WebSphere MQ Adapter for Microsoft Dynamics AX 2012
MA7R

User Guide
Version 1.0

Take Note!

Before using this User's Guide and the product it supports, be sure to read the general information under "Notices".

First Edition, February 2013

This edition applies to Version 1.0 of *WebSphere MQ Adapter for Microsoft Dynamics AX 2012* and to all subsequent releases and modifications until otherwise indicated in new editions.

Notices

The following paragraph does not apply in any country where such provisions are inconsistent with local law.

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

WebSphere® MQ

IBM®

AIX®

OS/400®

MVS™

z/OS®

The following terms are trademarks of the Microsoft Corporation in the United States and/or other countries:

Windows Server 2008 R2, Windows XP
Dynamics AX 2012

Table of Contents

About this book

IBM® WebSphere MQ adapter for Microsoft Dynamics AX 2012 describes and documents the features provided.

This book describes the following:

- Introduction to WebSphere MQ adapter for Dynamics AX
- Installation and Un-installation of WebSphere MQ adapter
- Configuration of WebSphere MQ adapter
- Troubleshooting
- How to use samples

For the latest information about WebSphere MQ Adapter for Microsoft Dynamics AX 2012, see the product readme.txt file, which is supplied.

Who this book is for

This book is primarily for Administrators of Microsoft Dynamics AX who can deploy and configure the adapter for integration of Dynamics AX 2012 with external systems over WebSphere MQ Transport.

This book can also be referred by programmers who write MQ applications that invoke Dynamics AX services asynchronously.

What skills and knowledge you need to understand this book

To understand the information in this book, you need the following skills, knowledge and experience:

- IBM WebSphere MQ
- Microsoft Dynamics AX 2012
- Application development skills using IBM WebSphere MQ API in C, C++, Java, C#.
- Some knowledge of the Java™ Message Service Specification, Version 1.1 and the WebSphere MQ implementation of JMS, WebSphere MQ classes for Java Message Service, might be beneficial, but is not absolutely necessary. You do not need to be a Java or JMS application programmer.

Terms used in this book

The following are the terms used in this book.

- The term adapter refers to IBM WebSphere MQ Adapter for Microsoft Dynamics AX 2012.
- The term Dynamics AX refers to Microsoft Dynamics AX 2012.
- The term AIF refers to Microsoft Dynamics AX Application Integration Framework.

- The term WMQ refers to IBM WebSphere MQ.
- The term WebSphere MQ JMS refers to WebSphere MQ classes for Java Message Service.

Supported Platforms and Software

IBM WebSphere MQ Adapter for Dynamics AX is developed using Microsoft Dynamics AX Integrated Development Environment and Microsoft .NET Framework 3.5.

Prerequisites

- IBM WebSphere MQ v7.1.0.1 Client or higher
- IBM WebSphere MQ Queue Manager v7.0.1 or higher
- Microsoft Dynamics AX 2012.
- Microsoft .NET Framework v3.5 or higher

Refer to the WebSphere MQ information center for prerequisites of IBM WebSphere MQ.
Refer to Microsoft Dynamics AX 2012 website for prerequisite details.

1. Introduction to WebSphere MQ Adapter for Dynamics AX

This document describes the functions available in this SupportPac. Some knowledge of WebSphere MQ and Microsoft Dynamics AX is assumed. For information about WebSphere MQ, refer to the *WebSphere MQ Information Center*. For information about Dynamics AX 2012, refer to the Microsoft Dynamics AX online documentation.

1.1 Microsoft Dynamics AX Overview

Microsoft Dynamics AX is a complete ERP solution for enterprises that provides a purpose-built foundation across five industries, along with comprehensive, core ERP functionality for financial, human resources and operations management. It empowers organizations to anticipate and embrace change so that business can thrive. All of this is packaged in a single global solution giving organizations rapid time to value.

Customers use different channels to invoke services provided by Dynamics AX. Internet clients (Web services based) over HTTP using Internet Information Server, Dynamics AX Client GUI, Office Add-in for Microsoft Excel, Word, Enterprise Portal, BizTalk Server are some of the channels that customers use. AX services can also be asynchronously invoked using Microsoft Message Queuing (MSMQ) and Files.

Microsoft Dynamics AX Application Integration Framework (AIF) enables companies to integrate and communicate with external business processes and partners through the exchange of XML over various transport media. AIF enables both business-to-business and application-to-application integration scenarios.

The capability to integrate Microsoft Dynamics AX with other systems inside and outside the enterprise is a common requirement. Application Integration Framework (AIF) provides this capability by enabling the exchange of data through formatted XML. This formatted XML is referred to as a document, and each document contains data and business logic. Documents are based on a *Document* class and defined using Microsoft Dynamics AX.

1.2 Dynamics AX Integration with other systems

AIF provides an extensible framework for the exchange of XML documents with external systems. The framework supports both synchronous and asynchronous transports. In synchronous mode, requests are tightly coupled to responses. This means that the submitter of the request must wait for a response from AIF before proceeding. In this case, AIF immediately processes the request and then sends a response. In asynchronous mode, requests are placed into a queue, called the gateway queue. Queued messages are processed at a later time and AIF sends a response when processing is completed. In this case, responses are delayed, but large volumes of messages can be processed more efficiently, and message processing can be controlled by changing various configuration settings.

AIF can be used to send data into Microsoft Dynamics AX. This kind of exchange is

called an inbound exchange. For example, during an inbound exchange, an external system may send a sales order so that the sales order can be saved to the Microsoft Dynamics AX database. AIF can also be used to retrieve data from Microsoft Dynamics AX. This kind of exchange is called an outbound exchange. For example, during an outbound exchange, an external system may send a request for a purchase order and receive the purchase order.

