AEMO
AUSTRALIAN ENERGY MARKET OPERATOR

Guide to MSATS Participant Batcher Software

3.03 Final

January 2019

Covers the setup and use of the MSATS Participant Batcher software

Important Notice

NO RELIANCE OR WARRANTY

This document does not constitute legal or business advice, and should not be relied on as a substitute for obtaining detailed advice about the National Gas or Electricity Law, the Rules or any other applicable laws, procedures or policies. While AEMO has made every effort to ensure the quality of the information in this Guide, neither AEMO, nor any of its employees, agents and consultants make any representation or warranty as to the accuracy, reliability, completeness, currency or suitability for particular purposes of that information.

LIMITATION OF LIABILITY

To the maximum extent permitted by law, AEMO and its advisers, consultants and other contributors to this Guide (or their respective associated companies, businesses, partners, directors, officers or employees) are not liable (whether by reason of negligence or otherwise) for any errors, omissions, defects or misrepresentations in this document, or for any loss or damage suffered by persons who use or rely on the information in it.
AEMO has prepared this Guide to MSATS Participant Batcher Software (Guide) to provide guidance on the use of the MSATS Participant Batcher Software under the National Gas or Electricity Rules (Rules), as at the date of publication.

TRADEMARK NOTICES

Microsoft is a trademark of Microsoft Corporation in the United States and/or other countries.
Oracle and Java are registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

DOCUMENTS MADE OBSOLETE

The release of this document changes any version of the Guide to Participant Batcher Software and earlier versions of Guide to MSATS Participant Batcher Software.

DISTRIBUTION

Available to the public.

DOCUMENT IDENTIFICATION

Business custodian Manager, Metering
IT Custodian Lenard Bull
Prepared by: Technical Writers, Market Systems
Last update: 1/02/2019 11:29 AM

VERSION HISTORY

Version 3.02. Updated for the Power of Choice project and the *B2B e-Hub*.
Version 3.03 Fixed broken link for AEMO site

FURTHER INFORMATION

AEMO's Support Hub
Phone: 1300 AEMO 00 (1300 236 600)

FEEDBACK

To suggest corrections or improvements to this document, please contact AEMO's Support Hub.

Contents

Introduction

Purpose

This guide explains how to use the MSATS Participant Batcher Software. It describes the interface and how to set up your preferences.

Audience

This guide is intended for registered participants' technical and software development staff, responsible for IT systems implementation.

How to use this guide

- This document is written in plain language for easy reading.
- Where there is a discrepancy between the Rules, and information or a term in this document, the Rules take precedence.
- Text in this format indicates a resource on AEMO's website.
- Text in this format indicates a direct link to a section in this guide.
- Glossary terms are capitalised and have the meanings listed against them in the Glossary on page 1.
- *Italicised terms* are defined in the National Electricity Rules (NER). Any rules terms not in this format still have the same meaning.
- Actions to complete in the web portal interface are **bold and dark grey**.

Assumed Knowledge

This guide assumes you have knowledge of:

- The operating system your company is using.
- The Java application environment.
- How the NEM systems operate from an external perspective.

What's in this guide

About MSATS Participant Batcher Software on page 3 provides an overview of the MSATS Participant Batcher Software, who can use it, how to use it, the software requirements and where to obtain it.

MSATS Batch File Interface on page 10 explains the MSATS file server and participant folders.

B2B Batch File Interface on page 13 explains the B2B file server and participant folders.

Quick Start Guide on page 16 provides the basic steps to install, configure, and run the Participant Batcher software.

Setup on page 17 explains in detail how to set up and configure the Participant Batcher software.

Initiation on page 38 explains how to run the Participant Batcher sample applications and convert them to production.

Maintenance on page 41 explains how to maintain the Participant Batcher software.

Needing Help? provides information about contacting AEMO's Support Hub, how to provide feedback, and related resources.

About MSATS Participant Batcher Software

What the software is for

The Participant Batcher software (Batcher) provides a simple batch interface to MSATS and B2B by removing the detail of the file handshaking and leaving participants to deal with the raw .zip files only.

The Participant Batcher software transfers files using FTP from and to the MSATS and B2B participant file shares. The Participant Batcher software does the entire message acknowledgement and file manipulations as required by the batch file interfaces.

Who can use the software

For more details, see MSATS Batch File Interface and B2B Batch File Interface.

The intended users for the Participant Batcher Software are:

- Participants having a Participant ID to access AEMO's IT systems.
- Participants in the retail market interfacing with the MSATS system.
- *B2B Hub participants* interfacing with the *B2B e-Hub*, including B2B users interfacing with the AEMO Network Outage Scheduler (NOS) and the High Speed Monitoring (HSM) system.

How do you use the software?

The Participant Batcher software is a “state” machine and the local file folders control the state. This means if the process stops for some reason, the Participant Batcher software automatically recovers using the stored state in the local folders. Each of the local folders has a series of subfolders (see File states) containing the final results of the file transfers, and a process in the application periodically moves files to the appropriate subdirectory.

For reception of data, participants receive files in the participantlocal outbox subdirectory .zip. After the Batcher successfully sends a file, it moves the file from the participantlocal inbox to the DONE subfolder.

The Batcher software is a batch application and so does not include a graphical user interface. All configuration of the Batcher is done in the .properties file. For more details, see Configuring the properties file.

Recommendations

The design of the Participant Batcher software assumes access to the local folders is highly reliable and efficient. So, the AEMO recommend setting up the structure of the application with the local folders hosted on the machine running the application. If space is a problem, the recommendation is to use a custom clean-up process to empty the various local subfolders by move. Participant Batcher has a feature to move files elsewhere, Configuring the properties file.

A change to the .properties file requires an application restart, as the .properties file is only read at application startup.

Structure

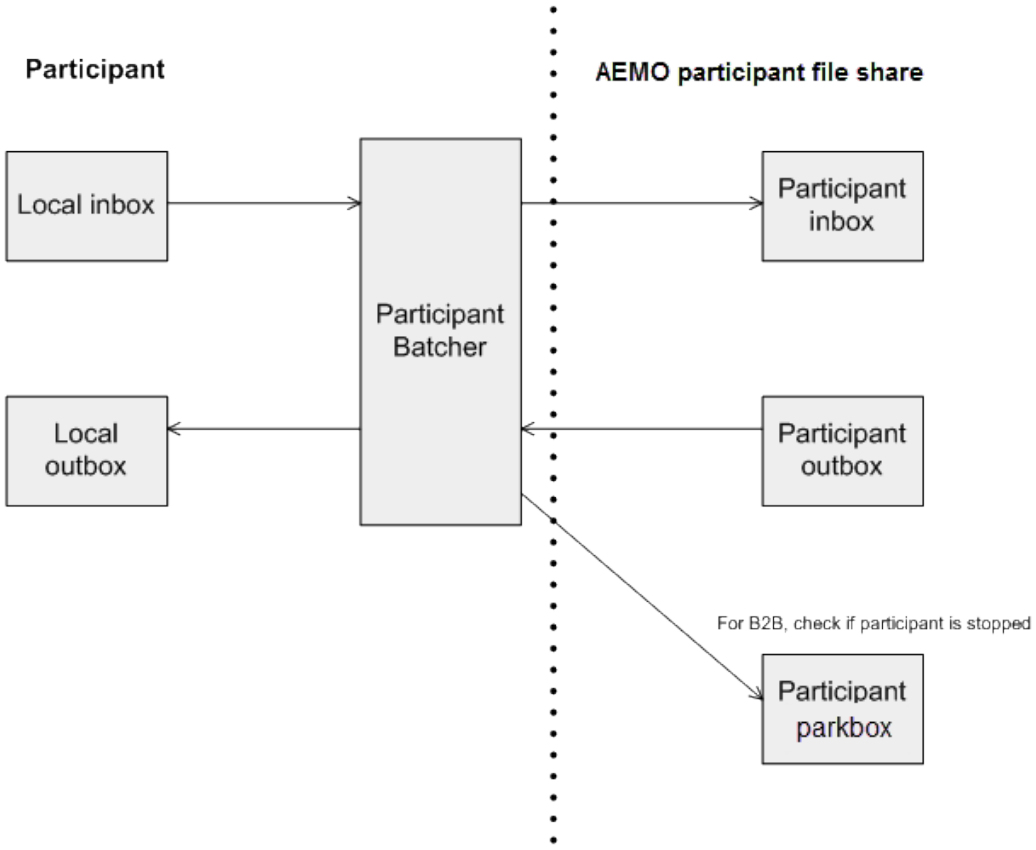
On the participant side, two folders are required:

- **Participant local inbox (pli)**: Where participants put .zip files for transferring to AEMO.
- **Participant local outbox (plo)**: Where the Batcher puts .zip and acknowledgement files to signal acceptance of data.

