



Neuron ESB: An Enterprise Service Bus for the Microsoft Platform

This paper describes Neuron ESB, Neudesic's ESB architecture and framework software. We first cover the concept of an ESB in general in Part 1 before introducing the Neuron ESB model in Part 2.

Part 2: The Neuron ESB

Neuron ESB is the code name for Neudesic's Enterprise Service Bus initiative. Neuron ESB is both an ESB architecture and ESB software. Neuron ESB is a full-featured ESB whose scope includes the following:

- Interconnection of applications, services, and legacy systems
- Intelligent routing
- EAI functionality
- Centralized management
- SLA monitoring and enforcement
- Deployment
- Compliance auditing

An ESB for the Microsoft Community

Neuron ESB is a Microsoft-friendly ESB. Whereas most ESBs seem to originate outside of the Microsoft community, Neuron ESB was developed by a Microsoft partner and provides great out-of-box support for the Microsoft technology stack.

The Neuron ESB framework software is .NET-based and has built-in support for WCF, WF, MSMQ, BizTalk Server, and SQL Server. Customers aren't required to use all of those technologies: Neuron ESB will leverage any combination of them a customer has an investment in.

Neuron ESB is a true ESB with a pure architecture. Its Microsoft-readiness is achieved without violating ESB tenets, which we've been careful to outline in Part 1 of this paper. The infrastructure services that ship with Neuron ESB have a Microsoft implementation, but customers are free to change the implementation of any service.

Architecture

The ESB design pattern connects enterprise business systems with underlying communication and integration technologies through a framework of infrastructure services that execute a common plan. Neuron ESB specifies an architecture for combining these elements, defines metadata for expressing configuration and policies, and includes framework software that has a Microsoft implementation.

WHITE PAPER (DRAFT 1, 10/06)

The Neuron ESB software consists of infrastructure services and .NET assemblies which are modular, replaceable, and extensible. Working in concert, they provide a connection and integration backbone for the enterprise and an operating environment for services.

Figure 1 illustrates the Neuron ESB architecture. The ESB provides a common messaging fabric that interconnects several categories of endpoints:

- Business endpoints include applications, services, and legacy systems.
- Integration services perform useful EAI functions that can be applied to messages as they are en route.
- Infrastructure services implement the ESB, providing endpoint connections, security enforcement, configuration management, message routing, and operations monitoring.

BUSINESS ENDPOINTS

Business endpoints can take several forms. Native applications directly target the ESB using the .NET or JMS API and are able to take full advantage of the ESB's advanced messaging features. Services, including first generation web services and second generation WS-* services, can be connected to the ESB. The ESB creates service connector ports to talk to services and clients using their preferred connection method, message format, and security model. The ESB can connect to legacy systems through the use of adapters or via integration hosts such as BizTalk Server. The ESB can communicate with other messaging systems through the use of WS-Eventing subscription management.

Integration Services

Integration services provide EAI functions such as schema validation, message transformations, activity monitoring, rules engine execution, and workflow orchestration. The integration services provided with Neuron ESB utilize BizTalk Server, the Microsoft Rules Engine, and Windows Workflow Foundation to provide this functionality. Since the EAI functions are exposed as services, customers are always free to change the implementation of any individual integration service.

Infrastructure services implement the ESB framework that connects business endpoints, integration services, and underlying technologies. Infrastructure services are physically decentralized but enforce a common plan for the enterprise that includes configuration, security, auditing, routing, and policy compliance.

Architecture continued :

