webMethods®

webMethods ACH Module

Installation and User's Guide

VERSION 6.1

webMethods, Inc.
South Tower
3877 Fairfax Ridge Road
Fairfax, VA 22030
USA
703.460.2500
http://www.webmethods.com

Document ID: ESTD-ACH-IUG-61-20051010

Contents

About This Guide 7
Document Conventions 7
Additional Information 8

Chapter 1. Concepts 9
What is Automated Clearing House (ACH)? 10
ACH Files 11
What is the webMethods ACH Module? 11
Sending Electronic Payments 11
Receiving Electronic Payments 13
webMethods ACH Module Architecture 14
webMethods ACH Module Features 16

Chapter 2. Installing the webMethods ACH Module 17
Overview 18
Requirements 18
Supported Platforms and Operating Systems 18
Required webMethods Components 18
Software Requirements 19
Hardware Requirements 19
Install the ACH Module 19
Uninstall the ACH Module 20

Chapter 3. Managing the webMethods ACH Module 21
Overview 22
Configuring the ACH Module 22
Configuring the Custom ACH Module Persistence Store 24
Viewing ACH Files in the ACH Module Persistence Store 25
Configuring Routing Rules 26
Creating Routing Rules 26
Editing Routing Rules 27
Deleting Routing Rules 27

Chapter 4. Sending and Receiving ACH Files 29
Sending and Receiving ACH Files without Trading Networks 30
Sending ACH Files 30
Creating a Batch 30
Creating an ACH File 31
Sending the ACH File 31
Receiving ACH Files 31
Sending and Receiving ACH Files with Trading Networks 31
Setting up to Send and Receive ACH Files with Trading Networks 32
Enabling the WmACHForTN Package 32
Creating an External ID Type in Trading Networks 32
Sending ACH Files with Trading Networks 34
Ensuring Profiles are Defined 34
Defining TN Document Types for the Incoming Data 34
Defining Processing Rules to Process the Incoming Data 34
Receiving ACH Files with Trading Networks 35
Ensuring Profiles are Defined 35
Defining TN Document Types for the ACH File and Batches 36
Defining Processing Rules to Process Incoming ACH Files and Batches 36

Chapter 5. Services 37
Summary of Services 38
wm.ach.batch:appendEntry 40
wm.ach.batch:createBatch 41
wm.ach.batch:getBatchControl 41
wm.ach.batch:getBatchHeader 42
wm.ach.batch:getNextEntry 42
wm.ach.batch:toString 43
wm.ach.converter:convertDocToString 43
wm.ach.converter:convertStringToDoc 44
wm.ach.queue:generateACHFile 44
wm.ach.queue:getBatchFromQueue 45
wm.ach.queue:query 46
wm.ach.queue:queueBatch 47
wm.ach.tn.trp:receive 48
wm.ach.tn.trp:send 49
wm.ach.trp:receive 50
wm.ach.trp:receiveStream 50
wm.ach.validate:validateDocRecord 51
wm.ach.validate:validateStringRecord 52
wm.ip.config:getConfig 52

wm.ip.config:setConfig 53
wm.ip.persist.store:get 53
wm.ip.persist.store:list 54
wm.ip.persist.store:register 54
wm.ip.persist.util:add 55
wm.ip.persist.util:get 56
wm.ip.persist.util:query 56
wm.ip.persist.util:updateStatus 57
wm.ip.route:addRule 58
wm.ip.route:deleteRule 59
wm.ip.route:getRules 59
wm.ip.route:list 60
wm.ip.route:register 60
wm.ip.route:route 61

Appendix A. webMethods ACH Module Sample 63
Overview 64
Before You Begin 64
Running the Sample 64

Appendix B. Standard Entry Class Codes 67
Overview 68
SEC Codes 68
Consumer Applications 68
Corporate Applications 70
Other Applications 71

Index 73

About This Guide

This guide describes how to install, configure, and use the webMethods ACH Module 6.1. It contains information for administrators who configure and manage a webMethods system and for application developers who want to create webMethods Integration Server services that exchange ACH files with trading partners.

To use this guide effectively, you should:

- Have a basic knowledge of ACH and ACH terminology.
- Be familiar with the webMethods Integration Server, the Integration Server Administrator, and webMethods Developer and understand the concepts and procedures described in the *webMethods Integration Server Administrator's Guide* and the *webMethods Developer User's Guide*.
- Have installed the webMethods Integration Server, Developer, and the webMethods ACH Module software.

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
Italic	Identifies variable information that you must supply or change based on your specific situation or environment. Identifies terms the first time they are defined in text. Also identifies service input and output variables.
Narrow font	Identifies storage locations for services on the webMethods Integration Server using the convention *folder.subfolder:service*.
`Typewriter font`	Identifies characters and values that you must type exactly or messages that the system displays on the console.
UPPERCASE	Identifies keyboard keys. Keys that you must press simultaneously are joined with the "+" symbol.
\	Directory paths use the "\" directory delimiter unless the subject is UNIX-specific.
[ ]	Optional keywords or values are enclosed in [ ]. Do not type the [ ] symbols in your own code.

Additional Information

The webMethods Advantage Web site at http://advantage.webmethods.com provides you with important sources of information about webMethods components:

- **Troubleshooting Information**. webMethods provides troubleshooting information for many webMethods components in the webMethods Knowledge Base.
- **Documentation Feedback**. To provide documentation feedback to webMethods, go to the Documentation Feedback Form on the webMethods Bookshelf.
- **Additional Documentation**. All webMethods documentation is available on the webMethods Bookshelf.

For more information about ACH and ACH terminology, go to http://www.nacha.org/About/what_is_ach_.htm and http://www.achrulesonline.org.

webMethods

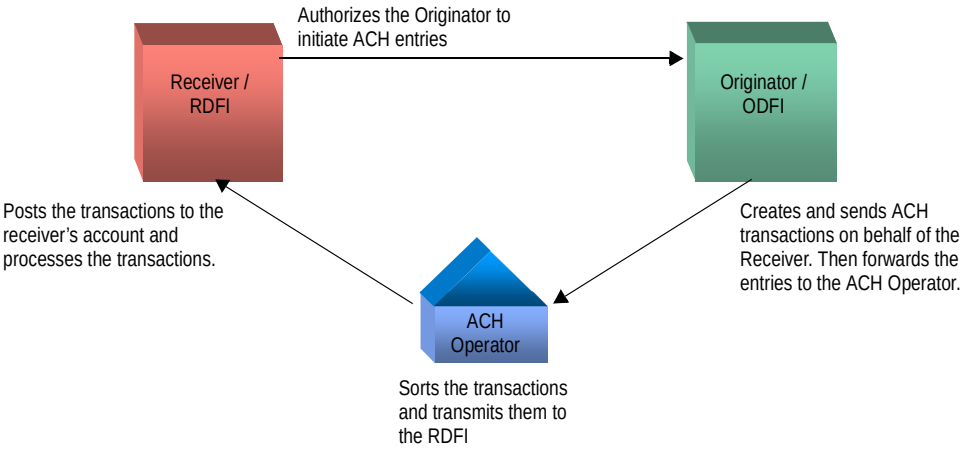
CHAPTER 1

Concepts

- What is Automated Clearing House (ACH)? 10
- What is the webMethods ACH Module? 11
- webMethods ACH Module Architecture 14
- webMethods ACH Module Features 16

What is Automated Clearing House (ACH)?

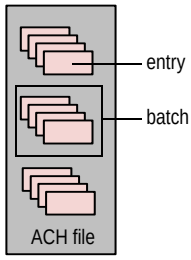
The *Automated Clearing House (ACH)* is a way to process payments with electronic transactions, replacing paper payments. ACH transactions are processed through the ACH network, which is a nationwide batch-oriented, electronic funds system that is governed by rules established by the *National Automated Clearing House Association (NACHA)* standards. The following diagram illustrates the ACH network:



- Originator initiates credit or debit transactions on behalf of a Receiver. For example, these transactions might be direct deposit of payroll, direct payment of consumer bills, or E-checks. The Originator is responsible for packaging the transactions into an ACH file. For more information, see "ACH Files" on page 11.
- Originating Depository Financial Institution (ODFI) receives payment instructions from the Originator and forwards the ACH file it receives to an ACH Operator. For example, an ODFI might be a bank.
- ACH Operator (for example, Federal Reserve) receives the ACH file. It sorts the transactions (entries) and transmits them to the Receiving Depository Financial Institution (RDFI).
- Receiving Depository Financial Institution (RDFI) posts the transactions (entries) to the Receiver's account.
- Receiver must authorize an Originator to initiate credit or debit transactions on their behalf. These transactions go through the ACH network and are posted to the Receiver's account. The Receiver receives the transactions and processes the transactions.

ACH Files

Electronic payment transactions are sent through the ACH network in an *ACH file*. The following diagram shows the structure of an ACH file:



- An *entry* (short for *entry detail record*) represents a single credit or debit transaction and is the basic unit of the fund transfer process. An entry contains information such as the recipient's ID, recipient's account number, the payment or debit type, and the amount of the payment or debit.

  An entry can be accompanied by multiple *addenda* records depending on the SEC code. Each addenda record supplies additional information about the entry.

- A *batch* consists of one or more entries having the same SEC code. Different batches can contain different types of payment-related information. For example, one batch can contain payments to a business' employees and another batch can contain payments to that business' creditors.

- An *ACH file* contains one or more batches. The file is sent to the recipient over the ACH network. The recipient then parses the file and processes the appropriate batch information it contains.

What is the webMethods ACH Module?