On AEMO's side, in the participant file share, three folders exist:

- **Participant Inbox**: Where the Batcher puts .zip files as input to AEMO and puts acknowledgement files to signal acceptance of data.
- **Participant Outbox**: Where AEMO puts .zip files containing data for Participant Batcher copying, and puts acknowledgement files to signal acceptance of data.
- **Participant Parkbox**: Where the *B2B e-Hub* indicates if a B2B participant is stopped – this application monitors the location before sending B2B files.

Figure 1 Participant Batcher interface with AEMO and participant systems



File states

The state transition changes separately for each file and the file extension records each state. The sending state is a file extension in the participantlocal inbox.

For more details, see:

- Table 1 explaining the Batcher sending states.
- Table 2 detailing the AEMO generated ACK file states temporarily stored in the participantlocal inbox.
- Table 3 detailing the receiving states which are a file extension in the participantlocal outbox.
- Table 4 explains the state when the file transfer is complete that is, the AEMO participant file share has no ACK or .zip files left.
- Table 5 explains the state when the files at the participant's end are transferred to the subfolders and renamed either .ACK or .zip as appropriate.

Table 1 Participant Batcher sending states

TMP	Your system is copying the file
.zip	file is ready to send
SENDINGFTP	transfer is in process to AEMO
SENTFTP	transfer succeeded
DONEAEMO	system acknowledged the file positively
NACKAEMO	system acknowledged the file negatively
RESENDAEMO	system acknowledged the file negatively with special code 111

Table 2 AEMO generated ACK file states

ACK	file from AEMO
ACK_DONE ACK	file after validation indicates that the .zip file was accepted
ACK_NACK ACK	file after validation indicates that the .zip file was rejected
ACK_RESEND ACK	file after validation indicates the .zip file was rejected with the special code 111

Table 3 Participant Batcher receiving states

GETTINGFTP	transfer from AEMO is processing
GOTFTP	transfer succeeded
ZIPvalidation	.zip validation of the XML schema was positive
NACKvalidation	NACK validation of the XML schema was negative

Table 4 Participantlocal inbox subdirectory contents

Transfer State	Done	Nack	Resend	Timeout
Success	File.ACK and Filezip			
Negative Acknowledge		File.ACK and Filezip		
Negative Ack (111)			File.RESEND	
Resend Timeout				File.TIMEOUT

Table 5 Participantlocal outbox subdirectory contents

Transfer State	Zip	Nack
Success	Filezip	
Negative Acknowledge		Filezip

Software requirements

The Participant Batcher software runs under the Java Platform, Standard Edition 8 (Java SE 8). Participants require the Java Runtime Environment 8 (Java JRE 8) available from Oracle Downloads.

The application can run on both Windows and Unix-like operating systems. As supplied, the software is compiled to run on 32-bit environments. AEMO has not tested for 64-bit environment operation.

Accessing the file server

1. FTP to one either pre-production or production:
   - MSATS production: ftp://146.178.211.205
   - MSATS pre-production: ftp://146.178.211.225
2. Sign in using the user ID and password provided by your company's Participant Administrator.

File naming convention

For more details, see Connecting to AEMO's Electricity IT Systems.

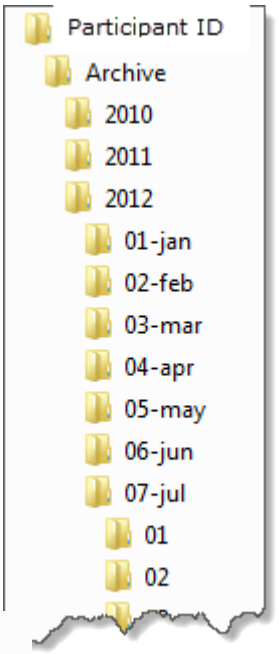
```
[0-9_a-z]{1,4}[hml][0-9_a-z]{1,30}[.](tmp|zip|ack|ac1)
```

The file naming convention used in MSATS and B2B is defined by the following regular expression:

- The first four characters represent the MSATS or B2B transaction group. For help, see MSATS transaction groups and B2B transaction group .
- The fifth character represents the priority, either: high (h), medium (m) or low (l).
- The remaining 30 characters ensure the message file has a permanently unique identifier.

Only use lower case characters. The batch handlers recognise and process incoming files by their four character transaction group. An invalid transaction group in the file name can cause the .zip file to be ignored.

Archive folders



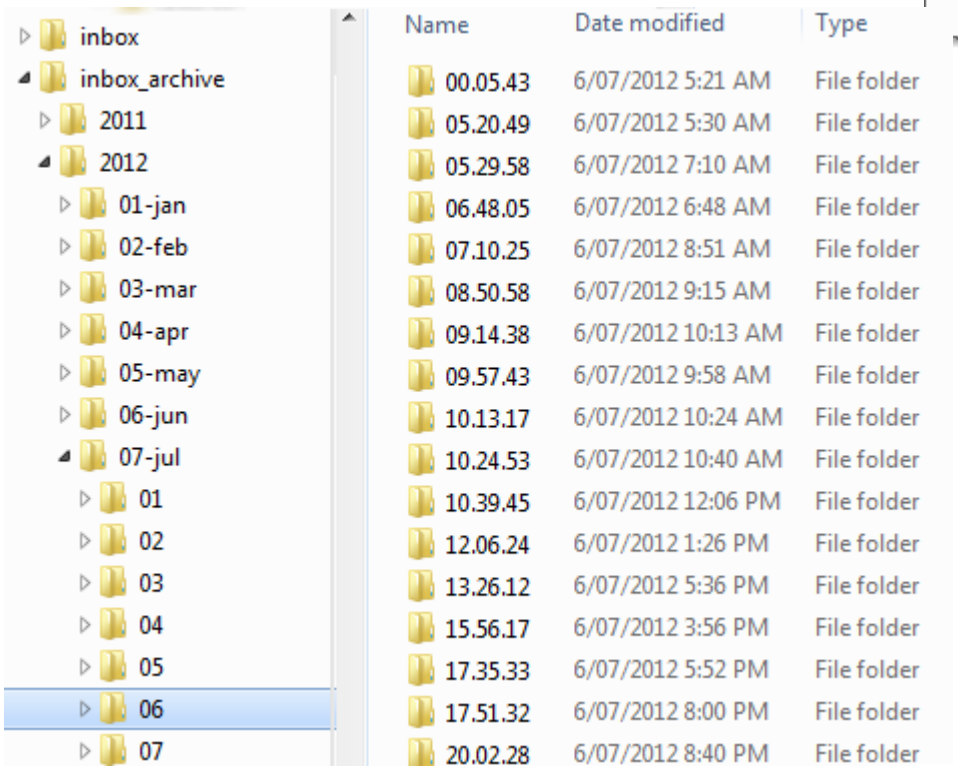
The archive folders are read-only and preserved for around13 months.

Each folder contains folders for different years, each yearly folder has folders for different months, and each monthly folder has folders for different days.

As part of the process, when a participant acknowledges a .zip file, MSATS moves the file from either the:

- Outbox into the Archive folder.
- Inbox into the Inbox Archive folder.

The folder is named hh.mm.ss for the time of day when the folder was created.



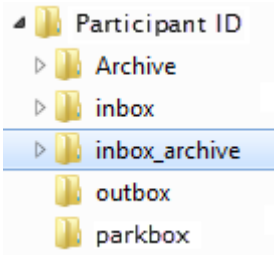
MSATS Batch File Interface

About the MSATS batch file interface

Every participant ID has a folder available for its exclusive use on the MSATS participant file server.

Some participants have variations of the folder structure, generally made by special arrangement to handle operational circumstances.

Each participant folder contains the following default folders:



Folder	Participant	Hub	Description
Archive	read only	write once	Contains files from outbox that have been acknowledged, by date and time in a nested set of folders.
inbox	writeable	read only	Participant delivers files intended for MSATS, including acknowledgements.
inbox-archive	read only	write once	Contains files from inbox that have been acknowledged, by date and time in a nested set of folders.
outbox	read only	writeable	MSATS delivers files meant for a participant, including acknowledgements.
parkbox	read only	writeable	Used when transitioning from one aseXML schema to another. For more details, see the Guide to Transition of aseXML.

