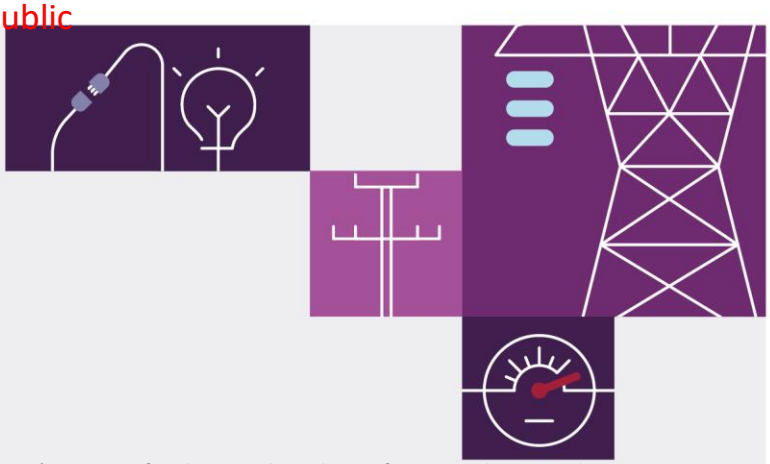
IDX - AEMO Gateway Software - Technical Specification – June 2026

1.02 June 2026

Pre-production: 18 May 2026

Production: 15 June 2026



Important notice

Purpose & audience

This document describes the technical changes required to participant's systems for the - Technical Specification . The Australian Energy Market Operator (AEMO) provides this information as a service targeting business analysts and IT staff in participant organisations. It provides guidance about the changes to their market systems under the National Electricity Rules, as at the date of publication.

How to use this document

- If you have questions about the business aspects of these changes, please see Consultations on AEMO's website.
- The references listed throughout this document are primary resources and take precedence over this document.
- Unless otherwise stated, you can find resources mentioned in this guide on AEMO's website.
- **Text in this format** is a link to related information. Some links require access to MarketNet.
- **Text in this format**, indicates a reference to a document on AEMO's website.
- **Text in this format** is an action to perform in the Markets Portal.
- 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.
- Glossary Terms are capitalised and have the meanings listed against them in the Glossary.
- Rules Terms have the meaning listed against them in the **National Electricity Rules**.

Privacy and legal notices

The material in this publication may be used in accordance with the **privacy and legal notices** on AEMO's website.

Trademark Notices

Microsoft, Windows and SQL Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the US and other countries.

© 2015 Google Inc, used with permission. Google and the Google logo are registered trademarks of Google Inc.

Distribution

Available to the public.

Document Identification

Prepared by: AEMO Digital

Last update: Friday, 5 June 2026 1:58 PM

Version History

1.02 Initial creation

Documents made obsolete

The release of this document changes only the version of IDX - AEMO Gateway Software - Technical Specification – June 2026.

Support Hub

To contact AEMO's Support Hub use Contact Us on AEMO's website or for urgent matters phone: 1300 AEMO 00 (1300 236 600.

Contents

1 Introduction

1.1 Audience

AEMO provides this information as a service targeting business analysts and IT staff in Registered Participant companies.

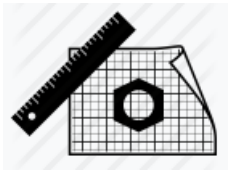
The primary audience for this technical specification is:

- Registered participants using this software for data exchange with IDX for Power Quality Data.

1.2 Objective

The IDX - AEMO Gateway Software - Technical Specification – June 2026 describes the AEMO Gateway Software details from a participant perspective.

1.3 Status

Version	Status
1.02	The design is ready for participants' builds.
1.01	The design is ready for participants' builds.
1.00	The design is ready for participants' builds.
0.01	 **Initial Draft for review. The design is not ready for participants' builds** Presents the IDX - AEMO Gateway Software - Technical Specification – June 2026 evolving design. Please send feedback to **Contact Us**. In the **Details of your enquiry** section, mention the EAS Knowledge Management team as the Resolver group.

1.4 Release dates

Scheduled for implementation in:

- Pre-production: 18 May 2026
- Production: 15 June 2026

1.5 Projects and enhancements

Changes and enhancements for this Release include:

No.	Functionality	Change	Affected interface	Reference
1	Power Quality Data (PQD)	New interface for managing Basic Power Quality Data (BPQD) and participant controls.	Markets Portal	Markets Portal
2	Industry Data Exchange (IDX) platform	New platform for data exchange between AEMO and Market Participants.	IDX	Industry Data Exchange

1.6 Rule and procedure changes

The following rules and procedures take precedence over technical specifications and guides.

Title	Project	Version/status	Effective
See IDX - Basic Power Quality Data - July 2026 for rule and procedure changes.			

1.7 Related technical specifications

Title	Description
Industry Data Exchange – Industry Data Exchange Platform – July 2026	Industry Data Exchange (IDX) platform to exchange B2B and B2M data with AEMO across electricity and gas markets.
Industry Data Exchange - Basic Power Quality Data - July 2026	Basic Power Quality Data (BPQD) API and schema.

1.8 Related documents

Once published, these resources take precedence over this technical specification

These guides and resources are updated according to this technical specification and published for the pre-production Release Date.

Title	Description	Status
API portal	Information about the PQD and IDX APIs	Not started
Markets Portal Help	Online help for managing BPQD and IDX	Not started

1.9 Approval to change

No approval or agreement to change required from participant change controllers. Agreement was sought in the Market Interface Technology Enhancements Working Group (MITEWG).

1.10 Market systems user group meetings

The Market Systems User Group (MSUG) is an industry user group established to discuss NEM wholesale and retail IT systems releases. Its purpose is to facilitate the continuing improvement of AEMO's IT systems by seeking feedback and collaboration from participants.

MSUG meetings are open to all interested parties, with invitations sent to all included on the distribution list. If you have a technical question for a project and want to attend the MSUG ask your company's support team to include your email address in their **AEMO Help Desk Bulletin (CRM)** distribution list.

1.11 Version numbers

AEMO releases new versions of this document as the technical requirements are streamlined.

Incremental version numbers such as 1.01, 2.01 and so on mean there is a minor change to the technical specification.

Major version numbers such as 1.00, 2.00 means there are substantial changes to the technical specification. Participants must carefully review these changes, detailed below.

1.12 Changes in this version

This version has the following changes:

- Revises information in the **Manual keystore file creation** process.

2 Proposed Timeline

The dates for the Market System User Group Meetings (MSUG) are tentative. We will provide an invitation one week prior to the meeting.

Milestone	Date	Description
Approval required	n/a	Final date for participant approval of this Release.
Revised Technical Specification	June 2026	AEMO releases new versions of this document as the technical requirements are streamlined. During the project this document is the source of truth. From the production release, the technical specification becomes final and the **related documents** become the source of truth. **Technical Specification Portal**
Related Documents publication	18 May 2026	Release of guides and resources mentioned in Related .
Next MSUG meeting	17 June 2026	Market Systems User Group Meeting (MSUG) to review the technical specification and ask AEMO technical SMEs questions. This date is tentative. The Knowledge Management Team provides the invitation prior to the meeting.
AEMO Gateway Software pre-production systems available	18 May 2026	Testing period begins for participants.
AEMO Gateway Software production systems available	15 June 2026	AEMO Gateway Software is released to participants.
Coordinated industry test	25 May 2026 – 15 June 2026	AEMO coordinated testing with participants.
IDX pre-production available	See **Industry Data Exchange Platform Technical Specification** for IDX pre-production dates.	
IDX production systems available	See **Industry Data Exchange Platform Technical Specification** for IDX production dates.	

3 Participant Impact

The AEMO Gateway software, developed during the foundation phase, is released as a single, comprehensive solution. Later releases introduce configuration artefacts tailored to specific business functions.

This approach minimises disruption and supports operational continuity as AEMO progressively introduces new IDX business functions and enhancements.

Milestone	Details	Description
Improvements for AEMO Gateway, pdrLoader, and pdrMonitor	See **AEMO Gateway**, **pdrLoader** and **pdrMonitor** for more information.	The AEMO Gateway, pdrLoader, and pdrMonitor have had functional improvements, bug fixes and security upgrades.

3.1 Staged releases

This release follows a staged delivery approach for the AEMO Gateway Software.

Participants should note the following for the release:

- Only the CLI install and IDX GUI installer are available for AEMO gateway Software and PDR Monitor.
- AEMO is testing the software for in-situ upgrades of existing installations for other use cases (for example, NEM wholesale participation). Until testing is complete and software is certified, AEMO does not recommend this version for in-situ upgrades of existing installations.

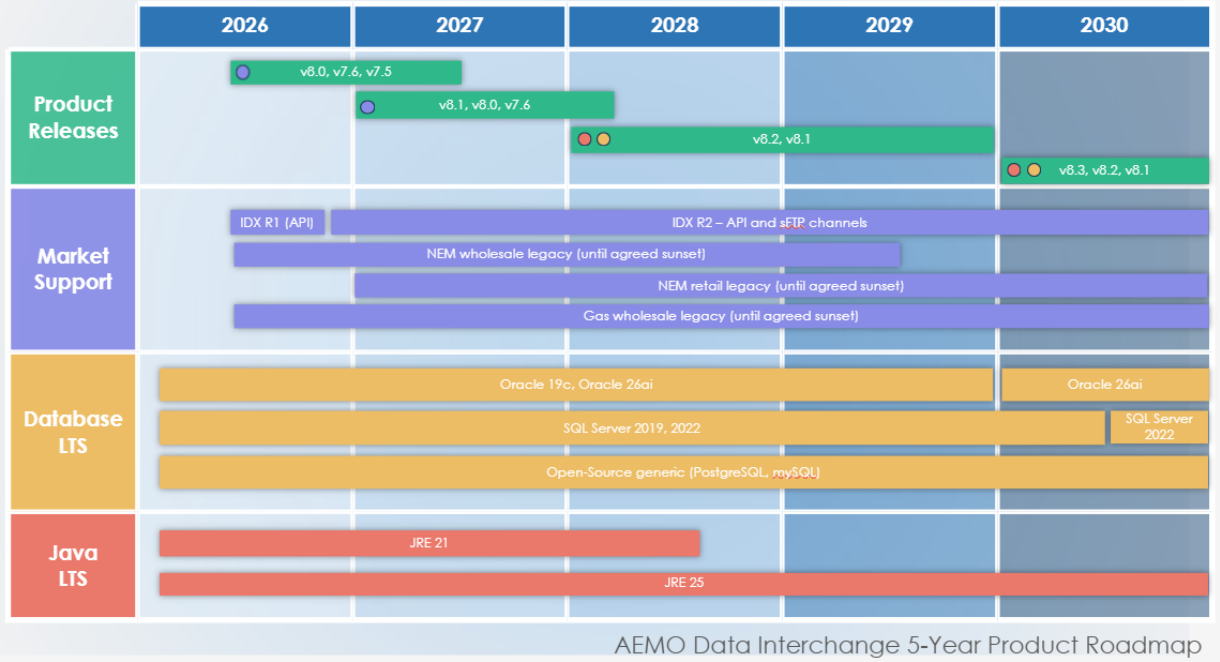
3.2 Legacy support

TBC

3.3 Data Interchange roadmap

The diagram presents the initial draft for the Data Interchange five-year product roadmap, showing how product releases and supporting technologies are planned to evolve from 2026 to 2030.

Participant Impact



Public

4 Need to Know

4.1 Assumed knowledge

This guide assumes you have knowledge of:

- The Java application environment.
- The operating system your organisation is using.
- IDX BPQD technical specification

4.2 Support

If you contact AEMO's Support Hub for support about this online help, ask them to direct your ticket to the Industry Data Exchange support resolver group.

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

IT assistance is requested through one of the following methods:

- Phone: 1300 AEMO 00 (1300 236 600)
- **Contact Us** form on AEMO's website

Please provide the following information when requesting 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
- A diagnostic support package (support dump), generated from pdrMonitor for Data Interchange related issues

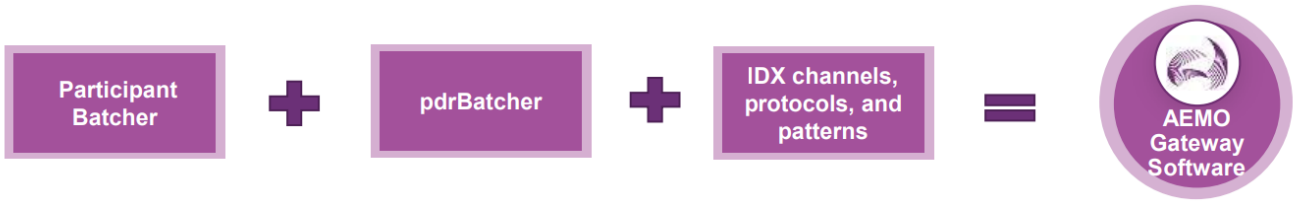
Public

5 About AEMO Gateway

5.1 What the software is for

The AEMO Gateway software provides an AEMO-supplied gateway that supports data exchange with the AEMO IDX Hub for power quality data using IDX patterns. It also supports current legacy data-exchange patterns.

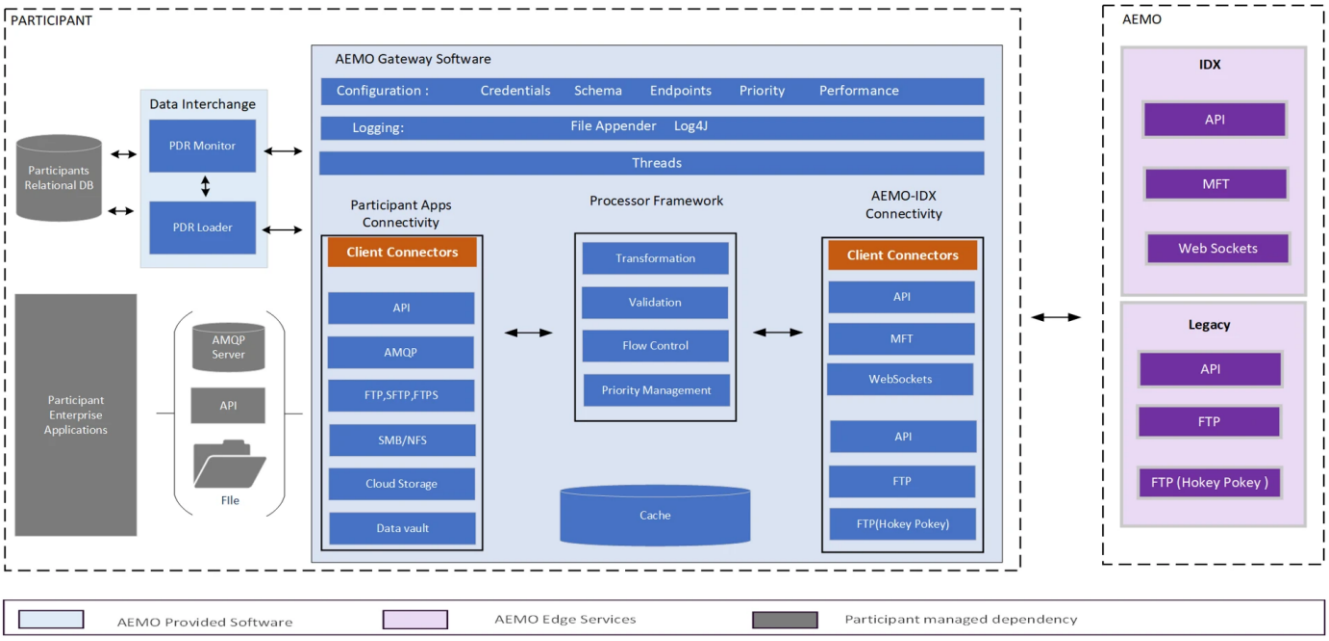
The software consolidates the existing MSATS Participant Batcher and pdrBatcher capabilities and supports interaction with the IDX patterns proposed for the BPQD interface.



