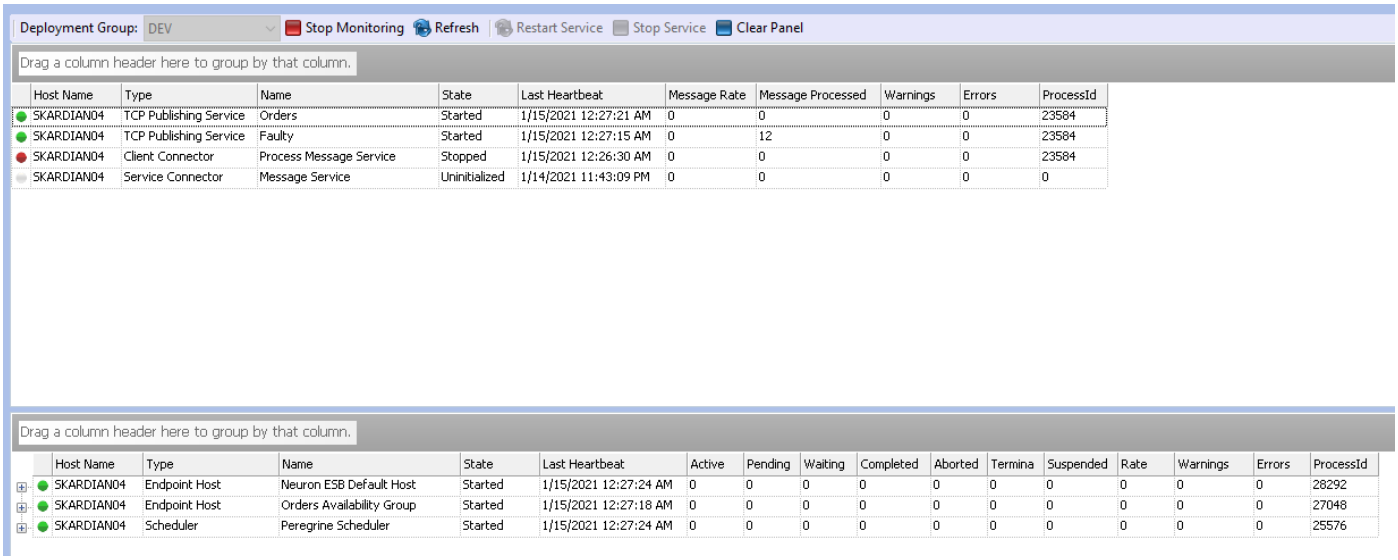
- Select Faulty Subscriber from the Subscriber dropdown list
- Select Faulty from the Topic dropdown list
- Select Neuron ESB Default Host from the Endpoint Host dropdown list



- Apply your changes
- Save the Neuron ESB configuration

With all of the entities that we will need created we need to setup the environment to make sure that the initial execution of the workflow will fail. To do that we need to stop the client connector that we created to act as our service.

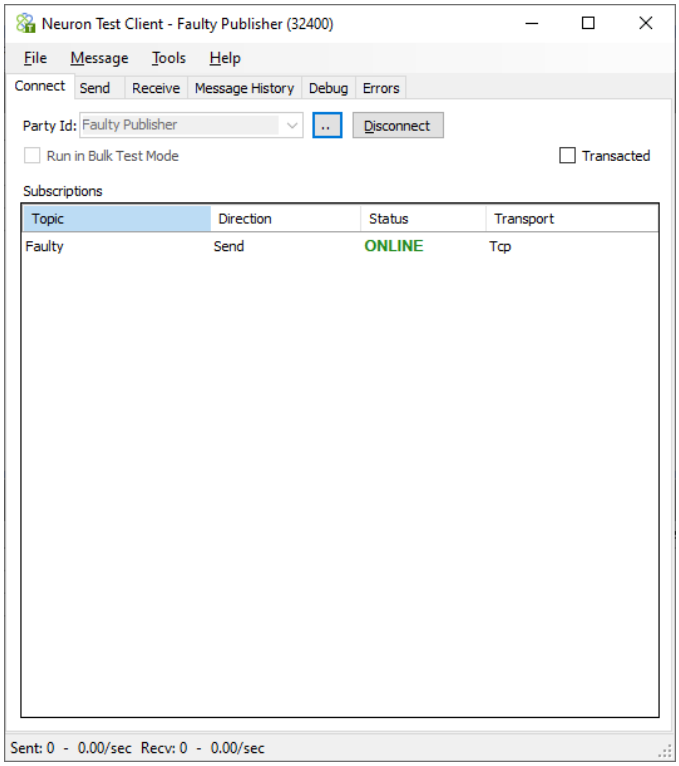
- Navigate to Activity -> Endpoint Health
- Click Start Monitoring
- Right click the Client Connector named Process Message Service and select Stop Service