The webMethods ACH Module allows you to use the Integration Server to participate in the ACH network. In the ACH network, the webMethods ACH Module can:

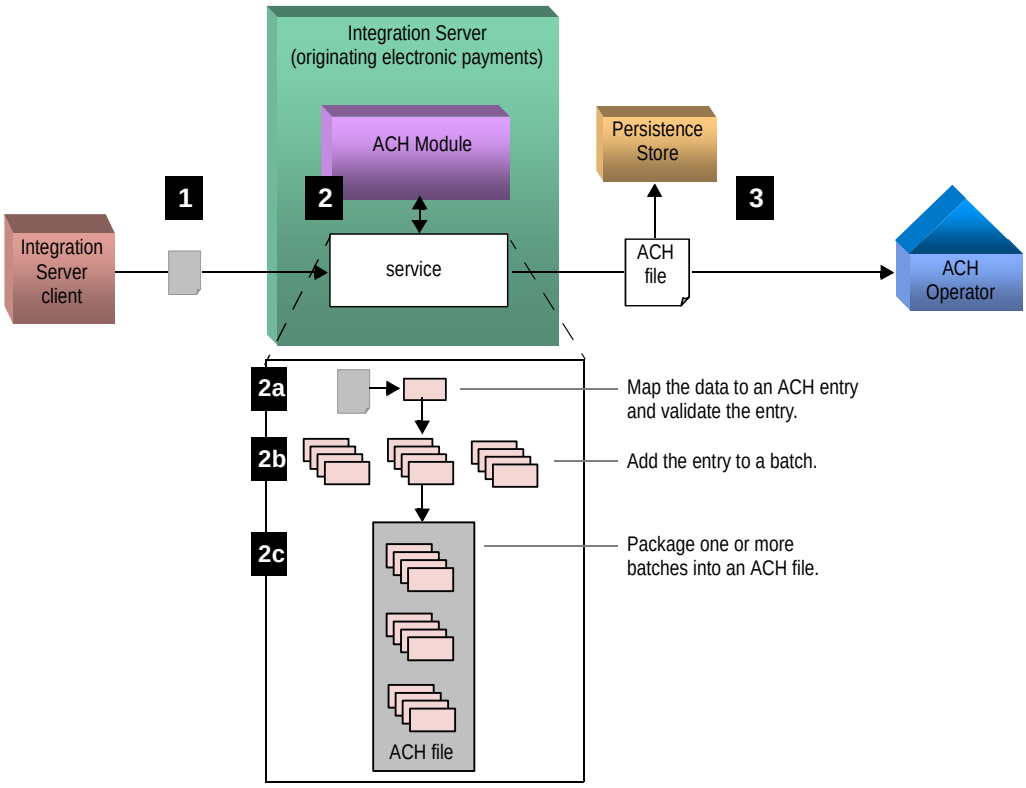
- Originate electronic payments by sending electronic payments described in an ACH file

- Receive and process electronic payments described in an ACH file

Sending Electronic Payments

To send electronic payments, the webMethods ACH Module creates an entry for each transaction, adds entries into a batch, packages the batches into an ACH file, and sends the ACH file to the appropriate ODFI.

The following diagram illustrates the use of the ACH Module to send electronic payments. See the table below the diagram for additional information.

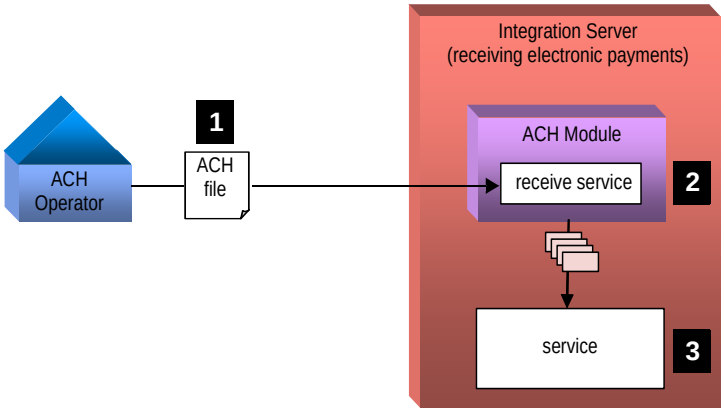


Step	Description
1	A client to the Integration Server sends data to the Integration Server that describes one or more electronic payment transactions.
2	A service that you create on the Integration Server receives the information from the client. The service formats the electronic payment information into the ACH format. For each electronic payment transaction, the service you create: a Maps the transaction information for an electronic payment into an ACH-format entry and validates the entry. You typically create an outbound map service to perform this mapping. When creating your outbound mapping, you can use IS document types that define the structure of ACH entries and that the ACH Module provides in the wm.ach.record folder. The ACH Module provides the wm.ach.validate:validateDocRecord and wm.ach.validate:validateStringRecord services to validate the ACH entry. b Adds the validated ACH entry to a batch using the wm.ach.batch:appendEntry service provided with the ACH Module.

Step	Description
	c Packages one or more batches into an ACH file and invokes a transport service to send the ACH file. To package the batches into an ACH file, use either the wm.ach.queue:generateACHFile or wm.ach.tn.trp:send service. Regardless of the service you use, the ACH file is saved to the ACH Module persistence store. To transport the document, you will need to create the transport service. For more information about creating this service, see "Sending ACH Files" on page 30 and "Sending ACH Files with Trading Networks" on page 34.
3	The ACH file is sent to the ACH Operator.

Receiving Electronic Payments

When receiving electronic payments, the webMethods ACH Module receives an ACH file and processes it. The following diagram illustrates the use of the ACH Module when receiving electronic payments. See the table below the diagram for additional information.



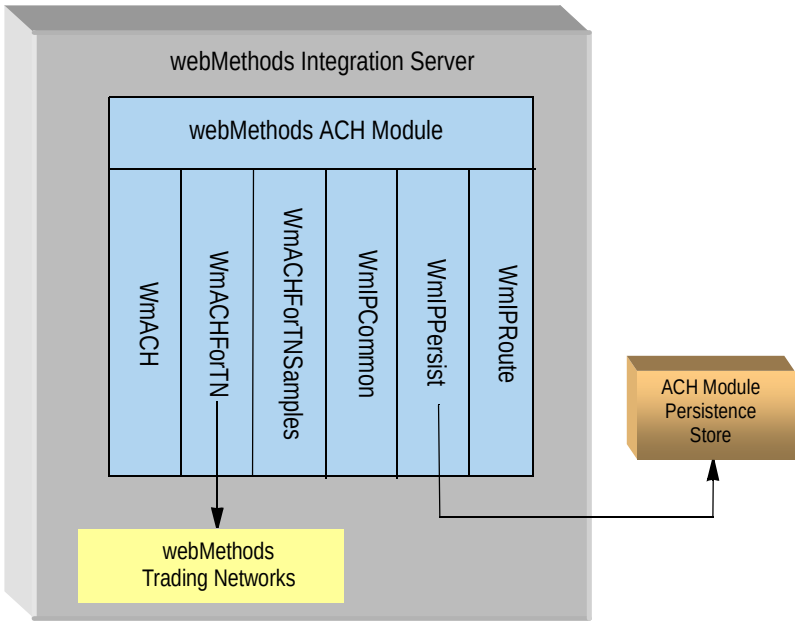
Step	Description
1	The ACH Operator sends an ACH file.
2	The RDFI receives the ACH file and sends the ACH file to the Integration Server by invoking either the wm.ach.trp:receive or wm.ach.tn.trp:receive service, which are provided with the ACH Module to process incoming ACH files. The wm.ach.trp:receive does not save the incoming ACH file to the ACH Module persistence store. However, if you use wm.ach.tn.trp:receive to send the ACH file to Trading Networks, you can save the incoming ACH file to the Trading Networks database.

Step	Description
	The receive service validates the incoming ACH file. It parses the individual batches from within the ACH file and sends each batch for processing. ■ When using the wm.ach.tn.trp:receive service to send the ACH file to Trading Networks, you set up processing in Trading Networks to process the batch. This includes creating a TN document type for the batch file and a processing rule that describes the processing to perform against the batch. The processing rule will likely use the Execute a Service processing action to invoke a service that will process the batch. See step 3 below for a description of a service that processes a batch. For more information about processing ACH files with Trading Networks, see "Receiving ACH Files with Trading Networks" on page 35. For more information about Trading Networks, see the *webMethods Trading Networks Concepts Guide* and the *webMethods Trading Networks User's Guide*. ■ When using the wm.ach.trp:receive service, the service uses routing rules that you define with the ACH Module. The routing rules identify the service to invoke to process the batch. See step 3 below for a description of a service that processes a batch. For more information about setting up routing rules using the ACH Module, see "Configuring Routing Rules" on page 26. For more information about processing ACH files without Trading Networks, see "Receiving ACH Files" on page 31.
3	The service you create extracts individual entries from within a batch and processes the transactions represented by the entry. The ACH Module provides the wm.ach.batch:getNextEntry service you can use to extract entries from a batch.

webMethods ACH Module Architecture

The webMethods ACH Module includes a set of packages that provide services, messages, and samples that enable you to send and receive ACH files over the ACH network. The webMethods ACH Module must be installed on the webMethods Integration Server. For detailed installation instructions and software requirements, see "Install the ACH Module" on page 19.

The following diagram illustrates how the webMethods ACH Module fits into the webMethods Integration Server. For information about each of these components, see the text following the diagram.



- webMethods ACH Module. The webMethods ACH Module includes the following public packages:
  - WmACH contains the services to send and receive ACH messages. It is designed as the generic interface through which ACH message communication occurs. The webMethods ACH Module parses all messages using a flat file template.
  - WmACHForTN is the package through which the webMethods ACH components implement Trading Networks. It follows the webMethods transport design to allow Trading Networks to call a generic interface, regardless of the actual implementation of the interface being used.
  - WmACHForTNSamples contains sample flow services, mappings, and records to demonstrate how the ACH Module can be used with Trading Networks. To run the sample, see Appendix A, "webMethods ACH Module Sample", on page 63.
  - WmIPCommon contains services required to configure the ACH Module (for example, the location of the persistence store folder and the routing rules folder).
  - WmIPPersist contains services required to save a message in either the file system or through custom-defined Integration Server services. Read more about the ACH Module persistence store in the bullet "ACH Module Persistence Store," below.
  - WmIPRoute contains services to define routing rules for messages. This package is used only if Trading Networks is *not* being used.

- webMethods Integration Server. This is the server underlying the webMethods components. Use the Integration Server Administrator to manage, configure, and administer all aspects of the Integration Server, such as users, security, packages, and services. For details, see the *webMethods Integration Server Administrator's Guide*.
- ACH Module Persistence Store. The webMethods ACH Module can saves ACH files to one of the following persistence stores:
  - File System - ACH files are saved into a specified directory.
  - Custom - By invoking user-defined Integration Server services, ACH files can be saved within a database or file system.

  If the WmACHForTN package is enabled, in addition to the ACH persistence store, ACH messages are also saved in the Trading Networks database.
- webMethods Trading Networks (or Trading Networks). By enabling the WmACHForTN package, ACH messages are sent to and received by Trading Networks. Trading Networks enables your enterprise to link with other companies (buyers, suppliers, strategic partners) and marketplaces to form a business-to-business trading network. For details, see the *webMethods Trading Networks Concepts Guide* and the *webMethods Trading Networks User's Guide*.

webMethods ACH Module Features

Using the webMethods ACH Module, you can:

- Send and receive ACH files using the Integration Server. ACH files can be created from data provided by other systems connected to the Integration Server.
- Route and store ACH files using a file system, custom services, or Trading Networks. This feature provides a centralized persistence store of all ACH files. You can save, delete, or track messages from the persistence store.
- Customize ACH Module configuration from the Integration Server Administrator. The webMethods ACH Module provides a user interface in which you can change the ACH Module configuration settings.
- Track stored ACH files from the ACH Module persistence store. You can view and track stored messages from the ACH Home page, which is provided with the webMethods ACH Module.

webMethods

CHAPTER 2

Installing the webMethods ACH Module

- Overview 18
- Requirements 18
- Install the ACH Module 19
- Uninstall the ACH Module 20

Overview

Important! The information in this chapter might have been updated since the guide was published. Go to the webMethods Advantage Web site at http://advantage.webmethods.com for the latest version of the guide.