MSATS file exchange protocol

The MSATS file exchange protocol relates to the inbox and outbox folders. Given the asynchronous nature of a file based interface, the MSATS file exchange protocol is designed to ensure that a file is only deleted once the receiver has processed it. The protocol is symmetrical in that the same rules and steps apply equally to the participants and the hub (the MSATS batch handlers). This is referred to as the “Hokey Pokey” protocol. The MSATS file exchange protocol is part of the file handling, and is quite distinct from the business processing of messages and transactions.

The writeable folder for a participant is known as the inbox. The writeable folder for the MSATS hub is known as the outbox. Both parties treat the other party's writeable folder as being read-only.

The protocol has six steps:

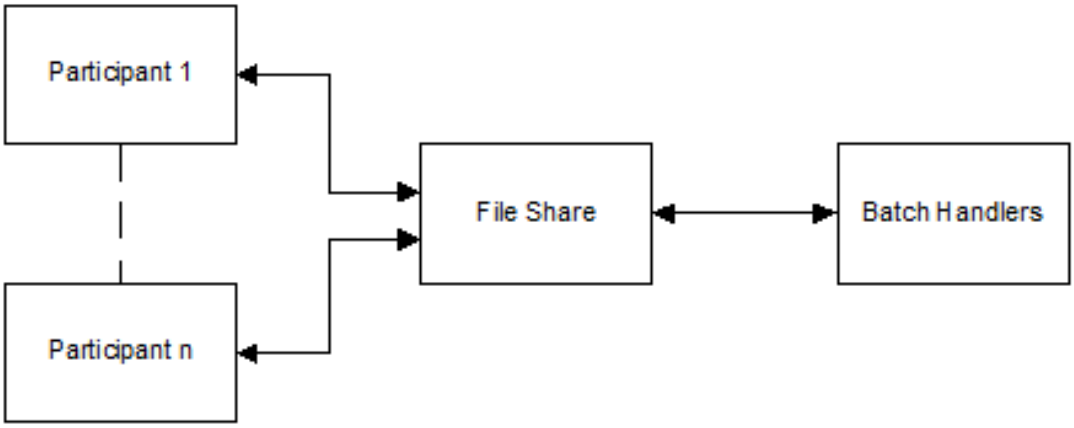
1. The sender generates a compressed content file (zip format) in their writeable folder with the extension of .tmp.
2. The sender renames the .tmp file to an extension of .zip.
3. The receiver detects the .zip file and processes it, then produces an acknowledgement content file in their writeable folder of the same name with the extension of .tmp.
4. The receiver renames the acknowledgement file to an extension of .ack.
5. The sender detects the acknowledgement file and deletes their original file with the extension of .zip.
6. The receiver detects the deletion of the original .zip file and deletes the corresponding acknowledgement file.

Notes:

- The detection of file deletions may be achieved by comparison of inbox and outbox contents by the correlation of file names.
- Files with the extension of .tmp are always ignored (to prevent attempting to process a partial file). The file renaming is assumed to be an atomic operation for the file system. The rename signifies that the file has completed being written and is available to be read.
- Acknowledgement files are not acknowledged.

Most aseXML messages carrying transactions are carried in the file with the extension of .zip, and aseXML message acknowledgements are carried in the file with the extension of .ack.

Figure 2 Participant interaction using the MSATS batch interface



MSATS transaction groups

For MSATS, the inbound handler processes participant's files on the basis of transaction group:

- **CATS** for CATS transactions.
- **MDMT** for MDM transactions.
- **NMID** for NMI discovery transactions.

B2B Batch File Interface

About the B2B batch file interface

B2B messages can use the same folders as MSATS messages since use the same naming About the B2B batch file interface. Some participants have a separate B2B folder containing inbox, outbox and parkbox just for B2B files. Archiving is into the same folder structure as used for MSATS, see Archive folders.

B2B delivery methods

For B2B, participants can choose from two delivery methods:

1. API protocol
2. FTP protocol

> Participants can set different protocols for different Transaction Groups but the protocol must be the same within a Transaction Group. For example, for a SORD transaction you cannot send in FTP and receive in API.

API protocol

Uses the API Gateway and web services, accessible over the internet or MarketNet.

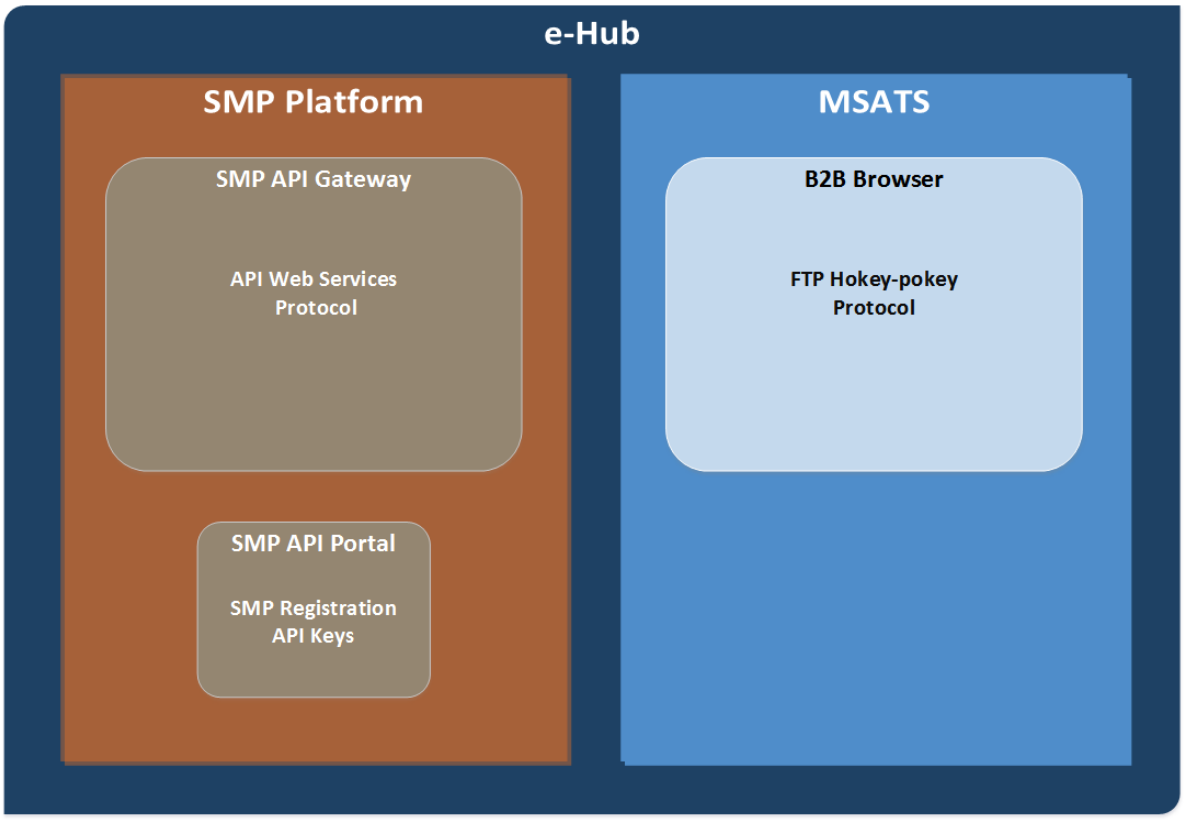
For more details, see Shared Market Protocol (SMP) Technical Guide and the National Electricity Rules.

FTP protocol

The FTP protocol is similar to the MSATS file exchange protocol, Hokey-Pokey, see MSATS file exchange protocol.

The key difference is that three parties are involved in the transfer, and the e-hub sends a transport-level acknowledgement. In normal operation, each transmission of a transaction message gets two transport-level acknowledgements; one from the hub (ac1) and the second from the receiver of the message (.ack).

Figure 3 B2B e-Hub



B2B transaction group

The B2B inbound handler recognises participant's files by the transaction group in the header.

Examples are:

CUST	Customer related
MTRD	Meter data
MRSR	Remote services request/response
NPNX	Notified party transaction
OWNP	One-Way Notification Process

OWNX	Notice of works (NOMW), PIN, and MFN
PTPE	Peer-2-Peer bilateral exchange(API gateway only)
SITE	Site access
SORD	Service orders, planned works

For more details about transaction groups, see the Guide to B2B and the **B2B Procedures**, especially the Technical Delivery Specification.

