webMethods Module for EDIINT Installation and User's Guide

Version 8.2 SP1

November 2013

WEBMETHODS

This document applies to webMethods Module for EDIINT 8.2 SP1 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Document ID: ESTD-EDIINT-IUG-82 SP1-20230703

Table of Contents

About this Guide

This guide describes how to exchange business documents over the Internet securely and reliably using the EDIINT protocol. In addition, it provides information for using webMethods Module for EDIINT to support EDIINT processing.

To use this guide effectively, you should be familiar with:

- EDIINT standards and terminology.
- webMethods Integration Server and Integration Server Administrator, and understand the concepts and procedures in the *webMethods Integration Server Administrator's Guide*.
- webMethods Trading Networks and webMethods Module for EDI, and understand the concepts and procedures described in the various Trading Networks and Module for EDI guides.
- Software AG Designer, and understand the concepts and procedures described in the *webMethods Service Development Help* for your release.
- My webMethods Server and its interface My webMethods, and understand the concepts and procedures described in the appropriate My webMethods documentation for your release.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Narrowfont	Identifies service names and locations in the format *folder.subfolder.service*, APIs, Java classes, methods, properties.
Italic	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.

Convention	Description
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at https://documentation.softwareag.com.

In addition, you can also access the cloud product documentation via https://www.softwareag.cloud. Navigate to the desired product and then, depending on your solution, go to "Developer Center", "User Center" or "Documentation".

Product Training

You can find helpful product training material on our Learning Portal at https://knowledge.softwareag.com.

Tech Community

You can collaborate with Software AG experts on our Tech Community website at https://techcommunity.softwareag.com. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software AG news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at https://github.com/softwareag and https://hub.docker.com/u/softwareag and discover additional Software AG resources.

Product Support

Support for Software AG products is provided to licensed customers via our Empower Portal at https://empower.softwareag.com. Many services on this portal require that you have an account. If you do not yet have one, you can request it at https://empower.softwareag.com/register. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.

- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

1 Concepts

What Is EDIINT?

EDIINT stands for "Electronic Data Interchange-Internet Integration," or "EDI over the Internet." The EDIINT standard, defined by the Internet Engineering Task Force (IETF), is a protocol that specifies how to exchange business documents, such as EDI and XML, over the Internet in a secure, reliable, non-reputable way. It does not specify how to validate or process the business documents that are transported.

There are three versions of the EDIINT standard:

- EDIINT AS1 (EDIINT Applicability Statement 1), which uses SMTP (e-mail) to transport documents
- EDIINT AS2 (EDIINT Applicability Statement 2), which uses HTTP (or HTTP/S) to transport documents
- EDIINT AS3 (EDIINT Applicability Statement 3), which uses FTPS (FTP over SSL) to transport documents

All versions support digital signatures, encryption, and signed receipts.

What Is webMethods Module for EDIINT?

webMethods Module for EDIINT (Module for EDIINT) adds support for the EDIINT exchange protocol. Documents using the EDIINT protocol are processed through Trading Networks. Therefore, if you want to use the EDIINT protocol, you must use Trading Networks.

Module for EDIINT supports EDIINT AS1 (SMTP), EDIINT AS2 (HTTP), and EDIINT AS3 (FTPS) messages, including MDN (receipt) exchange. The module uses the S/MIME version 2 cryptographic format exclusively both to package, encrypt, and provide a digital signature to outbound data, and to unpack, decrypt, and verify the authenticity of inbound data.

You can use EDIINT to transport both EDI and non-EDI formatted (for example, XML or custom format) documents.

- When you use the EDIINT transport for EDI documents, you must also have webMethods Module for EDI installed. Module for EDIINT passes EDI documents to Trading Networks, which in turn allows Module for EDI to process the EDI document using the functions of the Module for EDI packages (WmEDI and WmEDIforTN).
- When you use the EDIINT transport for non-EDI documents, Module for EDIINT passes the documents to Trading Networks, and Trading Networks processes them based on logic you define in Trading Networks.

Features

Module for EDIINT supports the following EDIINT features:

- Exchanges business documents securely using EDIINT AS1, EDIINT AS2, and EDIINT AS3.
- Uses the S/MIME version 2 cryptographic format exclusively both to package, compress, encrypt, and provide a digital signature to outbound data and to unpack, decrypt, and verify the authenticity of inbound data.

  Note: The S/MIME (Secure/Multipurpose Internet Mail Extensions) standard specifies formats and procedures for providing the cryptographic security services of message authentication, integrity, non-repudiation of origin, and confidentiality.
- Uses the SHA-1, SHA-2, SHA-256, SHA-384, and SHA-512 hash algorithm to sign outbound messages, and verifies inbound messages that were signed with either SHA-1, SHA-2, SHA-256, SHA-384, and SHA-512 or MD5.
- Enables you to set encryption types and key lengths for each of your trading partners using the extended fields in the partner's Trading Networks profile. The choices include: TripleDES, DES, RC2 (40 bits), RC2 (64 bits), and RC2 (128 bits).
- Provides the standard outbound encryption permutations (signed, encrypted, signed and encrypted, or plain) at the send-service level.
- Returns receipts for messages it receives and receives receipts for messages it sends. EDIINT receipts are known as MDNs (message disposition notifications). Module for EDIINT can send and receive synchronous or asynchronous, signed or unsigned MDNs.
- Enables you to include custom headers on EDIINT messages you send to trading partners.
- webMethods Module for EDIINT with Trading Networks on Microservices Runtime works with Integration Server 10.15 Core Fix 2 or higher.

AS2-Specific Features

Module for EDIINT supports the following EDIINT AS2 version 1.2 features:

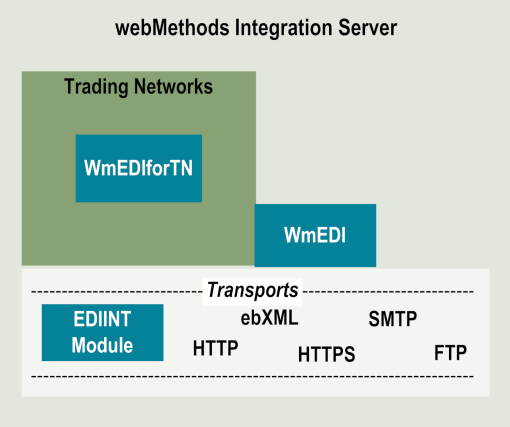
- **Multiple attachments.** Module for EDIINT can transmit EDIINT messages with multiple attachments, along with its payload, in a single EDIINT transmission. Also, the module processes EDIINT messages with multiple attachments and identifies and stores them separately. Module for EDIINT supports the following content types:

application/msword	application/xml	text/html
application/octet-stream	application/zip	text/plain
application/pdf	image/bmp	text/rtf
application/rtf	image/jpeg	text/xml
application/vnd.ms-excel	image/gif	
application/x-msexcel	image/tiff	

- **AS2 restart.** When transmission of an EDIINT message is interrupted, the module is capable of restarting the transmission from the point at which it failed, instead of starting the entire transmission over again.
- **HTTP chunking for transferring data.** Using chunked transfer coding, Module for EDIINT can begin transferring large amounts of data before the length of the content being transferred is known. For information about configuring chunk size, see *webMethods Integration Server Administrator's Guide*.

How Module for EDIINT Relates to Other webMethods Components

When you install Module for EDIINT, the WmEDIINT package is installed into Integration Server. The following diagram illustrates how the module fits into the webMethods product suite architecture.



- Integration Server is the underlying foundation.
- Trading Networks is a webMethods component that enables your enterprise to link with other companies (buyers, suppliers, strategic partners) and marketplaces to form a business-to-business trading network. To use Module for EDIINT, Trading Networks is required. For more information about Trading Networks, see the Trading Networks documentation for your release.
- Module for EDI comprises the following two packages:

  - The WmEDI package, which contains the basic functionality that provides support for the EDI standard.
  - The WmEDIforTN package, which allows the WmEDI package and Trading Networks to interact. This interaction allows you to use Trading Networks as a gateway for EDI document exchange. Module for EDI uses the functionality of Trading Networks to provide additional features, such as support for VANs, reconciling FAs, and batching the sending of EDI documents.
- Module for EDIINT contains the support for the EDIINT exchange protocol as described in "What Is webMethods Module for EDIINT?" on page 10.

Features Provided for EDIINT Processing

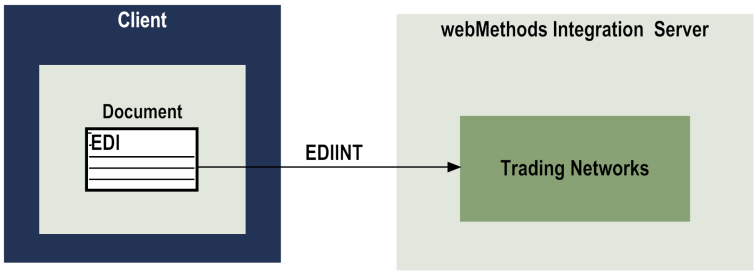
Module for EDIINT provides the following features to support EDIINT processing:

- Content handlers to recognize inbound EDIINT documents and MDNs.
- Built-in services that you use to send outbound EDIINT documents and MDNs and receive inbound EDIINT documents and MDNs.
- TN document types, which are automatically installed into Trading Networks, that allow Trading Networks to recognize inbound EDIINT documents and MDNs, and are set up so all EDIINT documents and MDNs are automatically saved to the Trading Networks database. Optionally, you can configure the module to store the EDIINT documents in the file system.
- Extended profile fields, which are automatically installed into Trading Networks, that are available in Trading Networks profiles for you to supply partner-specific information needed for EDIINT transport-level processing.
- Processing rules, which are automatically installed into Trading Networks, that you can use to perform the EDIINT transport-level processing.
- Delivery services, which are automatically registered with Trading Networks, that you can use to deliver EDIINT documents and MDNs.

For more information about how the module uses these features for EDIINT inbound and outbound processing, see "Inbound EDIINT Processing with AS1 and AS2" on page 13 and "Outbound EDIINT Processing with AS1 and AS2" on page 17, respectively.

Inbound EDIINT Processing with AS1 and AS2

For inbound processing, a client sends a document to Integration Server using the EDIINT exchange protocol. The document is processed in Integration Server using services provided with Module for EDIINT along with Trading Networks.



EDIINT Client

Use Module for EDIINT to create the client to send documents using EDIINT. If you are not using webMethods software on the client side, see documentation for the EDIINT software that you are using.

When the client sends the EDIINT document to Integration Server, it must associate the inbound document with a content type that Module for EDIINT recognizes, for example, `multipart/signed`. When Integration Server receives a document that has an EDIINT content type, it passes the document to the appropriate EDIINT content handler, which was installed when you installed the module.

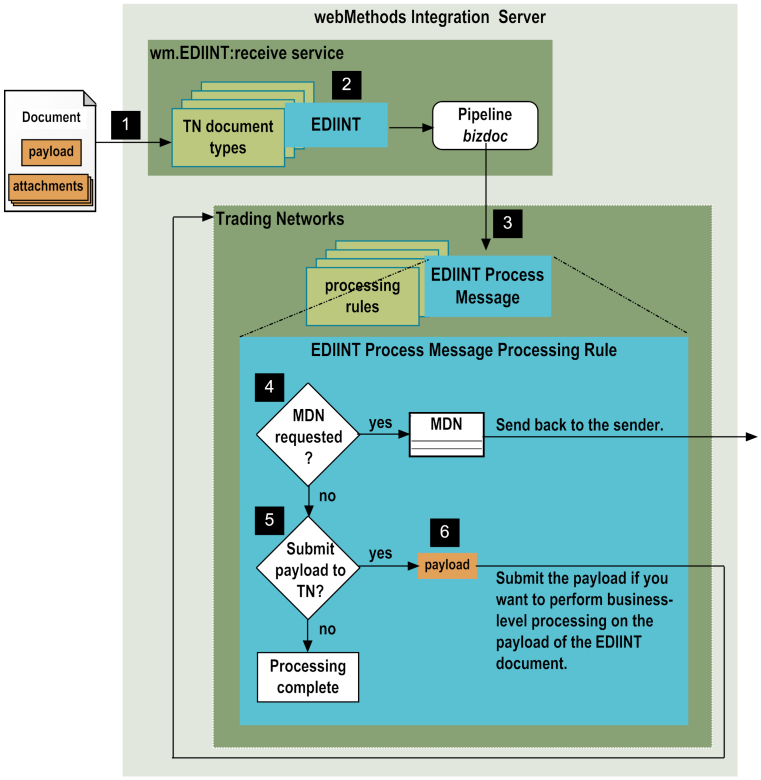
The EDIINT content handler passes the document to the service the client specifies. To use the EDIINT exchange protocol, the client must specify the wm.EDIINT:receive service. The wm.EDIINT:receive service is a built-in service provided with Module for EDIINT. For more information about processing the EDIINT document, see "Processing Inbound EDIINT Documents" on page 14.

Processing Inbound EDIINT Documents

Module for EDIINT provides all logic required to perform the transport-level processing for inbound EDIINT documents. You can use Trading Networks to add business-level processing for the payloads of your EDIINT documents.

Note:
For information about how Module for EDIINT processes inbound MDNs, see "Processing Inbound EDIINT MDNs" on page 16.

The following diagram illustrates how Module for EDIINT performs transport-level processing for an inbound EDIINT document. The diagram also shows how, if you want to perform business-level processing, you can submit the payload of the EDIINT document to Trading Networks for further processing.



Step	Description
1	The wm.EDIINT:receive service accepts the inbound document.
2	The wm.EDIINT:receive service uses the TN document types to determine the type of document. The document matches the EDIINT TN document type that is installed into Trading Networks when you install Module for EDIINT. After determining the TN document type, the wm.EDIINT:receive service forms a BizDocEnvelope for the inbound document and places it in the pipeline in the *bizdoc* variable. A BizDocEnvelope contains the original document and includes additional information that Trading Networks requires for routing and processing the document. One piece of information that Trading Networks can use in the selection of a processing rule is the user status. The EDIINT recognizer sets the user status to "ProcessMsg." After forming the BizDocEnvelope, the wm.EDIINT:receive service sends BizDocEnvelope to Trading Networks for processing.
3	Trading Networks determines the processing rule to use for the document. For inbound EDIINT documents, Trading Networks uses the EDIINT Process Message processing rule that is installed into Trading Networks when you install Module for EDIINT. It selects this processing rule because the TN document type is EDIINT and the user status is "ProcessMsg." This processing rule performs the **Execute a Service** action to invoke the wm.EDIINT.rules:processMsg service. The service processes the message by opening the

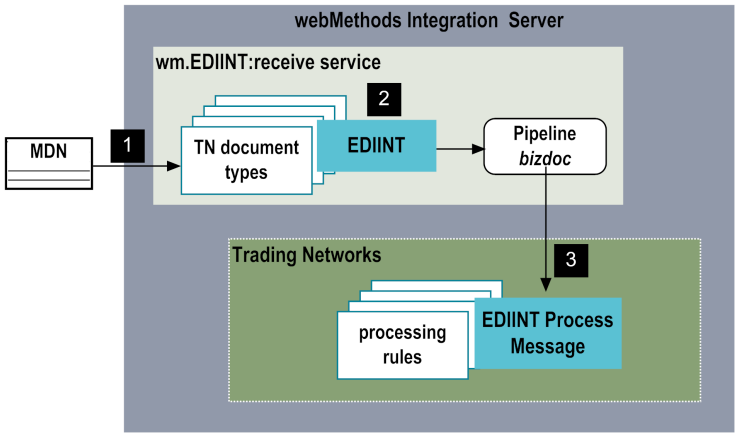
Step	Description
	MIME or S/MIME package and then decrypting and/or verifying the signature of the message.