If you are installing the ACH Module at the same time you are installing other webMethods components, such as the webMethods Integration Server, see the *webMethods Installation Guide* for instructions on installing those components.

Requirements

Supported Platforms and Operating Systems

The ACH Module supports the following platforms and uses the same JVM as its host Integration Server.

Platform and Operating System
Microsoft Windows 2000, 2003
Microsoft Windows XP Professional
Sun Solaris 2.8
HP/UX 11i

Required webMethods Components

The following table lists the webMethods components you must install before or at the same time you install the ACH Module. The table also lists the webMethods components you must install at some point for the ACH Module to operate fully.

Required for Installation	Required for Full Operation
Integration Server 6.1 or later	Developer 6.1 or later
Trading Networks Server 6.1 or later [1]	Trading Networks Console 6.1 or later [1]

[1] Optional. Trading Networks is required only if you are using the ACH Module with Trading Networks.

Software Requirements

The ACH Module has no software requirements beyond those of its host Integration Server.

Hardware Requirements

The ACH Module has no hardware requirements beyond those of its host Integration Server.

Install the ACH Module

Important! This section provides only instructions that are specific to installing the ACH Module. For complete instructions on using the webMethods Installer, see the *webMethods Installation Guide*.

Install the ACH Module 6.1 on the same machine as the Integration Server. The installer will automatically install the ACH Module in the Integration Server installation directory.

To install the ACH Module

1. Download webMethods Installer 6.1 from the webMethods Advantage Web site at http://advantage.webmethods.com.
2. If you are going to install the ACH Module on an already installed Integration Server, shut down the Integration Server.
3. Start the installer.
4. Choose the webMethods platform on which to install the ACH Module. If you are going to install the ACH Module on an existing Integration Server, choose the platform that matches the release of that Integration Server. For example, if you are going to install the ACH Module on a 6.1 Integration Server, choose the 6.1 platform.
5. Specify the ACH Module installation directory as the webMethods 6 installation directory (by default, webMethods6).
6. In the component selection list, navigate to webMethods Platform ▸ eStandards ▸ webMethods ACH Module and select the desired components:
   - Documentation 6.1 (Optional). Contains the documentation for this module.
   - Program Files 6.1 (Required). Contains the program files for this module.
   - Samples 6.1 (Optional). Contains sample documents and processing rules for this module.

- TNSupport 6.1 (Optional). Contains support for Trading Networks.
- Any required webMethods components you have not installed.

7 Complete the installation.

The webMethods ACH Module starts automatically when you start the Integration Server.

Uninstall the ACH Module

This section provides only instructions that are specific to uninstalling the ACH Module. For complete instructions on using the webMethods Uninstaller, see the *webMethods Installation Guide*

To uninstall the ACH Module

1 Shut down the Integration Server that hosts the ACH Module.

2 Start the webMethods Uninstaller, as follows:

System	Action
Windows	In the Add or Remove Programs window, select webMethods *release installation_directory* as the program to uninstall, where *release* and *installation_directory* are the release and installation directory of the Integration Server on which the ACH Module is installed.
UNIX	Navigate to the webMethods_directory/bin directory of the installation that includes the Integration Server on which the ACH Module is installed and enter `uninstall` (wizard) or `uninstall -console` (console mode).

3 In the component selection list, navigate to webMethods Platform ▸ eStandards ▸ webMethods ACH Module and select Program Files.

4 The uninstaller removes all ACH Module-related files that were installed into the *IntegrationServer_directory*\packages\WmACH directory. The uninstaller does not delete files created after you installed the ACH Module (for example, user-created or configuration files), nor does it delete the directory structure that contains the files.

5 If you do not want to save the files the uninstaller did not delete, navigate to the *IntegrationServer_directory*\packages directory and delete the WmACH directory.

webMethods

CHAPTER 3

Managing the webMethods ACH Module

- Overview 22
- Configuring the ACH Module 22
- Configuring the Custom ACH Module Persistence Store 24
- Viewing ACH Files in the ACH Module Persistence Store 25
- Configuring Routing Rules 26

Overview

The ACH Home page provides several functional areas that you can use to configure and manage the ACH Module. To access the ACH Home page, click ACH under Adapters in the Integration Server Administrator.

This chapter describes how to use the ACH Module in the following areas.

- About provides the copyright and version information for the ACH Module, as well as a link to the ACH Module documentation.
- Configuration enables you to configure the attributes specific to the ACH Module. For more information, see section "Configuring the ACH Module" on page 22.
- Persistence enables you to query and view the ACH files in the ACH Module persistence store. For more information, see sections "Configuring the Custom ACH Module Persistence Store" on page 24 and "Viewing ACH Files in the ACH Module Persistence Store" on page 25.
- Routing enables you to create the routing rules for inbound ACH files. For more information, see section "Configuring Routing Rules" on page 26.
- Payment Gateway Sample enables you to run the sample that is part of the WmACHForTNSamples package. For more information, see Appendix A, "webMethods ACH Module Sample", on page 63.

Configuring the ACH Module

The webMethods ACH Module enables you to specify configuration parameters on the ACH Home page.

To configure the ACH Module

1 On the ACH Home page, click Configuration. The Current Configuration page appears.

2 Enter the following information:

Field	Description
Default Destination ID	This value is used as a default destination ID in the header of all outbound ACH files. The ID is a 9-digit number.
Default Destination Name	This value is used as a default destination name in the header of all outbound ACH files.

Field	Description
Default Origination ID	This value is used as a default origination ID in the header of all outbound ACH files. The ID is a 9-digit number.
Default Origination Name	This value is used as a default origination name in the header of all outbound ACH files.
Max Batch Count	Maximum number of batches that can be queued and placed in the outgoing file.
Max Entry Count	Maximum number of entries per batch.
Max ADV Entry Count	Maximum number of entries per batch that can have an SEC Code of ADV.
Persistence Type	The type of ACH Module persistence store. Select one of the following from the drop-down list: ■ File - ACH files are saved to the file system. ■ IS Service - ACH files are saved to a custom store (such as a database) that the Integration Server can access. For configuration instructions, see "Configuring the Custom ACH Module Persistence Store" on page 24.
Persistence Folder Name	The location where the ACH files will be saved. ■ If the ACH Module persistence store is a file, specify the fully qualified path name of the directory in which the ACH Module will persist the batches. ■ If the ACH Module persistence store is a custom Integration Server service, specify the directory (namespace) that holds the services that access your custom persistence store. For example, *folderA.folderB*
Routing Folder Name	The location where routing rules will be stored.

Field	Description
Default Output Folder	The location where the outbound ACH file is stored.
Additional Persistence Query Fields	Enter one or more batch header fields, separated by commas. ACH Module will use these fields, in addition to the standard index fields, to index new queued batches. *Fields you specify will not be added to batches already in the persistence medium.* For each value you enter there will be an extra input field present on the Search Criteria page (see page 25). The standard index fields are message type, message version, sender ID, receiver ID, and effective entry date.

3 Click Save.

4 Reload the WmACH package so that the changes you made will take effect.

Configuring the Custom ACH Module Persistence Store

The ACH Module enables you to customize where ACH files are stored by providing specification references for your custom services. For information about specification references and services, see the *webMethods Developer User's Guide.*

You must create the following services yourself if you want to use a custom ACH Module persistence store. The specifications for these services are located in the WmIPPersist package.

- wm.ip.persist.spec:add - saves ACH files
- wm.ip.persist.spec:get - retrieves ACH files
- wm.ip.persist.spec:query - retrieves the unique key list of ACH files
- wm.ip.persist.spec:remove - removes ACH files from the ACH Module persistence store
- wm.ip.persist.spec:updateStatus - updates the status of ACH files

To configure a custom ACH Module persistence store

1 From the webMethods Developer, create a new service. For instructions, see the *webMethods Developer User's Guide*.

Important! You must name your service the same name as the specification. For example, if you implement the add specification reference, you must name your service add.

2 In the Developer editor, click the **Input/Output** tab.

3 In the **Specification Reference** field, type the specification's name or click the browse button to select it from the WmIPPersist package.

4 Open the Integration Server Administrator.

5 On the **ACH Home** page, click **Configuration**. The **Current Configuration** page appears.

   a Specify a persistence type of `IS Service`.

   b Enter a value in the **Persistence Folder Name** field. The namespace should be identical to the service created in step 1.

The custom ACH Module persistence store is now configured. The ACH Module will use this custom Integration Server service (rather than the file system) when it saves batches.

Viewing ACH Files in the ACH Module Persistence Store

The ACH Module enables you to query and view saved ACH files in its persistence store.

To view ACH files in the ACH Module persistence store

1 On the **ACH Home** page, click **Persistence**. The **Search Criteria** page appears.

2 Enter your search criteria. The following table describes the search parameters.

Input	Description
Message Type	The message type must be a SEC code. For a list of these, see Appendix B, "Standard Entry Class Codes", on page 67. If you leave this field blank, the search will return all message types except ADV.
Message Version	Version of the ACH file. `1.0` is the only valid value.
Sender ID	ID of the ACH file's originator. If you leave this field blank, the search will return all originator IDs.
Receiver ID	ID of the ACH file's receiver. If you leave this field blank, the search will return all receiver IDs.
Begin Date Range	The earliest effective entry date the search is to return. If you leave this field blank, the search will consider the beginning of the date range to be open-ended. The effective entry date is a field in the batch header record.

Input	Description
End Date Range	The latest effective entry date the search is to return. If you leave this field blank, the search will consider the end of the date range to be open-ended. The effective entry date is a field in the batch header record.
Maximum Results	Maximum number of ACH files to retrieve from the ACH Module persistence store.
(other fields)	Any fields specified in the Additional Persistence Query Fields field (page 24) display here as additional input fields. The query will consider an additional query field *only* if it was identified as an additional persistence query field at the time the batch was added to the batch queue.

3 Click Search. A list of ACH files matching the criteria you specified displays. Select a message from the list to see its contents.

Configuring Routing Rules

The webMethods ACH Module enables you to specify the Integration Server service that is invoked after an ACH file is received.

Routing rules are applicable only if the WmACHForTN package is *not* enabled. That is, these routing rules are not applicable if you are using Trading Networks to process an incoming file of batches.

Creating Routing Rules

To create a routing rule

1 On the ACH Home page, click Routing. The Routing Configuration page appears.

2 Click Create Routing Rule. The Routing Rule Properties page appears.

3 Enter the following information:

Field	Description
Message Type	The message type must be a SEC code. For a list of these, see Appendix B, "Standard Entry Class Codes", on page 67.
Version	The version of the ACH file. `1.0` is the only valid value.
Sender ID	The routing/transmit number of the sender.