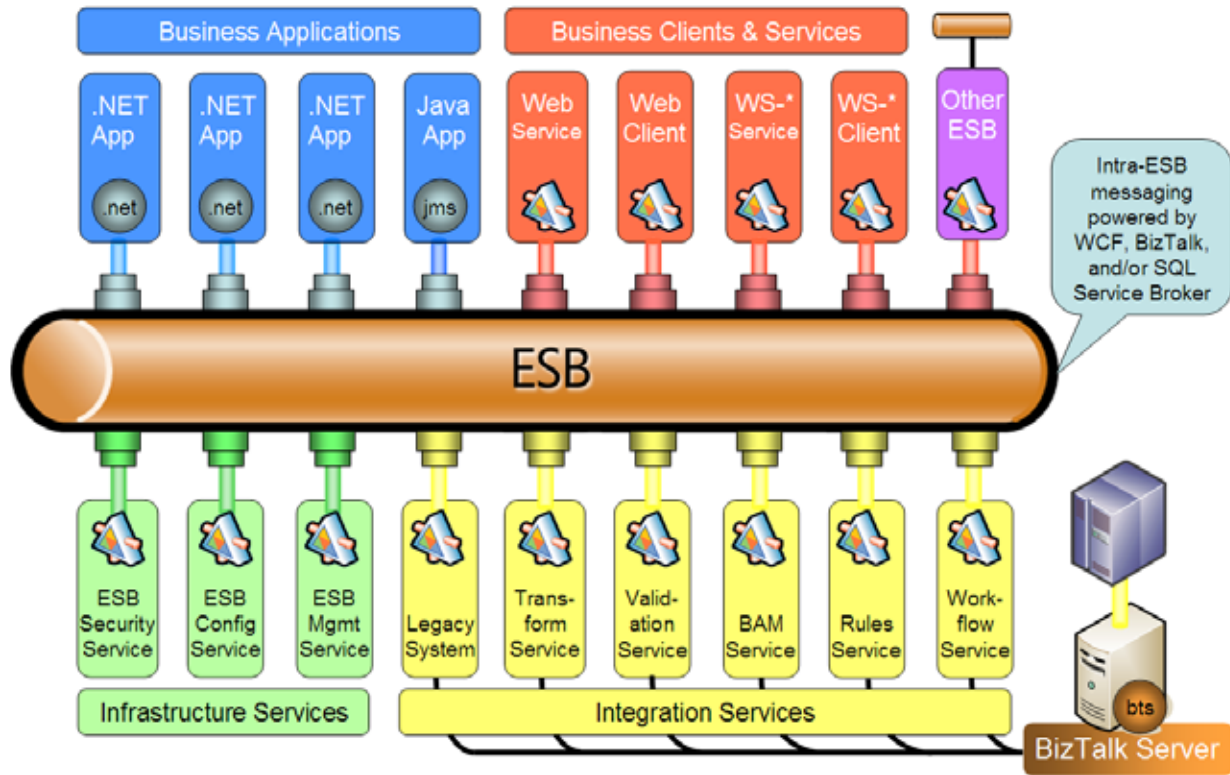


Figure 1: Neuron ESB Architecture

Core Messaging System

The core messaging system is responsible for connecting endpoints and routing messages between them. Applications share messages through publish-subscribe communication. The core messaging system has a unique design that is decentralized and modularly replaceable.

PUBLISH-SUBSCRIBE MESSAGING

Neuron ESB's core messaging system is inherently based on the principle of publish-subscribe topical communication. In publish-subscribe, senders and receivers of messages have no direct knowledge of each other. Subscribers send ("publish") messages over a logical topic name, and all interested parties receive a copy of those messages. Subscriptions are used to indicate an application's interest in a topic.

Topics can have a hierarchy. Neuron ESB topic names can be simple text strings such as orders or inventory; or multi-part names separated by a period or slash such as orders.new or inventory/re-orders/west. Subscribers can express their interest in a very granular way through the use of wildcarding. For example, a subscription to sales.orders.* will match messages published on sales.orders.new and sales.orders.cancelled but not sales.leads.

Figure 2 shows publish-subscribe in action. The service on the top left sends a message over the topic orders.new. Some systems receive copies of the message and others do not based on their standing subscriptions.

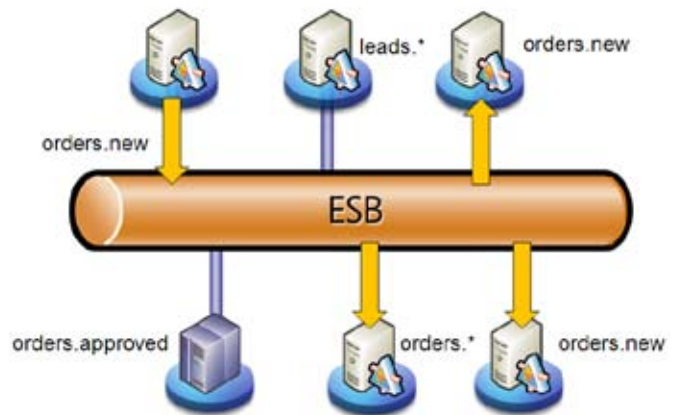


Figure 2: Publish-Subscribe communication

Core Messaging System continued:

The benefits of this arrangement cannot be overstated. Senders and receivers are loosely-coupled and the connections between them are easily changed. Adding a new application to the enterprise and hooking it into an existing flow of communication becomes a simple matter of creating a new subscription. Applications no longer need to know the addresses or other systems. Communication and routing details are no longer embedded in software and become part of central configuration, which can be changed while systems are running live. In an ESB setting, publish-subscribe provides you with an enterprise-wide event system.