5.1.1 Key features

The AEMO Gateway Software:

- Provides standard gateway capabilities, including efficient protocol conversion.
- Supports new participants during the transition by maintaining compatibility with legacy protocols.
- Offers backward compatibility with payload conversion capabilities to aid change and transition management.
- Aligns with the IDAM roadmap by software support for the latest IDAM patterns, ensuring secure and efficient identity and access management.



5.1.2 Connectivity

The connectors available to use are:

- Event Channel Connector: Client code to establish an event notification channel (Web Sockets) for outbound messages.
- Restful API: Includes client code for calling a Restful API.
- File Connector: A connector to manage File-based connections (SFTP, FTP, blob storage etc.) and Integration.
- AMQP Connector
- Other Connectors: As per Industry requirements.
- Support for Legacy Patterns: Support for legacy Hokey Pokey FTP file exchange patterns.

5.1.3 Payload Handling & Transformation

Payload transformation framework

- Supports payload transformation & file compression.
- Performs text manipulation (head, tail, replace, etc).
- Zip/Unzip payloads.

- Re-names the files etc.
- Framework supports custom transformation modules to be plugged in.

Validation framework

- Performs schema validations.
- Framework supports custom Validation modules to be plugged in.

Logging

Provides intuitive file-based logging for each step performed and a flexible logging framework based on participant preference (DB etc.).

Flow-Control

Manages control mechanisms to handle Flow-Control events.

Message retrieval

Manages outbound messaging pull based on event metadata and priority configuration

5.2 Software requirements

The AEMO Gateway:

- Is certified against OpenJDK 21 and runs on Java platforms using JRE11 or later.
- Runs on Windows and Unix-like operating systems.
- In a Windows 64-bit environment, requires the Java and wrapper for service must be either 32-bit or 64-bit.

The provided batch and shell scripts are examples only. They are not recommendations or requirements, and you may need to adapt them to suit your host operating system.

5.3 Hardware requirements

The minimum hardware requirements are a quad core machine with 8GB of ram.

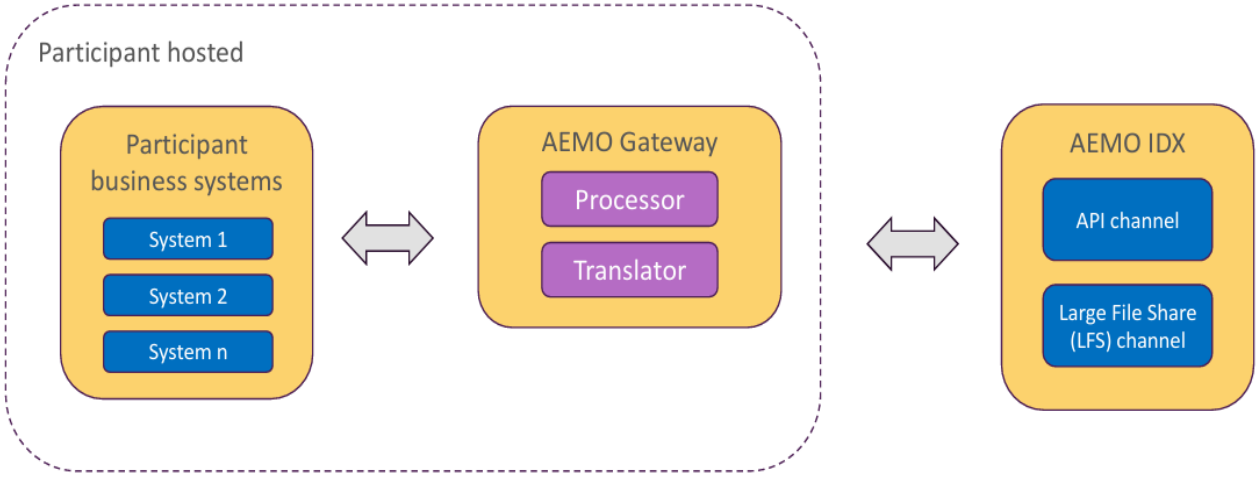
Message size, message transformation requirements, and throughput may change some of these requirements. Each organisation should understand their workload and non-functional requirements, such as performance and availability, to support a considered deployment design.

5.4 Who can use this software?

The AEMO Gateway software supports the following market participants who leverage the BPQD Interface to exchange Power Quality Data deployed on IDX Foundation:

- Local Network Service Provider (LNSP) - Recipients of BPQD.
- Metering Coordinator (MC) - Responsible for sending BPQD.
- Metering Data Provider (MDP) - Send BPQD on behalf of the MC.

5.5 How do you use the software?



The AEMO Gateway software transfers power quality data between the IDX BPQD API (OAuth 2.0 client credentials) and participant local directories.

Each participant and their business systems may have different enterprise integration requirements. Using the available connectors, participants can configure the software to connect to back-end systems to retrieve data or deliver data.

When you use local directories, each directory can send or receive all files or a selected subset of files. In the default operating mode, the software removes a file from the source directory after it successfully delivers the file to the target system. You can configure each processing thread to optimise performance and meet business requirements.

The software runs as a batch application and does not provide a graphical user interface. You configure the Gateway software by using a properties file.

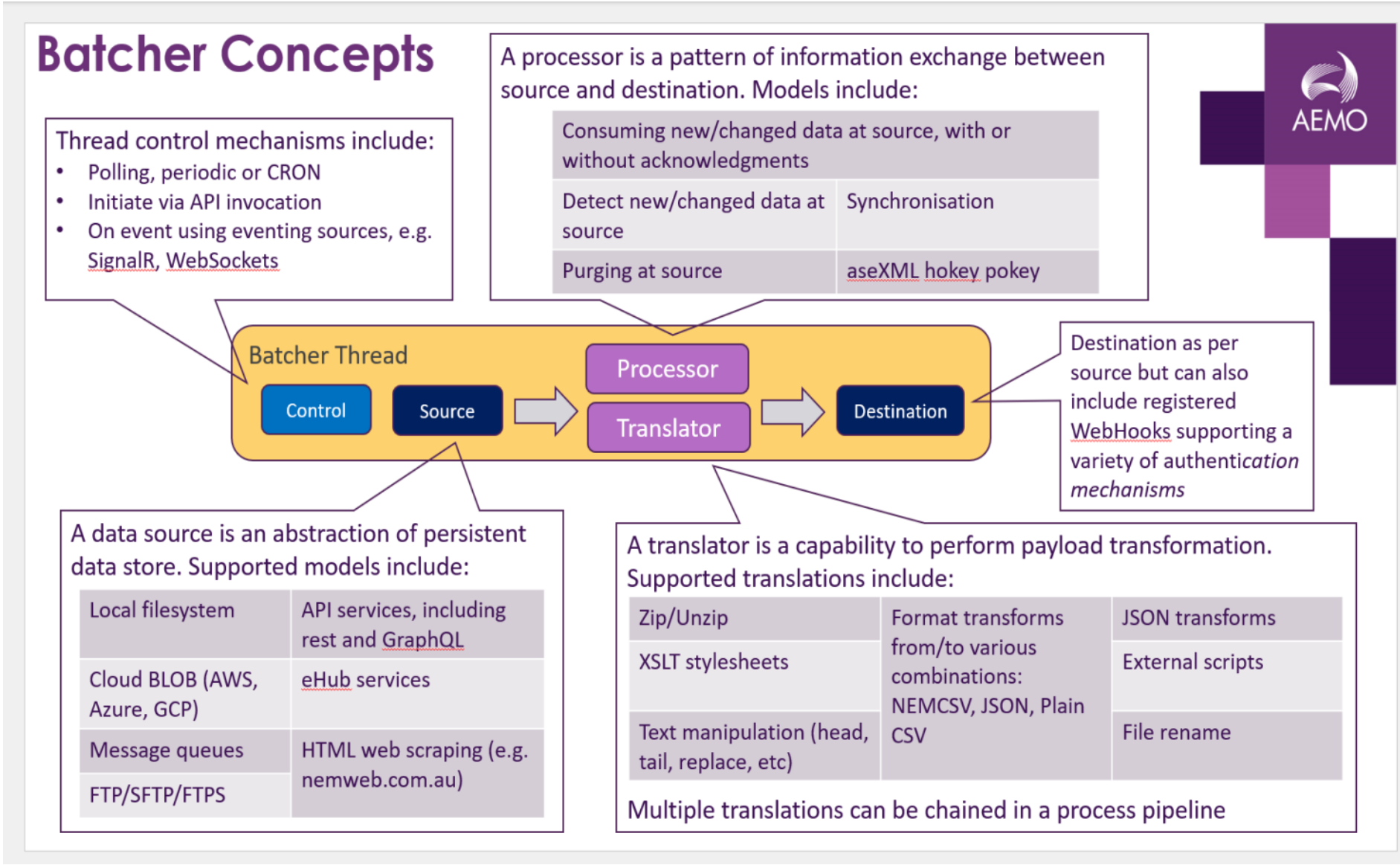
The application reads the properties file only at startup. To apply configuration changes, restart the application.

5.6 How it works

The AEMO Gateway Software is a highly configurable, multithreaded application that moves data from a source system to a target system.

AEMO Gateway comes preconfigured with a default configuration that provides a set of local staging directories. As default, the Gateway Enterprise Integration Point includes an Inbox, Inbox Archive with Done and Nack, and an Outbox. Participants can configure the install to meet their specific requirements.

The figure below provides a high-level overview of the Gateway Software architecture.


Batcher Concepts
A processor is a pattern of information exchange between source and destination. Models include:
Consuming new/changed data at source, with or without acknowledgments
Detect new/changed data at source
Synchronisation
Purging at source
aseXML hokey pokey
Thread control mechanisms include:
Polling, periodic or CRON
Initiate via API invocation
On event using eventing sources, e.g. SignalR, WebSockets
Batcher Thread
Control
Source
Processor
Translator
Destination
Destination as per source but can also include registered WebHooks supporting a variety of authentication mechanisms
A data source is an abstraction of persistent data store. Supported models include:
Local filesystem
API services, including rest and GraphQL
Cloud BLOB (AWS, Azure, GCP)
eHub services
Message queues
HTML web scraping (e.g. nemweb.com.au)
FTP/SFTP/FTPS
A translator is a capability to perform payload transformation. Supported translations include:
Zip/Unzip
Format transforms from/to various combinations: NEMCSV, JSON, Plain CSV
JSON transforms
XSLT stylesheets
External scripts
Text manipulation (head, tail, replace, etc)
File rename
Multiple translations can be chained in a process pipeline

6 Installation

6.1 Prerequisites

6.1.1 Environment prerequisites

The AEMO Gateway Software requires:

- A hosted infrastructure to support the installation of the software

6.1.2 Working directories

On the local machine, the AEMO Gateway Software requires working directories to support the default integration point for participant systems to exchange messages with AEMO Gateway. Participants can reconfigure the integration to their enterprise systems using any of the available connectors.

The installation process creates a default set of working directories (internal for AEMO Gateway Software).

To prevent generated content, such as log files, from consuming available disk space, configure a file-purging process on the local machine.

6.1.3 IDX connectivity

BPQD API

For machine-to-machine access to the PQD and IDX APIs, participants must create dedicated URM accounts. Assign only the IDX entities required for the intended API use. Do not use these accounts to access IDX services through Markets Portal interfaces.

Participants must notify AEMO of all service accounts used for machine-to-machine access so AEMO can enable OAuth 2.0 client-credentials authorisation.

TLS certificates for mTLS & non-repudiation

IDX requires two TLS certificates to support non-repudiation and certificate rollover. Participants must designate one certificate as primary and one as secondary. Only one certificate can hold each role at any time.

Both certificates must be AEMO-signed TLS certificates issued under the AEMO-RCA-G2 root.

Before assigning the primary and secondary roles to TLS certificates, participants must have:

- Access to the Markets Portal and the TLS Certificate Management interface with the appropriate URM permissions
- The ability to generate certificate signing requests (CSRs) and obtain two AEMO-signed TLS client certificates using the TLS Certificate Management application or the TLS Certificate Management API

Certificates appear in the participant's certificate inventory in the Markets Portal.

For the BPQD release, participants must request AEMO to set their certificates as primary or secondary.

6.1.4 Manual keystore file creation

To establish secure connectivity with the IDX Hub, participants must provide a PKCS#12 (.pfx) keystore for use with the AEMO Gateway Software.

The keystore supports the following security functions:

- Mutual TLS (mTLS) authentication with AEMO APIs.
- Validation of inbound IDX messages.
- Optional signing of outbound IDX payloads, assuming the same private key is used.

The AEMO Gateway Software uses a single PKCS#12 file as both a keystore and truststore file.

Keystore concept

A PFX file is not only a certificate file. It contains multiple entries:

- Identity (for mTLS and optional outbound signing outbound IDX messages)
  - Private key
  - Participant TLS certificate
  - Certificate chain (Intermediate CA and Root CA)
- Trust (for non-repudiation validation of inbound IDX messages)
  - IDX Hub public certificate
  - Its certificate chain
- Optional supporting trust

- Root certificates such as DigiCert

Required keystore structure

The final PFX file must contain multiple certificate entries:

1. Identity Entry

```
Private Key
-  Participant Certificate (leaf, e.g. NEMMCO-NonProd)
-  Intermediate CA
-  Root CA
```

2. Trust Entry – IDX Hub (non-repudiation validation)

```
IDX-HUB-<Environment>
-  Intermediate CA
-  Root CA
```

- Pre-production IDX Hub Certificate

  If you are connecting to the pre-production IDX Hub, **download** the IDX-HUB-NonProd leaf, intermediate, and root certificates.