Run the Exercise

To run this exercise

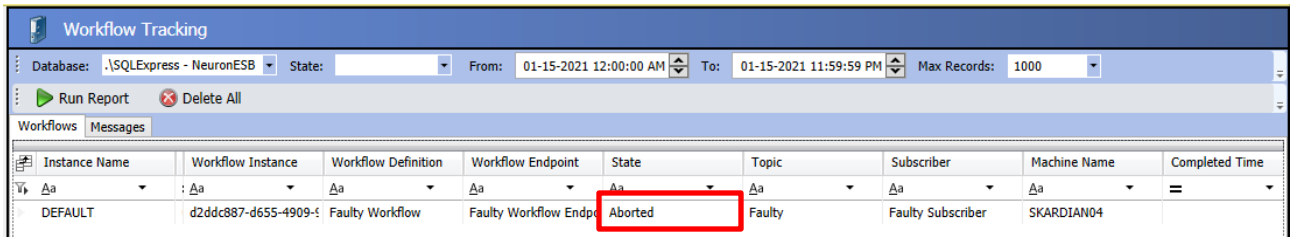
- Launch a test client and connect it as Faulty Publisher.



- On the send tab of our Test Client we will enter the following message

```
<Message>
      Please process me
</Message>
```

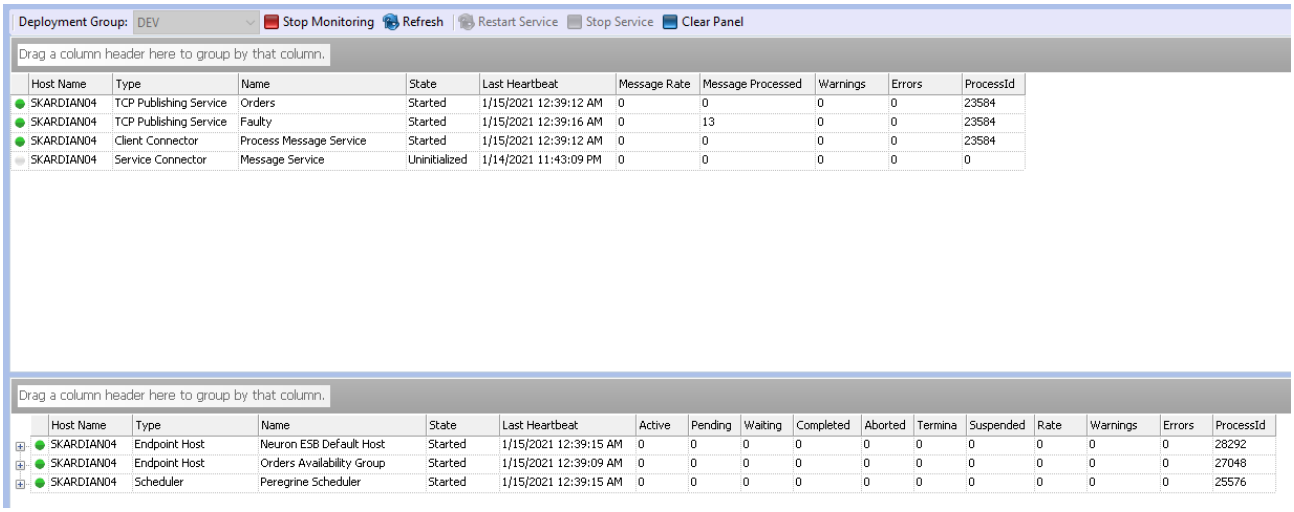
- Click Send
- Return to the Neuron ESB Explorer
- Navigate to Activity -> Workflow Tracking
- Click Run Report



Notice that we have an entry for our workflow instance, but the state is set to aborted. That is because the service that we are attempting to call is currently unavailable. We can resolve the issue that caused the workflow to abort by restarting the client connector.