While publish-subscribe is a versatile and powerful way to communicate, much software is written to use more traditional communication patterns such as one-way messaging, request-reply, and queued messaging. Neuron ESB supports these methods of communication as well. For example, an application can publish a request message to a topic and await a reply from one of the receivers.

Neuron ESB's routing capabilities are not limited to publish-subscribe. Messages can also be routed based on payload type or content. The various routing mechanisms are combinable. For example, a subscriber could subscribe to the topic orders.new but specify that only serialized Order v2 objects

TOPIC NETWORKS

The Neuron ESB core messaging system is as distributed and modular as the rest of the ESB. Many otherwise fine ESBs have an Achilles Heel: their core messaging system is intractably tied to a single underlying technology. This is bad for several reasons: it violates the modular, product-agnostic philosophy of an ESB; it limits the ability to harness best of breed technologies and adapt to new ones; and it tends to limit the enterprise to a single quality of service for messaging. In contrast, Neuron ESB provides a choice of communication technologies, allows them to be extended and combined, and can run multiple qualities of service in parallel.

The Microsoft platform offers many past and new technologies for communication and integration with some overlapping functionality between them. While it would have been easy to select one of them and build an ESB around it, we felt the higher road to take was to embrace the full Microsoft stack and offer customers the full range of capabilities available to them. To avoid tying the ESB to any single technology, Neuron ESB supports 4 communication technologies out-of-box with the ability to add additional ones. Neuron ESB can perform publish-subscribe messaging over WCF PeerChannel, MSMQ, BizTalk Server, or SQL Server 2005. WCF PeerChannel provides low-latency, volatile, best effort messaging that scales well and requires no server infrastructure. MSMQ, BizTalk Server, and SQL Server provide transacted, durable messaging with a variety of scalability and reliability characteristics. A lightweight .NET assembly with a simple interface integrates the ESB with these underlying technologies, many of which have their own channel models or adapter facilities.

Neuron ESB considers each publish-subscribe topic to be a separate network. Each topic network is separately configured, which allows customers to run a mix of communication technologies to power their ESB. Figure 3 shows an ESB with a mix of channel types in use. The ability to run several different qualities of service in parallel allows the right messaging characteristics to be used for each business conversation.

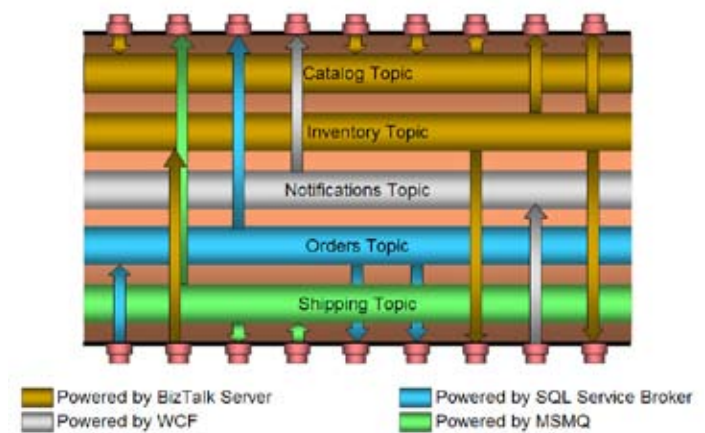


Figure 3: Topic Networks

The topic network model scales beautifully: each topic has its own messaging configuration and can specify different hosting servers.

Messages can be shared across topic networks through the use of bridges. Bridges receive messages on one topic and pass them on to another topic. Bridges are created by the ESB automatically based on configuration.

ENDPOINT CONNECTIONS

Business endpoints communicate with the ESB by going through a client stack. The client stack interacts with the ESB infrastructure services and connects an endpoint with a topic network for messaging. When an endpoint connects to the ESB, its client stack checks in with the ESB infrastructure services for authorization and obtains configuration information. The client stack then uses a channel factory to create an appropriate connection a topic network. In the case of WCF PeerChannel, the client joins a peer-to-peer cloud specific to a topic. In the case of the other messaging methods, messages are sent and received via queues.