- Production IDX Hub Certificate

  If you are connecting to the production IDX Hub, **download** the IDX-HUB-Prod leaf, intermediate, and root certificates.

3. Additional Trust Entries

   External CAs (for example DigiCert Intermediary):

```
DigiCert Global G2 TLS RSA SHA256 2020 CA1
```

   Download the DigiCert Intermediary certificate.

The keystore structure requires:

- **The identity chain includes the private key.**
- **The IDX Hub certificate is included separately.**
- **The DigiCert intermediary must be added to ensure MTLS trust.**

Missing any of these requirements may cause failure.

Keystore creation

Before you begin, identify your starting point.

Scenario A

You already have:

- A private key
- An AEMO-issued certificate

Proceed below to **Instructions** step 4. Prepare certificate chain.

Scenario B

You must create a new certificate by:

- Generating a private key
- Generating a CSR
- Submitting CSR via TLS Certificate Management App
- Downloading the certificate

Proceed below to **Instructions** step 1. Generate private key.

Instructions

To create a keystore file, you need to:

1. Generate private key

   Use a tool such as OpenSSL to generate a private key.

   Never share your private key.

```
openssl genpkey \
-algorithm RSA \
-pkeyopt rsa_keygen_bits:2048 \
-out private_key.pem
```

   Requirement:

   - RSA 2048-bit key

2. Generate CSR

   Use a tool such as OpenSSL to generate a CSR from your private key.

```
openssl req -new \
-key private_key.pem \
-out certificate.csr \
-subj "/CN=<YOUR_CERT_NAME, e.g. NEMMCO-NonProd>" \
-sha256
```

The Common Name (CN) must follow this format:

- ParticipantID>-<NonProd | Prod> e.g. NEMMCO-NonProd

3. Generate a certificate

   Submit CSR via TLS Certificate Management App as a new certificate, or a reissue from an existing certificate order. Download your new certificate:

   - **Create a new certificate**
   - **Reissue a certificate**

4. Prepare certificate chain (identity chain)

   Obtain the following files:

   - Participant certificate - participant.crt (e.g. NEMMCO-NonProd)
   - Intermediate CA - intermediate.crt (e.g. AEMO-ICA-TEST G1)
   - Root CA - root.crt (e.g. AEMO-RCA G2)

   Create chain file with the intermediate and root certificates:

```
cat intermediate.crt root.crt > ca-chain.crt
```

5. Create base PFX (identity chain)

   Use a tool such as OpenSSL to generate the PFX file containing your private key, participant.crt and the ca-chain.crt.

```
openssl pkcs12 -export \
-out aemo-gateway.pfx \
-inkey private_key.pem \
-in participant.crt \
-certfile ca-chain.crt \
-name "participant-key"
```

You will be prompted for a password.

Note the following:

- You must use this password during the Gateway installer process.
- The certificate order must be leaf, intermediate, then root.

You now have a basic Keystore file for mTLS connectivity with your private-key associated with the certificate chain:

- private_key.pem (private key)
- participant.crt (participant certificate (e.g. leaf -> NEMMCO-NonProd)
- intermediate.crt (intermediate CA)
- root.crt (root CA)

6. Add IDX Hub certificate (Trust chain)

Download the IDX Hub certificate for your environment:

- **Pre-production**
- **Production**

Add all three IDX Hub certificates in the chain which include leaf, intermediary, and root. Use a tool such as Keytool to import into your keystore PFX file.

The structure of the Leaf certificate:

```
keytool -importcert \
-file idx-hub-cert.crt \
-keystore aemo-gateway.pfx \
-storetype PKCS12 \
-alias idx-hub \
-noprompt
```

The structure of the Intermediate certificate:

```
keytool -importcert \
-file intermediate.crt \
-keystore aemo-gateway.pfx \
-storetype PKCS12 \
-alias idx-hub-ica \
-noprompt
```

The structure of the Root certificate:

```
keytool -importcert \
-file root.crt \
-keystore aemo-gateway.pfx \
-storetype PKCS12 \
-alias idx-hub-root \
-noprompt
```

Add the DigiCert trust certificate by **downloading** the DigiCert Intermediary certificate.

Use a tool such as Keytool to import into the keystore (your PFX file).

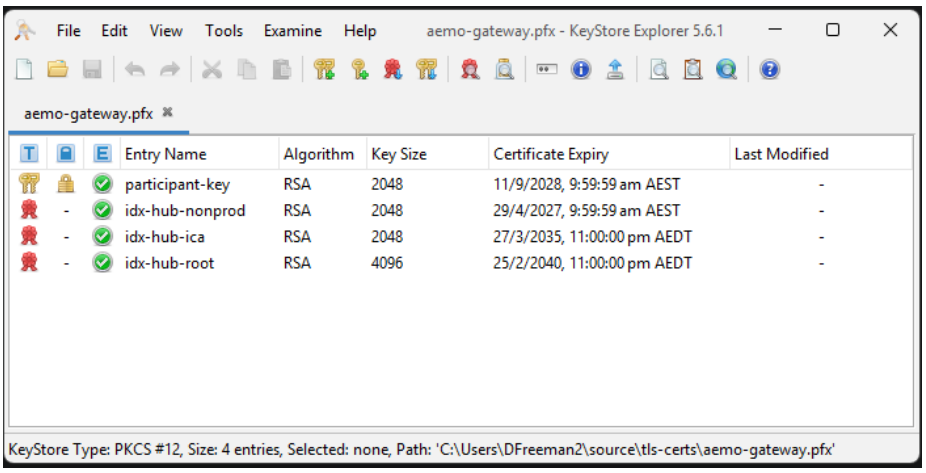
The structure of the DigiCert certificate:

```
keytool -importcert \
-file digicert-intermediary.crt \
-keystore aemo-gateway.pfx \
-storetype PKCS12 \
-alias digicert-intermediary \
-noprompt
```

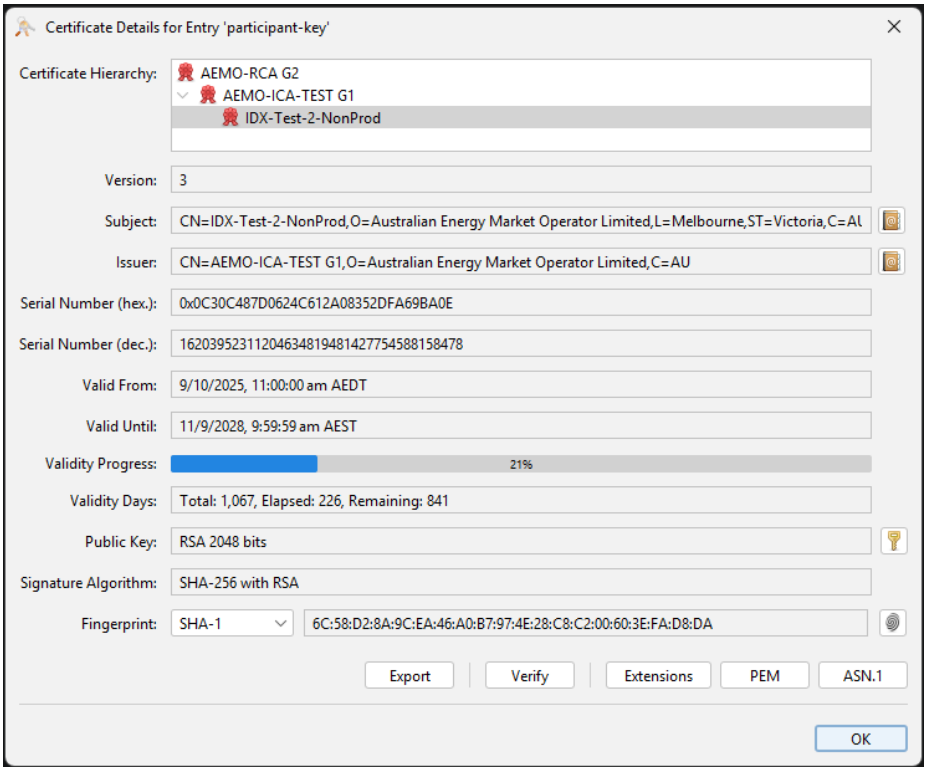
The IDX Hub certificate is required for payload non-repudiation validation.

This must be a separate entry, not part of an identity chain (e.g. participant-key).

Example end-state of the private-key (participant-key) and truststore certificates in the keystore/truststore file:



Example Private key (participant-key) and associate keystore certificate chain:



7. Validate the keystore

   a. Check contents

Use a tool such as Keytool to ensure the contents of your keystore file are correct.

Confirm that the keystore contains:

- Private key entry
- IDX Hub entry

   b. Inspect certificate structure

Use a tool such as OpenSSL to check your PFX certificate structure:

```
openssl pkcs12 -in aemo-gateway.pfx -nokeys -info
```

You should observe:

- Identity chain

```
Participant Private Key Entry (with Participant certificate,
Intermediate and Root)
```

- IDX Hub chain

```
IDX-HUB certificate
Intermediate certificate
Root certificate
```

- Additional Trust certificates

```
DigiCert Global G2 TLS RSA SHA256 2020 CA1
```

Use in AEMO Gateway Installer

Provide the following during installation:

- TLS keystore file: aemo-gateway.pfx
- Keystore password

The installer validates:

- Presence of a private key.
- Correct certificate chains.
- Ability to establish mTLS connectivity.

Common errors and resolutions

Common errors during this process are:

- Missing IDX Hub certificate
  - Symptoms: payload validation failures, message signature errors
  - Resolution: re-import the IDX Hub trust chain
- Incorrect certificate chain
  - Symptoms: TLS handshake failure, installer rejects keystore
  - Fix: ensure the order is Leaf → Intermediate → Root
- Trusted certificate not found
  - Symptoms: no trusted certificate found, during GUI installation
  - Fix: ensure DigiCert intermediary is added
- Private key mismatch
  - Symptoms: signing fails, IDX rejects messages

  - Fix: confirm the certificate matches the private key, or reissue the certificate
- Incorrect keystore format
  - Symptoms: installer cannot read file
  - Fix: ensure the file is PKCS#12 format (.pfx)

Checklist

Before using the keystore, participants must confirm that:

- The private key is included.
- The participant certificate is included.
- The Intermediate and Root chain is included.
- The IDX Hub certificate is included.
- The IDX Hub chain is included.
- The DigiCert intermediary is added.
- The correct password is recorded.
- The keystore has been validated using OpenSSL or Keytool

Key takeaways

The key takeaways are:

- The keystore acts as both an Identity store and a Trust store.
- It must contain multiple certificate chains.
- Certificate order and completeness are critical.
- Missing trust entries cause runtime failures.
- This process ensures:
  - mTLS connectivity
  - Message signing (optional)
  - Signature validation (optional)

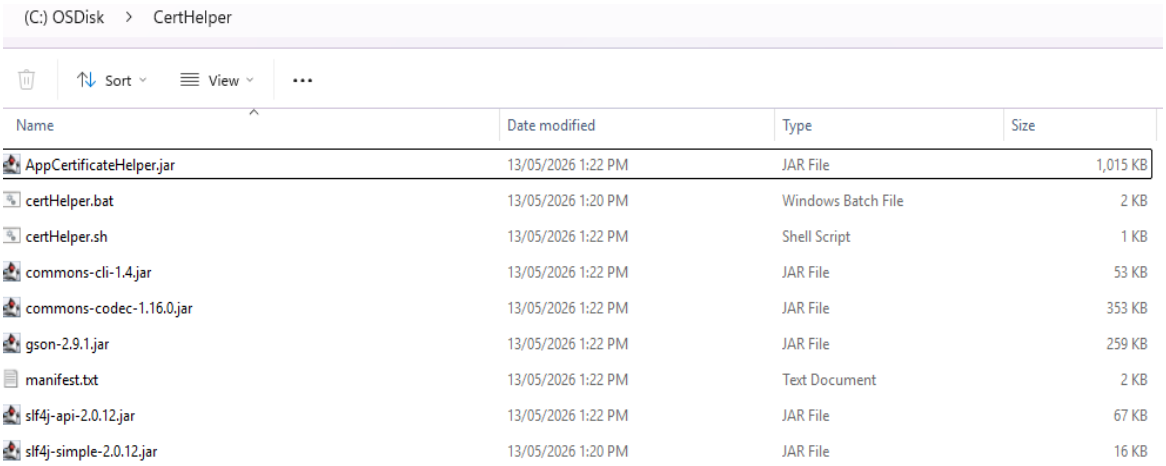
6.1.5 Certificate Helper tool

The Certificate Helper is not included in the first release. The tool will be available in a future release, with more information to come.

AEMO provides a Certificate Helper tool to assist participants with creating certificate signing requests (CSRs) and simplifying the process of preparing a suitable keystore file.

To access the help utility within the Certificate Helper, you need to:

1. Unzip the AppCertificateHelper.zip file to create the CertHelper folder:



2. Navigate to the CertHelper directory:

```
C:\CertHelper>certHelper.bat
```

3. From the command line prompt, run the following command:

```
certHelper.bat --help
```

```
C:\CertHelper>certHelper.bat --help
Executing Certificate Helper
Certficiate helper provides commands to create a certificate keystore suitable for an AEMO gateway installation

usage: au.com.aemo.Common.Certificate.AppCertificateHelper
 -al,--alias <arg>              The alias of a certificate
 -ch,--chain <arg>              Import certificate and add to chain in
                                keystore
 -cr,--create                   Create certificate and add to keystore
 -csr,--createAndSignReq <arg>  Create certificate and add to keystore
                                and create certificate signing request
 -e,--export <arg>              Export certificate to file
 -h,--help                      Show help information
 -i,--import <arg>              Import certificate and add to keystore
 -l,--list                      List certificates
 -p,--password                  Show keystore password
 -pr,--prepare <arg>            Prepares an existing keystore by adding
                                certificate chain and payload signing
                                certificates
 -r,--remove <arg>              Remove certificate from keystore
 -rc,--replyAndChain <arg>      Import CA reply from certificate signing
                                request and chain intermediate and root
                                certificates
 -rp,--reply <arg>              Import CA reply from certificate signing
                                request
 -sr,--certsignreq              Create certificate signing request

Process to create certificate
1. Create certificate and signing request using the --create option
2. Provide the signing request to AEMO using TLS management option in the Markets portal
3. Download signed request
4. Import the signed request using the --reply option with the file name of the signed request
5. Get TLS keystore file path and keystore password details using --list command which are required on installer

Process to update an existing keystore file with an existing signed certificate
1. Add CA and payload signing certificates using the --prepare option
2. Get TLS keystore file path and keystore password details using --list command which are required on installer

AEMO support contact details Email: Supporthub@aemo.com.au and Phone: 1300 236 600
```

Creating a new certificate

To create a new TLS certificate using the Certificate Helper tool:

1. Run the following prompt:

```
certHelper.bat --create
```

   - Participants are then prompted to provide the following:
     - Keystore file name
     - Participant Id
     - Environment: production (Prod) or non-production (NonProd)
     - Validity of the key (in days format)
     - Any extended attributes for the private key
     - Proxy hostname
     - Proxy port
2. Provide the signing request to AEMO using TLS Management Option in the Markets Portal.