Field	Description
Receiver ID	The routing/transmit number of the receiver.
IS Service	The Integration Server service to which an ACH batch matching the above criteria is sent for processing. Type the fully-qualified namespace of the Integration Server service. For example, *folderA.folderB:serviceName*. The service you specify must be able to accept and process the input variables *batchObject* and *errorObject*. These input variables must be of type Object.

4 Click Save Changes. The Routing Configuration page re-appears.

Editing Routing Rules

Complete the following steps to edit a routing rule. The IS Service field is the only one you can change.

To edit a routing rule

1 On the ACH Home page, click Routing. The Routing Configuration page appears.

2 Under the Edit column, click  for the routing rule you want to edit. The Routing Rule Properties page appears.

3 In the IS Service field, type the fully-qualified namespace of the Integration Server service to which an ACH batch matching the other criteria is sent for processing. For example, *folderA.folderB:serviceName*.

4 Click Save Changes. The Routing Configuration page appears.

Deleting Routing Rules

Complete the following steps to delete a routing rule.

To delete a routing rule

1 On the ACH Home page, click Routing. The Routing Configuration page appears.

2 Under the Delete column, click ✕ for the routing rule you want to delete. A dialog box appears confirming the delete.

3 Click OK.

webMethods

CHAPTER 4

Sending and Receiving ACH Files

- Sending and Receiving ACH Files without Trading Networks 30
- Sending and Receiving ACH Files with Trading Networks 31

Sending and Receiving ACH Files without Trading Networks

Note: To use the ACH Module without Trading Networks, you must use the Integration Server to disable the WmACHForTN package if it is installed.

Sending ACH Files

In Chapter 1, "Concepts", an overview of sending ACH files was shown in "Sending Electronic Payments" on page 11.

You create the logic to create an ACH file and send it. There are three parts to creating and sending ACH file when you are not using Trading Networks—creating and queueing the batch, generating an ACH file from the queued batches, and sending the ACH file.

Creating a Batch

Follow these steps in Developer to create and populate a batch with entries:

1. Create an empty batch using the wm.ach.batch:createBatch service.
2. In a loop, create entries and add the entries to a batch:
   a. Typically the send process involves the receipt of some type of data; for example, an Integration Server client might have sent payment information. This data might not be in ACH format. Create logic to map the data to ACH format. This mapping is referred to as *outbound mapping*. To help you with the outbound mapping, the ACH Module provides IS document types that define the structure of the ACH entries. The IS document types reside in the wm.ach.record folder within the WmACH package.
   a. After creating an ACH-formatted entry, use the wm.ach.batch:appendEntry service to add the entry to the batch.
3. After all entries have been added to the batch, add the batch to the queue with the wm.ach.queue:queueBatch service.

At this point the batch is queued in the persistence medium.

For an example of a flow service that creates and populates a batch, see the wm.ach.tn.sample.maps.outbound:mapPaymentXMLToCCD service in Developer.

Creating an ACH File

Do the following to generate an ACH file:

1. Call the wm.ach.queue:query service to return all the queued batches that match the criteria you specify through the service's input fields.
2. Add the batches in the QUEUED state to the ACH file with the wm.ach.queue:generateACHFile service.

Sending the ACH File

The ACH Module does not implement a transport service. Rather, the user must implement the transport service.

Receiving ACH Files

In Chapter 1, "Concepts", an overview of receiving ACH files was shown in "Receiving Electronic Payments" on page 13.

An incoming ACH file should be sent to the wm.ach.trp:receive service. The service validates the ACH file. Then the wm.ach.trp:receive service parses the batches that are in the ACH file. The next step is to invoke a service to process the batch. When the WmACHforTN package is disabled (indicating that you are *not* using Trading Networks), the wm.ach.trp:receive service routes the batch to the Integration Server service that is configured on the Routing page. For instructions about how to set up a routing rule, see "Creating Routing Rules" on page 26.

You create the Integration Server service that processes the batch. Each batch is associated with a single SEC code. In your service, process each entry in the batch. Use the wm.ach.batch:getNextEntry service to retrieve an entry.

Sending and Receiving ACH Files with Trading Networks

When the WmACHforTN package is enabled, the webMethods ACH Module sends incoming ACH files that it receives to Trading Networks for processing. In this way, Trading Networks replaces the ACH Module internal routing manager. (That is, Routing rules created from the ACH Home page are ignored.) Instead you use TN document types and processing rules to identify and route incoming ACH files. Additionally, you can also use Trading Networks when creating outgoing ACH files.

To use Trading Networks, you need to perform some preliminary setup; see "Setting up to Send and Receive ACH Files with Trading Networks" on page 32.

When using Trading Networks, the ACH Module stores ACH files in its own persistence store just as it does when you are not using Trading Networks. However, in addition, Trading Networks also provides the ability to track outgoing and incoming messages using its own persistence store; that is, you can save the ACH file and its batches to the

Trading Networks database. (The Trading Networks persistence store is separate from the ACH Module persistence store.) This enables queries to be run on particular ACH files and batches, such as which sender or receiver has sent an ACH file, or tracing all batches of a particular processing status on a particular date. All of these functions are available for both sent and received messages.

Setting up to Send and Receive ACH Files with Trading Networks

Before using the webMethods ACH Module with Trading Networks, you must do the following:

1 Enable the WmACHForTN package as explained in "Enabling the WmACHForTN Package" on page 32.

2 Set up the Trading Networks environment by doing one of the following:

- Execute the *IntegrationServer_directory*\packages\WmACHForTNSamples\pub\setup\TNSampleSetup.dat file. This automatically creates an external ID type and TN document types called "ACH File" and "CCD". To create TN document types for other batches, duplicate the CCD document type and make appropriate changes.
- Follow the instructions in "Creating an External ID Type in Trading Networks" on page 32.

Enabling the WmACHForTN Package

To enable the WmACHForTN package

1 Open the Integration Server Administrator.

2 In the **Packages** menu of the Navigation panel, click **Management**.

3 Navigate to the **WmACHForTN** package and click **OK** to enable the package. When the package is enabled, the server displays a **Yes** in the **Enabled** column.

Creating an External ID Type in Trading Networks

To create an external ID

1 From the **WmTN** package, run the **wm.tn.dictionary:addIDType** service. Set the *description* parameter to `R/T Identification`.

2 Set the TN property, tn.required.idType, to `R/T Identification`. To do so:

   a From the Integration Server Administrator, in the **Packages** menu of the Navigation panel, click **Management**.

   b Navigate to the **WmTN** package and click **Home**. The Settings > TN Properties screen displays.

   c Click **Edit TN Properties Settings**.

   d Add or modify the setting for the tn.required.idType as shown below:

   ```
   tn.required.idType = R/T Identification
   ```

   e Click **Save Changes**.

3 Update your Enterprise profile to include the R/T Identification external ID.

   a Start the Trading Networks Console.

   b Select **View** ▸ **Enterprise** to view the Enterprise profile.

   c In the External ID Type / Value table, click **Add New External ID**.

   d For the External ID Type, select **R/T Identification**.

   e Specify the correct value for **R/T Identification**.

   f Click **Save**.

4 Add an R/T Identification external ID to each of your partner profiles.

   a From the Trading Networks Console, Select **View** ▸ **Trading Partners**.

   b Select the row for a trading partner, right-click, and select **Edit Trading Partner**.

   c Click the **Corporate** tab in the profile.

   d In the External ID Type / Value table, click **Add New External ID**.

   e For the External ID Type, select **R/T Identification**.

   f Specify the correct value for **R/T Identification**.

   g Click **Save**.

Note: You can have multiple partners, but each must have a unique identifier.

Sending ACH Files with Trading Networks

In Chapter 1, "Concepts", an overview of sending ACH files was shown in "Sending Electronic Payments" on page 11.

To send ACH files using Trading Networks, you need to do the following setup—ensure profiles are defined for senders and receivers, define TN document types for the incoming data, define a processing rule to process the incoming data.

Ensuring Profiles are Defined

The overview described in "Sending Electronic Payments" on page 11 shows that an Integration Server client sends data (for example, payment information) to the Integration Server. If you want to use Trading Networks, the client would send the data directory to Trading Networks.

Within Trading Networks, this Integration Server client is the sender. The data that the sender sends to Trading Networks typically includes data to identify the sender and receiver. Be sure that you have Trading Networks profiles defined for the sender and receiver.

For information about creating Trading Networks profiles, see the *webMethods Trading Networks User's Guide*.

Defining TN Document Types for the Incoming Data

Create TN document types that Trading Networks can use to recognize the type of incoming data. How you define the TN document type depends on the format of the data that the sender is going to send.

For information about defining TN document types, see the *webMethods Trading Networks User's Guide*.

Defining Processing Rules to Process the Incoming Data

Define a processing rule that will execute when the incoming data is sent to Trading Networks. For example, you might set the criteria in the processing rule to match the TN document type you defined for the incoming data.

The processing action for the processing rule should put the incoming data into ACH format. To do so, use the Execute a Service processing action to invoke a service you create. This service should:

1. Create an empty batch using the wm.ach.batch:createBatch service.
2. In a loop, go through the incoming data for that corresponds to an ACH entry and create an ACH entry and add the entry to the batch.
   a. Create logic to map the incoming data to ACH format. This mapping is referred to as *outbound mapping*. To help you with the outbound mapping, the ACH Module provides IS document types that define the structure of the ACH entries.

The IS document types reside in the wm.ach.record folder within the WmACH package.

b After creating an ACH-formatted entry, use the wm.ach.batch:appendEntry service to add the entry to the batch.

3 After all entries have been added to the batch, add the batch to the queue with the wm.ach.queue:queueBatch service. At this point the batch is queued in the persistence medium. For an example of a flow service that creates and populates a batch, see the wm.ach.tn.sample.maps.outbound:mapPaymentXMLToCCD service in Developer.

4 Use the wm.ach.tn.trp:send service to send the ACH file. Note that the ACH Module does not implement a transport service. Rather, the user must implement this service. You can use the wm.ach.tn.trp:send in one of the following two ways:

- Directly send the ACH file by specifying the transport service in the *sendSvc* parameter of the wm.ach.tn.trp:send service.
- The wm.ach.tn.trp:send service first submits the ACH file to Trading Networks. So you can set up a TN document type for an outbound ACH file and create an associated processing rule that uses the Execute a Service processing action to invoked the transport service.

For information about defining processing rules, see the *webMethods Trading Networks User's Guide*.

Receiving ACH Files with Trading Networks

In Chapter 1, "Concepts", an overview of receiving ACH files was shown in "Receiving Electronic Payments" on page 13.