The remaining steps represent actions specified in the EDIINT Process Message processing rule.

Step	Description
4	The wm.EDIINT.rules:processMsg service determines whether the sender of the EDIINT document requested an MDN. ■ **If the sender requested an MDN**, the wm.EDIINT.rules:processMsg service determines whether the sender has requested a signed or an unsigned MDN. The service creates the appropriate type of MDN and sends the MDN back to Trading Networks for delivery. Trading Networks can deliver the MDN synchronously or asynchronously. For more information about how Trading Networks delivers the MDN, see "Sending Outbound EDIINT MDNs" on page 19. After sending the document to Trading Networks, continue with the next step. ■ **If the sender did not request an MDN** , continue with the next step.
5	The wm.EDIINT.rules:processMsg service invokes the wm.EDIINT.rules:processPayload service, which determines whether you want to send the payload of the EDIINT document to Trading Networks for processing. You define whether you want Module for EDIINT to send EDIINT payloads to Trading Networks for processing when you configure the module.
6	If the payload is to be sent to Trading Networks for processing (for example, to perform business-level logic), submit the payload. The payload can be either an EDI document or a non-EDI document (for example, an XML document). ■ **EDI documents.** If the payload is an EDI document, you must set up Module for EDI and Trading Networks to process the EDI document. For example, use Module for EDI to install TN document types for the EDI document and create a processing rule to process the EDI document. For more information about using Module for EDI with Trading Networks to process EDI documents, see *webMethods Module for EDI Concepts Guide*. ■ **Non-EDI documents.** If the payload is not an EDI document, you must define the TN document types that Trading Networks can use to recognize the payload and the processing rule that Trading Networks should use to process the document.

Processing Inbound EDIINT MDNs

The following diagram illustrates how Module for EDIINT processes an inbound EDIINT MDN.

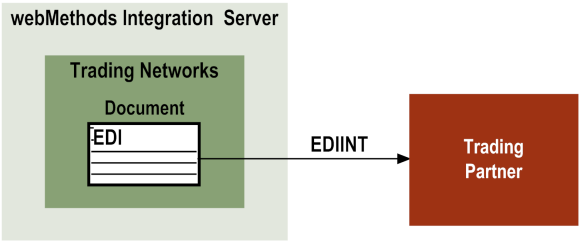


Step	Description
1	The wm.EDIINT:receive service accepts the inbound MDN.
2	The wm.EDIINT:receive service uses the TN document types to determine the type of document. The MDN matches the EDIINT MDN TN document type that is installed into Trading Networks when you install Module for EDIINT. After determining the TN document type, the wm.EDIINT:receive service forms a BizDocEnvelope for the inbound MDN and places it in the pipeline in the *bizdoc* variable. A BizDocEnvelope contains the MDN and includes additional information that Trading Networks requires for routing and processing the document. One piece of information that Trading Networks can use in the selection of a processing rule is the user status. The EDIINT recognizer sets the user status to "ProcessMDNMsg." After forming the BizDocEnvelope, the wm.EDIINT:receive service sends BizDocEnvelope to Trading Networks for processing.
3	Trading Networks determines the processing rule to use for the MDN. For inbound MDNs, Trading Networks uses the EDIINT Process MDN Message processing rule that is installed into Trading Networks when you install Module for EDIINT. Trading Networks selects this processing rule because the TN document type is EDIINT MDN and the user status is "ProcessMDNMsg." This processing rule performs the **Execute a Service** action to invoke the wm.EDIINT.rules:processMDN service, which processes the EDIINT MDN.

Outbound EDIINT Processing with AS1 and AS2

The documents you want to send using EDIINT can be EDI documents or non-EDI documents. The EDIINT standard specifies requirements for how to "package" a document for transport and how to transport the document.

To package the document and transport it, you use services provided with the module along with Trading Networks.

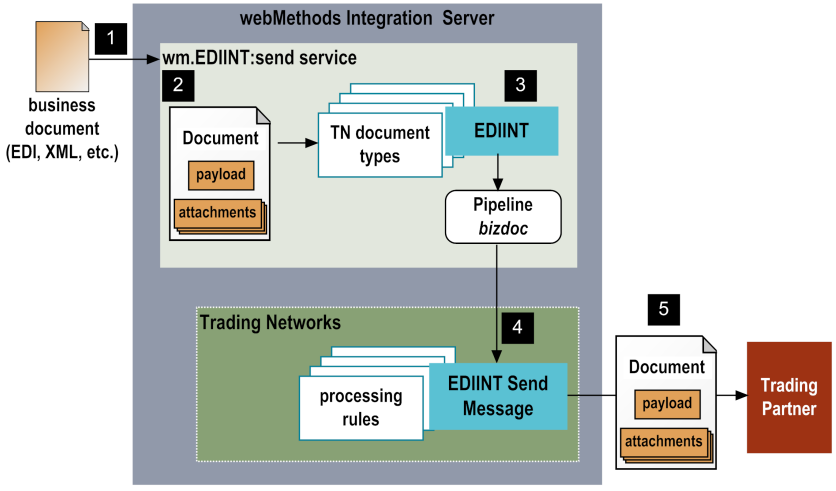


Sending Outbound EDIINT Documents

Module for EDIINT provides all logic required to perform the transport-level processing for sending outbound EDIINT documents.

Note:
For information about how to send outbound MDNs, see "Sending Outbound EDIINT MDNs" on page 19.

The following diagram illustrates how to use the module to send an outbound EDIINT document.



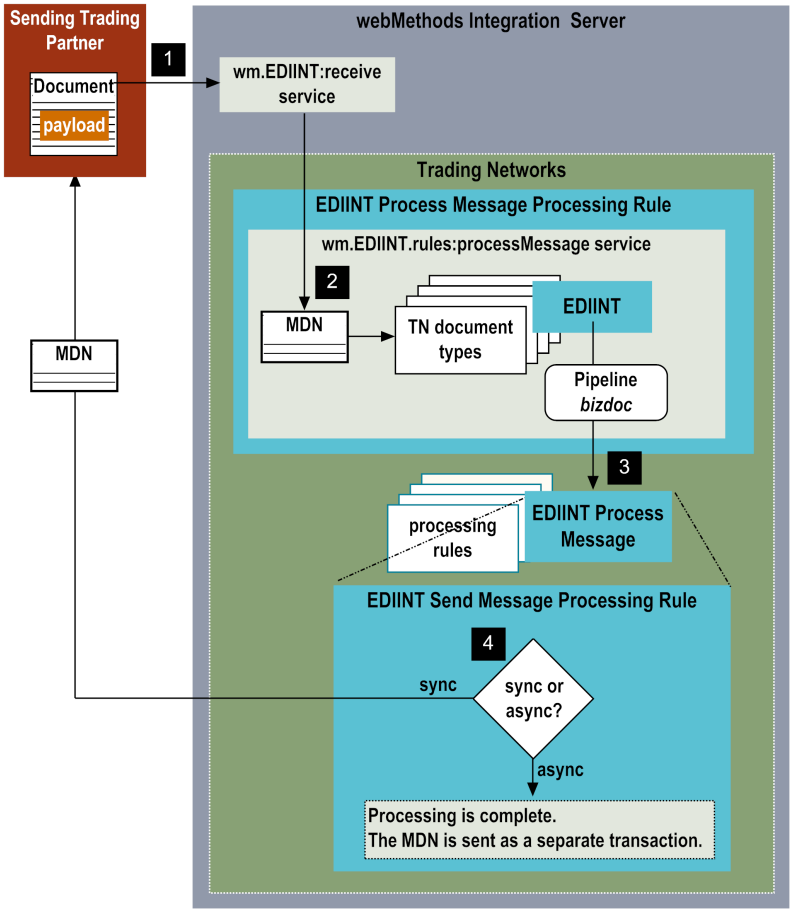
Step	Description
1	You invoke the wm.EDIINT:send service to send a document to Module for EDIINT to be packaged as an EDIINT document and delivered to the receiving trading partner.
2	Using the input information that you provide, the wm.EDIINT:send service creates the EDIINT document (that is, an EDIINT MIME or S/MIME message).
3	The wm.EDIINT:send service uses the TN document types to determine the type of document. The document matches the EDIINT TN document type that is installed into Trading Networks when you install Module for EDIINT. After determining the TN document type, the wm.EDIINT:send service forms a BizDocEnvelope for the inbound document and places it in the pipeline in the *bizdoc*

Step	Description
	variable. A BizDocEnvelope contains the original document and includes additional information that Trading Networks requires for routing and processing the document. One piece of information that Trading Networks can use in the selection of a processing rule is the user status. The EDIINT recognizer sets the user status to "SendMsg." After forming the BizDocEnvelope, the wm.EDIINT:send service sends BizDocEnvelope to Trading Networks for processing.
4	Trading Networks determines the processing rule to use for the document. For outbound EDIINT documents, Trading Networks uses the EDIINT Send Message processing rule that is installed into Trading Networks when you install Module for EDIINT. Trading Networks selects this processing rule because the TN document type is EDIINT and the user status is "SendMsg." This processing rule performs the **Execute a Service** action to invoke the wm.EDIINT.rules:sendMsg service.
5	The wm.EDIINT.rules:sendMsg service delivers the document by invoking the wm.EDIINT.delivery:deliveryDocument service, which in turn delivers the document to the appropriate trading partner.

Sending Outbound EDIINT MDNs

Module for EDIINT automatically sends an outbound MDN when it receives an inbound EDIINT document that requests an MDN. Based on how the sender of the inbound EDIINT document requests the MDN to be sent, the module can send the MDN either synchronously or asynchronously. When the module sends an MDN synchronously, it sends the MDN using the same HTTP connection as that of the inbound EDIINT document. Otherwise, it sends the MDN as a separate transaction.

The following diagram illustrates how Module for EDIINT sends an outbound MDN.



Step	Description
1	The sender sends an EDIINT document to the wm.EDIINT:receive service with a request for an MDN.
2	The wm.EDIINT:receive service accepts the inbound EDIINT document and passes the document to Trading Networks. Trading Networks processes the document using the EDIINT Process Message processing rule. For details about inbound processing, see "Processing Inbound EDIINT Documents" on page 14. The EDIINT Process Message processing rule invokes the wm.EDIINT.rules:processMsg service to process the inbound EDIINT document. Because an MDN is requested, the wm.EDIINT.rules:processMsg service creates the MDN and performs document recognition on the MDN using the Trading Networks TN document types. The MDN matches the EDIINT MDN TN document type. After determining the TN document type, Trading Networks forms a BizDocEnvelope for the MDN and places it in the pipeline in the *bizdoc* variable. The Trading Networks user status for the MDN is set to "SendMDNMsg." The wm.EDIINT.rules:processMsg service then passes the BizDocEnvelope into standard Trading Networks processing.
3	Trading Networks determines the processing rule to use for the MDN. For inbound MDNs, Trading Networks uses the EDIINT Send MDN Message processing rule that is

Step	Description
	installed into Trading Networks when you install Module for EDIINT. Trading Networks selects this processing rule because the TN document type is EDIINT MDN and the user status is "SendMDNMsg." The EDIINT Send MDN Message processing rule performs the **Execute a Service** action to invoke the wm.EDIINT.rules:sendMDN service.
4	The wm.EDIINT.rules:sendMDN service determines what type of MDN the sender has requested (synchronous or asynchronous) and then sends the MDN accordingly: ■ If the sender requested a synchronous MDN, the service returns a synchronous MDN to the sender using the same HTTP connection. ■ If the sender requested an asynchronous MDN, the service invokes the wm.EDIINT.delivery:deliveryDocument service to send an asynchronous MDN as a separate transaction.

Run-Time Processing with EDIINT AS3

To exchange EDIINT AS3 messages with a trading partner, you use an FTP server that is located either on your system or on your trading partner's system. You use just one FTP server. The partner with the FTP server is referred to as the *host partner*.

The partner who accesses the host partner's FTP server is referred to as the *client partner*. To retrieve the EDIINT AS3 messages or files, the client partner needs to log in remotely as an FTP client.

To enable the partners to exchange EDIINT AS3 messages, the host partner must provide the client partner with a particular set of specifications known as a choreography. The AS3 term *choreography* refers to the actions that occur between a client and an FTP server, and the FTP commands that enable those actions to occur. The choreography describes the means for delivering, retrieving, and deleting EDIINT AS3 messages. It includes information about how an upload is communicated to the server as finished and available for a trading partner to download, such as renaming the file extension. In addition, it states whether the partner who downloads the message must send a delete command to clean up the file, or whether the message is removed through other means within the server.

The three categories of actions are:

- Server logon actions (secure or un-secure)

  Module for EDIINT uses the secure FTP support provided by Integration Server. Using this support, clients connect to remote FTP servers using Secure Sockets Layer (SSL).

- Document upload and download actions
- MDN upload and download actions

Each partner must specify the choreography information in a Trading Partner Agreement (TPA), as described in "Creating Trading Partner Agreements (TPAs) for AS3 Support" on page 42.

Run-Time Processing for Host Partners

If you are a host partner (the partner with the FTP server on your Integration Server), you will send and retrieve EDIINT AS3 messages as follows:

Host Partner Sending an EDIINT AS3 Message to a Client Partner

1. The host partner invokes the wm.EDIINT:send service, which creates an EDIINT AS3 message and uploads it to the userFtpRoot*userhome*\AS3\inbox directory on the FTP server.
2. The client partner logs in to the host partner's FTP server and retrieves the message from the host partner's userFtpRoot*userhome*\AS3\inbox directory. The client partner can optionally delete the message.

Host Partner Retrieving an EDIINT AS3 Message or MDN from a Client Partner

1. The client partner logs in to the host partner's FTP server and uploads the EDIINT AS3 message or MDN to the host partner's userFtpRoot*userhome*\AS3\outbox directory.
2. EDIINT submits the EDIINT AS3 message or MDN to Trading Networks to be processed.
3. EDIINT places an MDN response message (if the TPA specifies it) in the host partner's userFtpRoot*userhome*\AS3\inbox directory.

Run-Time Processing for Client Partners

If you are a client partner (the partner accessing a remote FTP server on an Integration Server), you will send and retrieve EDIINT AS3 messages as follows:

Client Partner Sending an EDIINT AS3 Message to the Host Partner

1. The client partner invokes the wm.EDIINT:send service, which creates an EDIINT AS3 message and uploads it to the host partner's FTP server.
2. The host partner processes the EDIINT AS3 message and puts an MDN (if the TPA specifies it) on the host partner's FTP server.

Client Partner Retrieving an EDIINT AS3 message or MDN from the Host Partner

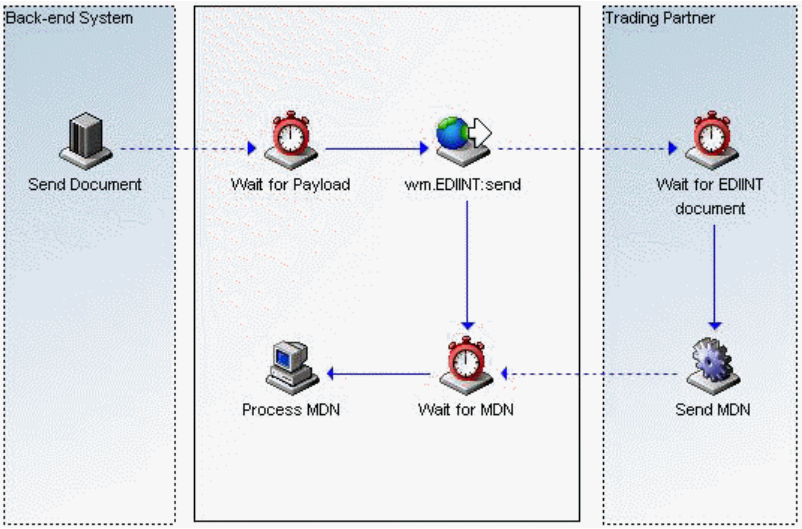
1. The client partner invokes the wm.EDIINT:retrieveAS3Message service, which logs in to the host partner's FTP server and downloads the EDIINT AS3 message or MDN. EDIINT deletes the message if the TPA specifies it.
2. EDIINT submits the EDIINT AS3 message or MDN to Trading Networks to be processed.