1.3 Integration Ports

In Microsoft Dynamics AX, integration ports provide simplified administration of services and Application Integration Framework (AIF). Integration ports replace the AIF endpoints and related concepts that were used in previous releases of Microsoft Dynamics AX. Each integration port can expose one or more services, and each integration port has a unique Uniform Resource Identifier (URI) that identifies the address of the port.

Each integration port also has a direction. An integration port can be either an inbound integration port or an outbound integration port. An inbound integration port is a destination for messages that originate from outside Microsoft Dynamics AX. An outbound integration port is a destination for messages that originate from your Microsoft Dynamics AX system. Inbound integration ports can be one of two types: basic or enhanced. Outbound integration ports are always enhanced ports.

1.4 Basic integration ports

Basic integration ports are exposed at a specific Windows Communication Foundation (WCF) endpoint on the Application Object Server (AOS) host. Only a developer can create a new basic integration port. When a developer creates an automatically deployed service group in the Application Object Tree (AOT), a basic inbound integration port is also automatically created. The basic inbound integration port is then associated with the service group that is being deployed. Some basic integration ports, such as those that are used for system services, are always deployed and enabled by default.

1.5 Enhanced integration ports

If you want advanced integration capabilities that you can use to customize the behavior of an integration port, you must create an enhanced integration port. Enhanced integration ports provide the following capabilities:

- Ability to host services on AOS or IIS
- Supported protocols include HTTP, NetTCP, MSMQ and File System
- Ability to pre and post processing of service requests and service responses
- Customize data contracts

1.6 Adapters

In Microsoft Dynamics AX services and Application Integration Framework (AIF), integration ports use adapters. These adapters enable Microsoft Dynamics AX to communicate by using various transport protocols.

Microsoft Dynamics AX provides four adapters that represent predefined bindings. HTTP and NetTCP adapters provide for synchronous message exchanges while MSMQ and File system adapters provide asynchronous message exchanges.

Custom adapters can also be developed. AIF defines a set of standard interfaces for developing custom adopters. Those interfaces define a set of methods that AIF will use for connecting to an external system and exchanging data. These interfaces are *AifIntegrationAdapter, AifReceiveAdapter* and *AifSendAdapter*.

1.7 WebSphere MQ Adapter for Microsoft Dynamics AX 2012

This section describes the architecture of WebSphere MQ Adapter for Microsoft Dynamics AX 2012.

WebSphere MQ (WMQ) offers an asynchronous, reliable, once and once and only once message delivery, secure connectivity between applications. WebSphere MQ can run on a number of well known platforms like AIX, Linux, z/OS and Windows. WebSphere MQ Adapter for Microsoft Dynamics AX 2012 is a custom adapter that makes WebSphere MQ as a reliable asynchronous transport for invoking Dynamics AX 2012 business services. As WebSphere MQ can run on a number of non-Windows platforms, applications running on non-Windows platforms can also invoke Dynamics AX 2012 services. Thus WebSphere MQ adapter opens up integration possibilities for a number of systems running on non-Windows platforms.

WebSphere MQ Adapter for Microsoft Dynamics AX 2012 implements the following interfaces:

1. AifIntegrationAdapter
2. AifReceiveAdapter
3. AifSendAdapter

The AifIntegrationAdapter interface is implemented as AifWmqIntegrationAdapter. This is the top level interface that is invoked by the Dynamics AX Application Integration Framework to retrieve other interfaces implemented by the WebSphere MQ adapter. This interface also does the following:

- Displays the custom panel for URI configuration
- Validation of the configured URI

The AifReceiveAdapter interface is implemented as AifWmqReceiveAdapter. This interface is responsible for receiving messages from a WebSphere MQ queue and forwards those messages to the Dynamics AX 2012 gateway queue.

The AifSendAdapter interface is implemented as AifWmqSendAdapter. This interface is responsible for getting response messages from the Dynamics AX gateway queue and sends them to a WebSphere MQ queue.

The architecture diagram below describes the communication between Microsoft Dynamics AX 2012 and WebSphere MQ.

The Application Integration Framework invokes the WebSphere MQ adapter's AifIntegrationAdapter implementation and retrieves the implementation of AifReceiveAdapter and AifSendAdapter by calling the getReceiveAdapter and getSendAdapter methods respectively.