Figure 4 shows how different kinds of endpoints make their connection to the ESB.

- Native applications use the .NET API create their own client stack, which connects to the infrastructure services and topic network using their native interfaces and message formats.
- Services (and clients) are hosted by service connectors that match a service's communication semantics and contracts.

Intelligent Routing continued:

A service connector contains a client stack and passes messages between the service and the topic network. Service connectors are spun up automatically by the ESB in response to configuration.

- Legacy systems are integrated using adapters or EAI products like BizTalk. Enclosing the adapter mechanism in a WS-* service is the preferred arrangement so that the ESB maintains its use of open standards to communicate with all endpoints. A service connector can then be used to treat the legacy system like any other service.

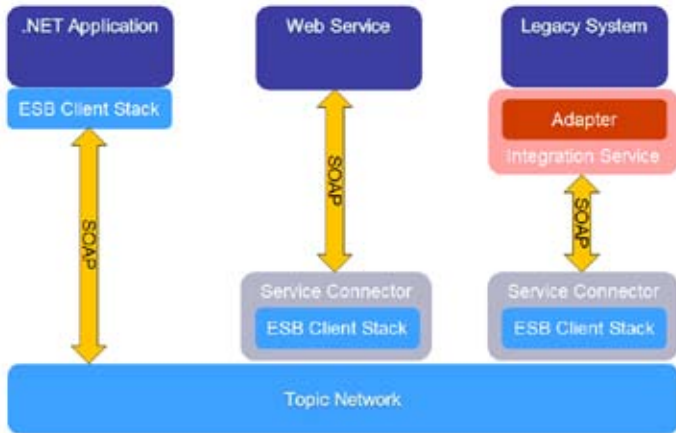


Figure 4: Endpoint Connections

Integration and EAI Services

Integration services provide useful functions such as EAI functionality and compliance auditing for messages that are en route.

- Validation services validate the schema of a message.
- Transformation services transform messages from one schema or format to another.
- Rules services apply rules to a message and may transform a message.
- Activity monitoring services record activity from the perspective of SLA monitoring, business activity monitoring, or compliance auditing.
- Workflow services execute a workflow or orchestration in response to an initiating message.
- Legacy integration services connect to a legacy system and expose it as a service.

Integration services are open-ended and customers may define new types beyond those listed above. 1-way services are used for notification purposes such as activity monitoring, auditing, and kicking off orchestrations. 2-way services are used to modify messages as a result of their processing.

Combinations of integration actions are known as a pipeline. The configuration for a subscriber can specify a pipeline of integration actions that occur at send time or receive time. At send time,

these actions are applied in after an endpoint sends a message but before it is routed over the ESB. At receive time, these actions are applied after the ESB receives a message but before it is delivered to the endpoint.

Figure 5 shows how pipelines might be used an order processing scenario. As orders are sent over the ESB, some parties have need of integration functions:

- When the Order Entry application sends a message, a send-time pipeline uses a rules engine to detect fraudulent orders.
- When the CRM service receives a message, a receive-time pipeline validates the message schema and transforms orders into contacts.
- When the order approval service receives an order, a receive-time pipeline converts orders into an older format the service understands.
- The order processing service needs no pipeline because it and the order entry application use the same order format.

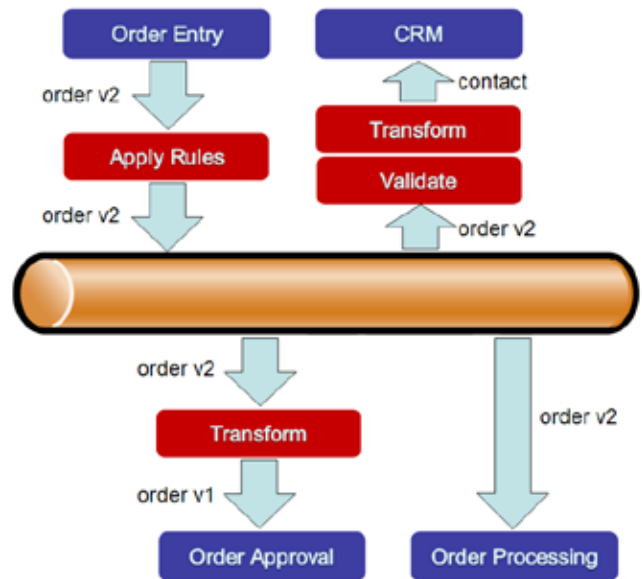


Figure 5: Pipelines invoke integration services to provide EAI functions

The integration services provided with Neuron ESB have BizTalk Server and Windows Workflow Foundation implementations. Since all EAI functionality is exposed as services, customers are free to change the implementation of any integration service or add new ones.