Client Partner Retrieving an EDIINT AS3 message or MDN from the Host Partner
3. If the TPA requires an MDN response message, EDIINT logs in to the FTP server and uploads the MDN to the host partner's FTP server.

For information about configuring your system to support EDIINT AS3, see "Configuring Your System to Support EDIINT AS3" on page 41.

Using a Business Process to Send Outbound EDIINT Documents

You can design a process model that waits for a document that you want to send using the EDIINT transport. You can assign a conversation ID to the outbound EDIINT document. When its corresponding MDN is returned, Module for EDIINT assigns the MDN the same conversation ID, so the MDN can rejoin the conversation.



Step	Description
1	The business process waits for a document, for example, from a back-end system. This is the document that you want to send using EDIINT.
2	The business process forms an EDIINT document with the back-end system document as the payload, and sends the EDIINT document to the trading partner. The step invokes the wm.EDIINT:send service to package the back-end system document as the payload of an EDIINT document. The *ConversationID* input parameter to the wm.EDIINT:send service is set to define the value to use for the conversation ID. It should be the same conversation ID that the back-end system document used.
3	The trading partner responds with an MDN. Module for EDIINT sets the conversation ID of the MDN to the value specified for the *ConversationID* input parameter in the preceding step. As a result, the MDN rejoins the correct business process.

2 Installing webMethods Module for EDIINT

Overview

This chapter explains how to install, upgrade, and uninstall webMethods Module for EDIINT. The instructions use the Software AG Installer and the Software AG Uninstaller wizards. For complete information about the wizards or other installation methods, or to install other webMethods products, see the *Installing webMethods Products On Premises* for your release.

Important:
webMethods Module 8.2 SP1 for EDIINT requires webMethods Trading Networks 8.2 or higher to be installed.

Requirements

For a list of the operating systems and webMethods products that webMethods Module 8.2 SP1 for EDIINT supports, see the *webMethods eStandards Modules System Requirements*, available in the webMethods area of the Software AG Documentation Website.

Installing Module for EDIINT

Note:
Module for EDIINT stores data in Trading Networks tables. If you are installing Module for EDIINT in a clustered environment, you must install Module for EDIINT and Trading Networks on each Integration Server in the cluster, and each installation must be identical. For information on running Trading Networks in a clustered environment, see information about configuring Trading Networks in the *webMethods Trading Networks Administrator's Guide* for your release.

To install Module for EDIINT

1. If you are going to install Module for EDIINT on an already installed Integration Server, shut down the Integration Server. Make sure that the version of Integration Server is 8.2 or higher.

2. Download the Software AG Installer from the Empower Product Support Web site at https://empower.softwareag.com.

3. Start the Software AG Installer wizard.

4. In the Release list, click **Software AG Products 8.x**. Provide your Software AG Empower user name and password. Click **Next**. The installer uses the user name and password to connect to the installer server and download the products for which you have purchased licenses.

5. Specify the installation directory as follows:

   - If you are installing Module for EDIINT on an existing Integration Server, specify the installation directory that contains the host Integration Server.
   - If you are installing both the host Integration Server and Module for EDIINT, specify the installation directory to use. (The default is Software AG.)

6. In the product selection list, select **eStandards > webMethods Module 8.2 SP1 for EDIINT**. The installer installs the WmEDIINT package in the *Integration Server_directory*\packages directory.

   You can select any required webMethods components you have not installed, such as Integration Server, Trading Networks, My webMethods Server, and Module for EDI. For a list of required products and their versions, see *webMethods eStandards Modules System Requirements*, available in the webMethods area of the Software AG Documentation Website.

7. Start the Integration Server on which you installed Module for EDIINT.

Upgrading to Module 8.2 SP1 for EDIINT

This upgrade procedure explains how to install Module 8.2 SP1 for EDIINT over a previous version of Module for EDIINT.

Note:
Software AG strongly recommends that you upgrade and migrate in a controlled test environment and test that installation for proper operation before upgrading and migrating your production environment.

To upgrade to Module 8.2 SP1 for EDIINT

1. Shut down all webMethods products and all other applications that are running on the machine on which you are going to install Module for EDIINT. In addition, if any business processes are running, wait for them to complete normally or use Monitor to stop them.

   Important:
   If all products, applications, and business processes are not shut down, the installer will not be able to replace key files that are locked by the operating system.

2. Back up the old Module for EDIINT installation directory:
   Integration Server_directory\packages\WmEDIINT.

3. Uninstall the previous version of the module. For more information, see the installation guide for that version.

Important:
The Software AG Uninstaller does not delete files that you have created or configuration files associated with the older version of the module, nor does it delete the directory structure that contains those files. If you do not want to save those files, go to the *Integration Server_directory*\packages directory and delete the WmEDIINT package directories.

4. Install Module for EDIINT as described in "Installing Module for EDIINT" on page 26. For the installation directory, specify the Module 8.2 SP1 for EDIINT installation directory.

5. Copy the *Integration Server_directory*\packages\WmEDIINT\config\properties.cnf file from your back-up installation to the same directory in your 8.2 SP1 installation.

6. Restart Module for EDIINT.

7. Migrate the data and TPAs from webMethods Trading Networks. To migrate data from Trading Networks, follow the instructions in the *Upgrading Software AG Products* for your release.

Note:
When migrating from version 6.5 to a later release, the value of S/MIME Type field is not migrated.

8. Run the service, wm.EDIINT.doc:migrateSMimeType in Integration Server Administrator to migrate the value of S/MIME Type field from version 6.5 to the SMIME Type field in the later versions of the module.

9. Run the service, wm.EDIINT.doc:migrateRequestSignedReceipt in Integration Server Administrator to migrate the value of Signed MDN field from version 6.5 to the Request Signed Receipt field in the later versions of the module.

10. Run the service, wm.EDIINT.doc:migrateCompression in Integration Server Administrator to migrate the value of Compressed field from version 6.5 to the Compression field in the later versions of the module.

Uninstalling Module for EDIINT

To uninstall Module for EDIINT

1. Shut down the Integration Server that hosts Module for EDIINT.

2. Start Software AG Uninstaller, as follows:

System	Instructions
Windows	In the **Add or Remove Programs** window, select **Software AG Products 8.x** *installation_directory*, where *installation_directory* is the

System	Instructions
	installation directory of the Integration Server on which Module for EDIINT is installed.
UNIX	Go to the *installation_directory*\bin directory of the installation that includes the Integration Server on which Module for EDIINT is installed and enter `uninstall` (wizard) or `uninstall -console` (console mode).

3. In the **Select products to uninstall** list, select **eStandards > webMethods Module 8.2 SP1 for EDIINT**.

4. Software AG Uninstaller does not delete files that you have created or configuration files associated with Module for EDIINT, nor does it delete the directory structure that contains those files. If you do not want to save those files, go to the *Integration Server_directory*\packages directory and delete the WmEDIINT package directories.

5. Restart the Integration Server from which you uninstalled Module for EDIINT.

3 Before You Can Transport Documents Using EDIINT

Overview

Before you can transport EDI or non-EDI documents using the EDIINT transport, perform the following tasks:

- Ensure that the partner profiles for senders and receivers contain the information that Module for EDIINT requires, as described in "Including EDIINT Information in Profiles" on page 32.
- If you want to use EDIINT AS1 (SMTP), or if you want to use EDIINT AS2 and have partners send MDNs via SMTP, configure the SMTP settings to enable Module for EDIINT to send and receive EDIINT documents via SMTP, as described in "Configuring SMTP Settings to Enable EDIINT Message Exchange" on page 39.
- If you want to use EDIINT AS3 (FTPS), configure your system as described in "Configuring Your System to Support EDIINT AS3" on page 41.

  Note: If you use HTTP, HTTPS, or FTPS, for instructions about how to add a port, see the *webMethods Integration Server Administrator's Guide* for your release. No EDIINT-specific settings are required to add an HTTP, HTTPS, or FTPS port.
- If you want to store large payloads and attachments in the Module for EDIINT file system instead of, or in addition to, the Trading Networks database, configure the module appropriately, as described in "Configuring Where Payloads Are Persisted" on page 49.
- If you want Module for EDIINT to send the payload of the document to Trading Networks for further processing, configure the module appropriately, as described in "Configuring Whether Trading Networks Is to Process Payloads" on page 49.
- If you want to configure EDIINT properties, see "Configuring EDIINT Properties" on page 52.

Including EDIINT Information in Profiles

To use the EDIINT transport when exchanging documents between partners (senders and receivers), the partners' profiles in Trading Networks must contain EDIINT information that Module for

EDIINT requires. For information about managing partner profiles, see the *webMethods Trading Networks Administrator's Guide* for your release.

The following sections identify the specific information that you supply to use the EDIINT transport. The sections do not describe all profile fields. For descriptions of profile fields that are not listed in the sections below, see the *webMethods Trading Networks Administrator's Guide* for your release.

External IDs Tab of the Profile

Provide values for the following fields on the **External IDs** tab of the Partner Profile page in My webMethods:

Corporate Tab field	Description
ID Type and **Value**	Select all versions of the EDIINT standard that the partner's corporation uses: **EDIINT AS1**, **EDIINT AS2**, and/or **EDIINT AS3**. You may also select any number of other external ID types, such as **User Defined**. Then, for each selected external ID type, specify a value to identify the partner. You may assign the same external ID value for multiple external ID types within the same profile. For example, if the partner uses **EDIINT AS2** and **EDIINT AS3**, you can specify the same identification value for both types. The external ID type and its value correspond to the values in the EDIINT document headers "From" and "To" (for AS1), "AS2-From" and "AS2-To" (for AS2), and "AS3-From" and "AS3-To" (for AS3).

Behavior of External ID Matching

If you used a version of Module for EDIINT prior to 6.5.2 and you want to use your existing profiles in 8.2 SP1, you have two options:

- Update your existing profiles so they use the external ID types **EDIINT AS1**, **EDIINT AS2**, or **EDIINT AS3** instead of the existing external ID types. For details, see "Updating Existing Profiles to Use EDIINT AS1, EDIINT AS2, and EDIINT AS3" on page 34.

  -OR-

- If you want to continue using the external ID types of your existing profiles without having to change your existing profiles, turn off the new **EDIINT ID Match** option to make the module behave as it did prior to version 8.2 SP1. For details, see "Using the EDIINT ID Match Option" on page 34.

  Important:
  You must either update your existing profiles or turn off the **EDIINT ID Match** option. Otherwise, you may get unpredictable results when trying to match external IDs.

Updating Existing Profiles to Use EDIINT AS1, EDIINT AS2, and EDIINT AS3

Prior to version 6.5.2, Module for EDIINT only looked at the external ID value to find a matching profile when sending or receiving documents; it ignored the external ID type. Thus, each external ID value was required to be unique; a value could not be duplicated within a single profile or in any other profile.

With version 8.0 and higher, Module for EDIINT looks at both the external ID value and the external ID type in order to find a matching profile. This means you may assign the same external ID value for multiple external ID types within the same profile or in any other profile (or the values may be unique).

By default, when you send an EDIINT document the wm.EDIINT:send service tries to match the document's "To" header (for example, `AS2-To: 987654321`) with the external ID type and value defined in a partner profile (for example, **EDIINT AS2** and `987654321`).

Similarly, when you receive an EDIINT document the wm.EDIINT:receive service tries to match the value of its sender ID input parameter (which specifies both type and value) with a partner profile's external ID type and value.

Important:
If you use the **EDIINT AS1**, **EDIINT AS2**, or **EDIINT AS3** types in your profiles, make sure the **EDIINT ID Match** option is selected (this is the default). If you turn off this option, you may get unpredictable results when trying to match external IDs. For details, see "Using the EDIINT ID Match Option" on page 34.

Using the EDIINT ID Match Option

If you do not want to use the external ID types **EDIINT AS1**, **EDIINT AS2**, or **EDIINT AS3** in your profiles, turn off the **EDIINT ID Match** option. Otherwise, leave this option turned on.

If you fail to set this option appropriately, you may get unpredictable results when trying to match external IDs. For example, if you define a profile with three external ID types that have identical values as follows:

External ID type	Value
EDIINT AS2	987654321
EDIINT AS3	987654321
User Defined	987654321

and you send an EDIINT document with the following headers:

```
AS2-From: 123456789
```

```
AS2-To: 987654321
```

the **EDIINT ID Match** option controls the external ID matching as follows:

When EDIINT ID Match is...	The matching external ID...
On	Is **EDIINT AS2**.
Off	Could be any of the three IDs; you cannot predict which one will match.

To set the EDIINT ID Match option

1. Open Integration Server Administrator if it is not already open.
2. In the **Solutions** menu of the navigation panel, click **EDIINT**.

   Integration Server Administrator opens a new browser window to display the Module for EDIINT home page.
3. In the navigation panel of the Module for EDIINT home page, click **Configuration**.
4. Specify how to match external IDs as follows:

Do this...	To have Module for EDI try to match...
Select the **EDIINT ID Match** check box	Both the external ID value and the external ID type.
Clear the **EDIINT ID Match** check box	Only the external ID value.

Note:
You can also set this option in My webMethods by configuring the EDIINTIDMatch EDIINT property. For more information, see "Configuring EDIINT Properties" on page 52.

Delivery Settings Tab of the Profile

In My webMethods, on the **Delivery Settings** tab of the Partner Profile page, specify the delivery methods that EDIINT requires. The wm.EDIINT:send service uses the information that you specify on this tab to send outbound EDIINT messages and MDNs.

If you are using...	Specify...
EDIINT AS1	At least one of the following delivery methods: ■ **Primary E-mail** ■ **Secondary E-mail** **Note:**

If you are using...	Specify...
	You must define the delivery method in both the sender and receiver profiles. The e-mail address in the sender's profile is used for the "From" address and the e-mail address in the receiver's profile is used for the delivery e-mail address ("To").
EDIINT AS2	At least one of the following delivery methods: ■ **Primary HTTP** ■ **Secondary HTTP** ■ **Primary HTTPS** ■ **Secondary HTTPS**

Note:
EDIINT AS3 is an available delivery method as well. For details, see "Configuring Your System to Support EDIINT AS3" on page 41.

Extended Fields Tab of the Profile

The following table lists the **Extended Fields** tab fields you should supply for EDIINT on the Partner Profile page in My webMethods.

In this Extended Fields tab field...	Specify...
AS1MDNURL	The e-mail address that is to accept inbound AS1 EDIINT MDNs (for example, `receiver@company.com`), if you are using EDIINT AS1. An inbound AS1 MDN is by definition asynchronous because it is not returned using the same connection as that of the originally sent document.