1.7.1 Validation of URI

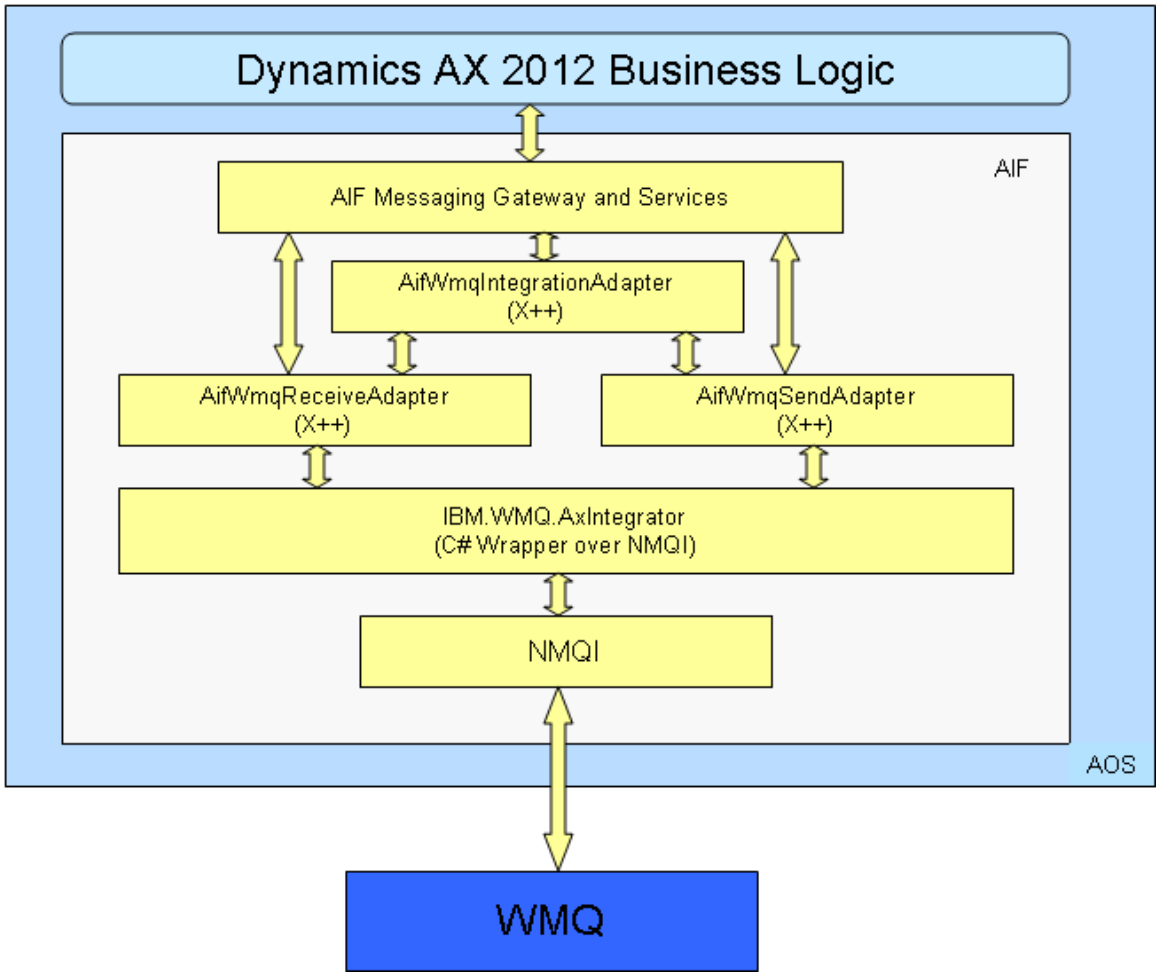
To validate the URI provided in the configuration panel, AIF calls the validateConfiguration method of the AifIntegrationAdapter interface. In this method the adapter does the validation of the URI.

1.7.2 Receive messages from a WebSphere MQ queue

When receiving messages from a WebSphere MQ queue, the AIF invokes the receiveMessage method of the AifReceiveAdapter interface. The receiveMessage method then does the following:

- Retrieves the configured URI.
- Creates a connection to WebSphere MQ queue manager specified in the URI using the connection information. The connection is creating using the low level MQ C# API.
- Opens the queue specified in the URI.
- Attempts to receive message.
- If a message is received, then some sanity checking on the message is done.
- The body of the message is wrapped in an AIF gateway message and the call returns.
- Further processing of the message and execution of required business logic is handled by AIF.

Any exception thrown either at the adapter layer or at the MQ .NET layer is reported back to the Dynamics AX Exception log. The exception log contains the full exception stack and helps in diagnosing the problems.



1.7.3 Send messages to a WebSphere MQ queue

After executing the required business logic, a response message may be provided by Application Integration Framework. Those response messages can be sent back to a WebSphere MQ application. When AIF wants to send a response message it invokes the sendMessage method of the AifSendAdapter interface implementation. The sendMessage method does the following to send a message to a WebSphere MQ queue:

- Retrieves the configured URI.
- Creates a connection to the WebSphere MQ queue manager specified in the URI using the connection information. The connection is creating using the low level MQ C# API.
- Opens the queue specified in the URI.
- Retrieves the XML body of AIF gateway message which is passed as a parameter to the sendMethod call by AIF.
- Creates a response MQ message.

- Retrieves the ResponseMessageId attribute from the XML message and sets that as a message property on the response MQ message. This allows the MQ application to selectively retrieve messages based on this property.
- Sets the XML message as the body of the MQ message and sends that message to MQ queue specified in the URI.

2. Installing WebSphere MQ Adapter for Microsoft Dynamics AX 2012.

This section describes how to install WebSphere® MQ Dynamics AX 2012 Adapter on Windows systems. This procedure can be used for installing a first or a subsequent installation.

Before you begin

To install a WebSphere MQ Dynamics AX 2012 Adapter, you must be logged on to Windows as an administrator.

All prerequisite software described above must be installed on the target system.

2.1 About this task

WebSphere MQ Adapter for Microsoft Dynamics AX 2012 is installed using an installation program. You can either use the installer in the form of a wizard with a graphical user interface, or you can start it from a command prompt. For more information about the installation program see http://www.acresso.com/products/is/installshield-overview.htm.

2.2 Installing WebSphere MQ Adapter using the installation wizard

Follow these instructions to perform an interactive installation of WebSphere MQ Adapter for Microsoft Dynamics AX 2012.

- Access the WebSphere MQ Adapter for Microsoft Dynamics AX 2012 installation image. The location might be the mount point of the CD, a network location, or a local file system directory.
- Locate setup.exe in the directory.
- Double-click the **Setup** icon to start the installation process. It is possible to run either by running setup.exe from the command prompt or double clicking setup.exe from Windows Explorer.

The installation uses an InstallShield X/Windows MSI installer.

To install WebSphere MQ Adapter, follow this procedure.

1. On Windows, log on as an administrator.
2. Run the setup.exe installer.
3. Wait for the installation wizard to open and display the following message:

```
Welcome to WebSphere MQ Adapter for Microsoft Dynamics AX 2012 installation
wizard
```

Click **Next**.
The wizard might ask you to read the license agreement.