An incoming ACH file should be sent to the wm.ach.tn.trp:receive service. The wm.ach.tn.trp:receive service sends the complete ACH file to Trading Networks. It then parses the batches that are in the ACH file, validates each batch, and sends each batch to Trading Networks as a separate document.

To receive ACH files with Trading Networks, you need to do the following setup—ensure profiles are defined for senders and receivers, define TN document types for the incoming ACH file and the batches within the ACH file, define processing rules to process each document; that is to process the ACH file and each batch.

Ensuring Profiles are Defined

Trading Networks requires that you have Trading Networks profiles defined for the sender and receiver of the ACH file and each batch. For information about creating Trading Networks profiles, see the *webMethods Trading Networks User's Guide*.

Defining TN Document Types for the ACH File and Batches

As described above, the wm.ach.tn.trp:receive service sends the complete ACH file to Trading Networks. You will need to create a TN document type that Trading Networks can use to recognize the ACH file.

Additionally, the wm.ach.tn.trp:receive service sends one document for each batch within the ACH file. Each batch is associated with a single SEC code. You need to create a TN document type for each type of batch.

For information about defining TN document types, see the *webMethods Trading Networks User's Guide*.

Defining Processing Rules to Process Incoming ACH Files and Batches

Define a processing rule to process an incoming ACH file and processing rules for each of the batches. You might set up the criteria in the processing rules to match on the TN document type of the incoming document (that is, either the ACH file or the TN document type for a batch).

To process the ACH file, you might just want to set the pre-processing action to save the document to the Trading Networks database, but perform no further processing actions on the document.

To process a batch, you can use the Execute a Service processing action to invoke a service that you create that processes each entry in the batch. Your service can invoke the wm.ach.batch:getNextEntry service to retrieve an entry from the batch.

For information about defining processing rules, see the *webMethods Trading Networks User's Guide*.

webMethods

CHAPTER 5

Services

- Summary of Services 38

Summary of Services

This section summarizes the services that come with the ACH Module.

Element	Package and Description
wm.ach.batch:appendEntry	WmACH. Appends an entry detail record and any related addenda records to a batch object.
wm.ach.batch:createBatch	WmACH. Creates a new batch object instance based on batch header information.
wm.ach.batch:getBatchControl	WmACH. Computes batch control information for a batch object.
wm.ach.batch:getBatchHeader	WmACH. Extracts batch header information from a batch object.
wm.ach.batch:getNextEntry	WmACH. Gets next entry record from ACH batch object.
wm.ach.batch:toString	WmACH. Returns the string representation of an ACH batch object. This includes the batch header record, entry detail records, and batch control record in the ACH record format.
wm.ach.converter:convertDocToString	WmACH. Converts an ACH record table to a record string in ACH record format. You may optionally instruct the service to validate the record.
wm.ach.converter:convertStringToDoc	WmACH. Converts an ACH record format string to a record table containing key/value pairs. You may optionally instruct the service to validate the record.
wm.ach.queue:generateACHFile	WmACH. Generates an ACH message file from queued batches.
wm.ach.queue:getBatchFromQueue	WmACH. Returns a batch object from the queue that matches the specified batch ID (key). The batch object returned is read only. Entries cannot be added to this batch object, nor can this batch object be added back to the queue.
wm.ach.queue:query	WmACH. Returns a set of batch keys from the batch queue that matches the specified criteria.
wm.ach.queue:queueBatch	WmACH. Adds an ACH batch object to the batch queue under the ACH Module persistence store. Also creates an index for each batch based on header information, to improve searching.
wm.ach.tn.trp:receive	WmACHForTN. Sends an ACH message stream to Trading Networks two times—first as the complete message stream, then as the stream's component batches.
wm.ach.tn.trp:send	WmACHForTN. Generates an ACH message file from queued batches based on the specified search criteria and sends the file to Trading Networks for persistence. The service then invokes the Integration Server service specified with the sendSvc parameter.

Element	Package and Description
wm.ach.trp:receive	WmACH. Parses an ACH message file, breaking it into separate batches, and sends the batches to the appropriate routing engine after validation. The routing engine invokes the appropriate processing rule depending on the matching criteria defined in the routing rules. If package WmACHForTN is enabled, then processing rules defined in Trading Networks are used; otherwise, routing rules defined through the ACH routing page are used.
wm.ach.trp:receiveStream	WmACH. Parses an ACH message stream, breaking it into separate batches, and sends the batches to the appropriate routing engine after validation. The routing engine invokes the appropriate processing rule depending upon the matching criteria defined in the routing rules. If package WmACHForTN is enabled, then processing rules defined in Trading Networks are used; else routing rules defined through the ACH routing page are used.
wm.ach.validate:validateDocRecord	WmACH. Performs validations specified by ACH rules on an ACH record.
wm.ach.validate:validateStringRecord	WmACH. Performs validations specified by ACH rules on an ACH record string.
wm.ip.config:getConfig	WmIPCommon. Obtains configuration information about an IP product (stored in a configuration file).
wm.ip.config:setConfig	WmIPCommon. Sets the configuration information for an IP product (stored in a configuration file).
wm.ip.persist.store:get	WmIPPersist. Obtains the object representing the ACH Module persistence store for a product.
wm.ip.persist.store:list	WmIPPersist. Obtains the list of product keys registered with the persistence module.
wm.ip.persist.store:register	WmIPPersist. Registers a product with the persistence module.
wm.ip.persist.util:add	WmIPPersist. Persists a message through the persistence module.
wm.ip.persist.util:get	WmIPPersist. Retrieves a message from the ACH Module persistence store.
wm.ip.persist.util:query	WmIPPersist. Retrieves a list of messages from the ACH Module persistence store based on search criteria specified.
wm.ip.persist.util:updateStatus	WmIPPersist. Updates the status of a stored message.
wm.ip.route:addRule	WmIPRoute. Adds a routing rule.
wm.ip.route:deleteRule	WmIPRoute. Deletes a routing rule from the list of rules.
wm.ip.route:getRules	WmIPRoute. Gets all routing rules defined for a product.

Element	Package and Description
wm.ip.route:list	WmIPRoute. Obtains a list of product keys registered with the routing module.
wm.ip.route:register	WmIPRoute. Registers a product with the routing module.
wm.ip.route:route	WmIPRoute. Invokes appropriate service with inputs specified, as defined by routing rules. Appropriate routing rule is obtained first by querying list of rules based on search criteria.

wm.ach.batch:appendEntry

WmACH. Appends an entry detail record and any related addenda records to a batch object.

Input Parameters

entry IData Entry detail record of a certain SEC type identified in the batchHeader input.

addendas IData[ ] (Optional) Addenda records specifying additional information about the entry.

batchObject Object Batch object created from batch header information.

Output Parameters

success String Results of the service. A value of `true` means the service terminated normally and a value of `false` means it terminated abnormally.

errors String Any errors produced during invocation of service.

Usage Notes

None

See Also

wm.ach.batch:createBatch

Examples

See the following service in the WmACHForTNSamples package:

wm.ach.tn.sample.maps.outbound:mapPaymentXMLToCCD

wm.ach.batch:createBatch

WmACH. Creates a new batch object instance based on batch header information.

Input Parameters

batchHeader IData Batch header record of a certain SEC type.

Output Parameters

batchObject Object Batch object created from batch header information.

Usage Notes

None

See Also

wm.ach.batch:appendEntry

Examples

See the following service in the WmACHForTNSamples package:

wm.ach.tn.sample.maps.outbound:mapPaymentXMLToCCD

wm.ach.batch:getBatchControl

WmACH. Computes batch control information for a batch object.

Input Parameters

batchObject Object Batch object created from Batch header information.

Output Parameters

batchControl IData Batch control record computed for *batchObject*.

Usage Notes

None

See Also

wm.ach.batch:getBatchHeader

Examples

None

wm.ach.batch:getBatchHeader

WmACH. Extracts batch header information from a batch object.

Input Parameters

batchObject Object Batch object created from batch header information.

Output Parameters

batchHeader IData Batch header record extracted from *batchObject*.

Usage Notes

None

See Also

wm.ach.batch:getBatchControl

Examples

None

wm.ach.batch:getNextEntry

WmACH. Gets next entry record from ACH batch object.

Input Parameters

batchObject Object Batch object created from batch header information.

Output Parameters

batchHeader IData Batch header record extracted from *batchObject*.

Usage Notes

Call this service when iteratively processing the entries of an inbound ACH batch.

Examples

None

wm.ach.batch:toString

WmACH. Returns the string representation of an ACH batch object. This includes the batch header record, entry detail records, and batch control record in the ACH record format.

Input Parameters

batchObject **Object** Batch object created from batch header information and ready to be queued.

Output Parameters

batchString **String** String representation of *batchObject* in ACH message format.

Usage Notes

Call this service when a batch has been created, all entries and addenda records have been added, and it is ready to be placed in the queue. The resulting string represents a part of the actual ACH message.

Examples

None

wm.ach.converter:convertDocToString

WmACH. Converts an ACH record table to a record string in ACH record format. You may optionally instruct the service to validate the record.

Input Parameters

recordName **String** Name of the ACH record.

recordTable **IData** The ACH record to be converted to string.

validate **String** A value of `true` means the service is to validate the record. A value of `false` means the service is to not validate the record.

Output Parameters

success **String** Results of the service. A value of `true` means the service terminated normally and a value of `false` means it terminated abnormally.

errors **String** Any errors produced during invocation of service.

recordString **String** Resulting ACH record format string.

Usage Notes

None

Examples

None

wm.ach.converter:convertStringToDoc

WmACH. Converts an ACH record format string to a record table containing key/value pairs. You may optionally instruct the service to validate the record.

Input Parameters

recordName	String Name of the ACH record.
recordString	String ACH record format string to be converted to key/value table.
validate	String A value of `true` means the service is to validate the record. A value of `false` means the service is to not validate the record.

Output Parameters

success	String Results of the service. A value of `true` means the service terminated normally and a value of `false` means it terminated abnormally.
errors	String Any errors produced during invocation of service.
recordString	String Resulting ACH record table.

Usage Notes

None

Examples

None

wm.ach.queue:generateACHFile

WmACH. Generates an ACH message file from queued batches.

Input Parameters

originId	String Origin for the ACH file, used in ACH file header block.
destinationId	String Destination for the ACH file, used in ACH file header block.
originName	String Origin name for the ACH file, used in ACH file header block.
destinationName	String Destination name for the ACH file, used in ACH file header block.
outputDirectory	String Path of folder where resulting ACH file should be created.
batchId	String[ ] Batch IDs for queued batches that need to be added to the generated ACH file.

Output Parameters

generatedFileName	String File name of the resulting ACH file.
success	String Results of the service. A value of `true` means the service terminated normally and a value of `false` means it terminated abnormally.
errors	String Any errors produced during invocation of service.