- Navigate to Activity -> Endpoint Health
- Click Start Monitoring

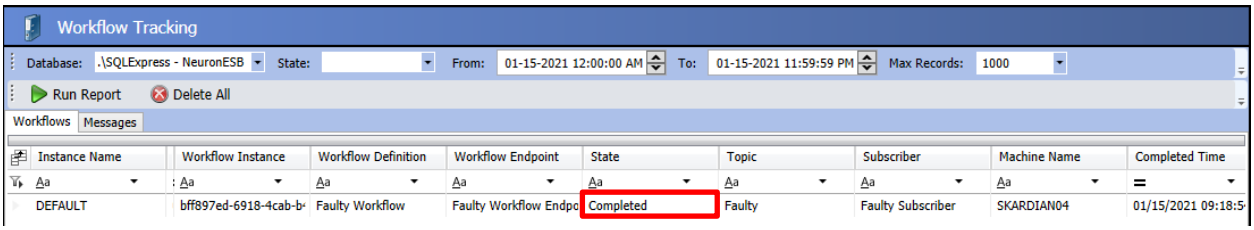
- Right click the client connector Process Message Service and select Restart Service



What about the workflow instances that failed while the service was unavailable? We can start those instances back up from Workflow Tracking, and they will resume execution from the last persistence point they encountered, or from the beginning if no persistence points were executed.

- Navigate to Activity -> Workflow Tracking
- Click Run Report
- Right click on the aborted workflow instance and select Start

Notice that the Workflow status now indicates that the Workflow Instance has completed successfully.



Review

Neuron ESB provides Workflow capabilities that allow companies to design fault tolerant, business resilient workflows to automate critical processes that may span hours, days, weeks or months and cross inter or intra company domains. Neuron ESB's Workflow offering is built upon Microsoft .NET Workflow Foundation 4.5, overlaying it with tools, infrastructure, hosting environment and services necessary to deliver enterprise level performance and scalability on the Microsoft .NET Platform.

Prior to using Workflows in Neuron ESB, make sure your Neuron ESB server has these installed/available:

- Management Objects (from the Neuron ESB installer)

- Neuron ESB Database must be configured for the active Deployment Group

Creating a workflow requires these artifacts to be created in Neuron ESB:

- Messaging
    - Topic
    - Subscriber (to associate with the workflow endpoint)
    - Publisher (to send messages to Neuron ESB that the subscriber will receive)
- Endpoint Host
- Workflow Endpoint
- Workflow Definition

Endpoint Hosts define which Neuron ESB servers will execute the workflows. If there is more than one Neuron ESB Server defined for a Deployment Group, you can select which servers are primary and which ones are failover servers.

When you deploy a Workflow Definition, you have to create a Workflow Endpoint. Workflow Endpoints are similar to Adapter or Service Endpoints – they are associated with a subscriber and subscribe to messages from a topic. Workflow Endpoints allow you to set the maximum number of concurrent workflows that will be executing. Finally, Workflow Endpoints are configured to be hosted by an Endpoint Host. An Endpoint Host can host multiple Workflow Endpoints.

There are three types of workflows:

- Normal
- Request/Reply
- Correlated

A Long-Running transaction is a Normal workflow that may take hours or even days to complete. When a Normal workflow publishes a multicast message and initializes a Correlation Set, the following receive activity will wait for the response. If the response isn't received within a short time, the workflow instance will be unloaded and saved to the database. When the response message is received, Neuron ESB will restart the correct workflow instance for further processing.

Quiz

1. When you create a new workflow, what are the three types of workflows that you can create?
2. For a Normal Workflow, name the three arguments available for use in the workflow.
3. For the Correlated Send and Receive pattern, which property set from the Publish Message activity needs to be assigned to the Receive Message activity?
4. Would you use the Request-Reply Workflow type to implement the Correlated Send and Receive pattern? Why?
5. You have a simple Request-Reply Workflow associated with a subscriber. Five messages are sent to that subscriber. How many instances of the workflow will you see in Workflow Tracking?

6. True or False. For a Correlated Workflow you need a Receive Message activity inside a While (or DoWhile) to determine when the workflow has processed all correlated messages.
7. A workflow executes in a Neuron Endpoint Host process that is a child of the Neuron ESB Windows Service. What screen in Neuron ESB Explorer do you use to configure one or more Neuron Endpoint Host processes?
8. In Neuron ESB Explorer, what ties together a Workflow Definition, Subscriber, Topic and Endpoint Host and allows for configuration of a Correlation Set when using a Correlated Workflow?
9. In order to use Workflow what two things are required?
10. True of False. When using the Publish Message activity in a workflow you need to make sure that the Subscriber party used in the Workflow Endpoint has a subscription to Send on the topic configured with activity.
11. Can a party used as a workflow subscriber also have a business process associated with it?
12. True or False. A Correlation Set can only contain one custom or one header property.

Appendix

The following Artifacts accompany this training:

- Neuron Workflows Answers.docx - Word Document with answers to the quiz
- Neuron Workflows Solution - Directory with the completed solution from the exercises in this guide.