3. From the TLS Management tool in Markets Portal, download the AEMO signed reply.
4. Run the following command, supplying the downloaded CSR reply file name:

```
certHelper.bat --reply <CSR-reply-file-name>
```

5. The certificate Helper then imports CSR reply and chains intermediate and root certificates within the keystore file.
6. Run the following command, to retrieve keystore name, path, and password:

```
certHelper.bat --list
```

Using an existing certificate

For participants to use an existing certificate, they need:

1. A signed certificate.
2. The certificate in a PKCS12 keystore file.
3. The password for the keystore file.
4. Internet access for the Certificate Helper to download required certificates.

For participants to utilise an existing certificate with the Certificate Helper, they need to:

1. Run the following command, supplying the existing keystore file name:

```
certHelper.bat --prepare <keystore-file-name>
```

2. Provide the password for the keystore file, and proxy hostname and port if required.
3. The Certificate Helper then downloads and chains the relevant CA certificates. The tool also downloads and imports the payload signing certificate to the keystore file.
4. Run the following command, to retrieve keystore name, path, and password:

```
certHelper.bat --list
```

6.2 AEMO Gateway GUI Installer

6.2.1 Downloading the application

The latest version is the AEMO Gateway Installer 8.0. For download locations, see **Data Interchange resources**.

Decompress the .ZIP file to a work folder to create a .JAR file. The .JAR file is the installation file referenced elsewhere in this document.

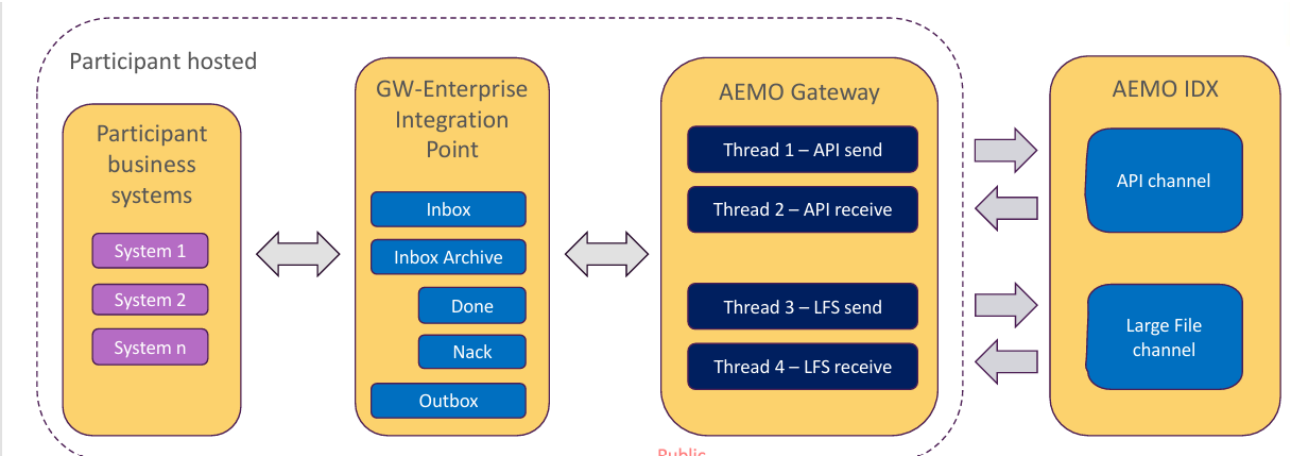
6.2.2 Environment prerequisites

The Gateway Software needs:

- A Java Runtime Environment (JRE) compatible with the certified version for this application.
- OAuth 2.0 credentials, including a Client ID and Client Secret, for authentication with IDX APIs.
- Valid TLS certificates to establish secure connections with IDX endpoints.
- A public signing certificate to support message signing and verification when interacting with the IDX Hub.

6.2.3 Overview

The base install of AEMO gateway is intended to provide a simplified filesystem-based integration as a starting point which can be adjusted by configuration to any of the supported channels as required.



Configuration Model for AEMO Gateway base install – data sources

6.2.4 Installation

To install and configure the AEMO Gateway Software application:

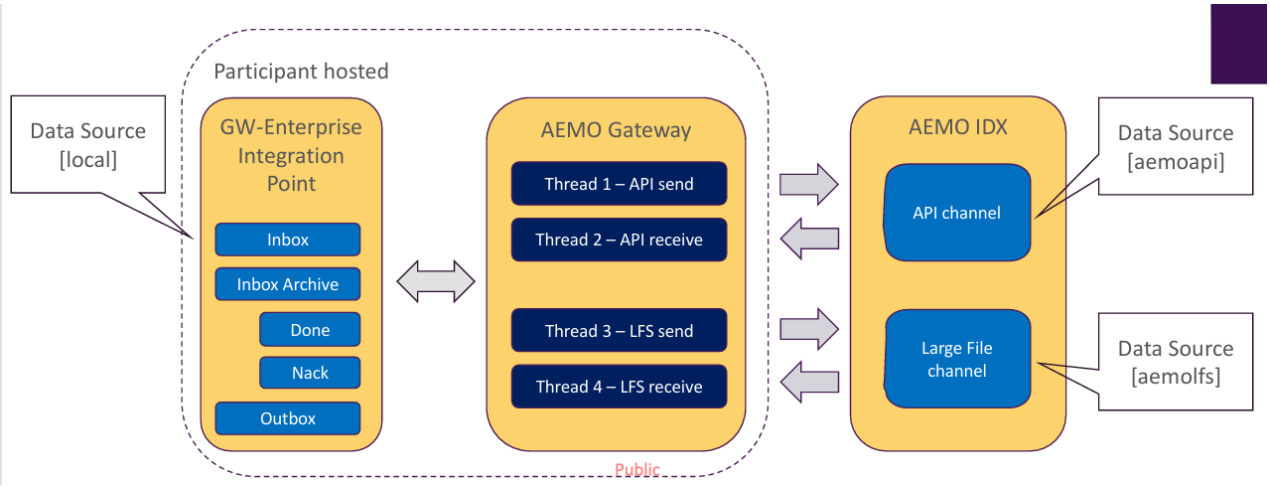
1. Validate the JRE installation is Open JDK 21 by running the following in a command prompt:

```
java -version
```

The response should be similar to:

```
openjdk version "21" 2023-09-19
OpenJDK Runtime Environment (build 21+35-2513)
OpenJDK 64-Bit Server VM (build 21+35-2513, mixed mode, sharing)
```

2. If a different java version is detected, refer to the documentation to install the certified JRE



version.

3. Validate the jar file signature. From the command line prompt, run the following command:

```
jarsigner -verify "Participant Data Replication Batcher IDX GUI
Installer v8.0.jar"
```

For more information, see **https://docs.oracle.com/en/java/javase/21/docs/specs/man/jarsigner.html**.

4. Start the installer using one of the following methods:

- Double-click the JAR file in a Windows environment, if *.jar files are associated with a Java Runtime Environment (JRE).
- Run the installer from the command line:
  - Open a command prompt.

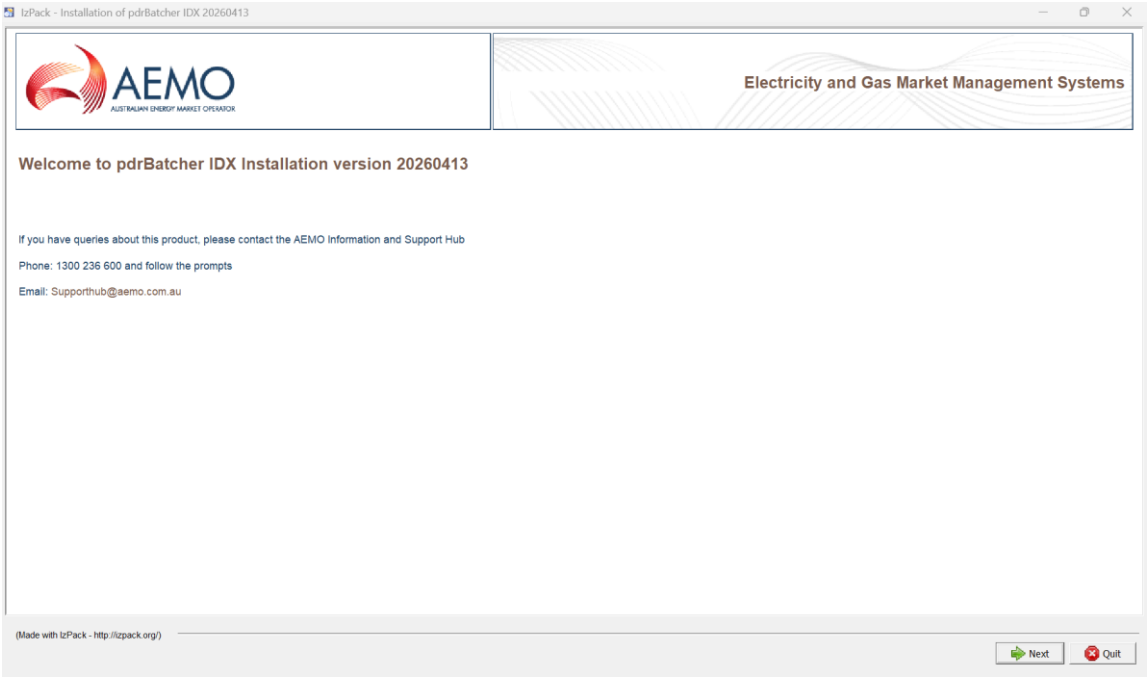
- Navigate to the folder where the installer is stored.
- Run:

```
java -jar <insert installer name here>.jar
```

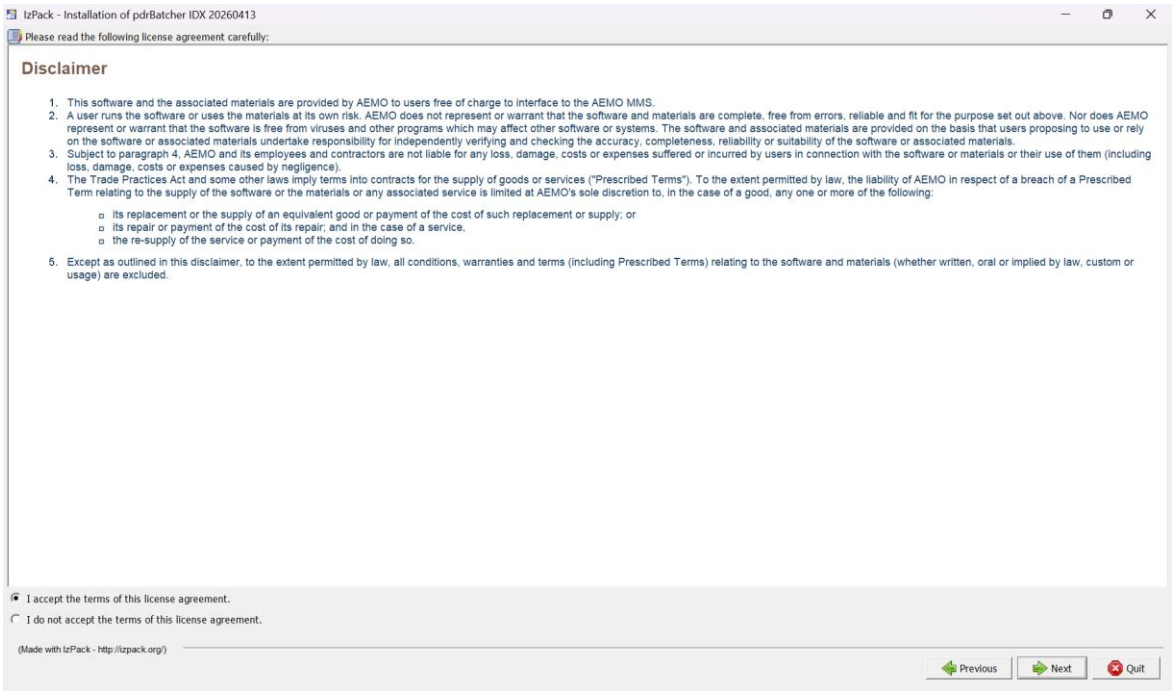
- For headless environments, such as Linux, run the installer in a terminal session using the -console flag:

```
java -jar <insert installer name here>.jar -console
```

- Provides the same capabilities as the GUI installer but runs entirely in text mode.



5. Accept the terms of the licence agreement and click **Next**.



6. Select the installation detail by clicking the required checkboxes and click **Next**.

- pdrBatcher Core: Mandatory, being the standard software installation.
- WebSocket: Select this option to enable WebSocket notifications for outbound messages. This reduces the latency of outbound data. WebSockets functionality is not yet available from the core platform. AEMO will provide details on when the WebSocket service becomes available.
- Windows Service: Select this option if the AEMO Gateway Software is to run as a Windows service. This option is unavailable for installations on unix-type operating systems.

- Secured API services: The secured API services option allows the installation to be configured with a self-signed certificate to encrypt API communications from PDR monitor to the Gateway Software. AEMO recommends you select this option for cyber security purposes. De-selecting this option results in plain text HTTP communications between PDR Monitor and the Gateway Software.

  The self-signed certificate has a 365-day expiry period. Participants need to re-generate a new certificate before the expiry date to allow monitor communications to continue.

  Participants may also choose to use a certificate issued by an external Certificate Authority, however AEMO does not support this as a part of the AEMO GUI installer.

  In this case, participants must register their certificate manually using command line tools.