AS2MDNURL	The URL that is to accept inbound AS2 EDIINT MDNs, if you are using EDIINT AS2.
	■ To accept MDNs via HTTP, specify a URL that includes the `/invoke/` element to invoke the wm.EDIINT:receive service. For example (where *host:port* would be an actual host and port number):
	`http:// host:port /invoke/wm.EDIINT/receive`
	■ To accept AS2 MDNs via SMTP, specify a URI similar to the following:
	`mailto:receiver@company.com`

In this Extended Fields tab field...	Specify...
	An inbound AS2 MDN could be synchronous (HTTP only) or asynchronous (SMTP or HTTP). A synchronous MDN is returned using the same HTTP connection as that of the originally sent document.
Encryption Algorithm	The encryption option to use for outbound EDIINT messages. Select from: ■ **TripleDES** (This is the default.) ■ **DES** ■ **RC2 40** (40 bits) ■ **RC2 64** (64 bits) ■ **RC2 128** (128 bits)
SMIME Type	The SMIME type to use for payloads sent and received by the trading partner. Use: ■ `plain`, for payloads are neither signed nor encrypted. ■ `signed`, for payloads are signed. ■ `encrypted`, for payloads are encrypted. ■ `signedAndEncrypted`, for payloads that are signed and encrypted. This is the default. For more information about how the SMIME Type extended profile field is used during inbound processing, see "How the SMIME Type Profile Field Affects Processing Payloads" on page 68. For more information about how the SMIME Type extended profile field is used during outbound processing, see "Setting the SMIME Type of the Outbound EDIINT Document" on page 72.
Compression	Whether the outbound message is compressed before signing and/or encrypting it. Use: ■ `True`, to compress the outbound message before signing and/or encrypting it. ■ `False`, to sign and/or encrypt the message without compressing it. This is the default.
Delivery Method	The delivery method you want to use to send the EDIINT document. ■ **For EDIINT AS1**, specify one of the following: ■ `PrimarySMTP` (corresponds to the Trading Networks **Primary E-mail** delivery method)

In this Extended Fields tab field...	Specify...
	■ `SecondarySMTP` (corresponds to the Trading Networks **Secondary E-mail** delivery method) ■ **For EDIINT AS2**, specify one of the following: ■ `PrimaryHTTP` ■ `SecondaryHTTP` ■ `PrimaryHTTPS` ■ `SecondaryHTTPS` ■ **For EDIINT AS3**, specify `AS3`.
Request MDN	Whether you want the receiver to return an MDN. Use: ■ `none`, to not request a return MDN. ■ `synchronousMDN`, to request a return synchronous MDN. ■ `asynchronousMDN`, to request a return asynchronous MDN.
Request Signed Receipt	Whether you want a signed MDN. Use: ■ `True`, to request a signed MDN. ■ `False`, to request a plain (unsigned) MDN.
FTPUserName	Your partner's user name if you are using EDIINT AS3 and you are the hosting partner (that is, your partner will access your FTP server), so that EDIINT can place EDIINT AS3 messages and MDNs in the userFtpRoot*username*\AS3\inbox directory. Your partner will download the messages and MDNs from this directory. For more information about the inbox directory, see "Creating Directories for Uploading/ Downloading" on page 41.

Certificates Tab of the Profile

The following table lists the information that you should supply for EDIINT on the **Certificates** tab of the Partner Profile page in My webMethods.

Use this sub-tab of the Certificates tab...	If you...
Sign/Verify	Want Module for EDIINT to be able to digitally sign outbound EDIINT documents.

Use this sub-tab of the Certificates tab...	If you...
	On the **Sign/Verify** tab of the profile, specify the certificates for your corporation, along with your private key. ■ You can set up a default signing certificate information in the **Enterprise** profile. ■ If you need to use a specific certificate to sign outbound documents for a particular partner, specify the certificate information on the **Sign/Verify** tab of that partner's profile. If you expect to receive an EDIINT document with a digital signature from a partner and you want Module for EDIINT to be able to verify the digital signature, specify "verify certificates" on the **Sign/Verify** tab in the partner's profile. You specify the certificates for your partner's corporation.
Decrypt/Encrypt	Expect to receive encrypted EDIINT documents from partners. On the **Decrypt/Encrypt** tab of the profile, specify the certificates for your corporation, along with your private key that you need to use to decrypt the documents. ■ You can set up default decrypting certificate information in the **Enterprise** profile. ■ If you need to use a specific certificate to decrypt inbound documents from a particular partner, specify the certificate information on the **Decrypt/Encrypt** tab of that partner's profile. If you want Module for EDIINT to be able to encrypt outbound EDIINT documents for a partner, specify "encrypt certificates" on the **Decrypt/Encrypt** tab in the partner's profile. You specify the certificates for your partner's corporation.

Configuring SMTP Settings to Enable EDIINT Message Exchange

To allow Module for EDIINT to receive and send documents using SMTP, you must configure inbound (for receiving) and outbound (for sending) SMTP settings. You must configure these settings if you want to use:

- EDIINT AS1
- EDIINT AS2 and have partners send MDNs via SMTP

Important:

Check your mailbox settings for message size limitations that could adversely affect your ability to receive or send large EDIINT documents.

Configuring Inbound EDIINT SMTP Settings

To configure your system to be able to receive inbound EDIINT documents via SMTP

1. Set up an e-mail account with an e-mail service provider that supports either the POP3 or IMAP protocols.

2. In Integration Server Administrator, add a **webMethods/Email** port that corresponds to the e-mail host that you established in the previous step. For instructions about how to add a port, see the *webMethods Integration Server Administrator's Guide* for your release.

   The following table lists information you should specify when adding the port:

In this section of the screen...	For this field...	Specify...
Package	**Package Name**	Whether you want the port available whenever the service is running: ■ **WmEDIINT** if you want the port disabled when the WmEDIINT package is disabled. ■ **WmRoot** if you want the port available whenever the server is running.
Server Information	All fields	Information about the e-mail host that you established in the previous step.
Security	**Require authorization within message**	`No`
	Run services as user	A user account with administrator authority, e.g., Administrator.
Message Processing	**Global Service**	`wm.EDIINT:receive`
	Default service	leave blank
	Invoke service for each part of multipart message	`No`
	Include e-mail headers when passing message to content handler	`Yes`

Note:
If a field in the Message Processing section of the screen is not listed in the table above, Module for EDIINT does not use it. Leave the field set to its default value.

3. Enable the port.

4. Edit the port's **Access Mode**, and click **Set Access Mode to Allow by Default**.

Configuring Outbound EDIINT SMTP Settings

To send outbound EDIINT documents or MDNs via SMTP, you must configure the name of the SMTP server you want to use for outbound EDIINT documents. You only need to perform this procedure if you want to use EDIINT AS1.

To enable outbound EDIINT SMTP transport

1. Open Integration Server Administrator.

2. In the **Solutions** menu of the navigation panel, click **EDIINT**. Integration Server Administrator opens a new browser window to display the Module for EDIINT home page.

3. In the navigation panel of the Module for EDIINT home page, click **Configuration**.

4. In the **SMTP Server** field, type the name of your SMTP server.

5. Click **Save Changes**.

Note:
You can also set the SMTP server in My webMethods by configuring the mailhost EDIINT property. For more information, see "Configuring EDIINT Properties" on page 52.

Configuring Your System to Support EDIINT AS3

Configuring your system to support EDIINT AS3 involves creating directories for uploading and downloading, and then creating Trading Partner Agreements (TPAs) for AS3 support.

Creating Directories for Uploading/Downloading

The host partner must create the following directories in Integration Server, under the default FTP root directory that Integration Server creates at startup:

userFtpRoot*userhome*\AS3

where *userhome* is the user's FTP home directory.

Note:

To rename the default userFtpRoot directory, use the Integration Server configuration parameter watt.server.userFtpRoot. For details, see the *webMethods Integration Server Administrator's Guide* for your release.

The host partner must then create the following directories under the AS3 directory:

- userFtpRoot*userhome*\AS3\inbox—The directory from which the host partner will download EDIINT AS3 messages.
- userFtpRoot*userhome*\AS3\outbox—The directory to which the host partner will upload EDIINT AS3 messages.

Creating Trading Partner Agreements (TPAs) for AS3 Support

A Trading Partner Agreement (TPA) in Trading Networks is a set of parameters that you can use to govern how documents are exchanged between two trading partners. One partner fulfills the sender role during document exchange, and the other partner fulfills the receiver role. Both the sender and receiver in a TPA must be a partner who has an existing profile in your Trading Networks system.

Each TPA must specify a unique combination of the following:

- A partner that represents the originator of a send or retrieve operation.
- The partner of the originator.
- The Agreement ID (the type of the TPA). To support EDIINT AS3 message exchange, use the predefined Agreement ID EDIINTAS3.

You might have multiple TPAs for a pair of trading partners. For example, if PartnerA is the originator of a send or retrieve operation, you would define the following TPA:

For this TPA field...	Specify...
Sender	PartnerB
Receiver	PartnerA
Agreement ID (type of TPA)	EDIINTAS3

Conversely, if PartnerB is the originator of a send or retrieve operation, you would define the following TPA:

For this TPA field...	Specify...
Sender	PartnerB
Receiver	PartnerA
Agreement ID (type of TPA)	EDIINTAS3

Creating a new TPA

To create a new TPA

1. In My webMethods, go to **Administration > Integration > B2B > Trading Partner Agreements**.

2. On the Agreements screen, create either a new TPA or a duplicate TPA. When you duplicate a TPA, you can update any or all of the fields. Also, you can use the duplicate TPA option to create a template TPA.

   - To create a new TPA (the TPA fields are empty), select **Add TPA**.

     Trading Networks displays the Trading Partner Agreement Details screen.

   - To create a duplicate TPA (the TPA fields are filled with values from an existing TPA), click the row containing the name of the TPA that you want to duplicate and then click  .

     Trading Networks displays the Agreement Details screen.

3. On the Trading Partner Agreement Details screen, complete the following fields for the TPA you want to create:

   Note:
   The **Sender**, **Receiver**, and **Agreement ID** fields must be unique for each TPA. After you create a TPA, you cannot change or update these fields of the TPA.

For this TPA field...	Specify...
Agreement ID	The Agreement ID `EDIINTAS3`, which indicates that the type of agreement between the two partners is an AS3 agreement using EDIINT. **Note:** You will not be able to continue creating a TPA unless you specify the **Agreement ID**.
Sender	The name of the trading partner that has the sender role in the transaction the TPA will govern. Type in the name of the partner or click **Edit** to select the sender from the Select Partner dialog. This list includes your own profile (**Enterprise**). For EDI, to create a template that you will duplicate to create other TPAs, you can use the default value of `Unknown`.
Receiver	The name of the trading partner that has the receiver role in the transaction the TPA will govern. Type the name of the receiver or click **Edit** to select the receiver from the Select Partner dialog. This list includes your own profile (**Enterprise**).

For this TPA field...	Specify...
	For EDI, to create a template that you will duplicate to create other TPAs, you can use the default value of `Unknown`.
Description	Optional. A description for the TPA. You can use up to 1024 characters of any type.
Data Status	Whether you want to be able to modify the values of the TPA data of the IS document type. The data status is only applicable when the agreement status is Agreed.
Initialization Service	The `wm.EDIINT.TPA:initService` service. This service populates the inputs to the variables in the IS document type `wm.EDIINT.TPA:EDIINTAS3` with default values. Type the name of the initialization service located on the server or click **Edit** to browse the services and select the one you want to use.
IS Document Type	The IS document type `wm.EDIINT.TPA:EDIINTAS3`. This IS document type defines the application-specific TPA data. The TPA data is used to govern the exchange of documents between the two partners. Alternatively, click **Find IS Document Type** to browse the IS document types and select `wm.EDIINT.TPA:EDIINTAS3`. Trading Networks displays the data tree input values of the selected IS document type in the bottom panel of the Agreement Details screen.

For more information about creating TPAs, see the *webMethods Trading Networks Administrator's Guide* for your release.

4. Click the **Set Inputs** icon and provide values for the following upload input parameters for the document type wm.EDIINT.TPA:EDIINTAS3:

For this upload parameter...	Specify...
AS3FTPServerLocation	Whether to upload/download EDIINT AS3 messages to a remote or local FTP server: ■ **remote**—Upload/download EDIINT AS3 messages to/from the remote FTP server in the **FTPUpload** and **FTPDownload** fields. ■ **local**—Upload/download EDIINT AS3 messages to/from the local FTP server. No other TPA fields will be used.
uploadService	The wm.EDIINT.delivery.defaultFTPUpload service, which uploads AS3 files.
serverhost	The name or IP address of the FTP server.

For this upload parameter...	Specify...
serverport	The port number on which the FTP server listens for requests. The default is 21.
dataport	Optional. The listener port number of the data transfer channel, for example, `3345`.
username	The FTP user on the remote FTP server.
password	The password for the user specified in **username**.
account	Optional. The user name for an account on the FTP server. Specify a value if your FTP host requires account information. The account is defined in the FTP protocol to further identify the user and password specified in **username** and **password**.
transfertype	The type of the FTP data transfer mode: ■ **active**—Active FTP data transfer mode. This is the default. ■ **passive**—Passive FTP data transfer mode.
encoding	Optional. The default character set for encoding data transferred during this session. Specify an IANA-registered character set, for example, `ISO-8859-1`. If you do not set **encoding**, the default JVM encoding is used.
timeout	Optional. The time (measured in seconds) to wait for a response from the FTP server before timing out and terminating the request. The default is to wait forever.
secureFTP	The type of the remote FTP server to connect to.
securedata	Whether to protect the FTP data channel: ■ **True**—Protect the FTP data channel. ■ **False**—Do not protect the FTP data channel.
auth	The authentication/security mechanism: ■ **SSL** ■ **TLS** ■ **TLS-P**
dirpath	Optional. The directory path to which EDIINT AS3 messages are uploaded. If you do not specify a directory path, the current directory will be used.

For this upload parameter...	Specify...
fileExtension	Optional. The file extension to be assigned to the uploaded EDIINT AS3 message file, for example, `msg`. EDIINT AS3 message file names are generated using the following naming convention: `MMddhhmmSSss` where `MM` is month, `dd` is day, `hh` is hour, `mm` is minutes, and `SSss` is seconds. For example, a generated file name with the extension `msg` might be 122012002222.msg.
renameTo	Optional. The new file extension. After uploading the EDIINT AS3 message file, move it to a specified directory, and optionally rename the file extension. For example, if your FTP *put* command places the AS3 file in the tmp directory and you want to move it to the outbox directory after uploading it, specify this command in the **renameTo** field: `/outbox/*` The wildcard character `*` is a placeholder for the file name. If you specified a file extension in the **fileExtension** field (for example, `msg`), you would specify this command in the **renameTo** field: `/outbox/*`.msg Optionally, you can rename the file extension in this field as well.
MDNDirpath	Optional. The directory to which MDNs are uploaded.
MDNFileExtension	Optional. The file extension to be assigned to the uploaded MDN file, for example, `mdn`. MDN file names are generated using the following naming convention: `MMddhhmmSSss` where `MM` is month, `dd` is day, `hh` is hour, `mm` is minutes, and `SSss` is seconds. For example, a generated file name with the extension `mdn` might be 122012002222.mdn.
MDNRenameTo	Optional. The new file extension. After uploading the MDN files, move it to a specified directory, and optionally rename its file extension.