4. If you are asked to read the license agreement and you accept the terms of the license agreement, click **I accept the terms in the license agreement**, and then click **Next**.
5. The Customer Information window is displayed. Provide the required information and click **Next**.
6. The Destination Folder window is displayed. If you do not want to install WebSphere MQ Adapter in the directory suggested, choose another directory. Click **Next**.
7. The Ready to Install the Program window is displayed. Click **Install**.
8. Installation begins and a status window is displayed. After installation is completed the InstallShield Wizard Completed window is displayed. Click **Finish** to close the window.

Microsoft Dynamics AX runs Windows 64 bit platforms only. Hence the WebSphere MQ Adapter for Microsoft Dynamics AX 2012 must be installed on a Windows 64 bit platforms only. On Windows 64 bit platforms WebSphere MQ adapter files will be installed in the C:\Program Files (x86) \IBM\WMQAXAdapter directory unless you choose to install it in a different directory.

The following table lists the installed directories, relative to the installation directory, and describes their contents.

File name	Installed Directory	Contents
WmqAxIntegrationAdapter.axmodel	Bin	The WebSphere MQ Adapter for Microsoft Dynamics AX 2012 model file. This file must be imported into Dynamics AX 2012.
IBM.WMQ.AxIntegrator.dll	Bin	.NET assembly required by WebSphere MQ adapter.
readme.txt		Readme file
License files		The license agreement for the product.
WebSphereMQadapter.pdf		Documentation of the product.
Samples	Samples\bin Samples\source	Compiled sample applications and source

2.3 Importing WmqAxIntegrationAdapter.axmodel into Dynamics AX 2012

This section describes the steps that are required to install *WmqAxIntegrationAdapter.axmodel* into Dynamics AX 2012.

Complete all the following actions in a Windows PowerShell prompt.

Optional Steps:
Run the following three *cmdlets* to import AX utility commands into the PowerShell environment. These *cmdlets* assist in the adapter installation process.

```
Import-Module "C:\Program Files\Microsoft Dynamics
AX\60\ManagementUtilities\AXUtilLib.PowerShell.dll"

Import-Module "C:\Program Files\Microsoft Dynamics
AX\60\ManagementUtilities\ Microsoft.Dynamics.ManagementUtilities.ps1"

Import-Module "C:\Program Files\Microsoft Dynamics
AX\60\ManagementUtilities\AXUtilLib.dll"
```

Mandatory Steps
Run the following command to install the WebSphere MQ adapter for Dynamics AX model file. Replace the installation path with the actual path where the WebSphere MQ Dynamics AX model file was copied during the installation process. Also replace the value of the **config** parameter with name of the Dynamics AX 2012 Application Object Server instance to which you want to install the adapter.

```
Install-AXModel -File "<WmqAx Installation path>\ WmqAxIntegrationAdapter.axmodel"
-config:<AOS Configuration Name>
```

Note: There are a number of variations to the above command. For more information, refer to the *Install-AXModel* command documentation on MSDN.

After successful installation, a compilation of AOT might be required and restart of AOS also might be required. Follow the instructions displayed by the Install-AXModel command.

Sometimes the Dynamics AX 2012 instance service might need to be restarted for updates to take effect.

2.4 Uninstalling WebSphere MQ Adapter for Microsoft Dynamics AX 2012

This section describes the uninstallation of WebSphere MQ Dynamics AX 2012 Adapter.

Before you begin

The following must be ensured:

1. All messages in the Dynamics AX 2012 gateway queue that were received or about to be sent via a port that uses WebSphere adapter must be removed.
2. All Inbound/Outbound ports that use the WebSphere MQ adapter must be deactivated and deleted.

To carry out the uninstallation complete the following steps:

Part I

Run the following command in the Windows PowerShell prompt to uninstall the WebSphere MQ Dynamics AX 2012 Adapter from Microsoft Dynamics AX.

```
Uninstall-AXModel -config:<AOS Instance> -Model WmqAxIntegrationAdapter
```

Part II

Complete all the following actions in a Dynamics AX 2012 Integrated Development Environment (MorphX IDE):

1. Open **Application Object Tree** (AOT).
2. Click and expand the **References** node in AOT.
3. Locate IBM.WMQ.AxIntegrator assembly. Right-click and select the **Delete** menu to remove the reference.

Part III

1. From the Microsoft Windows Control Panel, open the Programs and Features window.
2. Right-click **IBM WebSphere MQ Adapter for Dynamics AX** and select the **Uninstall** menu. Follow the instructions on the window to complete the uninstallation.

3. WebSphere MQ adapter URI

This section describes the Uniform Resource Identifier (URI) format supported by the WebSphere MQ Adapter for Microsoft Dynamics AX 2012.

The URI format supported is similar to the one described in IBM SupportPac MA93. Not all URI attributes are supported. This topic describes all attributes supported by the adapter.

The URI has two types of attributes; Mandatory attributes and Optional attributes. An URI must contain the mandatory attributes. Apart from mandatory attributes an URI must contain a queue name. An exception will be thrown if the specified URI does not have the mandatory attributes and the queue name.

3.1 Mandatory attributes

This section describes the fixed attributes of the URI supported by the WebSphere MQ adapter.

Attribute	Description
wmq:	This attribute identifies the WebSphere MQ URI.
msg/queue	This attribute indicates that a WebSphere MQ queue will be used for message exchange.

3.2 Optional attributes

This section describes the variable attributes of the URI supported by WebSphere MQ adapter

Attribute	Description
channelName	This attribute identifies the channel name to be used for connecting to a WebSphere MQ queue manager. If this attribute is not specified then SYSTEM.DEF.SVRCONN channel is used as default.
Hostname	This attribute identifies the name or IP address of the host on which queue manager is running. If this attribute is not specified then localhost is used.
Port	This attribute identifies the port number on which a queue manager is listening for client connections. Port 1414 is used as default if this attribute is not specified in the URI.
persistence	Indicates persistence type of messages sent by adapter. Possible values are: MQPER_PERSISTENT: Messages sent are persistent. MQPER_NOT_PERSISTENT: Messages sent are not persistent.