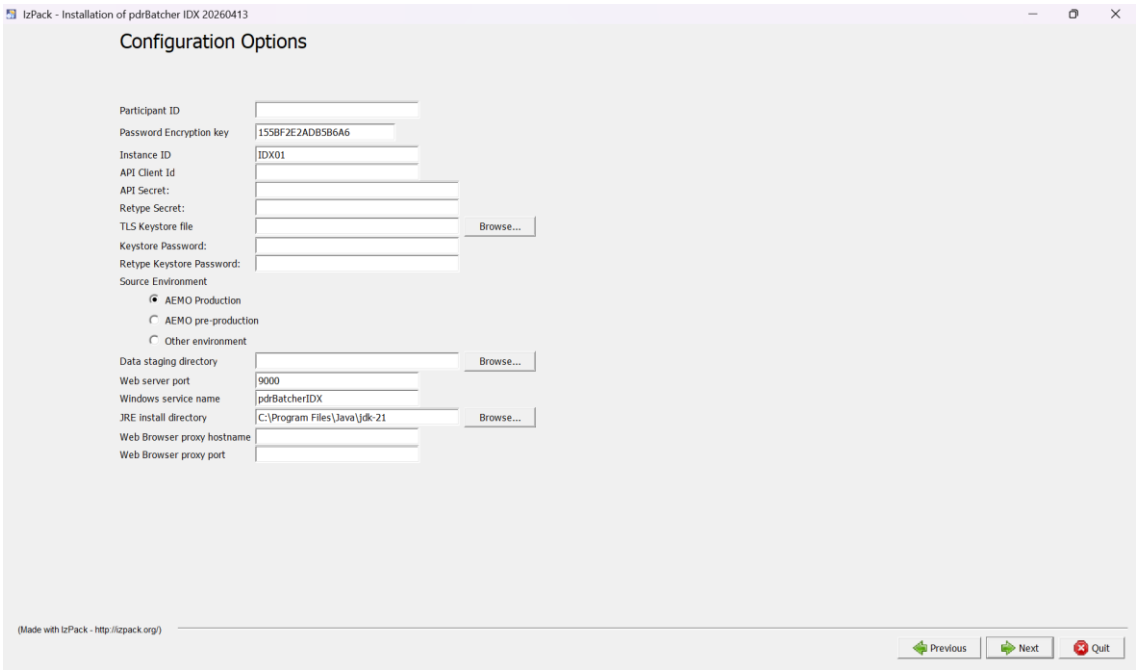
  Please refer to the certificate management section of the user guide for further information on certificate management.

- Advanced: Select this option to override advanced configuration properties such as the Windows Server name or the location of the Java Runtime Environment used to run the application. You can also select this option if your organisation uses a proxy for http traffic.

7. Select the location to install the application and click **Next**.

If you want to use an existing directory, click Browse and select the installation location.

8. Enter the required configuration details and click **Next**.

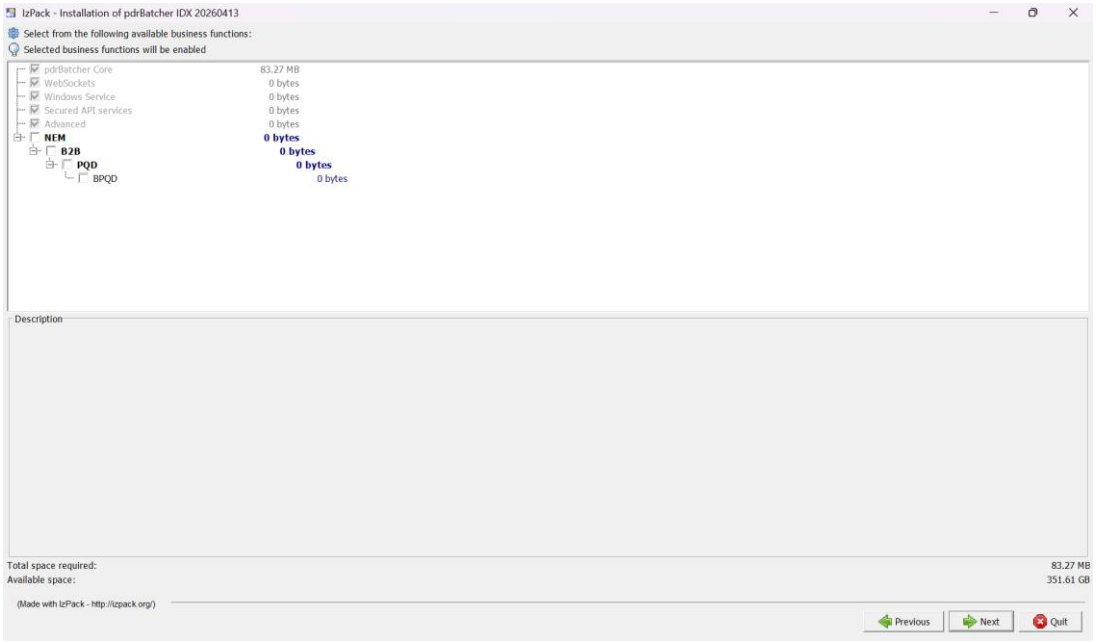


The required fields are:

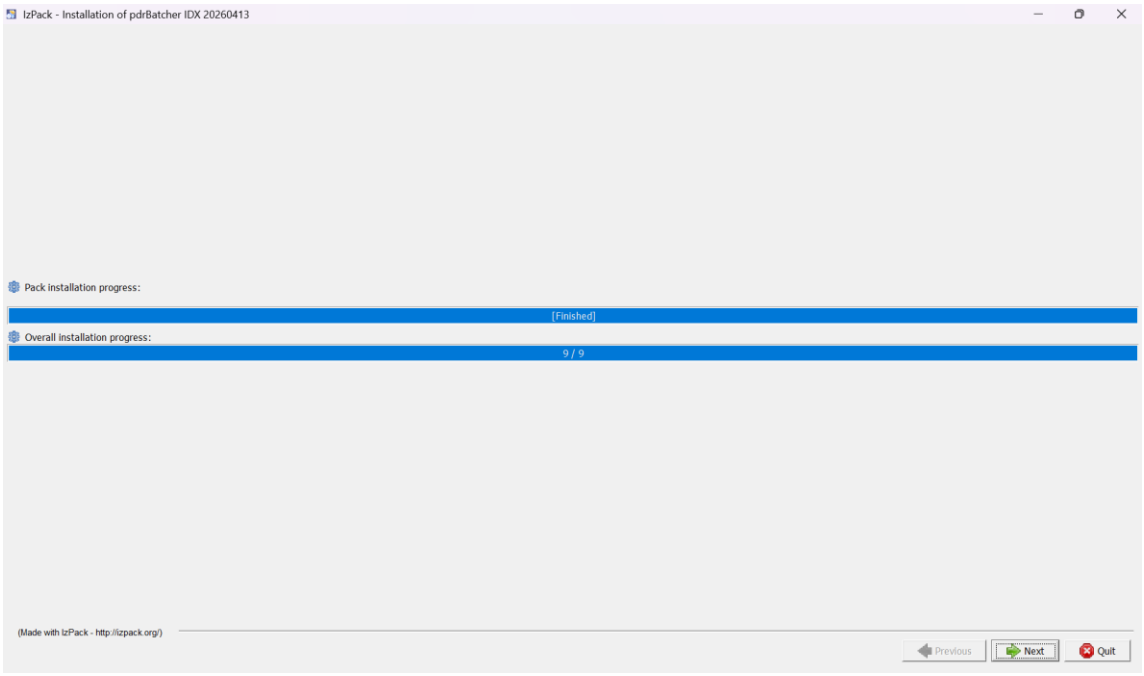
- Participant ID: Your participant identifier.
- Password Encryption Key: A 16-digit hexadecimal code (A-F,0-9) to secure your password.
- Instance ID: Instance identifier that uniquely identifies this Gateway Software installation.
- API Client Id: The OAuth ClientId that is used for connection to the AEMO IDX Hub. This is created and managed by the participant's Administrator in URM.
- API Secret: The OAuth secret that is associated with the ClientId used for connection to the AEMO IDX Hub. This is created and managed by the participant's Administrator in URM.
- TLS Keystore file: A keystore file that contains the required certificates for connectivity and validation of payload signatures. The Certificate Helper will assist in creating the required keystore file. The required certificates are:
  - A private key that has been signed by AEMO (refer TLS management in Markets Portal).
  - Intermediate and root certificates that were used for the certificate signing response.
  - Public key that is used by the AEMO IDX Hub for signing outbound payloads.
- Keystore Password: The password that protects the TLS keystore file and allows the AEMO Gateway to access the private key and certificate stored in the keystore.
- Source Environment: Select either AEMO production, AEMO pre-production, or Other environment. Selecting Other environment allows you to enter a customised IP address.

- Data Staging Directory: The data staging directory is a folder on the local filesystem that is used as an integration point between a participant's systems and the AEMO Gateway. A series of folders will be created under this directory (Inbox/Outbox) by the installer that can be used for sending and receiving data to the IDX hub.
- Web Server Port: Choose a port number for the web services interface. This must be unique across all installations of Gateway Software and pdrLoader on this server.
- Windows Service Name: Required if the Advanced installation option is selected. The default Windows service name is pdrBatcherIDX.
- JRE install directory: required if the Advanced installation option is selected. The home directory of the Java Runtime environment used to run the Gateway Software application.
- Web Browser proxy hostname: If your organisation uses a proxy to access AEMO services, enter the proxy host here otherwise leave blank. Refer to your organisation's platform support team to see if this is relevant.
- Web Browser proxy port: If your organisation uses a proxy to access AEMO services, enter the proxy port here otherwise leave blank. Refer to your organisation's platform support team to see if this is relevant.

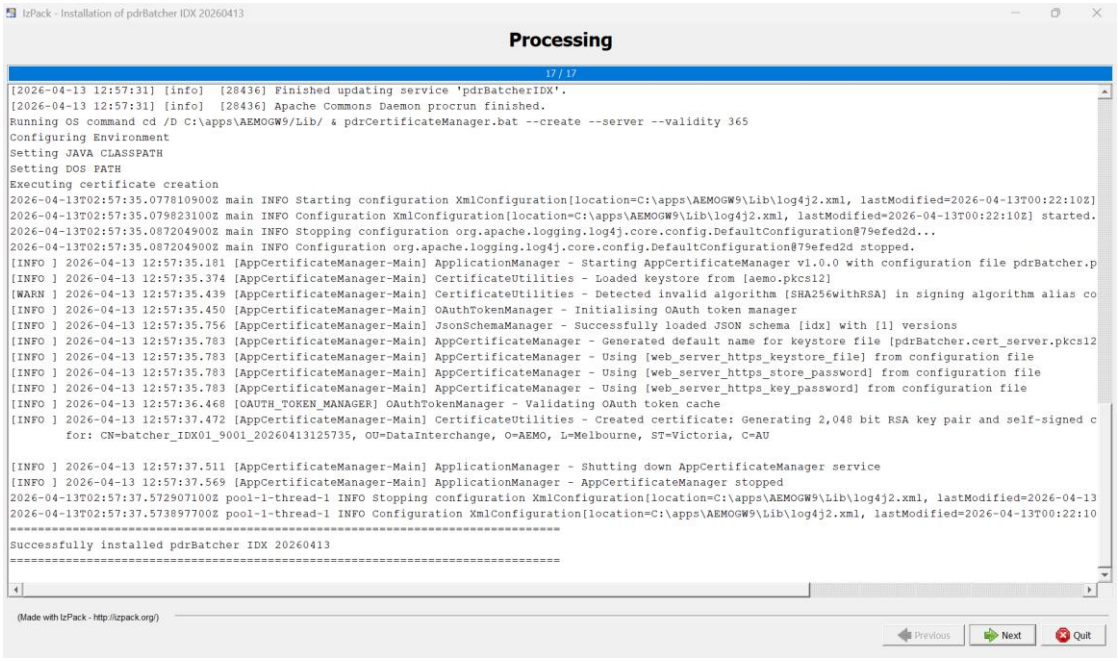
9. Select your relevant business function and click **Next**.



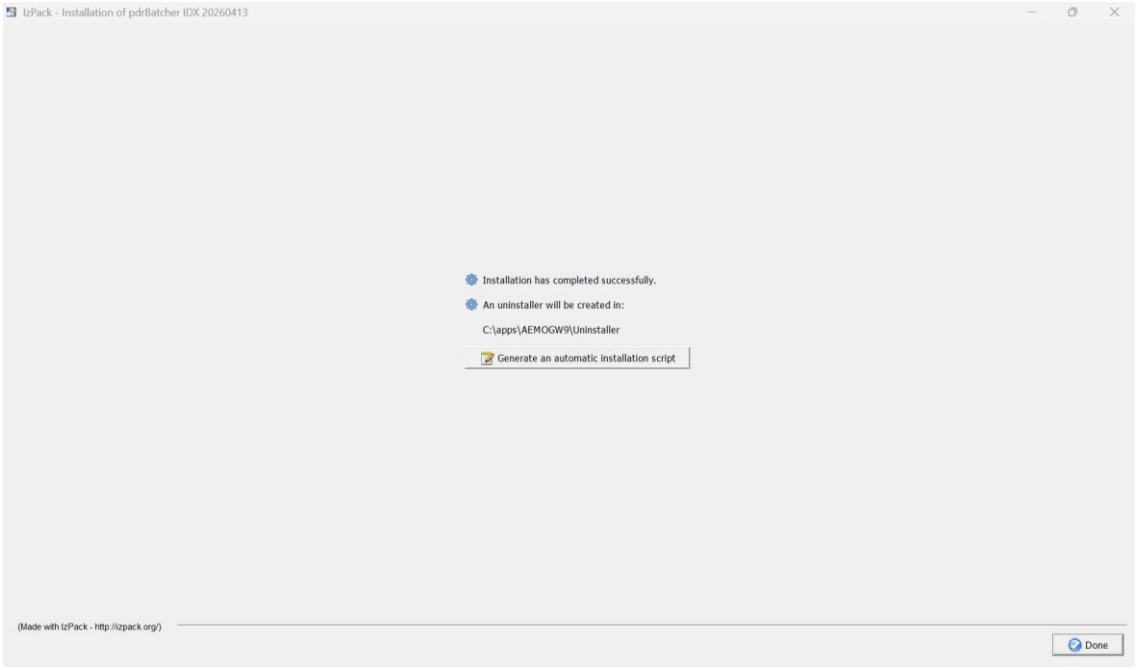
10. The required folders for the application are then installed, reflected in the installation progress display. Upon completion of this process, click **Next**.