Quick Start Guide

The basic steps to install, configure, and run the Participant Batcher software are:

1. Download the latest version of the ParticipantBatcherVn.n.zip file from AEMO's website and extract to a folder such as C:/pb. For help, seeSetup.
2. Configure the .properties file and place in the classes subdirectory. For help, see Configuring the properties file.

   If the file is unzipped to the C:\pb subdirectory on a Windows system, the sample application started by executing sample.bat works with just the editing of the ftpservername, username, password, log4j log file name, pninbox, pnoutbox, plinbox and ploutbox items in the sample.properties file. Exercise care to ensure there is no effect on production operations caused by running this application as a test.

4. Create Participant Batcher instances. For help, see Creating Instances.
5. Run the Participant Batcher. For help, see Initiation.
6. Perform regular maintenance to keep the Participant Batcher software running smoothly. For help, see Maintenance.

A change to the .properties file requires an application restart, as the .properties file is only read at application startup.

Setup

Features and recommendations

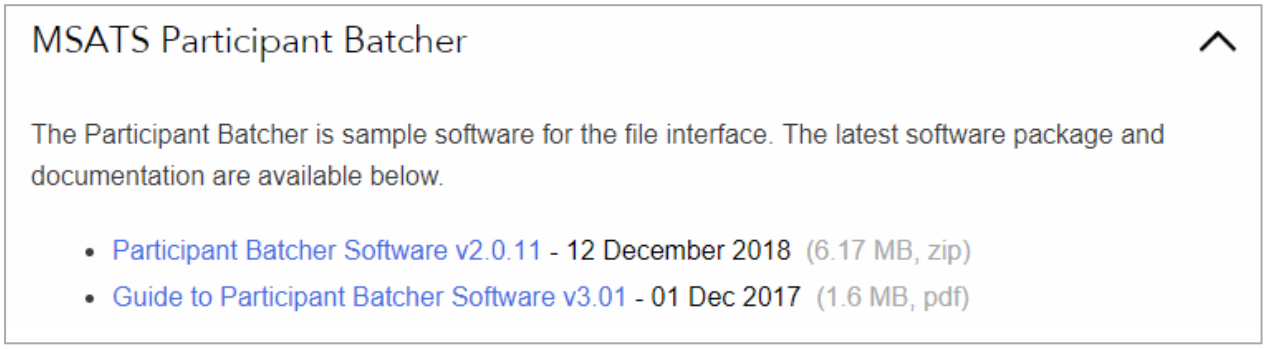
The installation, configuration, and initiation process depends on the issues you face. AEMO recommends the following:

- The Participant Batcher software handles both the “Participant Sending MSATS files (forward) process” and the “Participant Receiving MSATS files (backward) process”.
- To simplify the application management in a production environment, and to improve the application’s reliability, the Participant Batcher software handles only one inbox and one outbox. In other words, for each participant username and password, initiate a separate instance of this application. Multiple instances for different NEM security identities can be run concurrently.
- To handle file priority more efficiently, you can specify file masks to limit the files a particular instance of the Participant Batcher software can process. You can initiate multiple instances of this application for different file masks if you can justify such a configuration.

Properties configuration is the key to ensuring multiple instances of the Participant Batcher software run concurrently without conflicts or omissions. Therefore, avoiding any two .properties files having the same combined values of the above recommendations is important.

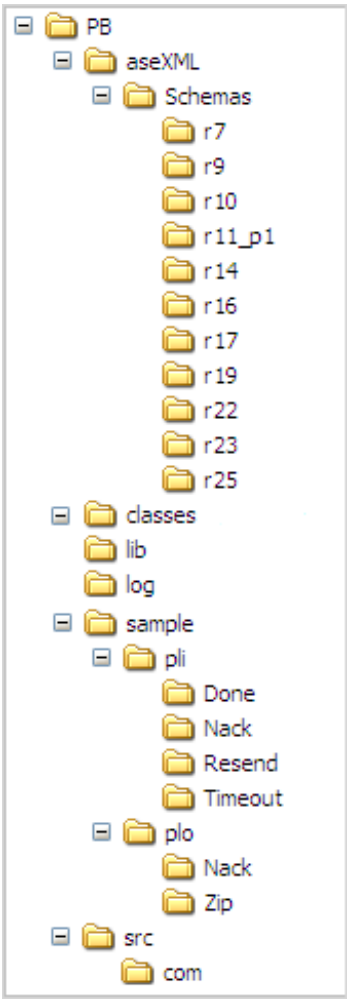
Downloading the application

The latest version of the application is in a single ..zip file available from **AEMO's website > Market Settlement and Transfer Solutions.** It looks similar to the following:



Extracting the distribution file

1. Extract the ParticipantBatcherVnn.zip file into a base folder. For example C:\PB retaining the folder structure recorded in the distribution file.



2. Assuming you decompressed it to C:\PB, the extraction creates the folder structure in the image opposite.

   Contained in each folder are the subfolders and files described in Table 6.

Table 6 Folder and subfolder structure

Folder	Contents
PB	README.txt
PB\aseXML\Schemas	aseXML schema folder structure
PB	All files required to run the application, arranged in the correct folder structure. This includes files that have not changed since the last build. All Java classes are bundled into a single file called ParticipantBatcher.jar. This jar file is signed and the values of parameters used in the build process are stored in the jar's manifest. pbEnvironment.bat sample.properties pbServiceInstall.bat pbServiceUninstall.bat crypt.bat stop.bat sample.bat
PB\classes	pb_key.properties files Place your created .properties files in this folder.
PB\lib	all.jar files: AbsoluteLayout.jar commons-codec-1.4.jar commons-httpclient.jar commons-logging-1.1.1.jar commons-net-2.0.jar edtftpj-1.5.2.jar jakarta-slide-commandline-2.1.jar jakarta-slide-webdavlib-2.1.jar JavaService.exe log4j-1.2.15.jar ParticipantBatcher.jar resolver.jar serializer.jar xercesImpl.jar

Folder	Contents
	xml-apis.jar
PB\log	Log and monitor files
PB\sample\pli	Participant local inbox where .zip files are placed for transferring to AEMO.
PB\sample\pli\done	Successfully sent and acknowledged .zip files plus the .ACK file.
PB\sample\pli\nack	Negatively acknowledged .zip files, plus the .ACK file
PB\sample\pli\resend	If negatively acknowledged with the 111 code, the file is placed here before it is retried.
PB\sample\pli\timeout	When sending a resend file has not succeeded after retries, it is moved here.
PB\plo	.zip files received are manipulated in this subdirectory.
PB\plo\nack	When a received .zip file fails XML validation it is moved here.
PB\plo\zip	When a received ..zip file passes XML validation it is moved here.
PB\src	A folder containing all source files used to compile executable code.

Configuring the properties file

About the Properties file

Each running instance of the Participant Batcher software must have a corresponding .properties file in the classes subdirectory. For help, see Folder and subfolder structure.

Take extra care to ensure no two .properties files have the same combination value of FTP username and password, processtype, and filetypes.

The .properties file has the configuration relevant for each instance as a set of properties. The following sections describe each property in groups. You need to ensure the value for each property is set appropriately to achieve your intended outcome.

For Unix-like systems, the Windows file paths must be converted to Unix paths.

A change to the .properties file requires an application restart, as the .properties file is only read at application startup.

FTP server name (ftpservername)

This property specifies the AEMO FTP server name. The server name can be in the format of a text URL for example, ftp://aemo.com.au or an accessible IP address.

For information about URLs for accessing AEMO's IT systems, see How to Connect to AEMO's Electricity IT Systems.

FTP server login username and password (username and password)

Any running instance of the Participant Batcher software only handles processes for a single NEM security identity. You must specify the username and password in the .properties file.

The distribution file includes one .properties file example (sample.properties in the pb subdirectory) and one runner batch file example (sample.bat). When you set up the application, the folder structure can differ from the above table. By modifying the .properties file and the pbEnvironment.bat file, you can distribute the subfolders over separate locations. The only limitation is the \pli and \plo subdirectory trees are sets.

Properties configuration is the key to ensuring multiple instances of the Participant Batcher software run concurrently without conflicts or omissions. As a starting point, use the default properties provided in the sample.properties file for initial configuration. Only start adjusting after you have an instance working. Each property above has a suggested starting value.additional information is in this style.