Management:

Neuron ESB provides a powerful management tool named ESB Explorer. Using ESB explorer, administrators can view, configure, monitor, and control the complete Enterprise Service Bus with a single tool. Figure 6 shows four views of the explorer: a visualization of the ESB, configuration of a topic; configuration of a subscriber; and activity monitoring.

hierarchy is configured as a topic network that communicates over WCF PeerChannel, MSMQ, BizTalk Server, or SQL Server.

- **Subscribers** which represent business applications or services. Each subscriber has subscriptions that control the topics over which they can send or receive messages.

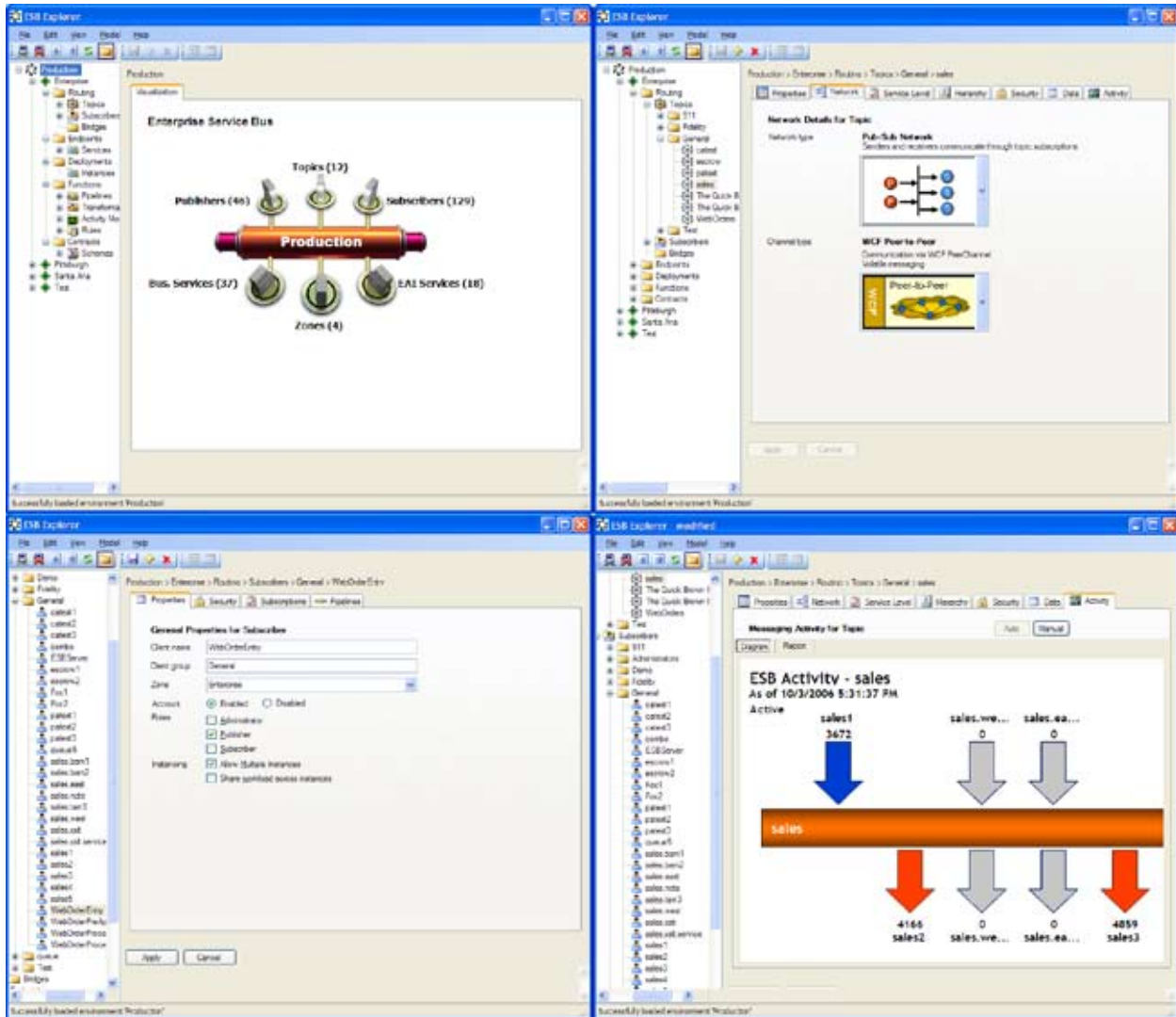


Figure 6: Management UI

CONFIGURATION METADATA

The ESB Explorer manages configuration for the following entities:

- Deployment entities
 - **Zones** that separate the ESB into separately deployable units. Each zone is powered by its own set of ESB servers. Zones allow regional areas of large enterprises to run independently of each other.
- Routing entities
 - **Topics** for publish-subscribe communication. Each topic
- **Bridges** that connect information across topics in the same zone or across zones.
- Endpoints
 - **Applications** that use the ESB. The configuration maps applications to subscriber roles and can also specify how they are monitored for health or how new instances can be deployed.
 - **Services** that connect to the ESB. The configuration maps services to subscriber roles and also specifies connection and contract information.

Configuration Metadata continued:

- **Deployments** which represent deployed instances of applications of services.
- Integration Functions
 - **Pipelines** which are packaged sequences of integration functions.
 - **Transformations** which transform messages en route.
 - **Rule sets** which apply rules to messages en route.
 - **Activity monitors** which track technical, compliance, or business activity.
 - **Orchestrations** which execute transactional business processes.
- Contracts
 - **Service contracts** that define how services and clients expect to be communicated.
 - **Schema** that define XML message structures.
- Operations
 - **Credentials** for security mechanisms such as certificates.