For this upload parameter...	Specify...
	For example, if your FTP *put* command places the MDN file in the tmp directory and you want to move it to the outbox directory after uploading it, specify this command in the **renameTo** field: `/outbox/*` The wildcard character `*` is a placeholder for the file name. If you specified a file extension in the **fileExtension** field (for example, `mdn`), you would specify this command in the **renameTo** field: `/outbox/*`.mdn Optionally, you can rename the file extension in this field as well.
storeUnique	Optional. A unique file name that EDIINT assigns after the file is uploaded.
transfermode	The type of the FTP file transfer mode: ■ **binary** (required for AS3) ■ **ascii**

5. Scroll down and provide values for the following download input parameters for wm.EDIINT.TPA:EDIINTAS3:

For this download parameter...	Specify...
downloadService	The wm.EDIINT.delivery.defaultFTPDownloadservice, which downloads AS3 files.
serverhost	The name or IP address of the FTP server.
serverport	The port number of the FTP server. The default is `21`.
dataport	Optional. The listener port number of the data transfer channel, for example, `3345`.
username	The FTP user on the remote FTP server.
password	The password for the user specified in **username**.
account	Optional. The user name for an account on the FTP server. Specify a value if your FTP host requires account information. The account is defined in the FTP protocol to further identify the user and password specified in **username** and **password**.

For this download parameter...	Specify...
transfertype	The type of the FTP data transfer mode: ■ **active**—Active FTP data transfer mode. This is the default. ■ **passive**—Passive FTP data transfer mode.
encoding	Optional. The default character set for encoding data transferred during this session. Specify an IANA-registered character set, for example, `ISO-8859-1`. If you do not set **encoding**, the default JVM encoding is used.
timeout	Optional. The time (measured in seconds) to wait for a response from the FTP server before timing out and terminating the request. The default is to wait forever.
secureFTP	The type of the remote FTP server to connect to.
securedata	Whether to protect the FTP data channel: ■ **True**—Protect the FTP data channel. ■ **False**—Do not protect the FTP data channel.
auth	The authentication/security mechanism: ■ **SSL** ■ **TLS** ■ **TLS-P**
dirpath	Optional. The directory path to which EDIINT AS3 messages are downloaded. If you do not specify a directory path, the current directory will be used.
filenamepattern	Optional. The EDIINT AS3 message file pattern, for example, `*.msg`.
MDNDirpath	Optional. The directory to which MDNs are downloaded. You can specify either the path relative to **dirpath** or the absolute path.
MDNFilenamepattern	Optional. The MDN message file pattern, for example, `*.mdn`.
deleteFile	Whether the file is to be deleted after downloading it.
transfermode	The type of the FTP file transfer mode: ■ **binary** (required for AS3) ■ **ascii**

6. Click **OK** to create the TPA.

Configuring Where Payloads Are Persisted

By default, all payloads are sent to Trading Networks during processing and persisted in the Trading Networks database. When payloads are large or contain attachments, this action can slow processing significantly.

Alternatively, large payloads and attachments can be persisted in the file system or in both the Trading Networks database and the file system.

Persisting Payloads in Both Trading Networks and the File System

To configure Module for EDIINT to persist the payload in both the Trading Networks database and the file system, you must:

- Assign priority to the EDIINT Process Message - Persist in File System processing rule over the default processing rule, EDIINT Process Message.
- Set the persistInTNAndFileSystem property to `true` and define a file location in the payloadDir property. For more information about configuring EDIINT properties, see "Configuring EDIINT Properties" on page 52.

Persisting Payloads in Only the File System

To configure Module for EDIINT to persist payloads in only the file system, you must:

- Assign priority to the EDIINT Process Message - Persist in File System processing rule over the default processing rule, EDIINT Process Message. EDIINT Process Message - Persist in File System invokes the processMsg_persistPayload service.
- Set the persistInTNAndFileSystem property to `false` and define a file location in the payloadDir property. For more information about configuring EDIINT properties, see "Configuring EDIINT Properties" on page 52.

Configuring Whether Trading Networks Is to Process Payloads

In addition to having Module for EDIINT perform transport-level processing for an entire inbound EDIINT document, you can perform further processing on the payload of the document by having the module send the payload to Trading Networks for separate processing. As installed, the module is configured to submit the payload to Trading Networks.

When Module for EDIINT is configured to send the payload to Trading Networks, the module submits the payload after it completes transport-level processing.

Use the Module for EDI to process payloads that have one of the following content types:

- application/edi-x12
- application/edifact

- application/xml
- application/edi-consent (which you can use to submit TRADACOMS payloads)

If you want to have Module for EDIINT submit payloads that have a different content type, you must provide your own service to process the payload and submit it to Trading Networks.

To enable or disable Trading Networks payload processing

1. Open Integration Server Administrator.
2. In the **Solutions** menu of the navigation panel, click **EDIINT**.

   Integration Server Administrator opens a new browser window to display the Module for EDIINT home page.
3. In the navigation panel of the Module for EDIINT home page, click **Configuration**.
4. Enable or disable Module for EDIINT to or from submitting payloads to Trading Networks by doing one of the following:
   - To enable, select the **Submit payload to TN** check box.
   - To disable, clear the **Submit payload to TN** check box.

   Note:
   You can also set this option in My webMethods by configuring the submitPayload EDIINT property. For more information, see "Configuring EDIINT Properties" on page 52.
5. In the **User Process Payload Service** field, specify a service that you created to process payloads and submit them to Trading Networks. You only need to specify a service if:
   - You selected the **Submit payload to TN** check box.

     -AND-
   - The content types of the inbound payloads are not application/edi-X12, application/EDIFACT, application/XML, or application/edi-consent.

   The service you specify in the **User Process Payload Service** field must accept the following input variables:

Input Variable	Description
stream	**InputStream** The payload.
contentType	**String** The content type of the payload.
EDIINTbizdoc	**Document** The bizdoc that contains the original EDIINT message. For the structure of *EDIINTbizdoc*, see the

Input Variable	Description
	wm.tn.rec:BizDocEnvelope service in the *webMethods Trading Networks Built-In Services Reference* for your release.
attachmentsPartName	**Document List** Collection of attachments from the original EDIINT message.

Variable	Description
attachmentPartName	**String** Internal name assigned to the attachment for unique identification.
attachmentPartStream	**InputStream** The attachment stream attached to the original message.

Note:
You can also specify this service in My webMethods by configuring the userProcessPayloadService EDIINT property. For more information, see "Configuring EDIINT Properties" on page 52.

6. Module for EDIINT ignores your settings in the following situations:

Conditions	Behavior of Module for EDIINT

■ You select the **Submit payload to TN** check box -AND-
■ You specify a service in the **User Process Payload Service** field -AND-
■ The inbound payload has a content type that is one of:
  ■ application/edi-X12
  ■ application/EDIFACT
  ■ application/XML
  ■ application/edi-Consent
	Module for EDI ignores the service you specify in the **User Process Payload Service** field.
	When you select the **Submit payload to TN** check box, Module for EDIINT always uses the Module for EDI to process payloads that have one of the following content types:
	■ application/edi-X12
	■ application/EDIFACT
	■ application/XML
	■ application/edi-Consent

■ You select the **Submit payload to TN** check box -AND- ■ You do not specify a service in the **User Process Payload Service** field -AND- ■ The content type of the inbound payload is not one of: ■ application/edi-X12	Module for EDIINT ignores the check in the **Submit payload to TN** check box. The module cannot submit a payload with an unsupported content type to Trading Networks. You must provide a service that submits payloads with unsupported content types.

Conditions	Behavior of Module for EDIINT
■ application/EDIFACT ■ application/XML ■ application/edi-consent	
■ You clear the **Submit payload to TN** check box -AND- ■ You specify a service in the **User Process Payload Service** field.	Module for EDIINT ignores the service you specify in the **User Process Payload Service** field. The module only invokes a service you specify to submit payloads to Trading Networks when you select the **Submit payload to TN** check box.

7. Click **Save Changes**.

Note:
For information about how to set up Module for EDI and Trading Networks to process EDI documents, see the *webMethods Module for EDI Installation and User's Guide*.

Configuring EDIINT Properties

Use the EDIINT properties to control some of the module's functions.

To configure EDIINT properties

1. In My webMethods, go to **Administration > Integration > B2B Settings > Configure Properties**.

2. In the Module for EDIINT Configuration Properties panel, specify values for the following properties:

Property	Definition
mailhost	Specifies the name of the mail host to be used for EDIINT AS1 (SMTP) transfers. The default is smtp.company.com.
EDIIINTIDMatch	Determines if you can use the external ID types for **EDIINT AS1**, **EDIINT AS2**, and **EDIINT AS3** in your profiles: ■ `True`—External ID types can be used in your profile. This is the default. ■ `False`—External ID types cannot be used in your profile.
submitPayload	When the payload is an EDI or XML document (that is, its content type is either application/xml, application/edi-x12,

Property	Definition
	application/edifact, or application/edi-consent), determines if the payload is submitted to Trading Networks for processing: ■ `True`—Submits the payload to Trading Networks. This is the default. ■ `False` —Does not submit the payload to Trading Networks.
userProcessPayloadService	Determines what service is configured to process the payload of a non-EDI or non-XML document (that is, when the content type of the payload is anything except application/xml, application/edi-x12, application/edifact, or application/edi-consent).
waitInSeconds	Determines how many seconds Module for EDIINT waits before sending an asynchronous MDN. The default is 30.
processPayloadIfMDNNotSent	Determines if the payload is to be processed when the asynchronous MDN is not sent successfully: ■ `True`—Processes the payload. ■ `False`—Does not process the payload. This is the default.
persistSentAttachments	Determines if the attachments and headers that were sent with an EDI document are added to the content part of the Trading Networks bizdoc: ■ `True` —Both attachments and headers are added to the content part of the bizdoc. This is the default. ■ `HeaderOnly`—Only headers are added to the content part of the bizdoc. ■ `False`—Neither headers nor attachments are added to the content part of the bizdoc.
payloadDir	Specifies the location of the file system in which to save the payload and the attachments of an incoming EDI document. There is no default. **Note:** When the **EDIINT Process Message - Persist in File system** processing rule invokes the wm.EDIINT:rules:processMsg_persistPayload service and a value is given to the *PayloadDir* parameter, that value has precedence over the value set in this property.

<table>
<tr><th>Property</th><th>Definition</th></tr>
<tr><td>AS2RestartTempFilePath</td><td>Specifies the location in which to store incoming AS2 messages when AS2 restart is enabled. The default is packages/WmEDIINT/pub/as2restart_transactions.</td></tr>
<tr><td>AS2RestartRetryCount</td><td>Specifies the number of times Module for EDIINT will attempt to restart message transmission that has been interrupted. The default is 5.</td></tr>
<tr><td>AS2RestartRetryIntervalInSeconds</td><td>Determines how many seconds Module for EDIINT waits between attempts to restart transmitting the interrupted message. The default is 2.</td></tr>
<tr><td>AS2RestartEnabled</td><td>Determines if the outgoing messages are AS2 restart enabled.

You can override this global property by setting the enableAS2Restart input parameter in the wm.EDIINT:Send service.

■ <code>True</code>— Outgoing message is AS2 restart enabled.

■ <code>False</code> —Outgoing message is not AS2 restart enabled. This is the default.</td></tr>
<tr><td>persistInTNAndFileSystem</td><td>Optional. In conjunction with the payloadDir property, specifies whether the EDI payload is persisted in the file system in addition to the Trading Networks database, as follows:

<table>
<tr><th>When payloadDir is...</th><th>And persistInTNAndFileSystem is...</th><th>The EDI payload is saved in...</th></tr>
<tr><td>Specified</td><td>True</td><td>Both Trading Networks and the file system.

Note:
Persisting payloads in both the Trading Networks and the file system slows performance even further than persisting them in only one of the options.</td></tr>
<tr><td>Not specified</td><td>True, false, or not specified</td><td>Trading Networks.</td></tr>
</table>
</td></tr>
</table>

Property	Definition		
	Specified	False or not specified	The file system. If there is an error saving to the file system, the payload is saved in Trading Networks.
	Not specified	Not specified	Trading Networks.

alwaysUseUserProcessPayloadService	Determines if the userProcessPayload service should always be used irrespective of content type.
	■ `True`—userProcessPayload service is used irrespective of content type.
	■ `False`—userProcessPayload service is used only if the content types of the inbound payloads are *not* any of the following:
	  ■ application/edi-X12
	  ■ application/EDIFACT
	  ■ application/XML
	  ■ application/edi-consent
	■ This is the default.		

receiverURLs	A comma-separated list of URLs. The default is /invoke/wm.EDIINT/receive. **Note:** The new URLs must be appended to the default URL, for example, wm.EDIINT.receiverURLs=/invoke/wm.EDIINT/receive,/invoke/EDIINT_AS2/done_Service		

Trading Networks Objects Provided for EDIINT

When you install Module for EDIINT, Trading Networks objects (that is, TN document types, document attributes, extended profile fields, and processing rules) are installed in Trading Networks for you. This section describes the Trading Networks objects provided with Module for EDIINT.

The information in this section about the Trading Networks objects is for reference only. You should not alter the definitions of any of the Trading Networks objects.

TN Document Types

The following table describes the TN document types provided for EDIINT processing and the document attributes associated with each. For more information about the document attributes, see "Document Attributes" on page 56.

TN document type name	Description	Associated document attributes
EDIINT	Trading Networks matches all EDIINT documents to this TN document type.	■ EDIINT Message Type ■ EDIINT Message ID ■ EDIINT Message Digest ■ EDIINT Delivery URL
EDIINT MDN	Trading Networks matches all EDIINT MDNs to this TN document type.	■ EDIINT Message Type ■ EDIINT Message ID ■ EDIINT Delivery URL ■ EDIINT MDN Original Message ID ■ EDIINT MDN Received MIC ■ EDIINT MDN Disposition

Document Attributes

The following table describes the document attributes provided for EDIINT processing. The TN document types described in "TN Document Types" on page 56 extract these attributes from the EDIINT documents and MDNs.

This attribute name...	Represents...	And is extracted from...
EDIINT Message Type	The protocol that the EDIINT document uses: AS1, AS2, or AS3.	■ EDIINT documents ■ EDIINT MDNs
EDIINT Message ID	The value of the EDIINT Message-ID header, which is also used for the value of the Trading Networks Document ID system attribute.	■ EDIINT documents ■ EDIINT MDNs
EDIINT Message Digest	The message digest calculated for the EDIINT document.	■ EDIINT documents

This attribute name...	Represents...	And is extracted from...
EDIINT Delivery URL	The destination URL or IP address from the EDIINT document.	■ EDIINT documents ■ EDIINT MDNs
EDIINT MDN Original Message ID	The value of the EDIINT Message-ID header from the original EDIINT document for which the MDN is a receipt.	■ EDIINT MDNs
EDIINT MDN Received MIC	The message digest calculated for the EDIINT MDN.	■ EDIINT MDNs
EDIINT MDN Disposition	The results from processing the original EDIINT document for which the MDN is a receipt.	■ EDIINT MDNs

Extended Fields

When you install Module for EDIINT, the **EDIINT** field group is added to the Trading Networks profiles. The **EDIINT** field group contains extended profile fields used for EDIINT processing. For more information about these fields, see "Extended Fields Tab of the Profile" on page 36.

Processing Rules

Module for EDIINT provides processing rules for EDIINT *transport-level* processing. If you want to do *business-level* processing on the payload of the EDIINT document, you need to:

- Configure Module for EDIINT to submit the payload to Trading Networks. For instructions, see "Configuring Whether Trading Networks Is to Process Payloads" on page 49.
- Create your own processing rules to process the payloads. For information about creating processing rules, see the *webMethods Trading Networks Administrator's Guide* for your release.

The following table describes the processing rules provided for EDIINT transport-level processing. For more information about the corresponding services for each processing rule, see "Services Invoked by Processing Rules" on page 58.