The username is usually, nemnet/account, for example:

```
username=nemnet/username
# either encryptedpassword or password
#encrypted# you need a password sYvJtcQFEEIyEw2Ui8CjBw==
password=somepassword
```

FTP server encrypted password (encryptedpassword)

As an alternative to password, the property encryptedpassword can be specified to improve your security. The key to the encryption is stored in the pb_key.properties file in the classes folder and is common to all instances using that classes folder tree.

The hash key must be exactly 16 hexadecimal characters of the form 0-9, A, B, C, D, E, and F. For improved security, change the value of the hash key supplied in the pb_key.properties file (use a scientific calculator to do this). The encrypted password is still sent from your site to the hub as clear text; however, a support person cannot easily discover the password from the .properties file.

MarketNet (NEMNet) passwords have a 90-day expiry, so you need to have a password change process in place to ensure your password does not expire. If your password expires or becomes locked out, please contact your company's Participant Administrator.

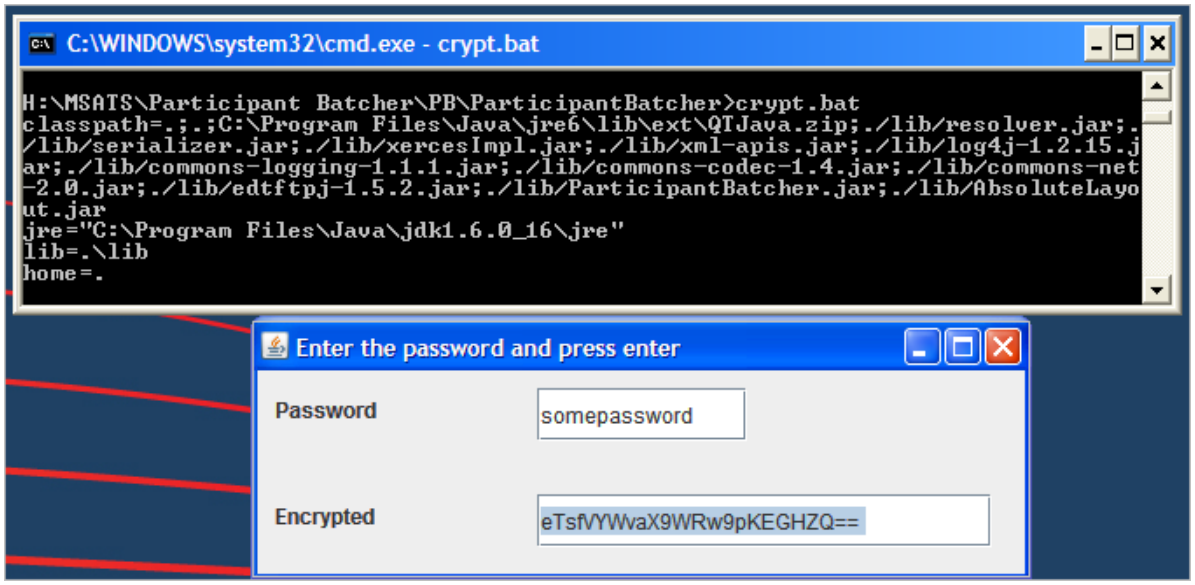
The pb_key.properties file contains one key of the form:

```
PB_KEY.properties file
#HASH
#Hash Key please keep very secure
# Hash key has to be exactly 16 Hexadecimal characters 0-
9,A,B,C,D,E,F
#    1234567890123456
hash=1234123412341234
```

To encrypt your password:

1. In the pb_key.properties file, change the hash key provided, to your own private 16 hexadecimal character hash key (using a scientific calculator).
2. Run Crypt.bat and enter your MarketNet password.
3. Press **Enter** on your keyboard.

4. Select the Encrypted password using your mouse and click CTRL+C on your keyboard to copy the encrypted password.



5. In the .properties file, use CTRL+V to paste the encrypted password next to password=.

6. Change password to encryptedpassword.

```
# either encryptedpassword or password
#encrypted# you need a password sYvJtcQFEElyEw2Ui8CjBw==
encryptedpassword=eTsfVYWvaX9WRw9pKEGHZQ==
```

Participants upgrading their Participant Batcher software must regenerate their encrypted password, re-enter it in the properties file, and then restart the Participant Batcher application.

FTP File Transfer Timeout (timeout)

This property allows the application to detect an FTP breakdown. An FTP operation that does not finish within the timeout value in milliseconds terminates with an error. Adjust the timeout value so under normal conditions it does not produce errors.

A value of 10000, equivalent to 10 seconds is a good starting point, for example:

```
# FTP File Transfer Timeout
timeout=10000
```

FTP file transfer mode (ftpmode)

This property allows the application to set the desired FTP transfer mode. Set this for compatibility with your firewall. The two values are active and passive, with the default mode being passive. To start, set the FTP transfer to work with passive and change if necessary. For further information refer to a network expert (also Communication mode (commode)

Communication mode (commode)

This property allows the application to choose among a variety of similar FTP libraries.

The two values possible are ftp or ftpedt:

- ftp corresponds to Apache Commons Net, http://commons.apache.org/proper/commons-net/apidocs/org/apache/commons/net/ftp/FTPClient.html
- ftpedt corresponds to edtFTPj, http://www.enterprisedt.com.

AEMO recommends running FTP in "passive" mode at all times, as per the default configuration for Participant Batcher. Using "active" FTP mode may work, however, there are potential issues (as participants have experienced) with the FTP port number being decided at the participant end and hence collisions can occur on the AEMO server.

FTP active data port range (lowftpport and highftpport)

These two properties allow the application to set the desired local port number range under the following conditions:

- The commode is ftpedt.
- The ftpmode is active.

The purpose of this is where multiple clients run on separate servers via a NAT (Network Address Translation) and ftpmode is set to active. Under these conditions the same data port can be chosen on the separate servers and when transferred via a network NAT a port collision occurs at the remote server. This can be made less probable by selecting a port range.

The preferred method to avoid these issues is to use the ftpmode of passive, for example:

```
# ftpmode either active or passive, active is default
ftpmode=passive
# choose the ftp library either ftp or ftpedt
commode=ftp
```

Business process type (processtype)

The Participant Batcher software handles the Participant Sending .zip files and the Participant Receiving .zip files processes. You must specify which business process you want the instance to handle.

The processtype value is a single digit with the following meanings:

1. = participant sending .zip files process.
2. = participant receiving .zip files process.
3. = both processes – this is the recommended setting.

Typically, the value of 3 is appropriate, for example:

```
# Business Process Type
# For User sending .zip files, let processtype=1
# For user receiving .zip files, let processtype=2
# For both processes, let processtype=3
processtype=3
```

When the application is handling both processes, with a processtype 3, the application receives files then sends files and continues to repeat the send-receive cycle. The sending and receiving of large numbers of files can cause delay. If the timing of sending and receiving files is a potential problem, then you can run separate instances for sending and receiving using processtype 1 and 2. Try 3 first; it's much easier and most traffic is from MSATS to you. The order of send and receive is alternated each cycle.

File types (filetypes)

To process high priority files on time, you can specify file masks for different instances. The filetypes value is a file masks list containing semicolon separated items.

For example:

```
filetypes=cats;nmid;mdmt;mtrd;cust;sord
```

The application processes only files that match the mask. A mask of cats matches any filename containing the string “cats”.

If you do not specify a value for filetypes, the application works on all files. For the simplest case of being non-selective, comment out the filetypes property.

B2B file types (b2bfiletypes)

A further file mask is required to know which files are B2B transfer files. If this is not specified the participant stopbox in the AEMO hub is not checked and the resend facility is used when sending B2B data to a stopped participant. The mask applies to the starting characters of file names only. A typical mask is:

b2bfiletypes=sord;mtrd;cust

Log4j logging configuration

You can specify the level to control the detail of logging. The levels are ERROR, WARN, INFO and DEBUG.

A sample log4j section is included in the sample.properties file and normally requires only two edits:

```
# Global logging configuration
log4j.rootLogger=INFO, fileout
```

```
# Rolling log file
output...log4j.appender.fileout.File=c:/pb/log/sample.log
```

The first sets the logging level, the second the location and filename of the log file. The recommendation is for you to set logging level to INFO, as this produces a moderate log file per day per instance that contains information about the overall process as well as any errors.

DEBUG is normally only used to help diagnose problems, as the log file becomes too large for routine checking.

Watermarks for the participant inbox (inboxhighmark and inboxlowmark)

Configure the high and low watermarks for the participant inbox folder on the AEMO hub, so it does not have too many unprocessed files. The intention is to have no more than the inboxhighmark and to receive more when the number falls to the inboxlowmark.