 - **Policies** to be enforced by the ESB.
 - **Service level agreements** the ESB is to monitor or enforce.

Configuration information is stored as XML. In fact, an entire ESB configuration can be emailed as a file attachment.

Once configuration changes have been made in the ESB Explorer, their effect can be modeled to ensure proper behavior before they are committed. Once changes are committed, the Neuron ESB server(s) are notified and reconfigure themselves. Nearly all configuration changes can be applied in a live environment and take effect immediately.

Anything that can be accomplished in the ESB Explorer can also be achieved in code using the .NET administration API.

How Neuron ESB Uses the Microsoft Technology Stack

The software that makes up the Neuron ESB framework is implemented in Microsoft .NET, much of it exposed as modular WCF services. Neuron ESB also integrates with many other parts of the Microsoft technology stack as described below, but they are optional.

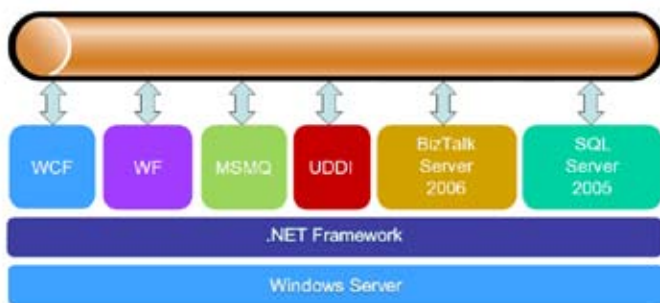


Figure 7: Areas of the Microsoft Technology Stack Integrated with Neuron ESB

WINDOWS COMMUNICATION FOUNDATION (WCF)

Services figure prominently in an ESB so it is natural that Neuron ESB leans heavily on WCF, Microsoft's premier technology for service-oriented communication. Neuron ESB uses WCF to implement and invoke its own infrastructure services. WCF is also used to connect first and second generation services and clients to the ESB. WCF's support a wide spectrum of protocols and standards allows Neuron ESB to connect well to any first or second generation service.

WCF PEERCHANNEL

WCF PeerChannel is one of the channel types provided for publish-subscribe communication. It provides volatile, best-effort messaging, meaning only subscribers currently connected to the ESB will receive messages. PeerChannel uses a unique combination of UDP and TCP to create server-less clouds of communication between systems and fast low-latency communication. Neuron ESB creates a unique cloud for each topic that has been configured to use PeerChannel. PeerChannel does not guarantee reliability, but Neuron ESB include an optional reliability layer that corrects out-of-sequence messages and missed messages using a message cache.