11. The processing form displays, configuring your software installation according to the selected option and settings. If an error occurs, click **Previous** and correct the required settings.



12. Once the installation is complete, the finish form displays. Click **Done** to close the installer.



The Generate an automatic installation script button is an advanced option used to generate a configuration file used for silent installs. It is recommended for advanced users running multiple installations.

6.2.5 Testing your installation

Once your installation is complete, you can test it is working correctly using the following steps:

1. Run the pdrConnectionTest script in the Lib folder and confirm connectivity to all defined data sources.

2. Start the application using the method most suitable for your installation and environment:

   - For a Windows service installation, start the service from the windows service panel.

   - For a Windows console installation, start the application by double-clicking the pdrBatcher.bat in the Lib subdirectory.

   - For a Linux installation, start the application by running the pdrBatcher.sh shell script in the Lib directory.

3. Check the contents of the pdrBatcher log file located in the Log folder and ensure that there are no messages with an ERROR status. In the event of ERROR messages, check all parameters, your environment, and repeat the installation if necessary.

6.3 Manual

6.3.1 Application

1. Validate the JRE installation is Open JDK 21 by running the following in a command prompt:

```
java -version
```

The response should be similar to:

```
openjdk version "21" 2023-09-19
OpenJDK Runtime Environment (build 21+35-2513)
OpenJDK 64-Bit Server VM (build 21+35-2513, mixed mode, sharing)
```

2. Unzip the ZIP file into a directory.

3. Run the following command, after unzipping, from a command line prompt in the Lib folder:

```
jarsigner -verify "AppPdrBatcher.jar"
```

4. Remove the delete.me files.

5. Edit the pdr_key.properties file in the \lib subdirectory and replace the default values with a 16-digit hexadecimal string of your own choice.

   Before starting to configure your installation, ensure you update the 16-digit hexadecimal key in the pdr_key.properties file. Changing the encryption key (hash) ensures any encrypted passwords in your properties file cannot be decrypted using the default and publicly available hash.

6. Edit the pdr_log4j.xml file in the \lib subdirectory and update the installation-directory parameter to redirect the log output file to the desired location.
   Escape Windows style directories using the backslash operator need to read properly from the properties configuration file.
   For example:

```
<Property name="installation-directory">c:\\pdr\\batcher</Property>
```

You can enter Unix paths using the forward slash operator directly as they do not require an escape sequence.
For example:

```
<Property name="installation-directory">/home/pdr/batcher/</Property>
```

7. Edit the pdrEnvironment batch file to ensure the PATH environment variable includes a reference to the directory where the Java executable is installed on your system.

The AEMO Gateway scripts assume JRE Open JDK 21. If you have another version of Java, ensure you change this batch file to reference the appropriate installation directories to suit. Placing the directory of the Java install at the start of the PATH environment variable ensures the AEMO Gatway always runs against the correct version of Java. Scripts are samples and not a recommendation or requirement.

The design and implementation of a deployment plan is the responsibility of each participant. For example, considering the pdrEnvironment.sh wrapper script in a Unix environment, a practice might be to adapt the script into the Unix services layer (generally in /etc/rc.d/init.d/* but may be different depending on the flavour of Unix) and configure it to execute at an appropriate run level at start-up.

6.3.2 Folders

Decompressing the distribution file creates a directory (\pdr\batcher) with five subdirectories, collectively called the local directories. One directory (\Lib) contains the application and the other two are working directories for AEMO Gateway:

1. Holding: Where the Gateway saves files being downloaded from the AEMO file server, then (when complete) the Gateway moves each file to the local directory for subsequent access.
2. Log: Contains log files associated with the transfer process.
3. Performance: Where the pdrBatcher stores performance and logging data for ingestion into pdrMonitor.

The application files in the \Lib directory include:

- The main application JAR file (AppPdrBatcher.jar), a range of third party JAR files upon which application depends, and one DLL file (NTEventLogAppender.dll), being the application files.
- Three properties files (pdrBatcher.properties, pdr_log4j.xml and pdr_key.properties), to provide parameters for the running instance.
- pdrBatcher.*, to run the application in Windows or Unix environments
- pdrEnvironment.* containing the definitions of the environment variables required to run the application.
- pdrConnectionTest.* to validate the connectivity to a data source defined in the properties configuration file.
- pdrKeyManager.* to show and cycle the HMAC encryption keys used to secure the internal API services consumed by monitor

- pdrCertificateManager.* to manage the certificate trust stores for consuming secured API services
- pdrThreadExecute.* A sample wrapper to show how to initiate a batcher thread configured to be controlled by API invocation
- pdrBatcherTransformTest.* A utility that allows for testing of transform configuration offline before applying these configuration to the properties file.
- Three files (pdrServiceInstall.bat, pdrServiceUninstall.bat, WinRun4J.jar, WinRun4J.exe, WinRun4J64.exe and WinRun4J.ini) to support the running of the application as a Windows service.
- pdrPasswordEncrypt.* to generate an encrypted password.

For more information on the files within the Lib directory, see **Command line tools**.

The only file in the directories other than \Lib is a placeholder file to cause the creation of required directories during the unzipping process, named delete.me. Remove these placeholder files.

Before attempting to run, edit the two properties files (pdrBatcher.properties and pdr_key.properties) to suit your situation.

When setting up the application, the directory structure does not have to be identical to the arrangement described above. By modifying the .properties file, the installer can distribute the directories over different locations. For example, if setting up a pdrLoader instance, make the AEMO Gateway output directory be the loader input directory.

6.3.3 Logging

The application uses SLF4J as its logging framework with a logging provider of Log4J2. You can adjust the logging configuration to your specific installation requirements.

The two configurations available are:

1. pdr_log4j.xml - for service applications.
   The default configuration uses a RollingFileAppender which creates a daily log file in the application Log sub-directory.

2. log4j2.xml - for command line facilities.
   The default configuration emits log messages to a console-based appender.

See **documentation** for more information about Log4J2.

This document does not explain Log4J2 configuration concepts in detail. If you need to enable DEBUG logging to assist with a support enquiry, follow these steps:

1. Edit pdr_log4j.xml and modify the file-log appender from INFO to DEBUG

```
<Loggers>
<Root level="DEBUG" additivity="false">
<!--
<appender-ref ref="console" level="INFO"/>
-->
<appender-ref ref="file-log" level="INFO"/>
<!--
<appender-ref ref="Win32EventLog" level="ERROR"/>
-->
<!--
<appender-ref ref="AsyncMailer" level="error"/>
-->
</Root>
</Loggers>
```

2. Restart the application service for the changes to take effect.

Please note DEBUG logging generates a large volume of log messages. Only use this mode when required.

The application writes logs using **SLF4J**, passing the messages to Log4J2. You can replace Log4J2 with another logging framework supported by SLF4J if required. This may involve downloading and installing additional jar files in the application's Lib folder.

If you use an alternate logging framework, ensure:

1. The framework is supported and certified for the same JRE as the application.
2. The SLF4J bridge matches the SLF4J API interface version (slf4j-api-*.jar).
3. Log4J2 artefacts are removed from the Lib folder:
   - log4j-slf4j2-impl-*.jar
   - log4j2.xml
   - pdr_log4j.xml
4. An appropriate logging configuration for the alternate framework is available. You may need to insert this configuration into the application service start up parameters.

6.3.4 Windows service

There are varieties 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 (Apache

ProcRun / Commons Daemon) as an example of how running the pdrBatcher as a Windows service can be achieved. Participants can adapt this example to the configuration and software of their own choice.

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

- pdrServiceInstall.bat installs a Java application as a Windows service.
- pdrServiceUninstall.bat removes a Java application running as a Windows service.
- prunsrv.exe is a Windows wrapper for the Java run time environment required to run the pdrBatcher under a 32-bit JRE environment.
- prunsrv64.exe is a Windows wrapper for the Java run time environment required to run the pdrBatcher under a 64-bit JRE environment.
- pdrServiceInstall.ini is a template file containing the configuration data defining the setup of the Windows service.

6.3.5 Docker containerisation

The Docker containerisation examples provide guidance for preparing and managing containerised deployment of the Data Interchange solution elements.

The build scripts are developed to run on a Linux system as an illustration of how to containerise the applications.

Customisation will be required to suit the specific deployment models and environments.

Additional licenced commercial products may be required to support containerised deployment.

These facilities are provided as an example and guide only. The following additional considerations for a production deployment are not covered and are the responsibility of the participant:

- Security
- Backup and redundancy
- Infrastructure as code
- Networking configuration to access MarketNet
- Container east-west communication traffic

Installation prerequisites

The installation prerequisites are:

- A Linux operating system or Windows Subsystem for Linux.
  The scripts use the following commands to manipulate configuration files:
  - dos2unix
  - xmlstarlet
  - xxd
- If these are not available in your Linux distribution, add them using the appropriate package manager, for example:

```
sudo apt install xmlstarlet
```

- Docker desktop - Review hardware specifications required to operate Docker desktop before proceeding with **Download**.

Installation instructions

1. Create a shell session in the Docker subfolder where the application zip was extracted.
2. Run the following command to ensure correct end-of-line encoding for shell scripts.

```
dos2unix *.sh
```

3. Run the script to prepare an application configuration that is consistent with the DockerFile:

```
./1_prepareDockerConfig.sh
```

   This results in an application configuration file:

```
pdrBatcher.properties
```

   and an encryption hash file:

```
pdr_key.properties
```

   and a log4J2 configuration file being prepared:

```
pdr_log4j.xml
```

   Adjust these as required for your specification deployment and requirements.

   Documentation for Log4J2 can be found here:

```
https://logging.apache.org/log4j/2.x/
```

4. Run the script to build the docker image:

```
./2_buildDockerImage.sh
```

   At the conclusion of this script, the available docker images are listed.

Ensure that the image just created is shown in this list

5. Adjust the relevant environment variables in script:

```
3_testContainer.sh
```

6. Run the container by executing command:

```
./3_testContainer.sh
```

Inspect the operation of the application (e.g. log files) with a shell in the running docker container:

```
docker exec -it batcher bash

tail -f /aemo/pdr/batcher/Log/pdr.log
```

7 Configurations Examples

7.1 Oauth2 configuration

The AEMO gateway provides the ability to configure Oauth2 token endpoints for use as an authentication mechanism for “data sources”. It provides services such as caching for tokens, including those defined with a limited scope. AEMO recommends that the client ID and client secret are maintained in a secrets management solution and referenced in the GW configuration.

```
oauth_config_list=aemoidam
oauth_token_eviction_cycle_secs=60 oauth_aemoidam_client_id=xxxx
oauth_aemoidam_client_secret=xxxx
oauth_aemoidam_request_url=https://apis.nem.sit.aemo.com.au/oauth/toke
n
oauth_aemoidam_provider=DEFAULT
oauth_aemoidam_media_type=application/x-www-form-urlencoded
oauth_aemoidam_refresh_before_expiry_secs=60
```

7.2 JSON schema configuration

The AEMO gateway provides the ability to configure JSON schemas which are used to validate IDX payloads. The JSON schemas are provided by AEMO. Multiple configurations are supported. The configuration names need to be structured to ensure that the business function and schema version can be identified.

```
json_schema_configurations=idx
json_schema_idx_base_cache=${batcher_install_dir}/schemas/IDX
json_schema_idx_pqd_1_schema=pqd-schema-v1-0-0.json
```

7.3 Data source configuration

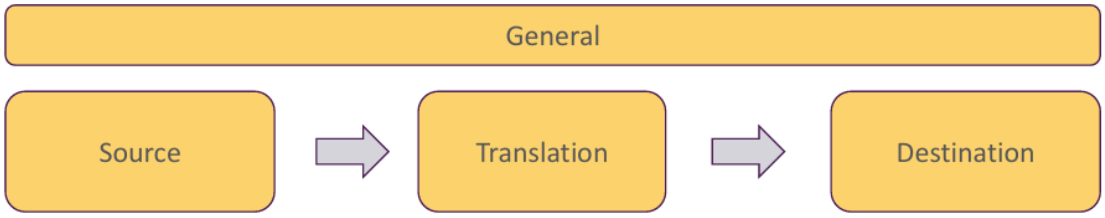
The base AEMO GW installation will be defined with the following data sources:

- aemoapi – interacts with the AEMO IDX platform API channel
- aemolfs – interacts with the AEMO IDX platform large file share (sFTP) channel
- Local – the enterprise integration point Note the references to the Oauth configuration and JSON schema configuration

```
batcher_aemoapi_mode=IDX
batcher_aemoapi_url=https://apis.nem.sit.aemo.com.au
batcher_aemoapi_auth_provider=OAUTH2
batcher_aemoapi_oauth_config=aemoidam
batcher_aemoapi_api_business_response_codes=200,201
batcher_aemoapi_default_ssl_context=false
batcher_aemoapi_flow_control_url=flowcontrol/v1/participants
batcher_aemoapi_flow_control_header_values=x
initiatingParticipantId=${participant_identifier}
batcher_aemoapi_flow_control_refresh_secs=30
batcher_aemoapi_bf_fire_and_forget=pqd
batcher_aemoapi_json_schema_config=idx
```

7.4 Thread source configuration

To describe thread concepts, we take a simplified example of an API send thread, excluding WebSocket eventing channel, and break the thread configuration into the following components:



General Mode is the message exchange pattern. The IDX_SEND supports both synchronous, asynchronous, and fire and forget patterns. We have configured this example thread for API polling at 30 second intervals:

```
batcher_thread_2_description=Generic API send thread
batcher_thread_2_mode=IDX_SEND batcher_thread_2_timeout=30
batcher_thread_2_polling_interval=30
batcher_thread_2_activity_monitor=Y
batcher_thread_2_schedule_mode=POLL
```

For this send thread, the source is [local] picking up files from the Inbox folder:

```
batcher_thread_2_source=local
batcher_thread_2_source_dir=${data_staging_dir}/Inbox
```

7.5 Thread source - translation - configuration

For this send thread, the translator performs the following actions:

1. Extracts data from the JSON message header and adds this data into eventing variables

2. Derives a MessageContextId based on the IDX specification 3. Signs the payload with a digital signature

```
batcher_thread_2_file_translator=JSON_VARIABLE -variable1 businessFunctionId -path1 $.data.header.businessFunctionId \
                    -variable2 market -path2 $.data.header.market \
                    -variable3 initiatingParticipantId -path3 $.data.header.initiatingParticipantId \
                    -variable4 priority -path4 $.data.header.priority \
                    -variable5 businessFunctionResourceId -path5 $.data.header.businessFunctionResourceId \
| ALIAS_VARIABLE -variable1 priorityCode -source1 "{jexl:priority.substring(0,1)}" \
            -variable2 messageContextUpper -source2 \
            {{businessFunctionId}_{businessFunctionResourceId}_{priorityCode}_{initiatingParticipantId}_{date:yyyyMMddHHmmsSSS \
            -variable3 messageContextId -source3 {lowercase:messageContextUpper} \
| SIGNATURE -sign -alias $(signingCertAlias) -algorithm SHA256withRSA -variable payloadSignature
```