You can extensively configure the log4j library, however, full documentation is beyond the scope of this document. For further information, documentation is available at http://logging.apache.org.

The process rules are:

- Assuming the number of files in the participant inbox is N, then if N is greater than the low-watermark for inbox, the application uploads no files.
- Otherwise (if N is at or below the low-watermark), the application uploads files to the participant inbox at the number of high-watermark minus N.

A typical set of numbers is:

```
# Watermarks for Participant Inbox
inboxhighmark=30
inboxlowmark=5
```

For this typical setting, on startup, the application transfers 30 files (high-watermark of 30 minus 0). After AEMO acknowledges 20 (so the number of files falls to the low-watermark of 10), the application sends another 20 (being the difference between high-watermark of 30 and current file count of 10), if they are available. Sending large numbers of files to the inbox achieves no purpose as the hub has processing limits and the number of files slows down operations such as getting file lists.

Watermarks for the participant local outbox (outboxhighmark and outboxlowmark)

Configure the high and low watermarks for the participantlocal outbox folder, so the files do not overload the folder. The intention is to have no more than the outboxhighmark and to receive more when the number falls to the outboxlowmark. The process rules are:

- Assuming the number of files in the participantlocal outbox is N, then if N is greater than the low-watermark, the application downloads no files.

- Otherwise (if N is at or below the low-watermark), the Participant Batcher software downloads files from the participant outbox up to the number of high watermark minus N.

A typical set of numbers is:

```
# Watermarks for Participant Local Outbox
outboxhighmark=500
outboxlowmark=100
```

The initial download is 500 files, then the reception of files stops until the application clean-up process moves at least 400 files to the .zip subdirectory and the process downloads to fill up to the 500 again. As the files clear to the NACK or .zip subdirectory, the space becomes free.

The clean process is moving files out of this folder, so normally if that process is fast enough this setting has no net effect. Setting the number to more conservative values can be used to limit the impact of file reception on your network traffic in combination with waiting times, Configuring the properties file.

Waiting time between each file for the main process (filewaittime)

To control bandwidth usage, you can specify a waiting time between the send or receive of each file. This ensures MarketNet can be shared among separate applications. To maximise speed, set the filewaittime to 1, to add one extra second after each .zip file is transferred in either direction.

All waiting times are expressed in milliseconds (ms), so 1000 is equivalent to 1 second.

For example:

```
# Waiting Time Between Each File for the Main Process
filewaittime=1
```

Waiting time between each cycle for the main process (processwaittime)

You can specify a process cycle frequency that allows management of the computer loading on the application computer and minimises the number of non-productive folder commands sent to AEMO. If the folder command on AEMO indicates work is required (sending, receiving, validating or acknowledging), then that work is done before this wait occurs. If the system is idle for example, no work to do, then this timer sets the time between checks.

A typical figure is 30000 for a maximum delay of 30 seconds, for example:

```
# Waiting Time Between Each Process for the Main Process
processwaittime=30000
```

Waiting time between each resend process (resendwaittime)

The resendwaittime property specifies the frequency for the resend process. A separate resend process runs in parallel with the main send and receive processes. When the receiving B2B participant is stopped, the AEMO B2B Handler defines a special code for negative acknowledgement. In this circumstance, the file being sent is put in the resend subdirectory of the participantlocal inbox.

The resendwaittime property controls how often the application checks the resend subdirectory for files. The objective is that the receiver in B2B is ready to accept after this time.

A typical number is 300000 (five minutes), for example:

```
# Waiting Time Between Each Resend Process
resendwaittime=300000
```

This allows time for the other participant to make a significant impact on their queue of files. If the files do not succeed after several retries, they are sent to the timeout subdirectory of the participantlocal inbox. To reduce timeouts, increase the resendwaittime property.

Waiting time between each clean-up process (cleanwaittime)

The cleanwaittime property specifies the frequency for the clean-up process. The clean-up process moves the received files into the .zip or NACK subfolders of the participantlocal outbox. The process also moves sent files and their ACK files to the DONE, NACK and RESEND subfolders of the participantlocal inbox. A typical number for the cleanwaittime property is 30000 (30 seconds), similar to the processwaittime.

This feature is not as relevant now because the B2B has a feature that indicates which participant is stopped. This can be checked before sending a B2B file, B2B file types (b2bfiletypes).

For example:

```
# Waiting Time Between Each Cleanup Process
```

```
cleanwaittime=30000
```

Waiting time between each monitoring process (monitorwaittime)

The monitorwaittime property specifies the frequency for the monitoring process. The application includes a check that the various working threads are active. A successful check causes an update to a file (Monitor file folder (monitordir)) and a failure means the file is not updated.

A typical number for the monitorwaittime property is 300000 (five minutes), for example:

```
# Waiting Time Between Each Monitoring Process
monitorwaittime=300000
```

By pointing application monitoring tools (not supplied in this application) to that file, the monitor can raise the alarm of the failure of this application.

Stop file wait time (stopfilewaittime)

The stopfilewaittime property specifies how long to wait for the separate threads to smoothly stop before aborting them. To ensure a smooth stop under normal conditions this value is set larger than all the wait times specified in this section. In practical terms setting it above processwaittime and monitorwaittime ensures file operations are not normally interrupted when stopping.

A typical number for the stopfilewaittime property is 300000 (five minutes).

```
# stop file wait time to wait until threads cleanly finish
stopfilewaittime=300000
```

Waiting time between each exception (exceptwaittime)

The exceptwaittime property specifies the wait between exceptions in the main process loop. In this case, the process only retries an FTP or file exception on a cycle set by this wait. This stops the application looping quickly, and uselessly, when FTP is not working.

A typical number for the wait time is 30000 (30 seconds), similar to the processwaittime, for example:

```
# Exception wait in the main process in mses, e.g. FTP error or file error
exceptwaittime=30000
```

folders (pninbox, pnoutbox, pnstopbox, plinbox and ploutbox)

You must specify five process folders and their subfolders for each instance:

1. Participant inboxpninbox.
2. Participant outboxpnoutbox.
3. Participant stopboxpnstopbox.
4. Participant local inboxplinbox.
5. Participant local outboxploutbox.

For performance reasons, prepare the two local folders on the same server running the application. Unzipping the ParticipantBatcher.zip file creates a sample set of the local folders. Each of the local folders must contain the correct subfolders.

Substituting “participantid” below, with your participant ID, the settings are:

```
# participant local inbox, participant local outbox
pninbox=/participantid/inbox
pnoutbox=/participantid/outbox
pnstopbox=/participantid/parkbox
plinbox=c:/pb/sample/pli
ploutbox=c:/pb/sample/plo
```

Monitor file folder (monitordir)

You must specify the file folder for the instance threads monitoring. In the folder specified in the monitordir property, the application creates and updates a file of the form instance_monitor.log. The filename is instance specific, so the folder is shareable among instances. Each monitor file does not grow in size and there is only one per instance.

The default is to use the same folder as the log file, for example:

```
# Monitor File folder
```

```
monitordir=c:/pb/log
```

Stop file folder (stopfiledir)

This folder is used if you want to stop the “sample” instance smoothly by creating a file of the form sample.stp. The folder must be specified and the stop file is deleted as part of the application stopping.

The default is to use the same folder as the log file, for example:

```
# stop file path to a file to make the application stop - sample.stp
stopfiledir=c:/pb/log
```

Resend try control file folder (resendcontroldir)

You must specify the folder for the resend process control file. In the folder specified in the resendcontroldir property, the application creates a file (instance.control), which is used to keep track of how many times the application attempts to resend a file without success, Maximum resend tries number (resendcontrolno)). The filename is instance specific, so the folder is shareable among instances.

The default is to use the same folder as the log file, for example:

```
# Resend Try Control File folder
resendcontroldir=c:/pb/log
```

Maximum resend tries number (resendcontrolno)

You must specify the maximum number of attempts to resend a file. When a file is resent without success, the application increments a counter (Resend try control file folder (resendcontroldir). If the count exceeds this limit, the file is moved to the timeout subdirectory. To decide the maximum number of attempts to resend a file, divide the time you are prepared to wait before giving up by the value of the resendwaittime property.