BIZTALK SERVER 2006

EAI functionality is a key ingredient in an ESB, and BizTalk Server provides Neuron ESB with unrivaled EAI functionality. Neuron ESB's integration with BizTalk strongly leverages BizTalk's capabilities in many areas.

Neuron ESB provides integration services that are WCF services externally and BizTalk integrations internally.

- The BizTalk Mapper Service transforms messages based on transformations created with the powerful BizTalk Mapper. This includes not only XSLT transformations but also BizTalk-specific functoids.
- The BizTalk Rules Service executes rules defined in the Microsoft Business Rules Composer that have been deployed as policy assemblies.
- The BizTalk BAM Service keeps BizTalk Business Activity Monitoring (BAM) databases up to date, providing business stakeholders with real-time access to key performance indicators.

BizTalk can also serve as the core messaging engine for some or all of Neuron ESB's topic networks. Neuron ESB endpoints send or receive messages through queues, and BizTalk serves as the publishing engine. In this context BizTalk provides the ESB with a powerful multi-threaded messaging engine with excellent scalability and reliability.

Solutions created in BizTalk such as an orchestration can communicate topically using the Neuron ESB BizTalk Adapter. The adapter allows BizTalk solutions to send or receive on the ESB using topical publish-subscribe messaging. Thus, BizTalk is able to communicate over any of Neuron ESB's channel types (such as WCF PeerChannel) providing new capabilities.

SQL SERVER 2005:

SQL Server 2005 provides many facilities for web service communication and message processing. Customers with an investment in SQL Server can leverage it as part of the ESB infrastructure.

SQL Server's Service Broker can serve as the core messaging engine for some or all topics. Neuron ESB endpoints send or receive messages through SQL message queues through HTTP endpoints, and SQL Server is the publishing engine. For distributing the messaging, the servers hosting publishing message queues, the publishing engine, and delivery message queues can all be different. Each topic can be powered by a different collection of servers if desired.

Since SQL stored procedures can now contain .NET code, a stored procedure may send or receive messages over the ESB using Neuron ESB's .NET API.

Other integrations to showcase SQL Server are in the works, include the following:

- Ability to integrate ESB messages as a source for SQL Reporting Services.
- Ability to fire a SQL trigger in response to an ESB message.

MSMQ

Microsoft Message Queue is one of the channel types that can power a topic network. Endpoints send and receive over queues. Neuron ESB provides the publishing engine that moves messages between queues based on subscriptions. Messages are stored in durable, transacted queues.

WORKFLOW FOUNDATION

Some of the capabilities normally associated with BizTalk have become available in Workflow Foundation. Neuron ESB supports the use of BizTalk and/or Workflow Foundation. Integration services for running workflow orchestrations or executing rules are available in both BizTalk and Workflow implementations.

Workflow foundation is also the recommended way to implement a Service Level Agreement (SLA) monitoring service or enforcement service. The Workflow can use the Neuron ESB .NET management API to repeatedly check health, throughput rates, and error levels and take action where needed. That action might include notifying a human being, integrating with an operations management system, or failover actions such as routing changes or deployment of new application instances.

MICROSOFT OFFICE

Neuron ESB integrates with Microsoft office to provide documents that can accept information in real-time and are always up to date. For example, an Excel integration maps the data in received messages to worksheet cells using XPath rules.

UDDI

Neuron ESB's repository of information includes endpoint management information such as the location and configuration of services. Since that information may already be contained in UDDI directories, the Microsoft UDDI toolkit is used to integrate with them.

WINDOWS SERVER

Neuron ESB's server software is a distributed collection of services that can be deployed to one or more Windows Servers. Windows is the support platform for all of the technologies listed above.

The core messaging system in Neuron ESB is based on integrated Windows security. Although the connections of endpoints to the ESB may exercise a range of security models, all entities are ultimately authenticated and authorized using Windows security.

SUMMARY

The Enterprise Service Bus phenomenon has been under-represented on the Microsoft platform. Neuron ESB provides a turnkey solution, an architecture and software for a full featured ESB with comprehensive support for Microsoft technologies.

David Pallmann is an Architectural Consultant for Neudesic LLC, a Microsoft Gold Partner consulting firm headquartered in Southern California. Copyright © 2006 by Neudesic LLC. All Rights Reserved.