Attribute	Description
	MQPER_PERSISTENCE_AS_PARENT MQPER_PERSISTENCE_AS_Q_DEF Default: MQPER_NONPERSISTENT
connectQueueManager	The name of the queue manager to which the requesting service SHOULD connect to. This corresponds to the QmgrName parameter used on the MQCONN() or MQCONNX() calls.
channelTableName	This is the name of the client channel table file.
channelTableLib	This is the Path to the client channel table.
Expiry	This is the message lifetime, specified as the decimal representation of a signed integer, and measured in tenths of seconds.
Priority	This indicates the priority associated with the message. It is specified as the decimal representation of an integer. It corresponds to the Priority field in the MQMD structure. • Valid values are integers between 0 (lowest priority) and 9 (highest priority). • A special value (-1) can be used to indicate that the message priority is taken from the definition of the first queue the message is put to. • The default value is -1 (which equates to the constant MQPRI_PRIORITY_AS_Q_DEF).
binding	This indicates the type of binding the adapter will use to connect a queue manager. Server: The adapter will connect to a queue manager in shared memory binding mode. Client: The adapter will connect to a queue manager in a client mode over a TCP socket.
clientConnectionMode	This indicates whether the adapter will connect to a queue manager in a .NET managed or unmanaged mode. Valid only when using Client bindings. Managed: Use .NET Managed mode Unmanaged: Use .NET Unmanaged mode.
user	Name of the user to be used for connecting to a queue manager.
encoding	Numeric encoding of message data. This property corresponds to the Encoding field in the MQMD structure. • This value is specified as the decimal representation of an integer corresponding to the valid MQENC_* values defined for the MQMD.Encoding field in the *Application Programming Reference* documentation. • If this property is not specified the sender can choose an appropriate encoding, or can allow WebSphere MQ to use its default value. • This property is only allowed for native bindings.
codedCharSetId	Character encoding of message data. It is specified as the

Attribute	Description
	decimal representation of an integer. This property corresponds to the CodedCharSetId field in the MQMD structure. If this property is not specified the service requester and service provider must use a value which corresponds to the character set of the message data.
correlId	A 24 byte length CorrelationID.

The following examples show some of the typical URI formats recognized by the adapter.

This example is of a simple URI with just the queue name. In this case the adapter will connect to the default queue manager in server binding mode.

```
wmq:/msg/queue/SALES.REQUEST
```

This is another example of an URI with a connection name and a queue manager name. In this case the adapter will connect to the queue manager QM running on host 'remotehost', listening at port 2414. The channel name used will be 'SYSTEM.DEF.SVRCONN.

```
wmq://remotehost:2414/msg/queue/SALES.REQUEST@QM
```

This is another example of a fully qualified URI. Here the adapter will connect to queue manager QM running on 'remotehost' listening on port 1414 using channel 'AX.SVRCONN.

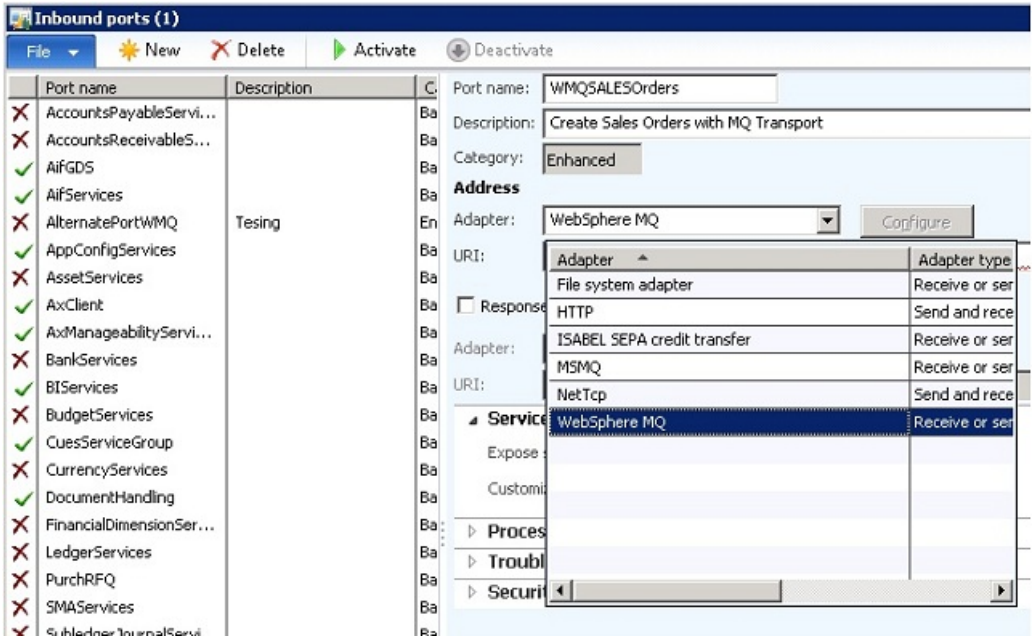
```
wmq://remotehost:1414/msg/queue/SALES.REQUEST@QM?connectQueueManager=QM
&channelName=AX.SVRCONN
```

4. Configuring WebSphere MQ Adapter for Microsoft Dynamics AX 2012

This section describes how to configure an Inbound Port using WebSphere MQ Dynamics AX 2012 Adapter in Microsoft Dynamics AX Services and Application Integration Framework.