Processing Rule Name	Description
EDIINT Process Message	Trading Networks invokes this processing rule for inbound EDIINT documents. This processing rule invokes the wm.EDIINT.rules:processMsg service to process the inbound EDIINT document.
EDIINT Process MDN Message	Trading Networks invokes this processing rule for inbound EDIINT MDNs. This processing rule invokes the

Processing Rule Name	Description
	wm.EDIINT.rules:processMDN service to process an inbound MDN message.
EDIINT Send Message	Trading Networks invokes this processing rule for outbound EDIINT documents. This processing rule invokes the wm.EDIINT.rules:sendMsg service to initiate the sending of an outbound EDIINT document.
EDIINT Send MDN Message	Trading Networks invokes this processing rule for outbound EDIINT MDNs. This processing rule invokes the wm.EDIINT.rules:sendMDN service to initiate the sending of an outbound EDIINT MDN message.
EDIINT Process Message - Persist in File system	Trading Networks invokes this processing rule for inbound EDIINT documents that are to be persisted in the file system. By default, inbound EDIINT documents are persisted in the Trading Networks database. This processing rule invokes the wm.EDIINT.rules:processMsg_persistPayload service to process an EDIINT document and persist it in the file system that the user configured.

Important:
With the exception of providing the value for *payloadDir* in the **EDIINT Process Message - Persist in File system** processing rule, you should not modify or customize these processing rules in any way.

Services Invoked by Processing Rules

The following table describes the services that are invoked by the processing rules described in "Processing Rules" on page 57. All services are located in the wm.EDIINT.rules folder except for deliveryDocument, which is located in the wm.EDIINT.delivery folder.

Service	Description	Processing Rule that Invokes It
processMsg	Processes an inbound EDIINT document.	EDIINT Process Message
processMsg_persistPayload	Processes an inbound EDIINT document and stores it in the file system that the user configured.	EDIINT Process Message - Persist in File system
processPayload	Processes the payload of an EDIINT document. If you configure Module for EDIINT to process payloads, the wm.EDIINT.rules:processMsg service invokes this service to submit the payload to	EDIINT Process Message

Service	Description	Processing Rule that Invokes It
	Trading Networks. For more information about configuring the module to process payloads, see "Configuring Whether Trading Networks Is to Process Payloads" on page 49.	
processMDN	Processes an inbound EDIINT MDN.	EDIINT Process MDN Message
sendMsg	Initiates the sending of an outbound EDIINT document.	EDIINT Send Message
sendMDN	Initiates the sending of an outbound EDIINT MDN document.	EDIINT Send MDN Message
deliveryDocument	Sends an outbound EDIINT document or MDN. The wm.EDIINT.rules:sendMDN and the wm.EDIINT.rules:sendMsg services invoke this service.	EDIINT Send Message -AND- EDIINT Send MDN Message

Restarting AS2 Message Transmission

When the optional AS2 Restart feature is enabled and transmission of an AS2 message is interrupted, the sender is able to restart the transmission from the point at which it failed, instead of starting the entire transmission over again. This is advantageous when large files are transmitted because the possibility of failure due to such occurrences as connection timeouts or loss of network connectivity is greater.

There are two ways to enable AS2 Restart when Module for EDIINT is in the sender role:

- Using the *enableAS2Restart* input parameter of the wm.EDIINT:send service. For more information, see "Invoking the Send Service to Restart Message Transmission" on page 60.
- Configuring the AS2RestartEnabled EDIINT property. For more information, see "Using EDIINT Properties to Restart Message Transmission" on page 60.

Note:
The *enableAS2Restart* input parameter of the wm.EDIINT:send service overrides the global AS2RestartEnabled EDIINT property.

When Module for EDIINT is the receiver, the module temporarily saves restart-enabled messages to its file system in the location defined in the AS2RestartTempFilePath property. As a message comes in, the module continues to append this temporary file and returns to the sender a Content-length header containing the number of bytes that have already been received, as an HTTP HEAD response. If message transmission fails, the sender restarts transmission from the point specified in the Content-length header.

Once the entire payload is received, the module begins processing it.

You can purge the payloads saved in the location specified in the AS2RestartTempFilePath property at any time by invoking the wm.EDIINT:purgeFiles service. For more information, see "wm.EDIINT:purgeFiles" on page 88.

Invoking the Send Service to Restart Message Transmission

Using the wm.EDIINT:send service to restart AS2 message transmission requires setting the *enableAS2Restart* input parameter to true. The following table explains how AS2 message transmission is restarted when the send service is invoked.

Step	Description
1	For every message that is restart-enabled, Module for EDIINT adds to the message headers an ETag header and to the bizdoc an ETag attribute, both of which contain a unique transfer ID, the ETag ID.
2	The module sends the receiver an HTTP HEAD request for the number of bytes that have already been transmitted.
3	The receiver (either another Module for EDIINT or a similar EDIINT product) responds to the HEAD request with a Content-length header specifying the number of bytes that have already been received for the particular ETag ID.
4	Module for EDIINT uses the value in the Content-length header to determine at what point to resume transmission.

For more information about defining send service parameters, see "wm.EDIINT:send" on page 91.

Using EDIINT Properties to Restart Message Transmission

Using EDIINT properties to control restarting message transmission requires setting the AS2RestartEnabled property to true. When Module for EDIINT is the sender and message transmission fails, the module attempts to restart it. When transmission for a single message continues to fail, the module attempts to restart the message transfer every so many seconds, as defined in the AS2RestartRetryIntervalInSeconds property, until the message is sent successfully or the number of retry attempts defined in the AS2RestartRetryCount property is reached.

When the number of attempts is exhausted before transmission is complete, the sender can restart the transmission manually. For more information, see "Restarting AS2 Message Transmission Manually" on page 60.

For more information about configuring EDIINT properties to enable AS2 Restart, see "Configuring EDIINT Properties" on page 52.

Restarting AS2 Message Transmission Manually

When Module for EDIINT is configured to restart AS2 message transmission automatically, you can optionally restart AS2 message transmission manually.

Note:
You can determine which transactions are restart-enabled by performing a search of the EDIINT ETag attribute for the value "is not blank". For more information about performing searches, see *Working with My webMethods*.

To restart AS2 message transmission manually

1. In My webMethods, go to **Monitoring > Integration > B2B > Transactions**.
2. Select the transaction that you want to restart.
3. In the Transactions Details panel, click **Resume AS2 Transfer**.

4 Creating a Client to Submit a Document Using EDIINT

Overview

The EDIINT standard requires the documents you send using the EDIINT transport be "packaged" in a specific way. If your client is running on the Integration Server that has webMethods Module for EDIINT installed, the client should invoke the wm.EDIINT:send service to package the document correctly and send it. If you are not using webMethods software for the client, see the documentation for the EDIINT software you are using to determine how to correctly package and send documents. This chapter describes how to create a client using the wm.EDIINT:send service of Module for EDIINT.

Note:
You can use the wm.EDIINT:send service to send both EDI documents and non-EDI documents.

Content Types to Use

You can use any of the EDIINT content types listed below. These content types are for both EDIINT documents and MDNs.

- application/edi-consent
- application/edi-x12
- application/edifact
- application/pkcs7-mime
- application/pkcs7-signature
- application/xml
- message/disposition-notification
- multipart/related
- multipart/report
- multipart/signed

Setting the Input Variables for the wm.EDIINT:send Service

The client should invoke the wm.EDIINT:send service to correctly package a document for EDIINT transport. For the list of input variables that the client should set for the wm.EDIINT:send service, see “ webMethods Module for EDIINT Services” on page 87.

5 Processing Inbound EDIINT Documents and MDNs

Processing Inbound EDIINT Documents

Module for EDIINT provides the wM.EDIINT:receive service as the entry point for inbound EDIINT documents. That is, clients that send EDIINT documents must invoke the wM.EDIINT:receive service. For more about creating a client that sends EDIINT documents, see "Creating a Client to Submit a Document Using EDIINT" on page 63.

For EDIINT transport-level processing, you do not need to create or customize services. the module provides all the logic needed to perform transport-level processing. If you want to perform business-level processing on the payload of the EDIINT document, you will need to add your own logic.

Before You Can Process Inbound EDIINT Documents

- Define a profile for the sender and receiver of the EDIINT document. For instructions about how to create profiles, see the *webMethods Trading Networks Administrator's Guide* for your release. For information about adding EDIINT information to profiles, see "Including EDIINT Information in Profiles" on page 32.
- If you want to process the payloads of the EDIINT documents, configure Module for EDIINT to submit the payloads to Trading Networks. For instructions, see "Configuring Whether Trading Networks Is to Process Payloads" on page 49.

Example of an EDIINT Document Posted by HTTP

The following is an example of an EDIINT document posted by HTTP:

```
Content-type: multipart/signed; micalg=SHA-1; protocol="application/pkcs7-
signature";  boundary="----=_Part_0_409673203.1011470256738"
Disposition-notification-to:
http://Administrator:manage@localhost:5555/invoke/wm.EDIINT/receive
Disposition-notification-options: signed-receipt-protocol=optional, pkcs7-
signature; signed-receipt-micalg=optional, SHA-1
AS2-From: 123456789
AS2-To: 987654321
Message-ID: <1687657971.1011470256928.JavaMail.zhouz@zhenzhou>
Content-Length: 2534
------=_Part_0_409673203.1011470256738
Content-Type: application/edi-x12
Content-Transfer-Encoding: binary
ISA*00*sssssssssss*00*rrrrrrrrrr*ZZ*123456789   *ZZ*987654321
*961007*2013*U*00200*000000001*0*T**
GS*PO*S1S1S1S1S1S1S*R1R1R1R1R1R1R1R*961007*2013*000000004*X*003050
ST*850*000040001
BEG*00*BE*2a*43324234v5523*961007*23tc4vy24v2h3vh3vh*ZZ*IEL*09*RE*09
...
SE*22*000040001
GE*1*000000004
IEA*1*000000001
```

```
------=_Part_0_409673203.1011470256738
Content-Type: application/pkcs7-signature; name=smime.p7s
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s
MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAMIAGCSqGSIb3DQEHAQAAMYIBuDCCAbQC
AQEwXjBZMQswCQYDVQQGEwJVUzEXMBUGA1UEChMOd2ViTWV0aG9kcyBJbmMxDzANBgNVBAsTBlBE
IEVESTEgMB4GA1UEAxMXRURJSU5UIHNhbXBsZSBTZW5kZXIgQ0ECAQEwCQYFKw4DAhoFAKCBsTAY
BgkqhkiG9w0BCQMxCwYJKoZIhvcNAQcBMBwGCSqGSIb3DQEJBTEPFw0wMjAxMTcyMDM4MDhaMCMG
CSqGSIb3DQEJBDEWBBQKbJZgrh/bit8BFmv1fuaWf40PjzBSBgkqhkiG9w0BCQ8xRTBDMAoGCCqG
SIb3DQMHMA4GCCqGSIb3DQMCAgIAgDANBggqhkiG9w0DAgIBQDANBggqhkiG9w0DAgIBKDAHBgUr
DgMCBzANBgkqhkiG9w0BAQEFAASBgKRrXO1tX3oFfLTgJwuoWKhygMQzdyNpX1Z4xU7kjDqYS8gs
yvaRSl0s7d4wA3N1CLGQUk87yRCFqoJPygrXyCI0kaGh1Ny61GxkPHuQ2cP54m11Wzgq9OGhaRba
TJu8HWB1ETFBML+wIBGunkRcR3s5mEpxINmflEYNZlxmf78ZAAAAAAAA
------=_Part_0_409673203.1011470256738-
```

Creating Outbound MDNs that Acknowledge Receipt of an EDIINT Document

For an inbound EDIINT document, Trading Networks executes the EDIINT Process Message processing rule, which in turn invokes the wm.EDIINT.rules:processMsg service. The wm.EDIINT.rules:processMsg service detects whether the sender of the EDIINT document requested an MDN.

If the sender requested an MDN, theprocessMsg service determines whether the sender requested a signed or an unsigned MDN. The processMsg service creates the appropriate type of MDN and submits the MDN back to Trading Networks for delivery to the sender.

To deliver the outbound EDIINT MDN, Trading Networks executes the EDIINT Send MDN Message processing rule, which in turn invokes the wm.EDIINT.rules:sendMDN service. The sendMDN service determines whether the sender requested a synchronous or asynchronous MDN.

- If the sender requested a synchronous MDN, the sendMDN service returns the MDN to the sender using the same HTTP connection.
- If the sender requested an asynchronous MDN, the sendMDN service invokes the wm.EDIINT.rules:deliveryDocument service to send the MDN as a separate transaction.

Setting Up to Process Payloads from EDIINT Documents

To perform business-level processing on the payloads of EDIINT documents

1. Configure Module for EDIINT to process payloads. For instructions, see"Configuring Whether Trading Networks Is to Process Payloads" on page 49.

2. Set the value of the SMIME Type extended field in the senders' profiles to specify the SMIME type you expect inbound EDIINT documents to use. The module uses the SMIME type you specify to determine whether to process the payload. For more information, see "How the SMIME Type Profile Field Affects Processing Payloads" on page 68. For more information about setting up profiles for EDIINT, see "Including EDIINT Information in Profiles" on page 32.

3. Set up Trading Networks to process the payload and perform the business-level logic.

   - If the payload is an EDI document , see the *webMethods Module for EDI Installation and User's Guide* for how to set up Module for EDI to process the EDI documents.
   - If the payload is not an EDI document , you must:
     - Define a TN document type for the payload. For instructions, see the *webMethods Trading Networks Administrator's Guide* for your release.
     - Define a processing rule for the payload document. For instructions, see the *webMethods Trading Networks Administrator's Guide* for your release.

How the SMIME Type Profile Field Affects Processing Payloads

Module for EDIINT determines the SMIME type used by the inbound EDIINT document, that is, whether the inbound document is plain, signed, encrypted, or signed and encrypted. The module then compares the SMIME type of the inbound document with the value of the SMIME Type extended profile field of the sender's profile. The table below describes the actions the module takes based on the outcome of the comparison.

Value of SMIME Type Extended Profile Field	SMIME Type of Inbound Document	Action Module for EDIINT Takes for the Inbound Document
plain	any value	Processes the payload.
signed	signed signed and encrypted	Processes the payload.
	plain encrypted	Logs an error message to the Trading Networks activity log and does not process the payload.
encrypted	encrypted signed and encrypted	Processes the payload.
	plain signed	Logs an error message to the Trading Networks activity log and does not process the payload.
signedAndEncrypted	signed and encrypted	Processes the payload.
	plain signed encrypted	Logs an error message to the Trading Networks activity log and does not process the payload.

Processing Inbound EDIINT MDNs

Module for EDIINT provides the wM.EDIINT:receive service as the entry point for inbound EDIINT MDNs. That is, clients that send EDIINT MDNs must invoke the wM.EDIINT:receive service. For more information about creating a client that sends EDIINT documents, see "Creating a Client to Submit a Document Using EDIINT" on page 63.

You do not need to create or customize services to process inbound EDIINT MDNs. The module provides all the logic needed to perform the processing.

Before You Can Process Inbound EDIINT MDNs

Define a profile for the sender and receiver of the EDIINT MDN. For instruction about how to create profiles, see the *webMethods Trading Networks Administrator's Guide* for your release. For information about adding EDIINT information to profiles, see "Including EDIINT Information in Profiles" on page 32.