For example, with a resendcontrolno value of 8 and a resendwaittime of 300,000 (five minutes), the application gives up after about 40 minutes:

```
# Maximum Resend Tries Number
resendcontrolno=8
```

To attempt retries for a day, set the resendcontrolno value to 288 for the same resendwaittime:

```
# Maximum Resend Tries Number
resendcontrolno=288
```

Custom clean-up (customclean1)

To control the movement of files from one folder to another at the cleanwaittime cycle time, multiple sets of customclean are set up. The purpose of this is to move files from the local folders to other servers in the participant's computer system. For example, from the .plo/zip for incoming .zip files (from some internal server) and then to the .pli folder for sending to AEMO. You can specify multiple custom clean-ups by defining property names with the last digit incremented (for example, customclean1, customclean2, etc.).

The form of the information is:

```
format customclean1=FromDir;ToDir;FileType;FileMask;FileMask;
```

Where FromDir is where the files are copied from and ToDir is where the files are copied to.

Extension is the file extensions for selection, and Mask is one or more masks separated by a semicolon to further select the files. For example:

```
customclean1=c:/pb/sample/plo/zip;g:/pb/cats;.zip;cats;nmid
customclean2=c:/pb/sample/plo/zip;g:/pb/mtrd;.zip;mtrd
```

This copies mtrd.zip files received to a network folder, and *nmid*..zip and cats.zip files to a separate network folder.

HTTP port (port)

You can specify a port for browser access to the running Participant Batcher instance. For example, if the instance has port = 80 in its .properties file and the user launches a browser on the server with a URL of localhost, a display appears with information of file transfer statistics, status of the instance, configuration information, and file lists.

If you are running more than one instance choose another port number for each instance, for example:

```
# If port is >0 enables that port as a web site for inspection in that participant batcher
port = 9876
```

The URL is then localhost:9876.

To keep a history across restarts, the application keeps a file instance.ser in the monitor subdirectory. This file is a fixed size. If you have issues with an upgrade and get error messages related to the .ser file, you can delete the instance.ser file to lose the history.

Be aware that port 80 is normally reserved as a default and can be used as part of your existing applications on this server. You must check the port is not used before you configure any instance to use any port. If port is not defined or set to zero, the internal web page is not produced and no port is used.

Schema validation (schemavalidation)

You must specify whether or not the running instance is to perform XML schema validation on incoming .XML files (in .zip files).

The possible values of the schemavalidation property are:

- True to perform schema validation.
- False to ignore schema validation.

The recommended value for the schemavalidation property is true:

```
# Schema Validation
# true - if schema validation should be carried out; false - otherwise
schemavalidation=true
```

XML schema namespace for the generated acknowledgement (ackschemanamespace)

You must specify the XML schema namespace to use for the generated .ACK file.

For example:

```
# XML schema namespace for the generated Acknowledgement
ackschemanamespace=urn:aseXML:r25
```

The recommended value for ackschemanamespace is usually the highest version of schema in the \aseXML\Schemas folder. The actual version used for acknowledging the B2B transaction groups (as specified for B2B file types) is the version of the incoming XML file being acknowledged.

XML schema location for the generated acknowledgement (ackschemalocation)

You must specify the XML schema location to use for the generated ACK file. For example:

```
# XML schema location for the generated Acknowledgement
ackschemalocation=
http://www.nemmco.com.au/asexml/schemas/r25/
aseXML_r25.xsd
```

XML schema caching locations (schemacache1)

To utilise the “schema local caching” feature to improve performance, you must specify the XML schema caching locations. You can specify multiple schema caches by defining property names with the last digit incremented for example, schemacache1, schemacache2, etc. Unzipping the ParticipantBatcher.zip creates schemas in a folder tree.

The ackschemalocation property must agree with the ackschemanamespace property, XML schema namespace for the generated acknowledgement (ackschemanamespace).

For example:

```
# XML schema caching locations - can be multiple
schemacache1=file:///c:/pb/aseXML/schemas/r7/aseXML_r7.xsd
schemacache2=file:///c:/pb/aseXML/schemas/r9/aseXML_r9.xsd
```

```
schemacache3=file:///c:/pb/aseXML/schemas/r10/aseXML_r10.xsd
schemacache5=file:///c:/pb/aseXML/schemas/r11_p1/aseXML_r11_
p1.xsd
schemacache6=file:///c:/pb/aseXML/schemas/r14/aseXML_r14.xsd
schemacache7=file:///c:/pb/aseXML/schemas/r16/aseXML_r16.xsd
schemacache8=file:///c:/pb/aseXML/schemas/r17/aseXML_r17.xsd
schemacache9=file:///c:/pb/aseXML/schemas/r21/aseXML_r21.xsd
```

Creating Instances

About creating instances

Before you initiate any instance, ensure all folders specified in its .properties file exist. Remember to check for relevant subfolders (e.g. done subdirectory in the participant local inbox folder). The folder file structure is like the sample folder with subfolders above. Use copy and paste in Windows Explorer to quickly create a new folder structure. Then copy the file sample.properties to the new instance name and edit the contents to suit.

Typically you need to edit at least:

ftpservername, username, password, log4j log file name, filetypes, pninbox, pnoutbox, plinbox, and ploutbox properties.

Once all of these are in place, you can initiate instances by calling each instance runner batch file. In this example, all instances are configured in the same folder, but must have separate local inbox and outbox folders.

To ensure there is only one instance of the Participant Batcher software running at any given time for a specific configuration, the software enforces an instance naming mechanism where each running instance has a name. The instance name reflects on all related files.

For example, if the name of one running instance is AEMO1:

1. Create and configure a .properties file in the classes subdirectory named AEMO1.properties.
2. Set up a batch runner named AEMO1.bat containing the string AEMO1 in the command line starting the batcher.
3. The software creates the instance log file named AEMO1.log.

Name your instances after the participant ID associated with your username and password combination and avoid having special instances for file or process types. If you are using the Participant Batcher to send B2B transactions, you can have an instance for MSATS and an instance for B2B files.

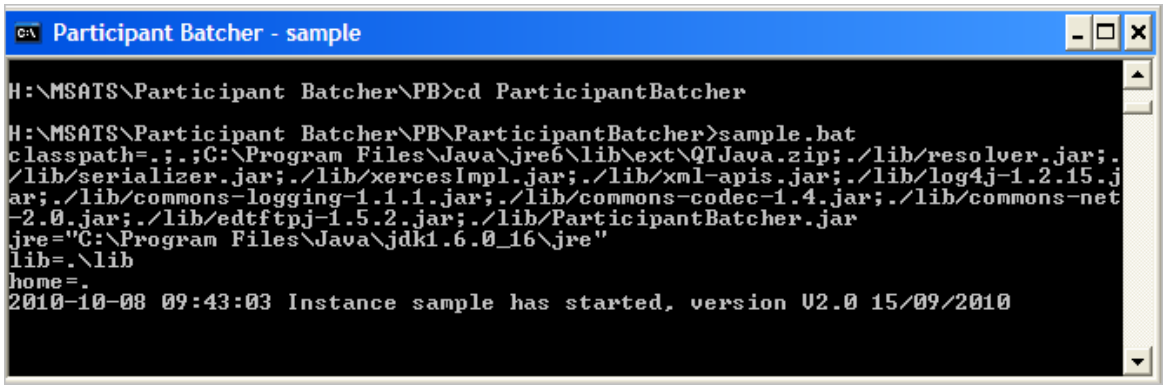
Checklist

1. Have you created the folder tree from ParticipantBatcher.zip? SeeExtracting the distribution file.
2. Does the classes folder have the sample.properties and pb_key.properties files?
3. Do you have a log folder?
4. Do you have a lib subdirectory containing all jar files listed in the README.txt?
5. Do you have the equivalent pli and plo folders with their related subfolders? SeeExtracting the distribution file.
6. Have you edited sample.properties, especially ftpservername, username, password, log4j log file name, pninbox, pnoutbox, plinbox, ploutbox, and filetypes in the properties file? See Configuring the properties file.
7. Have you edited the pbEnvironment.bat file? Ensure your Java path is correct. If you are running on Unix (or similar), recode *.bat files as equivalent shell scripts.
8. Are you sure your Java version is correct? To check, use java-version in the command line.
9. Have you established if you want a port token in the sample.properties file? See HTTP port (port).

Initiation

Testing the sample file

Run the sample.bat and expect to see a JVM screen similar to the following:



Extra information usually indicates an error condition, such as a folder not found. Check the log subdirectory at the sample.log file, correct the problem and try again.