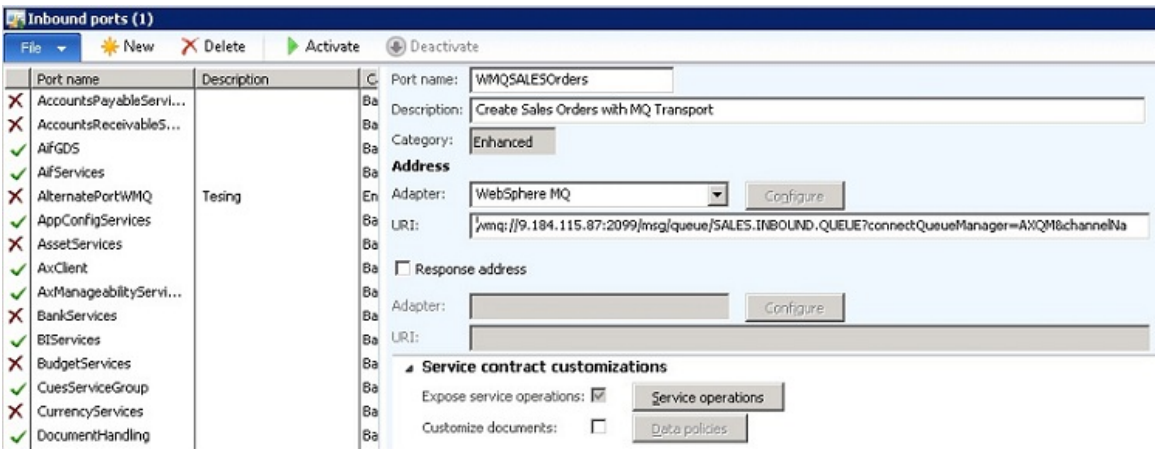
All of the following steps are completed from System Administration Area Page of the Microsoft Dynamics AX Console.

1. From the **Services and Application Integration Framework** group, select the **Inbound ports** option. The Inbound Ports window is displayed.
2. Click the **New** button. The right pane in the window becomes editable.
3. Configure the inbound port as follows:
   - Type a name in the **Port name** and **Description** fields.
   - By default the selected value in the **Category** field is **Enhanced**.
   - From the **Adapter** list, select **WebSphere MQ**.

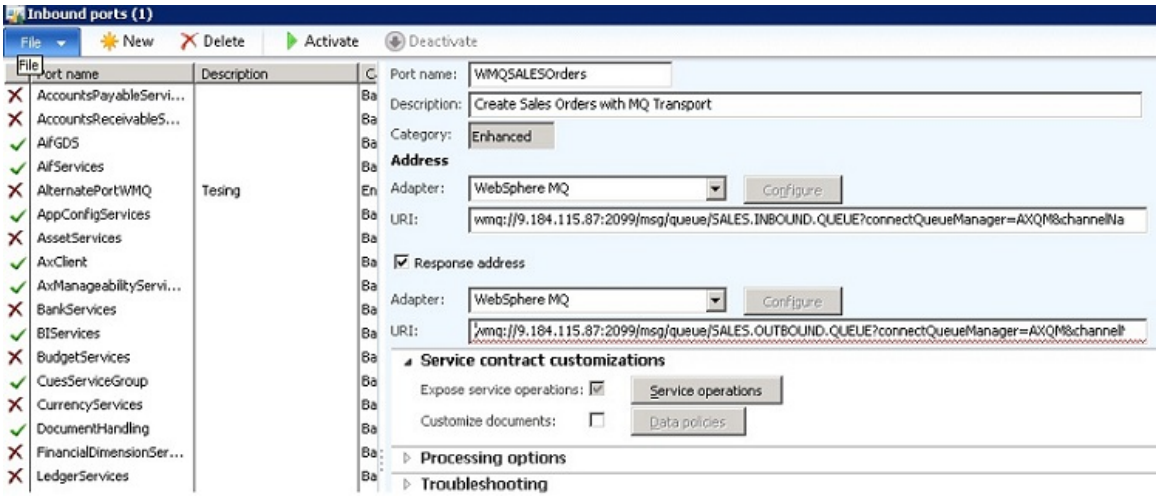


   - Enter a valid WebSphere MQ URI. The URI must point to a valid WebSphere MQ queue. For example a URI could be:

     ```
     wmq://mqhost:2099/msg/queue/SALES.REQUEST.QUEUE@QM1?connectQueueManager=QM1&&channelName=AX.SVRCONN
     ```



- If the AX service provides a response then select the **Response Address** check box and complete the two fields immediately below the check box:
    - In the **Adapter** field, select WebSphere MQ from the list.
    - In the **URI** field, enter a valid WebSphere MQ URI. The URI must point to a valid WebSphere MQ queue.



- In the **Service contract customizations** section, click **Service operations**. The “Select service operations” window is displayed.
- Choose a service that you want to invoke through WebSphere MQ adapter and click **OK** to close the window. For example, to create sales orders choose the **SalesSalesOrderService.create** service.
- Select the **Document customizations** check box. The **Data policies** button is enabled.
- Click **Data policies**. The “Document data policies” window is displayed.
- Select the fields that an incoming message must contain and click **OK**. For example an incoming “Sales order creation” message must contain “Customer name” and “item id”. AX flags an error if a message does not contain these fields.

The configuration of the adapter is now complete. To activate the port, click the **Activate** button. A window showing the status of the activation process is displayed.

The log indicates either a successful activation or a failure. Appropriate action must be taken if there is a failure.

5. Usage scenarios

This section provides an overview of how a MQ customer application or WebSphere Message Broker node will use the WebSphere MQ Dynamics AX adapter to invoke Dynamics AX services asynchronously.