Example of an EDIINT MDN Posted by HTTP

The following is an example of an EDIINT MDN posted by HTTP:

```
AS2-From: 987654321
AS2-To: 123456789
Message-ID: <2038921766.1012252564086.JavaMail.zhenzhou@zhenzhou>Content-Type:
multipart/signed; protocol="application/pkcs7-signature"; micalg=SHA-1;
boundary="----=_Part_20_-1967424986.1012252564076"
------=_Part_20_-1967424986.1012252564076Content-Type: multipart/report; Report-
Type=disposition-notification; boundary="----=_Part_19_568293921.1012252564056"
------=_Part_19_568293921.1012252564056Content-Type: text/plainContent-Transfer-
Encoding: 7bit
MDN for -
Message ID: <128678451.1012252560430.JavaMail.zhenzhou@zhenzhou>
From: 123456789
To: 987654321
Received on: 2002-01-28 at 16:16:04 (EST)
Status: processed
Comment: This is not a guarantee that the message has been completely processed
or understood by the receiving translator
------=_Part_19_568293921.1012252564056Content-Type: message/disposition-
notificationContent-Transfer-Encoding: 7bit
Reporting-UA: webMethods Integration ServerOriginal-Recipient: 987654321Final-
Recipient: 987654321Original-Message-ID:
<128678451.1012252560430.JavaMail.zhenzhou@zhenzhou>Received-content-MIC:
qZvJD2+2H/OAQYa3+uIZUIyNUaw=, SHA-1Disposition: automatic-action/MDN-sent-
automatically; processed
------=_Part_19_568293921.1012252564056--
------=_Part_20_-1967424986.1012252564076Content-Type: application/pkcs7-
signature; name=smime.p7sContent-Transfer-Encoding: base64Content-Disposition:
attachment; filename=smime.p7s
MIAGCSqGSIb3DQEHAqCAMIACAQExCzAJBgUrDgMCGgUAMIAGCSqGSIb3DQEHAQAAMYIBujCCAbYCAQE
wYDBbMQswCQYDVQQGEwJVUzEXMBUGA1UEChMOd2ViTWV0aG9kcyBJbmMxDzANBgNVBAsTBlBEIEVEST
EiMCAGA1UEAxMZRURJSU5UIHNhbXBsZSBSZWNlaXZlciBDQQIBATAJBgUrDgMCGgUAoIGxMBgGCSqGS
Ib3DQEJAzELBgkqhkiG9w0BBwEwHAYJKoZIhvcNAQkFMQ8XDTAyMDEyODIxMTYwNFowIwYJKoZIhvcN
AQkEMRYEFP0/GE3KNoRkF6KUtnqDOm40bUxEMFIGCSqGSIb3DQEJDzFFMEMwCgYIKoZIhvcNAwcwDgY
```

```
IKoZIhvcNAwICAgCAMA0GCCqGSIb3DQMCAgFAMA0GCCqGSIb3DQMCAgEoMAcGBSsOAwIHMA0GCSqGSI
b3DQEBAQUABIGAJBb3whwo+h0PsmEyPMXQHIpjFS5fa5w8PIipHQ9nfJVoTTbp5VTL4zT1E34vjESok
tGBYmYnD+gTTe2aEB3PoIqCym25Lv2MZuvcSVNoa2hS4hrCnDwmYNqbFySlV2ZAqodgBElztd71eeIg
nXLU1/R65gFOJw72Wto0xi8Q930AAAAAAAA=
------=_Part_20_-1967424986.1012252564076--
```

6 Using EDIINT to Deliver Outbound Documents

Before You Can Deliver Outbound EDIINT Documents

Define a profile for the sender and receiver of the EDIINT document. For instructions about how to create profiles, see the *webMethods Trading Networks Administrator's Guide* for your release. For information about adding EDIINT information to profiles, see "Including EDIINT Information in Profiles" on page 32.

Setting the SMIME Type of the Outbound EDIINT Document

You can define the SMIME type that you want webMethods Module for EDIINT (Module for EDIINT) to use for an outbound EDIINT document; that is, whether you want to send the outbound EDIINT message:

- Without signing or encrypting (plain)
- Signing only (signed)
- Encrypting only (encrypt)
- Signing and encrypting (signedAndEncrypted)

You specify the SMIME type that you want the module to use by using one of the following:

- The *Type* input parameter to the `wm.EDIINT:send` service
- The SMIME Type extended profile field, and setting the *Type* input variable to the wm.EDIINT:send service to `getFromProfile`. For more information about setting the SMIME Type extended profile field, see "Extended Fields Tab of the Profile" on page 36.

Using the wm.EDIINT:send Service to Send EDIINT Documents

Module for EDIINT provides the wm.EDIINT:send service to send EDIINT documents. This service performs all the necessary EDIINT transport-level processing.

To use the wm.EDIINT:send service:

- Set the wm.EDIINT:send service input variables as necessary. You must create a java.io.InputStream from the EDI or XML data and pass it to the input variable *data/stream*. For a description of key input variables, see "Setting the Input Variables for the wm.EDIINT:send Service" on page 64.

  Also, you must create a java.io.InputStream for each attachment you send with an EDIINT message.

  For a complete description of this service, see " webMethods Module for EDIINT Services" on page 87.

- Invoke the wm.EDIINT:send service from a service that you create.

Important:
Do not invoke the wm.EDIINT:send service directly from the Trading Networks **Execute a Service** processing action. Doing so will produce processing errors. The service that you create to invoke the wm.EDIINT:send service can be invoked directly from the **Execute a Service** processing action.

7 Viewing and Managing Information About EDIINT Documents and MDNs

Viewing Information about EDIINT Documents and MDNs

Because EDIINT documents and MDNs are processed through Trading Networks, Module for EDIINT takes advantage of Trading Networks features. For example, you can use the Trading Networks system attribute, **User Status**, to assign a user-defined status to a document. The module assigns statuses to EDIINT documents and MDNs as it processes the document. For more information about viewing the user status associated with a document, see "Viewing the User Status Associated with a Document" on page 76.

If you submit the payload of an inbound EDIINT document to Trading Networks for business-level processing, you can view information about the envelope(s) that the payload contains. For more information, see "Viewing Related Documents" on page 78.

Viewing the User Status Associated with a Document

You can view the **User Status** associated with a document on the Transactions page of My webMethods. For instructions on how to view information about documents, see the *webMethods Trading Networks User's Guide* for your release.

The following table describes the values of the **User Status** system attribute for EDIINT transport-level processing.

Type	User Status	Description
Inbound EDIINT document	ProcessMsg	The EDIINT document has been received and processing is starting.
	ProcessMsg PAYLOAD	The EDIINT document was processed successfully. Module for EDIINT is configured to process payloads and the wm.EDIINT.rules:processPayload service has been invoked.
	ProcessMsg ERROR	One of the following: ■ The EDIINT document contained an invalid sender ID or receiver ID. ■ The message could not be decrypted or verified.
	ProcessMsg DONE	The EDIINT document was processed successfully, and Module for EDIINT was configured not to submit the payload to Trading Networks.

Type	User Status	Description
Inbound EDIINT MDN	ProcessMDNMsg	The EDIINT MDN has been received and processing is starting.
	ProcessMDNMsg DONE	The MDN was processed.
	ProcessMDNMsg ERROR	One of the following: ■ The MDN contained an invalid sender ID or receiver ID. ■ The signature of the MDN could not be verified. ■ The MDN contained errors. ■ The MDN digest did not match that of the original EDIINT document.
	ProcessMDNMsg IGNORED	An identical MDN was previously received.
	ProcessMDNMsg DONE/NAK	The message digest of the inbound document does not match that of the outbound document.
Outbound EDIINT document	SendMsg	Processing to send the EDIINT document has started.
	SendMsg DONE	One of the following: ■ The message was sent and an MDN was not requested. ■ The message was sent and an MDN was returned.
	SendMsg WAITMDN	The message was sent and an MDN was requested but not yet received.
	SendMsg ERROR	One of the following: ■ The message could not be sent. ■ The returned MDN contained errors.
Outbound EDIINT MDN	SendMDNMsg	Processing to send the EDIINT MDN has started.
	SendMDNMsg DONE	The MDN was sent successfully.
	SendMDNMsg ERROR	The MDN could not be sent.

Viewing Related Documents

If you submit the payload of an inbound EDIINT document to Trading Networks for business-level processing, you can view information about the envelope(s) that the payload contains.

By default, Trading Networks persists to the Trading Networks database any EDIINT document that is submitted to it. If the document, which is stored in the pipeline variable *bizdoc*, has not been persisted (or if the wm.EDIINT.rules:receive service fails to place the document into *bizdoc*), the following occurs, depending on the content type of the payload:

Payload Content Type	Action If the bizdoc Has *Not* Been Persisted
application/XML	No information about the payload is displayed.
application/edi-X12 application/EDIFACT application/edi-consent	Information about the main envelope is not displayed, but information about each individual envelope that has been persisted is displayed.
other	If you specified a user-defined service to handle another content type (as described in "Configuring Whether Trading Networks Is to Process Payloads" on page 49), Module for EDIINT passes the *bizdoc* to the user-defined service, along with the payload's data stream and content type. The user-defined service must handle the display of payload information.

For instructions about how to view related documents in My webMethods, see the *webMethods Trading Networks User's Guide* for your release.

Resubmitting EDIINT Outbound Transactions

You should resubmit an outbound transaction if the document was not recognized when it was originally received. You will need to create or modify a TN document type definition to recognize the document before processing the document again.

When you resubmit a document, Trading Networks performs the following processing:

- Uses the TN document type definitions to recognize the document
- Performs a processing rule lookup to determine the rule to use
- Performs the pre-processing and processing actions identified in the matching processing rule

For instructions about how to resubmit documents in My webMethods, see the *webMethods Trading Networks User's Guide* for your release.

8 Error Handling

Overview

This chapter describes the logging and exception handling in Module for EDIINT. For a list of error codes and supporting information, see Error Codes.

Message Logging

Module for EDIINT uses Integration Server's logging mechanism to log informational, warning, and error messages to the server log of Integration Server. To view the server log, use the Integration Server Administrator. For information about viewing and configuring the server log, see the *webMethods Integration Server Administrator's Guide* for your release.

The module adds errors, warnings, and informational messages to the server log, using the format, EDIINT.00000n.nnnnnnnE, where:

- EDIINT is the product code that indicates the message is issued by the module.
- 00000*n* is the major error code, where *n* can be any of the following values:

Value	Category	Explanation
1	General	Contains generic messages
2	Document	Contains messages pertaining to EDIINT documents
3	Validation	Contains messages with validation errors
4	Transport	Contains messages pertaining to the transportation of EDIINT documents
5	Other	Contains other messages
6	Compression	Contains messages pertaining to compression

- *nnnnnnn*E is the minor error code.

Error Codes

EDIINT.000001.000002E Invalid input for EDIINT large document size {0}.

Explanation: Number format error while handling large EDI documents. An incorrect value is set for the tn.BigDocThreshold property field.

Action: Ensure that the value set in the tn.BigDocThreshold property field contains only integers.

EDIINT.000001.000003E Invalid input - {0}.

Explanation: Generic error during validation of user inputs.

Action: Ensure that the inputs provided are valid and meet the specifications for the data type. For example, in the case of String, ensure that the input is not null or blank.

EDIINT.000001.000004E Invalid URI - {0}, missing ":".

Explanation: The URI format does not meet URI specifications.

Action: Ensure that the URI format meets specifications. For example, localhost:80 is incorrect. http://localhost:80 is correct.

EDIINT.000001.000005E Unrecognized input object - {0} {1}.

Explanation: The module encountered an incorrect data type.

Action: Ensure that inputs are one of the following data types: Byte Array, String, or Stream.

EDIINT.000001.000006E messageDigestHolder is empty.

Explanation: Message digest is missing from the messageDigestHolder collection.

Action: Ensure that message digest is part of the messageDigestHolder collection.

EDIINT.000001.000008E Invalid AS2 Identifier: [{0}].

Explanation: The AS2 identifier does not contain valid ASCII characters.

Action: Ensure that all characters in the AS2 identifier are part of the ASCII character set.

EDIINT.000001.000012E Unsupported protocol - {0}.

Explanation: The transport protocol used is not supported.

Action: Ensure that the transport protocol is one of the following: HTTP, FTP, mailto, file.

EDIINT.000002.000003E Cannot register EDIINT Attribute - {0}.

Explanation: Module for EDIINT is unable to register the specified attribute with Trading Networks.

Action: Ensure that Trading Networks is installed and running and the Trading Networks pool is created correctly.

EDIINT.000002.000004E Cannot register EDIINT MDN Attribute - {0}.

Explanation: Module for EDIINT is unable to register the EDIINT MDN Attribute with Trading Networks.

Action: Ensure that Trading Networks is installed and running and the Trading Networks pool is created correctly.

EDIINT.000002.000012E Error in creating / updating EDIINT MDN BizDocEnvelope - {0}.

Explanation: EDIINT BizDocEnvelope could not be created or updated.

Action: Ensure that Trading Networks is installed and running and the Trading Networks pool is created correctly.

EDIINT.000002.000013E Cannot register EDIINT services - {0}.

Explanation: The module is unable to register services with Trading Networks.

Action: Ensure that Trading Networks is installed and running and the Trading Networks pool is created correctly.

EDIINT.000002.000014E Cannot register EDIINT Processing rules - {0}.

Explanation: The module is unable to register processing rules with Trading Networks.

Action: Ensure that Trading Networks is installed and running and the Trading Networks pool is created correctly.

EDIINT.000002.000020E Cannot get partner's internal ID for {0}, {1}.

Explanation: The module is unable to retrieve the partner profile from Trading Networks.

Action: Ensure that Trading Networks is installed and running and the Trading Networks pool and the trading partner profiles are created correctly.

EDIINT.000002.000024E Missing Message-ID.

Explanation: The message-ID is not present in the EDIINT message.

Action: See the Trading Networks activity logs of the corresponding error message bizdoc for details.

EDIINT.000002.000025E Document type name must be EDIINT and user status must start with SendMsg:.

Explanation: The document type name is not EDIINT or the user status does not begin with SendMsg:.

Action: Ensure that the document type name is EDIINT and the user status begins with SendMsg:.

EDIINT.000002.000026E Cannot relate to submitted payload.

Explanation: The module is unable to relate the payload to the corresponding bizdoc.

Action: See the Trading Networks transaction analysis logs to ensure the EDIINT messages are being saved correctly in Trading Networks.

EDIINT.000002.000030E Cannot relate to submitted individual EDI envelope.

Explanation: The module is unable to relate the bizdoc with envelopeDocuments in the input pipeline.

Action: Ensure that the pipeline is populated with envelopeDocuments.

EDIINT.000002.000031E Cannot retrieve EDI document summary information - envelopeDocuments.

Explanation: The module is unable to locate envelopeDocuments and relate it to the bizdoc in the input pipeline.

Action: Ensure that envelopeDocuments is populated in the pipeline.

EDIINT.000002.000032E Cannot relate the submitted envelope documents at index [{0}] to orignal bizdoc.

Explanation: The module is unable to locate the envelopeDocument document ID and relate it to the bizdoc in the input pipeline.

Action: Ensure that document IDs are populated for all the envelopeDocuments in the pipeline.

EDIINT.000002.000033E Cannot retrieve docId from envelopeDocuments.

Explanation: The module is unable to locate the envelopeDocument document ID and relate it to the current bizdoc in the input pipeline.

Action: Ensure that document IDs are populated for all envelopeDocuments in the pipeline.

EDIINT.000002.000034E Cannot retrieve EDI BizDocEnvelope {0}.

Explanation: The module is unable to locate the envelope document in Trading Networks for the input document ID.

Action: Ensure that the documents in Trading Networks are populated with correct document IDs.

EDIINT.000002.000037E Cannot delete file from User Outbox folder {0}.

Explanation: The module is unable to delete the file from the User Outbox folder.

Action: Ensure the correct access permissions to the FTP root directory used by the Integration Server are set for the user.