7.6 Thread source - destination - configuration

For this send thread, the destination is [aemoapi]. A number of eventing variables derived in the translation stage are used to deliver the payload to the destination:

```
batcher_thread_2_dest=aemoapi
batcher_thread_2_dest_dir={lowercase:businessFunctionId}/v1/{businessF
unctionResourceId}
batcher_thread_2_dest_api_header_values=x-market:{market}\ ,x-
initiatingParticipantId:{initiatingParticipantId}\ ,x-
messageContextId:{messageContextId}\ ,x-signature:{payloadSignature}
```

7.7 Azure Blob Storage configuration

Instructions: Replace {parameters} with appropriate values.

```
batcher_azure_account={account name}
batcher_azure_url=https://{account name}.blob.core.windows.net
batcher_azure_API_key={Azure API key - use this if accessing as
"owner"}
batcher_azure_API_SAS_token={Azure SAS token - use this if accessing
as permissioned user, not owner}
batcher_azure_timeout_connection=10
batcher_azure_timeout_transfer=120
batcher_azure_proxy_host={proxy host name if proxy in place, otherwise
omit property}
batcher_azure_proxy_port={proxy host port if proxy in place, otherwise
omit property}
batcher_azure_mode=azure
batcher_azure_dir={main container/folder}
```

7.8 AWS S3 configuration

Instructions: Replace {parameters} with appropriate values.

```
batcher_aws_region={AWS region code e.g. ap-southeast-2}
batcher_aws_API_access_key={AWS access key}
batcher_aws_API_secret_key={AWS secret key}
batcher_aws_timeout_connection=10
batcher_aws_timeout_transfer=120
batcher_aws_proxy_host={proxy host name if proxy in place, otherwise
omit property}
batcher_aws_proxy_port={proxy host port if proxy in place, otherwise
omit property}
batcher_aws_mode=aws
batcher_aws_dir={main container/folder}
```

7.9 Google Storage configuration

Instructions: Replace {parameters} with appropriate values. Use either the batcher_google_service_account_key_file or batcher_google_service_account_key_contents sourced from Secrets Manager.

```
batcher_google_url=https://storage.googleapis.com/
batcher_google_service_account_key_file={the full path to your service
acocunt JSON key file}
batcher_google_service_account_key_contents={the text contents of your
service account JSON key file}
batcher_google_oauth_token_expiry_mins=1440
batcher_google_timeout_connection=10
batcher_google_timeout_transfer=120
batcher_google_proxy_host={proxy host name if proxy in place,
otherwise omit property}
batcher_google_proxy_port={proxy host port if proxy in place,
otherwise omit property}
batcher_google_dir={main storage bucket}
```

7.10 Credentials-Referencing data vault variables

7.10.1 Docker properties

```
data_vault_configurations=docker
data_vault_docker_provider=PROPERTIES_FILE
data_vault_docker_reference=/run/secrets/AEMO.properties
```

7.10.2 Azure Key Vault

```
data_vault_configurations=azure
data_vault_azure_provider=AZURE_KEY_VAULT
data_vault_azure_reference={fully qualified URL to vault e.g.
https://{vault_name}.vault.azure.net/}
data_vault_azure_auth_method=VMMI
```

7.10.3 AWS Secrets Manager

```
data_vault_configurations=aws
data_vault_aws_provider=AWS_SECRETS_MANAGER
data_vault_aws_reference={secret_name - as defined in the Secrets
Manager, must be hosted in same region as which app runs}
data_vault_aws_auth_method=VMMI
data_vault_aws_service_context={role_name - under which app runs.
Defined in Identity and Access Management roles with appropriate
permissions}
```

7.10.4 GCP Secrets manager

```
data_vault_configurations=gcp
data_vault_gcp_provider=GOOGLE_SECRETS_MANAGER
```

```
data_vault_gcp_reference={fully qualified URL to the secret manager
e.g.
https://secretmanager.googleapis.com/v1/projects/{your_context}/secret
s/}
data_vault_gcp_auth_method=VMMI
```

8 Security Testing

We are conducting external testing of the AEMO Gateway software because of its broad operational footprint and security-critical role. The testing focuses on:

- Application architecture and local attack surface
  - Background batch processor for IDX/BPQD file exchange.
  - Local REST APIs and/or IPC channels exposed between internal components.
  - Processor framework workflows (source → validation → translation → destination)
- Source code assisted review
  - Review of Java code for insecure patterns and input handling flaws.
  - Dependency/SBOM review for vulnerable libraries.
  - Weak credential storage and insecure keystore handling.
  - Unsafe deserialization, file parsing, and command execution risks.
  - Transformation and validation logic (JSON/CSV/aseXML/XSLT/ZIP).
- Runtime penetration testing
  - Authentication/authorisation behaviour.
  - File exchange security for IDX/BPQD payloads.
  - Behaviour under malformed or hostile payloads.
  - Local privilege escalation and unsafe process execution.
  - Runtime protections: code integrity, anti-tampering, modification resistance.
- Packaging and supply chain security
  - Verification of package signing, installation integrity, and update chain.
  - Assessment of GUI installer, CLI installation and deployment.
  - Validation that distribution artefacts are resistant to tampering.

9 Industry Testing Support

AEMO scheduled coordinated industry testing to commence from 25 May 2026, with formal testing concluding on 12 June 2026.

9.1 Prerequisites

The assumptions and prerequisites for participants involved in industry testing include:

- Participants perform testing on any internal application changes prior to connecting to the AEMO pre-production environment.
- Participants have appropriately skilled resource capability for execution and support requirements during industry testing.
- Items listed in section **3.2 Prerequisite activities** within the Coordinated Industry Test Plan.

9.2 Testing

The following BPQD IDX documents are available for more information on:

- **Coordinated Industry Test Strategy**
- **Coordinated Industry Test Plan**

9.3 Support

AEMO supports participants during industry test by:

- Assisting participants to resolve any connectivity issues within the pre-production environment through requests raised directly via the AEMO Support Hub to the Industry Data Exchange support resolver group.
- Providing Q&A sessions for participant queries.

10 Data Interchange Improvements

10.1 pdrBatcher/AEMO Gateway Software

The latest version of the pdrBatcher is now referenced as the AEMO Gateway Software v8.0.0 release, which is an upgrade from the pdrBatcher v7.6.0 installation.

10.1.1 Functional improvements

The functional improvements in AEMO Gateway Software v8.0.0 release include:

- Encrypting variables in the properties file using ${encrypted:XXXX} syntax.
  batcher_{DS}_password_encrypted is discontinued. Replaced by ${encrypted:xxx} on batcher_{DS}_password.
- Support for OAuth2 authentication.
  Change to data source configuration for Google cloud to reference OAuth2 framework.
- Renaming application package from au.com.nemmco.* to au.com.aemo.*.
- Each thread now using its own subfolder within the holding directory.
- SYNCHRONISE mode now supporting delete operations at destination using configuration.
  New configuration properties:
  - batcher_thread_{N}_mode_param1
- Property renaming batcher_thread_{N}_source_ack_translator to batcher_thread_{N}_ack_translator.
- Adding extensibility documentation and examples.
- Extensibility methods always execute in a daemon thread to reduce risk of custom code impacting overall performance.
- Deprecated property:
  - batcher_thread_{N}_max_size_kB

  replaced with
  - batcher_thread_{N}_max_size_bytes
- File sizes can now be entered in human readable format.
- Azure blob supporting OAuth2 client credentials against EntraId.

- pdrConnectionTest having command line parameters --configuration and --sources allowing the specific data source to test.
- Adding SFTP and FTPS as data source providers to make these service types easier to configure. The system continues to support the legacy configuration of SFTP/FTPS under FTP.
- Adding DEBUG log capability for transform configuration. New property batcher_translator_debug_dir and new --debug parameter for transform test CLI tool.
- An example for configuring structured logging for JSON output.
- Adding a feature for archiving transactions. New properties:
  - batcher_transaction_archive_data_source
  - batcher_transaction_archive_root_dir
  - batcher_transaction_archive_data_dir
  - batcher_thread_{N}_transaction_archive
  - batcher_thread_{N}_transaction_archive_data_dir
- Adding a stop file status API for monitor reporting.
- Adding parameters to control thread check and reporting interval:
  - batcher_thread_check_interval
  - batcher_thread_report_interval
- Adding a command line tool for testing a sample file for validation against registered schemas.
- Setting content-type based on filename when uploading to cloud blob.
- Windows service wrapper migrating from WinRun4J (legacy) to Apache Procrun.
- Various new file translation functionalities added:
  - XML_VARIABLE: Creates a variable based on an XSLT path specification
  - ALIAS_VARIABLE: Creates a variable based on an expression
  - JSON_TO_XML: Converts JSON to XML
  - XML_TO_JSON: Converts XML to JSON
  - SIGNATURE: Creates or verifies a digital signature

10.1.2 Bug fixes

The bug fixes in AGS includes:

- File translators are now loaded in a predictable order, ensuring it applies the correct filemask.
- Source and destination connections are now tested if there is an error in processing a file.

10.1.3 Security upgrades

Security upgrades to numerous third-party libraries.

10.1.4 Upgrading AEMO Gateway

Refer to the **pdrBatcher implementation instructions** for details on how to perform an upgrade install.

10.2 pdrLoader

The pdrLoader is an optional, participant-hosted component that works alongside the AEMO Gateway Software to load data received from AEMO into a participant's local relational database.

The pdrLoader v7.7.0 release is an upgrade from the v7.6.0 installation.

10.2.1 Functional improvements

The functional improvements in pdrLoader v7.7.0 release include:

- Encrypting variables in the properties file using ${encrypted:XXXX} syntax.
  Property db_password_encrypted is discontinued and replaced by ${encrypted:xxx} on db_password.
- Renaming application package from au.com.nemmco.* to au.com.aemo.*
- pdrFileEventInt changing from abstract class to interface.
- A null option for db_user and db_password to allow for externally identified database access.
- Adding extensibility documentation and examples.
- Extensibility methods always execute in a daemon thread to reduce risk of custom code impacting overall performance.
- Improving efficiency of SPARSE data row filter. This change to the row filter interface may impact any custom row filter implementations.

- pdrConnectionTest having command line parameters --configuration and --sources allowing the specific data source to test.
- An example for configuring structured logging for JSON output.
- Adding parameters to control thread check and reporting interval:
  - loader_thread_check_interval
  - loader_thread_report_interval
- Upgrade GUI installer extracting database connection parameters from existing properties configuration.
- Adding installation options to disable manifest and automate configuration processes.
- Windows service wrapper migrating from WinRun4J (legacy) to Apache Procrun.

10.2.2 Bug fixes

The bug fixes in the pdrLoader includes:

- Defect fix - validate any update to reconciliation point in time is historic.
- Archive re-request only increments in table PDR_MANIFEST_LOG where the rejection message is other than Maximum number of files in trickle directory reached.

10.2.3 Security upgrades

Security upgrades to numerous third-party libraries.

10.2.4 Upgrading pdrLoader

Refer to the **pdrLoader implementation instructions** for details on how to perform an upgrade install.

10.3 pdrMonitor

The pdrMonitor is an optional monitoring and support utility used to observe and troubleshoot the operation of the AEMO Gateway Software and related components.

The pdrMonitor v1.4.0 release is an upgrade from any existing v1.3.0 installation.

10.3.1 Functional improvements

The functional improvements in the pdrMonitor v1.4.0 includes:

- Encrypting variables in the configuration file using ${encrypted:XXXX} syntax.
- Property DbPasswordEncrypted is discontinued and is replaced by ${encrypted:xxx} on DbPassword.
- Renaming application package from au.com.nemmco.* to au.com.aemo.*.
- An example for configuring structured logging for JSON output.
- Exposing B2B performance data in application transaction reporting.
- Adding flow control visualisation for batcher.
- Adding user event auditing and reporting.
- Adding new dashboard widgets for B2B flow control and stopfile reporting.
- Adding context sensitive help links.
- Adding visualisation and description for batcher and loader settings tab.
- Adding support for http2 protocol. New configuration:
  - /WebServer/Provider
- Upgrade GUI installer extracting database connection parameters from existing XML configuration.
- Adding new dashboard widgets for application and security events.
- Windows service wrapper migrating from WinRun4J (legacy) to Apache Procrun.
- Configuration to control thread check and reporting status into the log file.

10.3.2 Bug fixes

Corrects the defect that did not allow an IP address to be typed into monitor browser for adding a new connection/source.

10.3.3 Security upgrades

The security upgrades in pdrMonitor v1.4.0 include:

- Security upgrades to numerous third-party libraries.

- Add session timeout warning to browser interface.

10.3.4 Upgrading pdrMonitor

Refer to the **pdrMonitor implementation instructions** for details on how to perform an upgrade install.

11 Data Interchange Implementation Instructions

11.1 pdrBatcher/AEMO Gateway Software

For detailed installation instructions, see AEMO Gateway GUI installer.

11.1.1 Full install

To install pdrBatcher v8.0.0 as a new implementation, select the following file from the Data Interchange software:

File	Filename	Benefit
Participant Data Replication Batcher GUI Installer v8.0.0	Participant Data Replication Batcher GUI Installer v8.0.0.zip	Requires the minimum data entry for localisation. All implementation steps are automatic. A manual installation is possible by exiting the installer and using a text editor to set the configuration properties.

11.1.2 Upgrading from v7.6.0 to v8.0.0

The release of pdrBatcher v8.0.0 introduces a number of new optional configurations in the .properties file. For details, see Properties file updates below.

Properties file updates

For help configuring these properties, see **Participant Data Replication Batcher User Guide**.

Property	Detail
batcher_error_dir	Error folder for files that are unable to be processed.
OAuth2 provider definitions	Refer to the documentation in pdrBatcher.properties.
Data Source definitions	Numerous new configurations which add support for various connector types such as AMQP, etc. Refer to the documentation in pdrBatcher.properties.