5.1 MQ application invoking sales order creation service

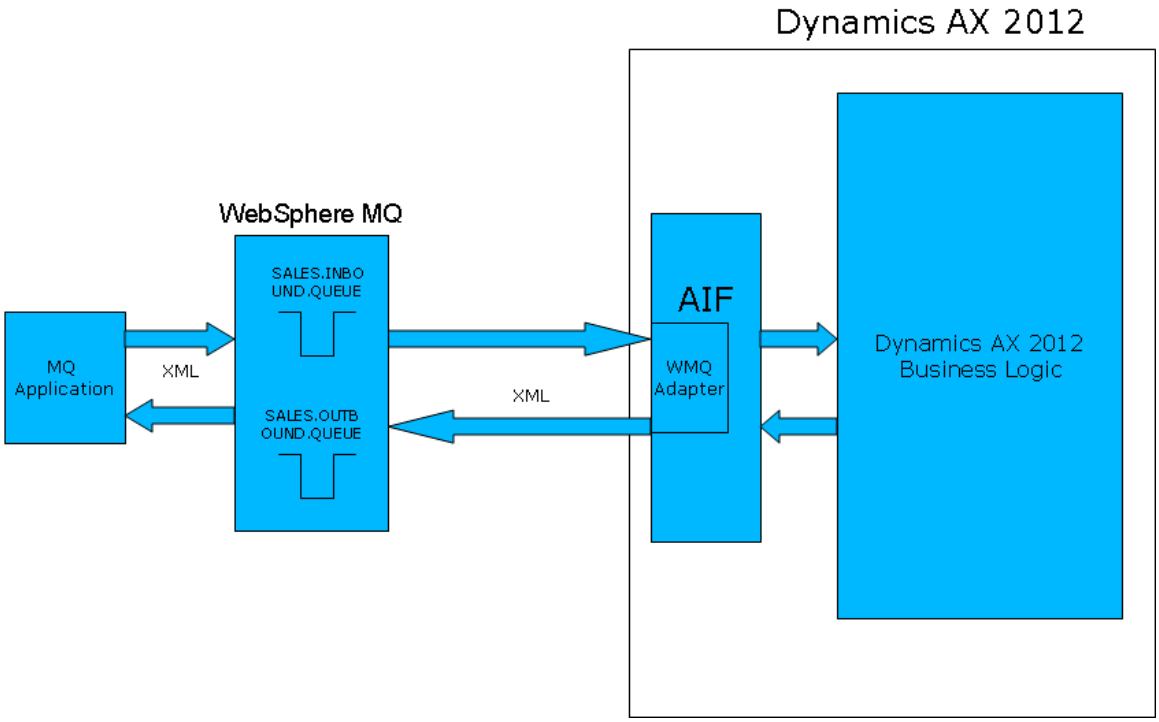
In the following example an MQ application (meaning an application developed using the MQ API) connects to a queue manager for sending and receiving messages.

5.1.1 Configuration

This section describes the required setup.

5.1.1.1 WebSphere MQ configuration

1. A WebSphere MQ queue manager
2. Two server connection channels (SVRCONN), one for the use by MQ client application and the other for WebSphere MQ adapter.
3. Two local queues. SALES.INBOUND.QUEUE is the queue where MQ application puts sales order creation request messages. SALES.OUTBOUND.QUEUE is the queue where MQ application receives response messages.



5.1.1.2 Microsoft Dynamics AX 2012 configuration

This section describes the configuration to be done on Dynamics AX 2012.

1. An activated Inbound port with:
    - Address Name as WebSphere MQ
    - Address URI representing a WebSphere MQ queue
    - Response Address Name as WebSphere MQ
    - Response Address URI representing a WebSphere MQ queue
2. A Dynamics AX batch job task for the following classes:
    - AifGatewayReceiveService
    - AifGatewaySendService
    - AifInboundProcessingService
    - AifOutboundProcessingService
3. The batch job is scheduled to run at an interval of 1 minute.

The MQ application connects to a queue manager and puts a message into SALES.INBOUND.QUEUE with a message body as shown in the following sample request message. The message contains the information required to create a sales order. After sending the request message, the application receives a response message on SALES.OUTBOUND.QUEUE.

Sample Request Message in XML

```
<?xml version="1.0" encoding="utf-8"?>
<Envelope
xmlns="http://schemas.microsoft.com/dynamics/2011/01/documents/Message">
  <Header>
    <MessageId>{a06aa615-b7c3-4fc8-81d0-1bbbbc538ad7}</MessageId>
<Action>http://schemas.microsoft.com/dynamics/2008/01/services/SalesOrderServic
e/create</Action>
  </Header>
  <Body>
    <MessageParts>
      <SalesOrder
xmlns="http://schemas.microsoft.com/dynamics/2008/01/documents/SalesOrder">
        <SalesTable class="entity">
          <CustAccount>100002</CustAccount>
          <PurchOrderFormNum>PO</PurchOrderFormNum>
          <ReceiptDateRequested>2012-11-11</ReceiptDateRequested>
          <SalesLine class="entity">
            <ItemId>SU10031</ItemId>
            <SalesQty>18</SalesQty>
            <SalesUnit>KG</SalesUnit>
          </SalesLine>
        </SalesTable>
      </SalesOrder>
    </MessageParts>
  </Body>
</Envelope>
```

The MQ application receives the response message from SALES.OUTBOUND.QUEUE. A sample response message header is shown in the following example:

```
  <Header>
    <MessageId>{698e143f-9d21-4a3e-b0d5-d7fe564ee41f}</MessageId>
    <RequestMessageId>{07ba240f-f2c0-429a-b9da-831a573f3854}</RequestMessageId>
  </Header>
```

The <RequestMessageId> in the response message will be same as <MessageId> of the request message sent by the application. Using <MessageId> and <RequestMessageId> applications correlate request and response messages.

5.1.1.3 Sample source code for MQ application

This section describes the source code of the sample application that creates a sales order.