EDIINT.000002.000038E Cannot delete the file {0} as file name does not start with {1}.

Explanation: The module is unable to delete the file from the User Outbox folder because the file is named incorrectly.

Action: Rename the file and delete again.

EDIINT.000002.000062E Cannot create EDIINT ID Types - {0}.

Explanation: The module is unable to create EDIINT ID types in the Trading Networks database.

Action: Ensure that Trading Networks is installed and running and the Trading Networks pool is created correctly.

EDIINT.000004.000006E Invalid input entered for getContentLength - {0}.

Explanation: The input entered for getContentLength is invalid.

Action: Set getContentLength to either "true" or "false."

EDIINT.000005.000002E Error occurred while loading the EDIINT properties file - {0}.

Explanation: The module is unable to load the properties.cnf file.

Action: Ensure that the property.cnf file is present within the configuration folder of the WmEDIINT package and that the correct access permissions are set for the property.cnf file.

EDIINT.000005.000003E EDIINT properties file - '{0}' does not exist.

Explanation: The module is unable to load the properties.cnf file.

Action: Ensure that the properties.cnf file is present within the configuration folder of the WmEDIINT package and the correct access permissions are set for the properties.cnf file.

EDIINT.000005.000005E Error occurred while saving the EDIINT properties file - {0}.

Explanation: The module is unable to save the properties.cnf file.

Action: Ensure the correct access permissions are set for the configuration folder of the WmEDIINT package.

EDIINT.000005.000007E The WmEDIINT property file {0} does not exist.

Explanation: The module is unable to find the properties.cnf file in the configuration directory of the WmEDIINT package.

Action: Ensure that the properties.cnf is present in the configuration directory of the WmEDIINT package.

EDIINT.000005.000008E The WmEDIINT property file {0} is read only.

Explanation: The module does not have write permission for the properties.cnf file.

Action: Give the module write permission for the properties.cnf file.

EDIINT.000005.000009E Access denied, as User "{0}" does not have the privilege to view {1}.

Explanation: User does not have access permission to view the specified properties.

Action: Set the access permissions to allow the user to view the properties.

EDIINT.000005.000010E Access denied, as User "{0}" does not have the privilege to edit {1}.

Explanation: User does not have access permission to edit the specified properties.

Action: Set the access permissions to allow the user to edit the specified properties.

EDIINT.000006.000001E Cannot handle datasource of type: {0}.

Explanation: mimeSrc is invalid.

Action: Ensure the MIME data source is either of these two types:1) com.wm.app.tn.mime.MimeData or 2) java.util.InputStream

EDIINT.000006.000002E Datasource is null.

Explanation: mimeSrc is null.

Action: Input the correct MIME data source to the service.

EDIINT.000006.000003E Cannot parse the message.

Explanation: The module is unable to parse the input stream.

Action: Ensure the input stream is entered correctly.

EDIINT.000006.000004E Cannot process this object. Write this MimeData to stream and then process that stream.

Explanation: The module is unable to parse the MIME object.

Action: See the server logs for details. Try writing the mime data to stream and then process the stream.

EDIINT.000006.000005E Input parameter mimeSrc is not of type MimeData.

Explanation: The input for MimeSrc is not of type MimeData.

Action: Ensure the input for MimeSrc is of type MimeData.

EDIINT.000006.000007E Cannot process message that is not javax.mail.internet.MimeMessage or javax.mail.internet.MimeBodyPart. Current message is: {0}

Explanation: The MIME part of the message is not of type javax.mail.internet.MimeMessage or javax.mail.internet.MimeBodyPart.

Action: Ensure the input of the MIME part of the message is either of these two types: 1) javax.mail.internet.MimeMessage or 2) javax.mail.internet.MimeBodyPart.

EDIINT.000004.000012E Internal Id is null

Explanation: The internal ID in the input is either null or empty.

Action: Ensure the correct value for the internalId input parameter is provided.

EDIINT.000004.000011E Content-Range {0} is not in the expected format

Explanation: The Content-Range is not in the expected format so it cannot be processed.

Action: Ensure that the correct Content-Length is passed so the correct Content-Range can be generated.

EDIINT.000004.000013E Access denied to do this operation

Explanation: User does not have privileges to perform this operation.

Action: Contact the Administrator to obtain privilege.

EDIINT.000004.000014E Exception in portal API {0}

Explanation: There is an exception in executing the portal API.

Action: Ensure that the input parameters are set correctly.

A webMethods Module for EDIINT Services

Overview

Use the services in the wm.EDIINT folder to send and receive EDIINT (AS1, AS2, or AS3) messages and MDNs.

Summary of Services in this Folder

The following services are available in the public folder:

Service	Description
wm.EDIINT:purgeFiles	Purges the temporary files used for AS2 restart transactions when they are no longer needed.
wm.EDIINT:receive	Receives inbound EDIINT (AS1/AS2/AS3) messages or MDNs and submits the message to Trading Networks to be unwrapped and decrypted, and to have its signature authenticated.
wm.EDIINT:restart	Restarts transmission of an EDIINT AS2 message from the point at which the transmission failed.
wm.EDIINT:retrieveAS3Message	Downloads EDIINT AS3 messages or MDNs from a partner's remote FTP server and submits the message to Trading Networks to be unwrapped and decrypted, and to have its signature authenticated.
wm.EDIINT:send	Constructs an outbound EDIINT message according to the configuration of the input parameters, and then submits the message to Trading Networks.

wm.EDIINT:purgeFiles

Purges the temporary files used for AS2 restart transactions when they are no longer needed.

Input Parameters

daysOld **String** The age of the files to get purged (in days). All files older than the value specified in *daysOld* are purged on invoking the purgeFiles service. The day a file is sent is considered day zero.

For example, when the value of *daysOld* is 2, the service purges all temporary files sent more than two days ago.

Valid values are integers greater than or equal to zero.

Output Parameters

filesDeleted **String** The number of files purged.

wm.EDIINT:receive

Receives inbound EDIINT (AS1/AS2/AS3) messages or MDNs and submits the message to Trading Networks to be unwrapped and decrypted, and to have its signature authenticated.

The EDIINT content handler populates the parameters described below.

Input Parameters

protocol **String** The EDIINT protocol to use.

Value	Meaning
`smtp`	EDIINT AS1 message or MDN.
`http`	EDIINT AS2 message or MDN.
`ftp`	EDIINT AS3 message or MDN.

message-ID **String** The EDIINT message ID of the EDIINT message or MDN. This value becomes the Trading Networks system attribute, **Document ID**.

contentType **String** The content type of the EDIINT message or MDN.

stream **Object** The data InputStream representing the inbound EDIINT message or MDN.

AS2-From **String** The sender ID from the EDIINT AS2 message/MDN. This should match the sender's Trading Networks external ID.

AS2-To	**String** The receiver ID from the EDIINT AS2 message/MDN. This should match the receiver's Trading Networks external ID.
AS3-From	**String** The sender ID from the EDIINT AS3 message/MDN. This should match the sender's Trading Networks external ID.
AS3-To	**String** The receiver ID from the EDIINT AS3 message/MDN. This should match the receiver's Trading Networks external ID.
From	**String** The sender ID from the EDIINT AS1 message/MDN. This should match the sender's Trading Networks external ID.
To	**String** The receiver ID from the EDIINT AS1 message/MDN. This should match the receiver's Trading Networks external ID.
ReceiptDeliveryOption	**String** The address to which to send an asynchronous MDN, if requested.
Etag	**String** Optional. The unique ID of the Etag header that identifies the transaction to be restarted after AS2 message transmission fails.

Output Parameters

bizdoc	**Document** The resulting document formatted as an IS document (IData object). For the structure of a business document, see *webMethods Trading Networks Built-In Services Reference*.

Usage Notes

For information about how to use this service, see "Viewing and Managing Information About EDIINT Documents and MDNs" on page 75.

wm.EDIINT:restart

Restarts transmission of an EDIINT AS2 message from the point at which the transmission failed.

Input Parameters

internalID	**String** The internal ID of the document that failed to be sent completely due to a network interruption or other failure.

Output Parameters

None.

wm.EDIINT:retrieveAS3Message

Downloads EDIINT AS3 messages or MDNs from a partner's remote FTP server and submits the message to Trading Networks to be unwrapped and decrypted, and to have its signature authenticated.

The service locates the partner's remote FTP server using the values defined in IS document type wm.EDIINT.TPA:EDIINTAS3TPA. If the IS document type specifies that the retrieved file(s) are to be deleted, the service deletes the file(s).

Input Parameters

sender **Document** Optional. Identification of the partner from whom to retrieve the message.

Note:
When *sender* is not specified, the service will access the remote FTP servers of all partners with whom you have a Trading Partner Agreement.

Parameter	Description
id	**String** The partner's external ID.
idTypeDesc	**String** Optional. The partner's external ID type. Default: AS3.

receiver **Document** Identification of the partner who retrieves the message.

Parameter	Description
id	**String** The partner's external ID.
idTypeDesc	**String** Optional. The partner's external ID type. Default: AS3.

Output Parameters

None.

wm.EDIINT:send

Constructs an outbound EDIINT message according to the configuration of the input parameters, and then submits the message to Trading Networks.

Input Parameters

type

String The SMIME type that you want to use for the outbound EDIINT message. Specify one of the following:

Value	Meaning
`plain`	Neither sign nor encrypt the outbound EDIINT message.
`signed`	Sign the outbound EDIINT message.
`encrypted`	Encrypt the outbound EDIINT message.
`signedAndEncrypted`	Sign and encrypt the outbound EDIINT message.
`getFromProfile`	Whether to sign and/or encrypt the outbound EDIINT message using the value of the SMIME Type extended field from the receiver's profile.

compressed

String (optional) Whether the EDIINT message that you are sending is compressed before it is signed and/or encrypted. Specify one of the following:

Value	Meaning
`true`	Compress the outbound message before signing and encrypting.
`false`	Do not compress the outbound message before signing and encrypting. This is the default.
`getFromProfile`	Whether to compress the outbound message using the value of the Compression extended field from the receiver's profile.

deliveryMethod

String The delivery method you want to use to send the EDIINT document. The wm.EDIINT:send service obtains the Trading Networks profile for the receiver (specified by the *receiverID* input parameter) and delivers the EDIINT document to the receiver's system using the information specified for the delivery method. Specify a delivery method that is defined in the receiver's Trading Networks profile.

- **For EDIINT AS1**, specify one of the following:
  - `PrimarySMTP` (corresponds to the Trading Networks **Primary E-mail** delivery method)
  - `SecondarySMTP` (corresponds to the Trading Networks **Secondary E-mail** delivery method)

- **For EDIINT AS2**, specify one of the following:
  - `PrimaryHTTP`
  - `SecondaryHTTP`
  - `PrimaryHTTPS`
  - `SecondaryHTTPS`
- **For EDIINT AS3**, specify `AS3`.

You can also specify `getFromProfile` if you want the service to obtain the delivery method from the Delivery Method extended field from the receiver's profile.

data

Document The payload that you want to send.

Parameter	Description
contentType	The content type to assign to the outbound message.
stream	The java.io.InputStream that you map from the EDI or XML data.
otherHeaders	(optional) The name and value of the header for outbound messages. This header information is saved for each inbound payload document in Trading Networks in the `payloadMimeHeaders` content part.

attachments

Document array Any attachments to a message.

Parameter	Description
stream	The java.io.InputStream for the attachment you want to add.
contentType	The content type of the attachment. For example, `application/zip` if the attachment is a zip file.
otherHeaders	(optional) The name and value of the headers you want to add to the attachments. This information will be saved in Trading Networks as a content type for both sender and receiver.

requestMDN

String Whether you want the receiver to return an MDN. Specify one of the following:

Value	Meaning
`none`	Do not request a return MDN.

`synchronousMDN`	Request a return synchronous MDN.
`asynchronousMDN`	Request a return asynchronous MDN.
`getFromProfile`	Whether to request a return MDN using the value of the Request MDN extended field from the receiver's profile and what type of MDN to return.

Note:
If you specify PrimarySMTP, SecondarySMTP, PrimaryFTPS, or SecondaryFTPS for *deliveryMethod*, you can only receive an asynchronous MDN.

requestSignedReceipt **String** Whether you want the MDN to be signed.

Note:*requestSignedReceipt* is ignored when *requestMDN* is none.

Value	Meaning
`true`	Request a signed MDN.
`false`	Request a plain (unsigned) MDN.
`getFromProfile`	Whether the MDN is signed using the value of the Request Signed Receipt extended field from the receiver's profile.

senderID **Document** Identification of the sender of the EDIINT message.

Parameter	Description
id	The sender's external ID. That is, the identification that you want for the sender in the message.
idTypeDesc	Optional. The external ID type for the sender ID you specified in *id*. This is an external ID type as defined in Trading Networks. **Note:** By default, the service uses the appropriate external ID type based on the value you specify for the *deliveryMethod* parameter. For example, if you specify AS3 for the *deliveryMethod* parameter, the service uses the EDIINT AS3 external ID type. Specify a value for *idTypeDesc* only if you want to override this default.

receiverID **Document** Identification of the receiver of the EDIINT message.

Parameter	Description
id	The receiver's external ID. That is, the identification that you want for the receiver in the message.
idTypeDesc	Optional. The external ID type for the receiver ID you specified in *id*. This is an external ID type as defined in Trading Networks. **Note:** By default, the service uses the appropriate external ID type based on the value you specify for the *deliveryMethod* parameter. For example, if you specify AS3 for the *deliveryMethod* parameter, the service uses the EDIINT AS3 external ID type. Specify a value for *idTypeDesc* only if you want to override this default.

ConversationID **String** (optional) Conversation ID for the outbound EDIINT message.

The *conversationID* parameter is an identifier that links all documents that are part of the same business process (also called a conversation). That is, all documents in the same business process need to have the same *conversationID*. Trading Networks can extract *conversationID*s from EDI documents and use them to pass documents to webMethods Process Engine after Trading Networks performs the actions identified by a processing rule. For more information, see the *webMethods Trading Networks Administrator's Guide* for your release.

In this field, you might want to specify the same conversation ID as that of the payload that you are sending. Module for EDIINT automatically assigns the same conversation ID that is assigned here to a return MDN.

Note:
Leave this field blank unless you own a license for webMethods Process Engine.

TNFlags **Document** Data to pass as input to the wm.tn.route:routeBizDoc service (for example, TN_parms).

Parameter	Description
prtIgnoreDocument	**String** Whether the wm.tn.route:routeBizDoc service invokes the pub.prt.tn:handleBizDoc service. Set this parameter to true if the WmPRT package is not installed.

Value	Meaning

`true`	wm.tn.route:routeBizDoc invokes the pub.prt.tn:handleBizDoc service.
`false`	wm.tn.route:routeBizDoc does not invoke the pub.prt.tn:handleBizDoc service.

enableAS2Restart **String** (optional) Whether transmission of an AS2 message restarts automatically if the transmission fails. Specify one of the following:

Value	Meaning
`true`	Restart message transmission if transmission fails.
`false`	Do not restart message transmission if transmission fails.

enableHTTP Chunking **String** Enables HTTP chunking.

Value	Meaning
`true`	HTTP chunking enabled.
`false`	HTTP chunking disabled.

customHeaders **Document** Custom headers you can include on your EDIINT message.

Parameter	Description
Name	**String** Specify a name for the custom header.
Value	**String** Specify the information to display in the custom header.

Output Parameters

None.

Usage Notes

For information about how to use this service, see "Creating a Client to Submit a Document Using EDIINT" on page 63 and "Viewing and Managing Information About EDIINT Documents and MDNs" on page 75.