Usage Notes

- Call this service when a set of queued batches needs to be sent. The generated ACH file can then be sent to the ACH network.
- Batch IDs for queued batches can be obtained by using index fields and the wm.ach.queue:query service to query the batch queue.

See Also

wm.ach.queue:query

Examples

None

wm.ach.queue:getBatchFromQueue

WmACH. Returns a batch object from the queue that matches the specified batch ID (*key*). The batch object returned is read only. Entries cannot be added to this batch object, nor can this batch object be added back to the queue.

Input Parameters

key	String Batch ID for a queued batch.

Output Parameters

batchObject	String Batch object retrieved from the queue.
success	String Results of the service. A value of `true` means the service terminated normally and a value of `false` means it terminated abnormally.
errors	String Any errors produced during invocation of the service.

Usage Notes

key can be obtained by querying the batch queue based on index fields.

See Also

wm.ach.queue:query

Examples

None

wm.ach.queue:query

WmACH. Returns a set of batch keys from the batch queue that matches the specified criteria.

Input Parameters

fromDate	String The earliest effective entry date the search is to return. If you leave this parameter blank, the search will consider the beginning of the date range to be open-ended. The effective entry date is a field in the batch header record.
toDate	String The latest effective entry date the search is to return. If you leave this parameter blank, the search will consider the end of the date range to be open-ended. The effective entry date is a field in the batch header record.
msgType	String Type of the ACH message. The message type must be one of the SEC codes listed in Appendix B, "Standard Entry Class Codes", on page 67. If you leave this parameter blank, the search will return all message types except ADV.
senderId	String Origin of the ACH message. If you leave this parameter blank, the search will return all originator IDs.
receiverId	String Destination of the ACH message. If you leave this parameter blank, the search will return all receiver IDs.
additionalQueryFields	IData Additional keys used to narrow the search results. The keys must be one of the batch header fields. The query will consider an additional query field *only* if it was identified as an additional persistence query field at the time the batch was added to the batch queue. Separate keys with commas. See usage notes below.

Output Parameters

batchId	String[ ] Batch IDs for batches retrieved from the queue.
resultCount	String Number of batches matching the criteria.
errors	String Any errors produced during invocation of service.

Usage Notes

Before using the *additionalQueryFields* parameter for querying, it must be configured through the wm.ip.config:setConfig service or the ACH Current Configuration page. Any number of batch header fields

can be used. Queue services uses these additional index fields, along with the standard index fields (message type, message version, sender ID, receiver ID, and effective entry date), to create an index when queuing batches.

See Also

wm.ip.config:setConfig

Examples

None

wm.ach.queue:queueBatch

WmACH. Adds an ACH batch object to the batch queue under the ACH Module persistence store. Also creates an index for each batch based on header information, to improve searching.

Input Parameters

originId	String Origin for the ACH file, used in ACH file header block.
destinationId	String Destination for the ACH file, used in ACH file header block.
originName	String Origin name for the ACH file, used in ACH file header block.
destinationName	String Destination name for the ACH file, used in ACH file header block.
batchObject	Object Batch object created from the batch header information and ready to be queued.

Output Parameters

success	String Results of the service. A value of `true` means the service terminated normally and a value of `false` means it terminated abnormally.
errors	String Any errors produced during invocation of service.

Usage Notes

This service uses the persistence mechanism implemented in the WmIPPersist package. Users can choose to store persisted messages in the file system or implement their own custom Integration Server service to persist batches.

See Also

wm.ach.queue:query

Examples

None

wm.ach.tn.trp:receive

WmACHForTN. Sends an ACH message stream to Trading Networks two times—first as the complete message stream, then as the stream's component batches.

Specifically, the service does the following:

1. Receives an ACH message stream.
2. Sends the stream to Trading Networks as an "ACH file" document type.
3. Breaks the stream into separate batches.
4. Validates the batches.
5. Sends each validated batch to Trading Networks as a separate document type (identified by the SEC code of the batch).

Input Parameters

achStream Object ACH message stream.

Output Parameters

None

Usage Notes

Before you use this service, package WmACHForTN must be enabled, and appropriate document types for batches and processing rules must be defined in Trading Networks.

Sending the message stream to Trading Networks two ways gives the user the option as to how to view it—either as a complete file or as individual batches. To view individual batches, use the Trading Networks feature Transaction Analysis.

See Also

wm.ach.trp:receive

wm.ach.trp:receiveStream

Examples

None

wm.ach.tn.trp:send

WmACHForTN. Generates an ACH message file from queued batches based on the specified search criteria and sends the file to Trading Networks for persistence. The service then invokes the Integration Server service specified with the *sendSvc* parameter.

Input Parameters

fromDate	String Begin date range.
toDate	String End date range.
msgType	String Type of the ACH message. The message type must be one of the SEC codes listed in Appendix B, "Standard Entry Class Codes", on page 67.
senderId	String Origin of the ACH message.
receiverId	String Destination of the ACH message.
sendSvc	String Fully qualified name of the Integration Server service that sends the message file to the ACH operator.

Output Parameters

None

Usage Notes

Call this service when a set of queued batches needs to be sent to the ACH network. The ACH Module does not implement any transport service. The user must implement a transport service that expects the following as input:

- Object(InputStream) achStream

The fully qualified name of this service must be specified as the input parameter *sendSvc*.

sendSvc is invoked after submitting the ACH file document to Trading Networks. The user can also define a Trading Networks processing rule for the "ACH file" document type that implements a transport service.

To view a persisted file, use the Trading Networks feature Transaction Analysis.

See Also

wm.ach.queue:query

Examples

None

wm.ach.trp:receive

WmACH. Parses an ACH message file, breaking it into separate batches, and sends the batches to the appropriate routing engine after validation. The routing engine invokes the appropriate processing rule depending on the matching criteria defined in the routing rules. If package WmACHForTN is enabled, then processing rules defined in Trading Networks are used; otherwise, routing rules defined through the ACH routing page are used.

Input Parameters

fileName String File name of the received ACH file.

fileId String Internal ID of the parent document under Trading Networks. The ACH Module uses this parameter internally. Leave this parameter null.

Output Parameters

success String Results of the service. A value of `true` means the service terminated normally and a value of `false` means it terminated abnormally.

errors String Any errors produced during invocation of service.

Usage Notes

Routing rules/Trading Networks processing rules should be defined for batches before calling this service. If package WmACHForTN is enabled, call wm.ach.tn.trp:receive instead of this service.

See Also

wm.ach.trp:receiveStream

wm.ip.config:getConfig

Examples

None

wm.ach.trp:receiveStream

WmACH. Parses an ACH message stream, breaking it into separate batches, and sends the batches to the appropriate routing engine after validation. The routing engine invokes the appropriate processing rule depending upon the matching criteria defined in the routing rules. If package WmACHForTN is enabled, then processing rules defined in Trading Networks are used; else routing rules defined through the ACH routing page are used.

Input Parameters

inputStream Object ACH message stream.

fileId String Internal ID of the parent document under Trading Networks. The ACH Module uses this parameter internally. Leave this parameter null.

Output Parameters

success String Results of the service. A value of `true` means the service terminated normally and a value of `false` means it terminated abnormally.

errors String Any errors produced during invocation of service.

Usage Notes

Routing rules/Trading Networks processing rules should be defined for batches before calling this service. If package WmACHForTN is enabled, then wm.ach.tn.trp:receive should be called instead of this service.

See Also

wm.ip.config:getConfig

wm.ach.trp:receiveStream

Examples

None

wm.ach.validate:validateDocRecord

WmACH. Performs validations specified by ACH rules on an ACH record.

Input Parameters

recordName String Type of the ACH record to be validated.

recordTable IData The ACH record to be validated.

Output Parameters

success String Results of the service. A value of `true` means the service terminated normally and a value of `false` means it terminated abnormally.

errors String Any errors produced during invocation of service.

Usage Notes

None

Examples

None

wm.ach.validate:validateStringRecord

WmACH. Performs validations specified by ACH rules on an ACH record string.

Input Parameters

recordName String Name of the ACH record.

recordString String The ACH record string to be validated.

Output Parameters

success String Results of the service. A value of `true` means the service terminated normally and a value of `false` means it terminated abnormally.

errors String Any errors produced during invocation of service.

Usage Notes

None

Examples

None

wm.ip.config:getConfig

WmIPCommon. Obtains configuration information about an IP product (stored in a configuration file).

Input Parameters

productKey String Identifier representing the product. When using the ACH Module, the product key must be `ACH`.

Output Parameters

config IData[ ] Key/value pairs obtained from the configuration information.

Usage Notes

None

Examples

None

wm.ip.config:setConfig

WmIPCommon. Sets the configuration information for an IP product (stored in a configuration file).

Input Parameters

productKey **String** Identifier representing the product. When using the ACH Module, the product key must be `ACH`.

Output Parameters

None

Usage Notes

Key/value pairs should be set in the pipeline.

Examples

None

wm.ip.persist.store:get

WmIPPersist. Obtains the object representing the ACH Module persistence store for a product.

Input Parameters

productKey **String** Identifier representing the product. When using the ACH Module, the product key must be `ACH`.

Output Parameters

store **Object** The ACH Module persistence store used to persist messages.

Usage Notes

None

See Also

wm.ip.persist.store:register

Examples

None

wm.ip.persist.store:list

WmIPPersist. Obtains the list of product keys registered with the persistence module.

Input Parameters

None

Output Parameters

productKeyList String[ ] List of product keys.

Usage Notes

None

See Also

wm.ip.persist.store:register

Examples

None

wm.ip.persist.store:register

WmIPPersist. Registers a product with the persistence module.

Input Parameters

productKey String Identifier representing the product. When using the ACH Module, the product key must be `ACH`.

persistenceType String Possible values are `File` and `IS Service`. See usage notes below.

msgType String Possible values are `stream`, `byteArray`, and `string`.

folder String Folder where messages for this product will be stored.

Output Parameters

None

Usage Notes

If the persistence type is `File`, messages will be stored in the file system.

If the persistence type is `IS Service`, messages will be stored in a custom store (such as a database) that the Integration Server can access. You could then use the services of an adapter (such as the webMethods JDBC Adapter) to store messages in, query messages in, and remove messages from, the custom store.

See Also

wm.ip.persist.util:add

Examples

None

wm.ip.persist.util:add

WmIPPersist. Persists a message through the persistence module.

Input Parameters

productKey	String Identifier representing the product. When using the ACH Module, the product key must be `ACH`.
msgType	String Type of the message. When using the ACH Module, the message type must be one of the SEC codes listed in Appendix B, "Standard Entry Class Codes", on page 67. Used to create the index.
version	String Version of the message. Used to create the index. `1.0` is the only valid value.
senderId	String Origin of the message. Used to create the index.
receiverId	String Destination of the message. Used to create the index.
additionalQueryFields	IData Additional key/value pairs used to create the index. These pairs are later used to narrow down search results through the query service.
msg	String Message to be stored.