If the JVM screen does not contain errors, run the application for a few minutes, stop it, and look at the log file. If the log file does not indicate warnings, you have succeeded; otherwise, correct the problem and try again. Usually you can correct all problems by adjusting the entries in the sample.properties file. See Configuring the properties file.

Converting to production

To convert the working sample to production:

1. Rename the instance to something meaningful. See Creating Instances.
2. Copy the sample.properties file and sample.bat file.
3. Copy the sample subdirectory tree.

4. Edit the new properties file to do the work intended, perhaps changing the ftpservername and filetypes.
5. Start the application and wait until the Participant Inbox and Outbox on the AEMO hub are empty – this may take some time if the system is active. At a quiet time, put a .zip file (perhaps a NMI discovery request) into the participantlocal inbox and check that it ultimately disappears.

   If the filename is not unique (for example, you have used the file previously), the copy to the done subdirectory fails and the file does not disappear. Change the filename for each test cycle.

6. If the application behaviour appears satisfactory, you need to implement management of the growth of files. A solution to move, delete, or archive files from the following subfolders is essential:
   - Done
   - Zip
   - Log
   - Timeout
   - Participant local inbox\N
   - Participant local outbox\Nack
7. As the last step, add monitoring on the instance by looking at the timely update of the instance_monitor.log file.

Monitoring the instance_monitor.log file is not sufficient to cover all failure modes. Other recommendations include:

- Scanning the content of the log file looking for warnings and errors.
- Scanning the timeout subdirectory for any files.
- Scanning the two nack subfolders for any files.

Running as a Windows service

There are a variety of commercial and open source solutions allowing Java applications to run as a Windows service. The distribution includes a configuration for a popular open source solution as an example of how running the Participant Batcher as a Windows service is achieved. The example software is 32-bit, so requires a 32-bit JRE. Participants need to adapt this example to software of their own choice.

The distribution includes three files relevant to running as a Windows service:

1. pbServiceInstall.bat installs a Java application as a Windows service.
2. pbServiceUninstall.bat removes a Java application running as a Windows service.
3. JavaService.exe is a 32-bit Windows wrapper for the 32-bit JRE, being the minimum to run the Participant Batcher software.

Installing

To install the Participant Batcher as a Windows service:

1. Edit the pbEnvironment.bat file and update the HOME and JRE environment variables to suit your environment. You did this to get the sample.bat console application working. Ensure the path to the JRE DLL is valid.

   ```
   %JRE%\bin\server\jvm.dll
   ```

   For example:

2. From the command line in the Participant Batcher folder (for example, c:\pb), run the pbServiceInstall.bat file supplying the instance name as a parameter (for example, sample).

   ```
   C:\>pbServiceInstall.batsample
   ```

   The following parameter creates a sample instance running as a service:

3. Use the Windows services interface to start the instance. Note that stdout and stderr log files are created in the log folder along with the usual log file. The stdout contains the usual instance console output and stderr has the console error messages. Multiple services are catered for.

Uninstalling

To uninstall the Participant Batcher as a Windows service:

- From the command line in the home folder (for example, c:\pb), run the pbServiceUninstall.bat file supplying the name of the service as a parameter. For example:

  ```
  C:\>pbServiceUninstall.batsample
  ```

Maintenance

The following housekeeping tasks are needed to keep a Participant Batcher installation running smoothly:

- Purge files older than a specified date from the log folder.
- MarketNet passwords have a 90-day expiry, so you need to have a password change process in place to ensure your password does not expire. In the event that your password expires or becomes locked out, please contact your Participant Administrator (who may need to contact the AEMO Information and Support Hub).
- AEMO retail systems are updated on a 6-monthly release cycle, which includes changes to participant interfaces. These releases are typically scheduled for production late May and mid-November. Plan to have resources available to implement the MSATS releases around this time and keep your systems on a supported configuration.

Upgrading

For information about upgrading to the current version of the Participant Batcher software, see the README.txt found in the distribution file.

Needing Help?

Participant Batcher assistance

Java errors?

- Check Participant Batcher software is the latest release.
- Check Java version is compatible.

Not transferring files, or timing out?

- Check network configuration and performance, including firewalls.
- Try disconnecting and reconnecting.
- Modify properties to suit available network conditions.
- Check with local IT support.

Not connecting or locked out?

- Check credentials (passwords do expire).
- Check successful connection using a command-line FTP client from the same environment as where Participant Batcher runs.
- Check using passive FTP, since active FTP is not as reliable.

Observing abnormal behaviour?

- Check release of Participant Batcher is the latest.
- Check logs.
- Check with local IT support.

- Check reproducible in a test environment.

When requesting support for the supplied software, provide at least (after checking each thoroughly):

- Version of Participant Batcher.
- Logs showing the abnormal behaviour (zipped).
- .Properties file.
- Diagram of architecture.

Watch out for market notices and bulletins about network changes potentially affecting access, but unplanned events do occur. In preparation for handling network and system failures, plan and test alternate configurations (including failovers).

AEMO's Support Hub

Contacting AEMO's Support Hub

IT assistance is requested through one of the following methods:

- Phone: 1300 AEMO 00 (1300 236 600)

  For non-urgent issues, normal coverage is 8:00 AM to 6:00 PM on weekdays, Australian Eastern Standard Time (AEST).

  See AEMO's Support Hub for more information.

AEMO recommends participants call AEMO's Support Hub for all urgent issues, whether or not you have logged a call in the Customer Portal.

Information to provide

Please provide the following information when requesting IT assistance from AEMO:

- Your name
- Organisation name
- Participant ID

- System or application name
- Environment: production or pre-production
- Problem description
- Screenshots

For AEMO software-related issues please also provide:

- Version of software
- Properties or log files
- Replication Manager support dump and instance name (if Data Interchange problem)

Feedback

Your feedback is important and helps us improve our services and products. To suggest improvements, please contact AEMO's Support Hub.

Related resources

The resources listed in this section contain related information that may assist you. You can find resources on AEMO's website.

B2B Procedures, the Business to Business (B2B) Procedures prescribe the content of, the processes for, and the information to be provided to support, B2B Communication.

Guide to Transition of aseXML, provides information for participants transitioning to another aseXML schema version.

Guide to User Rights Management, assists Participant Administrators to manage their Participant User's access to AEMO's systems. It also explains how to set up single user IDs for use with the Set Participant function in AEMO's web portals.

How to Connect to AEMO's Electricity IT Systems, explains the interfaces available for electricity participants and how to access them.

Java SE Downloads, http://www.oracle.com/technetwork/java/javase/downloads/index.html

List of National Electricity Market Procedures, guidelines and documents, provides a list of procedures authorised under the NER.

LOG4J: Logging Services, http://logging.apache.org/log4j/1.2/manual.html

Shared Market Protocol (SMP) Technical Guide, provides participants with the technical specifications for the delivery of B2B transactions using the *B2B e-*

Hub APIs. This detail assists participants developing their own systems to utilise these APIs.

MSATS user guide group

The MSATS user guide group forms a detailed guide to using MSATS. Each document is targeted towards a specific audience and explains how to navigate and use the menus for each web portal function.

The MSATS user guide group does not detail jurisdictional and configurable rules regarding the use of the web portal, batch interfaces, and systems interfacing with MSATS.

The following table provides a description of each document in the group and its intended audience. The documents are located on AEMO's website > **Electricity Retail and Metering**.

Name	Description	MSATS	B2B	PAs	Ombudsman
Guide to MSATS and B2B Terms	Assists participants of the Retail National Electricity Market (NEM) to understand the terms used in the retail electricity market procedures and the Market Settlement and Transfer Solution (MSATS) participant IT system.	✓	✓	✓	✓
Introduction to MSATS	Contains an overview of the MSATS web portal, explains the MSATS framework, provides assistance with gaining access, and a list of MSATS reference information.	✓	✓	✓	✓
Guide to MSATS Web Portal	Explains how to use the MSATS participant web portal functions (contains a short section on using the batch interface).	✓		✓	
Guide to MSATS B2B	Explains how to use the B2B function and includes a glossary of B2B terms.		✓	✓	

Name	Description	MSATS	B2B	PAs	Ombudsman
Guide to B2B e-Hub Self-Accreditation	Explains how to obtain accreditation to become a *B2B e-Hub Participant*.		✓	✓	
MSATS Ombudsman Enquiry User Interface Guide	Explains how to use the Ombudsman Enquiry system.				✓
Guide to User Rights Management	Explains how to create and maintain participant users.			✓	

Index

O

P

Q

R

S

T

U

W

X