This sample code loads the sales order creation request XML message from a file. Alternatively the XML message can be built on the fly. It is important to note here that the value of MessageId attribute in the XML must be unique. Hence this sample code generates a new GUID.

```
public void CreateSalesOrderInDynamicsAX()
{
   Hashtable mqProps = new Hashtable();
   MQQueueManager qm = null;

   try
   {
     mqProps.Add(MQC.CHANNEL_PROPERTY, "AX.CHANNEL");
     mqProps.Add(MQC.TRANSPORT_PROPERTY, MQC.TRANSPORT_MQSERIES_MANAGED);
     mqProps.Add(MQC.HOST_NAME_PROPERTY, "axhost");
     mqProps.Add(MQC.PORT_PROPERTY, 2099);

     // Create queue manager
     qm = new MQQueueManager("AXQM", mqProps);

     int reqOpenOptions = MQC.MQOO_OUTPUT | MQC.MQOO_FAIL_IF_QUIESCING;
     MQQueue reqQ = qm.AccessQueue("SALES.INBOUND.QUEUE", reqOpenOptions);

     String XMLPath = "C:\\DynxXML\\msg.xml";
     XmlDocument doc = new XmlDocument();
     doc.Load(XMLPath);

     // Generate a new GUID everytime
     XmlNodeList elemList = doc.GetElementsByTagName("MessageId");
     elemList[0].InnerText = "{" + Guid.NewGuid() + "}";

     MQMessage putMsg = new MQMessage();
     putMsg.Format = MQC.MQFMT_STRING;
     putMsg.WriteString(doc.InnerXml);
     // Send request
     reqQ.Put(putMsg);
     reqQ.Close();

     MQMessage respMsg = new MQMessage();
     MQGetMessageOptions gmo = new MQGetMessageOptions();
     gmo.WaitInterval = MQC.MQWI_UNLIMITED;
     gmo.Options = MQC.MQGMO_WAIT;

     int respOpenOptions = MQC.MQOO_INPUT_SHARED | MQC.MQOO_FAIL_IF_QUIESCING;
     MQQueue respQ = qm.AccessQueue("SALES.OUTBOUND.QUEUE", respOpenOptions);

     // Wait for response
     respQ.Get(respMsg,gmo);

     respQ.Close();
```

```
    qm.Disconnect();
  }
  catch (Exception ex)
  {
    Console.WriteLine(ex);
  }
 }
```

6. Troubleshooting

This section provides information to help you detect and deal with problems.

6.1 About this task

This section contains the following information:

- Problem determination with the adapter in Dynamics AX
- FFDC and Trace configuration
- Tips for troubleshooting

6.2 Problem determination for the adapter

This section provides information to help you detect and deal with problems in the adapter.

The adapter comprises of two parts:

1. The adapter component (WmqAxIntegrationAdapter.axmodel)
2. The supporting .NET assembly (IBM.WMQ.AxIntegrator.dll)

The adapter is developed using X++ language which is the default programming language of Dynamics AX. The adapter invokes methods exposed by the supporting assembly for communicating with WebSphere MQ. This .NET assembly is developed in C# language.

As the two parts of the adapter are developed using different languages the problem determination technique is different for both parts. The adapter uses the X++ standard exception handling for any errors that arise and log those errors to Dynamics AX exception log. The exception log will contain the complete exception stack. The exception stack will include the exceptions thrown by the supporting .NET assembly as well as the exceptions thrown by the underlying WebSphere MQ .NET client. The exceptions thrown by the supporting .NET assembly and WebSphere MQ .NET client are indicated as *System.Reflection.Exception.*

6.2.1 Adapter module tracing

Dynamics AX 2012 Exception log will contain all the exceptions thrown by the adapter as well as the AxIntegrator assembly. The relevant portion of the exception log or the whole exception log can be saved to file and sent for troubleshooting.

Tracing can be enabled using the *Tracing Cockpit* in Dynamics AX 2012 Development Workspace. Follow the instructions in 6.1.2 AxIntegrator tracing to collect trace information.

6.2.2 AxIntegrator tracing

AxIntegrator writes trace to the standard MQ trace files. Hence use the **strmqtrc** and **endmqtrc** commands to start and end tracing.

Windows uses the following commands for the client trace facility:

strmqtrc to start tracing
endmqtrc to end tracing

The output files are created in the MQ_DATA_PATH/trace directory.
Trace files are named AMQppppp.qq.TRC where the variables are:

ppppp
The process ID of Dynamics AX 2012 Application Object Server instance. The process will have name as *Ax32Serve.*exe.

qq
A sequence number starting at 0. If the full file name exists, this value is incremented by one until a unique trace file name is found. A trace file name can exist if a process is reused.

Note:
The process identifier can contain fewer, or more, digits than shown in the example. There is one trace file for each process running as part of the entity being traced.

To format or view a trace file, you must be either the creator of the trace file, or a member of the *mqm* group.

6.2.2.1 How to start and stop a trace

Enable or modify tracing using the **strmqtrc** control command. To stop tracing, use the **endmqtrc** control command. Refer to the WebSphere MQ information center for more details on the **strmqtrc** and **endmqtrc** commands.

6.2.2.2 Selective process tracing

Use the **-p** option of the **strmqtrc** command control to restrict trace generation to specified named processes. For example, to trace all threads that result from Dynamics AX 2012 AOS, use the following command:

```
strmqtrc -p Ax32Serv.exe
```

6.3 Must gather documents for troubleshooting

The documents that must be collected for troubleshooting are shown in the following is list:
1. Trace log from Dynamics AX 2012
2. Relevant exception log from Dynamics AX 2012
3. AxIntegrator trace file. This trace file can be identified using the process id of Ax32Serv.exe. It would also help to provide the entire MQ trace files.

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To make comments about the functions of IBM products or systems, talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

- By mail, to this address:
  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom

- By fax:
  – From outside the U.K., after your international access code use 44–1962–816151
  – From within the U.K., use 01962–816151

- Electronically, use the appropriate network ID:
  – IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  – IBMLink™: HURSLEY(IDRCF)
  – Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.