Output Parameters

msgId	String Unique key used to identify the stored message.

Usage Notes

None

See Also

wm.ip.persist.util:get

Examples

None

wm.ip.persist.util:get

WmIPPersist. Retrieves a message from the ACH Module persistence store.

Input Parameters

productKey **String** Identifier representing the product. When using the ACH Module, the product key must be `ACH`.

msgId **String** Unique key used to identify the stored message.

Output Parameters

msg **String** Retrieved message.

Usage Notes

None

See Also

wm.ip.persist.util:add

Examples

None

wm.ip.persist.util:query

WmIPPersist. Retrieves a list of messages from the ACH Module persistence store based on search criteria specified.

Input Parameters

productKey **String** Identifier representing the product. When using the ACH Module, the product key must be `ACH`.

msgType **String** Type of the ACH message. When using the ACH Module, the message type must be one of the SEC codes listed in Appendix B, "Standard Entry Class Codes", on page 67.

If you leave this parameter blank, the search will return all message types except ADV.

version **String** Version of the ACH message. `1.0` is the only valid value.

senderId **String** Origin of the ACH message. If you leave this parameter blank, the search will return all originator IDs.

receiverId **String** Destination of the ACH message. If you leave this parameter blank, the search will return all receiver IDs.

additionalQueryFields	IData Additional keys used to narrow the search results. The keys must be one of the batch header fields. The query will consider an additional query field *only* if it was identified as an additional persistence query field at the time the batch was added to the batch queue. Separate keys with commas. See usage notes below.
maxResults	String Maximum number of messages to be retrieved from the ACH Module persistence store.

Output Parameters

resultInfo	IData[ ] List of message headers containing message keys.
resultCount	String Number of messages retrieved matching the criteria.
resultMessage	String Message describing the nature of result.

Usage Notes

Before using the *additionalQueryFields* parameter for querying, it must be configured through the wm.ip.config:setConfig service or the ACH Current Configuration page. Any number of batch header fields can be used. Queue services uses these additional index fields, along with the standard index fields (message type, message version, sender ID, receiver ID, and effective entry date), to create an index when queuing batches.

See Also

wm.ip.persist.util:add

Examples

None

wm.ip.persist.util:updateStatus

WmIPPersist. Updates the status of a stored message.

Input Parameters

productKey	String Identifier representing the product. When using the ACH Module, the product key must be ACH.
msgId	String Unique key used to identify the stored message.
status	String Status of the message.

Output Parameters

None

Usage Notes

None

Examples

None

wm.ip.route:addRule

WmIPRoute. Adds a routing rule.

Input Parameters

productKey	String Identifier representing the product. When using the ACH Module, the product key must be `ACH`.
msgType	String Type of the ACH message. When using the ACH Module, the message type must be one of the SEC codes listed in Appendix B, "Standard Entry Class Codes", on page 67.
version	String Version of the ACH message. `1.0` is the only valid value.
senderId	String Routing/transmit number of the sender.
receiverId	String Routing/transmit number of the receiver.
status	String Status of the message.
isService	String Name of the Integration Server service to be invoked when a message matching the above criteria is received.

Output Parameters

None

Usage Notes

None

Examples

None

wm.ip.route:deleteRule

WmIPRoute. Deletes a routing rule from the list of rules.

Input Parameters

productKey String Identifier representing the product. When using the ACH Module, the product key must be ACH.

key String Position of the rule to be deleted.

Output Parameters

None

Usage Notes

None

Examples

None

wm.ip.route:getRules

WmIPRoute. Gets all routing rules defined for a product.

Input Parameters

productKey String Identifier representing the product. When using the ACH Module, the product key must be ACH.

Output Parameters

routingRules IData[ ] List of routing rules.

Usage Notes

None

Examples

None

wm.ip.route:list

WmIPRoute. Obtains a list of product keys registered with the routing module.

Input Parameters

None

Output Parameters

productKeyList String[ ] List of product keys.

Usage Notes

None

See Also

wm.ip.route:getRules

Examples

None

wm.ip.route:register

WmIPRoute. Registers a product with the routing module.

Input Parameters

productKey String Identifier representing the product. When using the ACH Module, the product key must be ACH.

folder String Folder where routing rules will be stored.

Output Parameters

None

Usage Notes

None

Examples

None

wm.ip.route:route

WmIPRoute. Invokes appropriate service with inputs specified, as defined by routing rules. Appropriate routing rule is obtained first by querying list of rules based on search criteria.

Input Parameters

productKey	String Identifier representing the product. When using the ACH Module, the product key must be `ACH`.
msgType	String Type of the message. When using the ACH Module, the message type must be one of the SEC codes listed in Appendix B, "Standard Entry Class Codes", on page 67.
version	String Version of the message. `1.0` is the only valid value.
senderId	String Origin of the message.
receiverId	String Destination of the message.
status	String Status of the message.
input	IData Input to the Integration Server service to be invoked.

Output Parameters

None

Usage Notes

None

Examples

None

APPENDIX A

webMethods ACH Module Sample

- Overview 64
- Before You Begin 64
- Running the Sample 64

Overview

The WmACHForTNSamples package demonstrates how to map a payment file to an ACH message file using Trading Networks. The package also helps illustrate how Trading Networks handles the two files.

Before You Begin

In this sample scenario, you will use the IS document types and services in the various folders of the packages that make up the webMethods ACH Module. This guide assumes that you are familiar with the Integration Server, Modeler, and Monitor.

Make sure that you have installed the webMethods ACH Module, including the WmACHForTNSamples package, on each Integration Server. For information about installing the webMethods ACH Module, see Chapter 2, "Installing the webMethods ACH Module" in this guide.

To set up Trading Networks to run the sample

1. From the Trading Networks Console select File ▸ Import.
2. Import the file *IntegrationServer_directory*\packages\WmACHForTNSamples\pub\setup\TNSampleSetup.dat.

Running the Sample

To run the sample, perform the following steps.

1. On the ACH Home page, click Payment Gateway Sample. The Submit Payment File page appears.
2. Click Browse and select the *IntegrationServer_directory*\packages\WmACHForTNSamples\pub\data\SamplePaymentFile.xml file.
3. After clicking Send, open the Trading Networks Console to view the results.

   Use the Trading Networks feature Transaction Analysis to view the file.

Trading Networks invokes the service wm.ach.tn.sample.maps.outbound:mapPaymentXMLToCCD to process the SamplePaymentFile.xml file. The service produces a formatted ACH message file, which is used as input to service wm.ach.tn.trp:receive.

To see how Trading Networks processes a file containing errors, run the service wm.ach.tn.sample:processACHFile from Developer, and specify an input file of *IntegrationServer_directory*\packages\WmACHForTNSamples\pub\data\achfile2.txt. Then use the Trading Networks feature Transaction Analysis to view the results; you will

see a status of `DONE WITH ERRORS`. To find more information about the errors, you can view the Activity Log.

webMethods

APPENDIX B

Standard Entry Class Codes

- Overview 68
- SEC Codes 68

Overview

The contents of this appendix were taken from the web site ACH Rules Online (http://www.achrulesonline.org).

The ACH Network supports a variety of payment applications. An Originator initiating entries into the system will code the entries in such a manner as to indicate the type of payment, such as a debit or credit, and whether an entry is consumer or corporate in nature (that is, the funds transfer affects either a consumer account or a corporate account at the RDFI). Each ACH application is identified and recognized by a specific three-digit code, known as a *Standard Entry Class* (SEC) code, which appears in the ACH record format. The SEC code identifies the specific computer record format that will be used to carry the payment and payment-related information relevant to the application.

This appendix presents a list of SEC codes and the different products each code supports.

SEC Codes

SEC codes are used in three type of applications—consumer, corporate, and other.

Consumer Applications

- ARC—Accounts Receivable Entry

  This Standard Entry Class Code enables Originators to convert to a Single Entry ACH debit a consumer check received via the U.S. mail or at a dropbox location for the payment of goods or services. The consumer's source document (that is, the check) is used to collect the consumer's routing number, account number, check serial number, and dollar amount for the transaction.

- CIE—Customer Initiated Entry

  Customer Initiated Entries are limited to credit applications where the consumer initiates the transfer of funds to a company for payment of funds owed to that company, typically through some type of home banking product or bill payment service provider.

- MTE—Machine Transfer Entry

  The ACH Network supports the clearing of transactions from Automated Teller Machines (that is, MTEs).

- PBR—Consumer Cross-Border Payment

  This Standard Entry Class Code is used for the transmission of consumer cross-border ACH credit and debit entries. This SEC Code allows cross-border payments to be readily identified so that financial institutions may apply special handling requirements for cross-border payments, as desired. The PBR format accommodates detailed information unique to cross-border payments (for example, foreign exchange conversion, origination and destination currency, and country codes).

- POP—Point-of-Purchase Entry

  This ACH debit application is used by Originators as a method of payment for the in-person purchase of goods or services by consumers. These Single Entry debit entries are initiated by the Originator based on a written authorization and account information drawn from the source document (a check) obtained from the consumer at the point-of-purchase. The source document, which is voided by the merchant and returned to the consumer at the point-of-purchase, is used to collect the consumer's routing number, account number, and check serial number that will be used to generate the debit entry to the consumer's account.

- PPD—Prearranged Payment and Deposit Entry

  - Direct Deposit

    Direct deposit is a credit application that transfers funds into a consumer's account at the Receiving Depository Financial Institution. The funds being deposited can represent a variety of products, such as payroll, interest, pension, dividends, etc.

  - Preauthorized Bill Payment

    Preauthorized payment is a debit application. Companies with billing operations may participate in the ACH through the electronic transfer (direct debit) of bill payment entries. Through standing authorizations, the consumer grants the company authority to initiate periodic charges to his or her account as bills become due. This concept has met with appreciable success in situations where the recurring bills are regular and do not vary in amount -- insurance premiums, mortgage payments, and installment loan payments being the most prominent examples. Standing authorizations have also been successful for bills where the amount does vary, such as utility payments.

- POS/SHR—Point of Sale Entry/Shared Network Transaction

  These two Standard Entry Class Codes represent point of sale debit applications in either a shared (SHR) or non-shared (POS) environment. These transactions are most often initiated by the consumer via a plastic access card.

- RCK—Re-presented Check Entry

  A Re-presented Check Entry is a Single Entry ACH debit application used by Originators to re-present a check that has been processed through the check collection system and returned because of insufficient or uncollected funds. This method of collection via the ACH Network, compared to the check collection process, provides Originators with the potential for improvements to processing efficiency (such as control over timing of the initiation of the debit entry) and decreased costs.

- TEL—Telephone-Initiated Entry

  This Standard Entry Class Code is used for the origination of a Single Entry debit transaction to a consumer's account pursuant to an oral authorization obtained from the consumer via the telephone. This type of transaction may only be originated when there is either (1) an existing relationship between the Originator and the Receiver, or (2) no existing relationship between the Originator and the Receiver, but the Receiver has initiated the telephone call. This SEC Code facilitates access to the ACH Network by providing an alternative authorization method, oral authorization via the telephone, for certain types of consumer debit entries.

- WEB—Internet-Initiated Entry

  This Standard Entry Class Code is used for the origination of debit entries (either recurring or Single Entry) to a consumer's account pursuant to an authorization that is obtained from the Receiver via the Internet. This SEC Code helps to address unique risk issues inherent to the Internet payment environment through requirements for added security procedures and obligations.

Corporate Applications

- CBR—Corporate Cross-Border Payment

  This Standard Entry Class Code is used for the transmission of corporate cross-border ACH credit and debit entries. This SEC Code allows cross-border payments to be readily identified so that financial institutions may apply special handling requirements for cross-border payments, as desired. The CBR format accommodates detailed information unique to cross-border payments (for example, foreign exchange conversion, origination and destination currency, and country codes).

- CCD—Cash Concentration or Disbursement

  This application, Cash Concentration or Disbursement, can be either a credit or debit application where funds are either distributed or consolidated between corporate entities. This application can serve as a stand-alone funds transfer, or it can support a limited amount of payment related data with the funds transfer.

- CTX—Corporate Trade Exchange

  The Corporate Trade Exchange application supports the transfer of funds (debit or credit) within a trading partner relationship in which a full ANSI ASC X12 message or payment related UN/EDIFACT information is sent with the funds transfer. The ANSI ASC X12 message or payment related UN/EDIFACT information is placed in multiple addenda records.

Other Applications

- ACK/ATX—Acknowledgment Entries

  These optional Standard Entry Class Codes are available for use by the RDFI to acknowledge the receipt of ACH credit payments originated using the CCD or CTX formats. These acknowledgments indicate to the Originator that the payment was received and that the RDFI will attempt to post the payment to the Receiver's account. Acknowledgment entries initiated in response to a CCD credit entry utilize the ACK format. Acknowledgments initiated in response to a CTX credit entry utilize the ATX format.

- ADV—Automated Accounting Advice

  This Standard Entry Class Code represents an optional service to be provided by ACH Operators that identifies automated accounting advices of ACH accounting information in machine-readable format to facilitate the automation of accounting information for Participating DFIs.

- COR—Automated Notification of Change or Refused Notification of Change

  This Standard Entry Class Code is used by an RDFI or ODFI when originating a Notification of Change or Refused Notification of Change in automated format. It is also used by the ACH Operator that converts paper Notifications of Change to automated format.

- DNE—Death Notification Entry

  This application is utilized by a Federal Government agency (for example, the Social Security Administration) to notify a depository financial institution that the recipient of a government benefit payment has died.

- ENR—Automated Enrollment Entry

  This optional SEC Code allows a depository financial institution to transmit ACH enrollment information to Federal Government Agencies via the ACH Network for future credit and debit applications on behalf of both consumers and companies.

- TRC/TRX—Truncated Entries

  This Standard Entry Class Code is used to identify batches of truncated checks. For more information on check truncation, please see the National Association for Check Safekeeping Guidelines available from NACHA.

- XCK—Destroyed Check Entry

  This application can be utilized by a collecting institution for the collection of certain checks when those checks have been destroyed.

Index

A

ACH file
- creating 31
- sending 31

ACH Home page
- introduction 22
- routing configuration 26

ACH Module
- architecture 14
- configuring 22
- introduction 11
- overview 14

addenda records 11

ADV (SEC Code)
- defined 71
- setting maximum per batch 23

architecture 14

B

batch containing entries, creating 30

C

codes, Standard Entry Class (SEC) 67

configuring
- ACH Module 22
- custom ACH Module persistence store 24

conventions used in this document 7

creating
- ACH file 31
- batch containing entries 30

custom ACH Module persistence store, configuring 24

D

documentation
- additional 8
- conventions used 7
- feedback 8

E

enabling WmACHForTN package 32

examples 64

F

files
- SamplePaymentFile.xml 64
- TNSampleSetup.dat 64

H

hardware requirements 19

home page 22

I

index fields
- specifying your own 24
- standard 24

installation
- ACH Module 19
- requirements 18

M

managing the ACH Module 21

messages
- receiving with Trading Networks 35
- receiving without Trading Networks 31
- sending with Trading Networks 34
- sending without Trading Networks 30

O

operating systems supported 18

P

packages
- in ACH Module architecture 15

WmACH
introduction 15
reloading 24
WmACHForTN
and routing rules 26
disabling to run ACH Module without Trading Networks 30
enabling 32
enabling to run ACH Module with Trading Networks 31
introduction 15
persisting messages in Trading Networks 16
WmACHForTNSamples
illustrative examples 64
introduction 15
WmIPCommon, introduction 15
WmIPPersist
introduction 15
specifications in 24
WmIPRoute, introduction 15
WmTN 32
persistence store (ACH Module)
see also "persistence store (Trading Networks)"
custom configuration 24
in ACH Module architecture 16
specifying location
through UI 23
with service 54
specifying type
through UI 23
with service 54
viewing messages in
through UI 25
with service 56
persistence store (Trading Networks)
see also "persistence store (ACH Module)"
introduction 31
platforms supported 18
program code conventions in this document 7

R

receiving messages
with Trading Networks 35
without Trading Networks 31
reloading WmACH package 24
requirements
hardware 19
running the sample 64
software 19
webMethods components 18
routing rules
and the WmACHForTN package 26
and Trading Networks 26
creating
through UI 26
with service 58
deleting
through UI 27
with service 59
editing 27

S

SamplePaymentFile.xml file 64
samples 64
SEC (Standard Entry Class) codes 67
sending ACH file 31
sending messages
with Trading Networks 34
without Trading Networks 30
services
wm.ach.batch:appendEntry 12, 30, 35, 40
wm.ach.batch:createBatch 30, 34, 41
wm.ach.batch:getBatchControl 41
wm.ach.batch:getBatchHeader 42
wm.ach.batch:getNextEntry 14, 31, 36, 42
wm.ach.batch:toString 43
wm.ach.converter:convertDocToString 43
wm.ach.converter:convertStringToDoc 44
wm.ach.queue:generateACHFile 13, 31, 44
wm.ach.queue:getBatchFromQueue 45
wm.ach.queue:query 31, 46
wm.ach.queue:queueBatch 30, 35, 47
wm.ach.tn.sample.maps.outbound:mapPaymentXMLToCCD 30, 35
wm.ach.tn.trp:receive 13, 35, 36, 48
wm.ach.tn.trp:send 13, 35, 49
wm.ach.trp:receive 13, 31, 50
wm.ach.trp:receiveStream 50

wm.ach.validate:validateDocRecord 12, 51
wm.ach.validate:validateStringRecord 12, 52
wm.ip.config:getConfig 52
wm.ip.config:setConfig 53
wm.ip.persist.spec:add 24
wm.ip.persist.spec:get 24
wm.ip.persist.spec:query 24
wm.ip.persist.spec:remove 24
wm.ip.persist.spec:updateStatus 24
wm.ip.persist.store:get 53
wm.ip.persist.store:list 54
wm.ip.persist.store:register 54
wm.ip.persist.util:add 55
wm.ip.persist.util:get 56
wm.ip.persist.util:query 56
wm.ip.persist.util:updateStatus 57
wm.ip.route:addRule 58
wm.ip.route:deleteRule 59
wm.ip.route:getRules 59
wm.ip.route:list 60
wm.ip.route:register 60
wm.ip.route:route 61
wm.tn.dictionary:addIDType 32
software requirements 19
Standard Entry Class (SEC) codes 67
supported
operating systems 18
platforms 18

T

TNSampleSetup.dat file 64
Trading Networks
in ACH Module architecture 16
persistence store 31
receiving messages with
through UI 35
with service 48
required component version numbers 18
routing rules 26
sending messages to
with service 34, 49
setting up to use 32
troubleshooting information 8
typographical conventions in this document 7

U

uninstallation
ACH module 20

W

webMethods components, required 18
wm.ach.batch:appendEntry 12
wm.ach.batch:appendEntry service 30, 35, 40
wm.ach.batch:createBatch service 30, 34, 41
wm.ach.batch:getBatchControl service 41
wm.ach.batch:getBatchHeader service 42
wm.ach.batch:getNextEntry 14
wm.ach.batch:getNextEntry service 31, 36, 42
wm.ach.batch:toString service 43
wm.ach.converter:convertDocToString service 43
wm.ach.converter:convertStringToDoc service 44
wm.ach.queue:generateACHFile 13
wm.ach.queue:generateACHFile service 31, 44
wm.ach.queue:getBatchFromQueue service 45
wm.ach.queue:query service 31, 46
wm.ach.queue:queueBatch service 30, 35, 47
wm.ach.tn.sample.maps.outbound:mapPaymentXMLToCCD service 30, 35
wm.ach.tn.trp:receive 13
wm.ach.tn.trp:receive service 35, 36, 48
wm.ach.tn.trp:send 13
wm.ach.tn.trp:send service 35, 49
wm.ach.trp:receive 13
wm.ach.trp:receive service 31, 50
wm.ach.trp:receiveStream service 50
wm.ach.validate:validateDocRecord 12
wm.ach.validate:validateDocRecord service 51
wm.ach.validate:validateStringRecord 12
wm.ach.validate:validateStringRecord service 52
wm.ip.config:getConfig service 52
wm.ip.config:setConfig service 53
wm.ip.persist.spec:add service 24
wm.ip.persist.spec:get service 24
wm.ip.persist.spec:query service 24
wm.ip.persist.spec:remove service 24
wm.ip.persist.spec:updateStatus service 24

wm.ip.persist.store:get service 53
wm.ip.persist.store:list service 54
wm.ip.persist.store:register service 54
wm.ip.persist.util:add service 55
wm.ip.persist.util:get service 56
wm.ip.persist.util:query service 56
wm.ip.persist.util:updateStatus service 57
wm.ip.route:addRule service 58
wm.ip.route:deleteRule service 59
wm.ip.route:getRules service 59
wm.ip.route:list service 60
wm.ip.route:register service 60
wm.ip.route:route service 61
wm.tn.dictionary:addIDType service 32
WmACH package
  introduction 15
  reloading 24
WmACHForTN package
  and routing rules 26
  disabling to run ACH Module without Trading Networks 30
  enabling 32
  enabling to run ACH Module with Trading Networks 31
  introduction 15
  persisting messages in Trading Networks 16
WmACHForTNSamples package
  illustrative examples 64
  introduction 15
WmIPCommon package, introduction 15
WmIPPersist package
  introduction 15
  specifications in 24
WmIPRoute package, introduction 15
WmTN package 32