Property	Detail
Event Source definitions	Event sources support channels such as WebSockets for connectivity to IDX. Refer to the documentation in pdrBatcher.properties.
Request rate limiter configuration	Refer to the documentation in pdrBatcher.properties.
JSON schema configuration	Refer to the documentation in pdrBatcher.properties.
XML schema configuration	Refer to the documentation in pdrBatcher.properties.
batcher_thread_check_interval	The interval (seconds) to check for failed threads. Default is 5 seconds.
batcher_thread_report_interval	The interval (seconds) to report thread summary in log file. Default is 3600 seconds.
batcher_thread_{N}_min_size_bytes	Allows the specification of a minimum file size filter on a processing thread.
batcher_thread_{N}_source_api_scope	The scope to request when the source uses OAuth for authentication.
batcher_thread_{N}_dest_api_scope	The scope to request when the destination uses OAuth for authentication.
batcher_thread_{N}_service_name	The service name associated with this thread when called as a synchronous API service.
batcher_thread_{N}_service_roles	The JWT roles required to access this thread.
batcher_thread_{N}_service_request_limit_filter	The label for the request rate limiter to apply. Optional.
batcher_thread_{N}_file_status_translator	Ability to perform a translation on an completed transaction for status reporting back to source. Allows merging of data within return acknowledgment transaction. Configuration option per batcher_thread_{N}_file_translator.
batcher_thread_{N}_transaction_archive	Set to Y to enable transaction audit for this thread.
batcher_thread_{N}_transaction_archive_data_dir	The transaction data directory for this thread, which overrides batcher_transaction_archive_data_dir.
batcher_thread_{N}_retention_time_mins	The retention time for a thread operating in PURGE mode.

Property	Detail
batcher_thread_{N}_file_integrity_check	The integrity check to apply to an incoming file. Valid settings are: - NONE: Pass through, no check performed - ZIP: Performs a check that a compressed file is valid. Supported compression types include zip, gz, .7z, .tgz, tar.gz Note: A zip validator does not inspect the content of the zip file unless further validators are chained in a pipeline. See below. - JSON: Performs a check that a file contains valid JSON - XML: Performs a check that a file contains valid XML. Valid parameters (case sensitive): schema: The XML schema configuration to use. Refer to configuration xml_schema_configurations. - DEFAULT: Performs a check based on the file extension, matching to the appropriate validator or passing through if there are no validators for that file type.
batcher_thread_{N}_parameters	A set of parameters that can be defined for use in operations within the thread. Defined as a list of comma delimited name value pairs. Example: x-market=NEM,x-initiatingParticipantId=XXXX
batcher_thread_{N}_participant_list	The comma delimited list of participant ID to process. Relevant to aseXML and IDX message exchange patterns. Overrides batcher_{DS}_participant_list for thread {N}.
batcher_thread_{N}_fire_and_forget	Sets whether to acknowledge inbound messages with a generated message ack back to source or assume a fire and forget pattern which suppresses generation of a message ack. Applies only to IDX message exchange patterns.
batcher_thread_{N}_business_function_list	The comma delimited list of business functions to process. Relevant to IDX message exchange patterns. Overrides batcher_{DS}_business_function_list for thread {N}.
web_server_https_keystore_file	The keystore file which contains certificates to support SSL configuration of the web server.
web_server_https_keystore_type	The type of keystore file. Default is PKCS12.
web_server_https_store_password	The key store password associated with KeyStoreFile. Defaults to encryption key stored in pdr_key.properties.
web_server_https_key_password	The certificate password. Defaults to encryption key stored in pdr_key.properties.
web_server_https_protocol	The SSL protocol to use. Options include TLS, TLSv1.2. Default is TLS.

Properties file deletions

No properties are made obsolete for v8.0.0.

GUI installation process

The GUI installation process is a simple, clean way to upgrade. Alternatively, you can use the Manual Installation process below.

Follow the steps below to upgrade an existing Participant Data Replication Batcher Software v7.6.0 to v8.0.0 installation:

1. If it is not already installed, install JRE 21.
2. Validate the jar file signature. From the command line prompt, run the following command:

```
jarsigner -verify "Participant Data Replication Batcher GUI Installer v8.0.0.jar"
```

   For more information, see **https://docs.oracle.com/en/java/javase/21/docs/specs/man/jarsigner.html**
3. Install the pdrBatcher v8.0.0 to the current installation folder (for example, C:\Pdr\Batcher).

   The current installation is backed up to a folder alongside the current installation with a timestamp appended to the folder name.
4. If running as a Windows service, ensure the Windows service name entered into the GUI installer matches the existing service name.
5. Check the pdrBatcher.properties file changes and re-apply any customised configurations. See **properties file updates and deletions**.
6. Run the pdrBatcher.bat or the Windows service and check the pdr.log in the Log folder for any errors.

Manual installation process

Follow the steps below to manually upgrade an existing Participant Data Replication Batcher Software v7.6.0 to v8.0.0 installation:

1. Backup your pdrBatcher v7.6.0 installation folder.
2. Remove the pdrBatcher v7.6.0 Windows service by running:

```
pdrServiceUninstall.bat <insert_Windows_service_name_here>.
```

3. Remove all existing jar files from the Lib folder.
4. Copy the following content from the pdrBatcher v8.0.0 installation media into your Lib folder:
   - All jar files
   - New windows service wrapper:
     - prun*.exe
     - pdrServiceInstall.ini
5. If it is not already installed, install JRE 21 and update the JRE path variable in:
   - pdrEnvironment.bat
   - pdrServiceInstall.bat
6. Create an Error folder under the main installation directory and add property:
   batcher_error_dir=${batcher_install_dir}/Error
7. Consider aligning your pdrBatcher.properties to the standard configuration file unless you have specific customisation requirements. For participants with customisation requirements, it is recommended that these customisations be merged into a standard v8.0.0 properties file definition.
8. Install a new Windows service by running:
   pdrServiceInstall.bat <insert_Windows_service_name_here>.
9. Check the NEMNET password is correctly encrypted before running the service by running the following command to avoid any password lockout:
   pdrConnectionTest.bat
10. Run pdrBatcher.bat or the Windows service and check the pdr.log in the Log folder for any errors. For help, see **Guide to Participant Data Replication Batcher**.

Custom code extensions

The pdrBatcher v8.0.0 release updates the code package from au.com.nemmco.* to au.com.aemo.*.

Any references to the interfaces which define those extensions must be recompiled against the interfaces in the new package. Refer to the SDK folder in the installation directory for instructions and examples. Participants requiring assistance to migrate any existing plug-in configurations should log a support ticket with AEMO Support Hub.

11.2 pdrLoader

For detailed installation instructions, see Participant Data Replication Loader GUI Installer Guide **or** Guide to Participant Data Replication Loader**.**

11.2.1 Full install

To install pdrLoader v7.7.0 as a new implementation, select the following file from the Data Interchange software:

File	Filename	Benefit
Participant Data Replication Loader GUI Installer v7.7.0.	Participant Data Replication Loader GUI Installer for <Database> v7.7.0.zip.	Requires the minimum data entry for localisation. All implementation steps are automatic. A manual installation is possible by exiting the installer and using a text editor to set the configuration properties.

11.2.2 Upgrading from v7.6.0 to v7.7.0

A successful upgrade involves understanding the changes between the versions. See improvements **for more information.**

The v7.7.0 release has shipped with updated JDBC drivers across most database platforms. Please confirm the compatibility of the JDBC driver with your local database infrastructure.

A SSL encrypted connection to your local database would be considered best security practice, and participants wanting to configure this should discuss signed certificates across their infrastructure with their system administrator.

Participants who do not require encrypted communications can generally apply configuration to the JDBC connection string to indicate a non-encrypted communication channel.

Refer to the OEM JDBC driver documentation for how to configure the connection string to your organisation’s requirements.

Properties file updates

The release of pdrLoader v7.7.0 introduces a number of new optional configurations in the .properties file.

For help configuring the application properties, see **Guide to Participant Data Replication Loader**.

Property	Detail
loader_thread_check_interval	The interval (seconds) to check for failed threads. Default is 5 seconds.
loader_thread_report_interval	The interval (seconds) to report thread summary in log file. Default is 3600 seconds.

Properties file deletions

Properties made obsolete in v7.7.0.

Property	Detail
db_password_encrypted	This parameter is discontinued. Encrypted configuration is denoted by enclosing the value as follows: ${encrypted:XXX}

GUI installation process

The GUI installation process is a simple, clean way to upgrade. Alternatively, you can use the **Manual installation** process.

Follow the steps below to upgrade an existing Participant Data Replication Loader Software v7.6.0 to v7.7.0 installation:

1. If it is not already installed, install JRE 21.

2. Validate the jar file signature. From the command line prompt, run the following command:

```
jarsigner -verify "Participant Data Replication Loader GUI Installer v7.7.0.jar"
```

   For more information, see **https://docs.oracle.com/en/java/javase/21/docs/specs/man/jarsigner.html**

3. Install the pdrLoader v7.7.0 to the current installation folder (for example, C:\Pdr\Loader).

   The current installation is backed up to a folder alongside the current installation with a timestamp appended to the folder name.

4. If running as a Windows service, ensure the Windows service name entered into the GUI installer matches the existing service name.

5. Check the pdrLoader.properties file changes and re-apply any customised configurations. See **Upgrading from v7.6.0 to v7.7.0**.

6. Run the pdrLoader.bat or the Windows service and check the pdr.log in the Log folder for any errors.

Manual installation process

Follow the steps below to manually upgrade an existing Participant Data Replication Loader software:

1. Backup your pdrLoader v7.6.0 installation folder.

2. Backup database tables PDR_REPORT_ROW_FILTERS and PDR_REPORT_TRANSFORMS.

3. Remove the pdrLoader v7.6.0 Windows service by running:

```
pdrServiceUninstall.bat <insert_Windows_service_name_here>.
```

4. Remove all existing jar files from the Lib folder.

5. Copy the following content from the pdrLoader v7.7.0 installation media into your Lib folder:

   - All jar files
   - New windows service wrapper:
     - prun*.exe
     - pdrServiceInstall.ini

6. If not already, install JRE 21 and update the JRE path variable in:

   - pdrEnvironment.bat
   - pdrServiceInstall.bat

7. Update the definition of_db_password to wrap the encrypted value, for example:

   db_password=uj3KvUYmchoKABGdPDggBQ==

   then becomes:

   db_password=${encrypted:uj3KvUYmchoKABGdPDggBQ==}

8. Consider aligning your pdrLoader.properties to the standard configuration file unless you have specific customisation requirements. For participants with customisation requirements, it is recommended that these customisations be merged into a standard v7.7.0 properties file definition.

9. Apply the database update scripts applicable to your database platform in the following order:

    c. <database_platform>_alter_pdr.sql

    d. populate_parser_config_metadata_delta.sql

    e. commit

10. Install a new Windows service by running:
    pdrServiceInstall.bat <insert_Windows_service_name_here>.

11. Check the database password is correctly encrypted before running the service by running the following command to avoid any password lockout:
    pdrConnectionTest.bat

12. Run pdrLoader.bat or the Windows service and check the pdr.log in the Log folder for any errors. For help, see **Guide to Participant Data Replication Loader**.

Custom code extensions

The pdrLoader v7.7.0 release updates the code package from au.com.nemmco.* to au.com.aemo.*.

Any references to the interfaces which define those extensions must be recompiled against the interfaces in the new package. The pdrFileEventInt has also been changed from abstract class to interface in this release. Refer to the SDK folder in the installation directory for instructions and examples. Participants requiring assistance to migrate any existing plug-in configurations should log a support ticket with AEMO Support Hub.

Rollback considerations

The pdrLoader v7.7.0 release updates the code package from au.com.nemmco.* to au.com.aemo.*.

Some of the references in the PDR_* management table provide mapping to various classes.

In the event that the installation needs to be reverted to v7.6.0, then the references in tables PDR_REPORT_ROW_FILTERS and PDR_REPORT_TRANSFORMS need to be reverted to the nemmco namespace.

11.3 pdrMonitor

For detailed installation instructions, see Participant Data Replication Monitor GUI Installer Guide **or** Guide to Participant Data Replication Monitor**.**

11.3.1 Full install

To install pdrMonitor v1.4.0 as a new implementation, select the following file from the Data Interchange software:

File	Filename	Benefit
Participant Data Replication Monitor GUI Installer v1.4.0	Participant Data Replication Monitor GUI Installer for <Database> v1.4.0.zip	Requires the minimum data entry for localisation. All implementation steps are automatic. A manual installation is possible by exiting the installer and using a text editor to set the configuration properties.

11.3.2 Upgrading from v1.3.0 to v1.4.0

A successful upgrade involves understanding the changes between the versions. See pdrMonitor improvements **for more information.**

The release of pdrMonitor v1.4.0 introduces new configurations in the XML configuration file.

Properties files updates

For help configuring the application properties, see **Guide to Participant Data Replication Monitor**.

Property	Detail
/Database/DbPassword	Encrypted passwords are represented as ${encypted:XXX} where XXX is the encrypted password.

Property	Detail
/WebServer/Provider	The web server provider. Supported values are: - au.com.aemo.Common.Web.JDK.WebServerJDK Supports http/1.1 and https/1.1 (default) - au.com.aemo.Common.Web.Jetty.WebServerJetty Supports http/1.1 and https/1.1 - au.com.aemo.Common.Web.Jetty.WebServerJettyHttp2 Supports https/2 with fall back to https/1.1 Modern browsers do not support http/2 (plain text). Note: https/2 requires a much larger number of threads due to the parallel nature of the protocol. Refer NoThreads configuration parameter.
/Authentication/AuthorisationProvider	The provider changes from: au.com.nemmco.Pdr.Monitor.Security.pdrMonitorAuthorisationProvider to au.com.aemo.Pdr.Monitor.Security.pdrMonitorAuthorisationProvider
/ThreadStatusCheckInterval	The interval (seconds) to check for failed threads. Default is 5 seconds.
/ThreadStatusReportingInterval	The interval (seconds) to report thread summary in log file. Default is 3600 seconds.

Properties file deletions

Properties made obsolete in v7.7.0.

Property	Detail
db_password_encrypted	This parameter is discontinued.

GUI installation process

The GUI installation process is a simple, clean way to upgrade. Alternatively, you can use the **Manual installation** process.

Follow the steps below to upgrade an existing Participant Data Replication Monitor Software v1.3.0 to v1.4.0 installation:

1. If it is not already installed, install JRE 21.

2. Validate the jar file signature. From the command line prompt, run the following command:

```
jarsigner -verify "Participant Data Replication Monitor GUI Installer v1.4.0.jar"
```

   For more information, see **https://docs.oracle.com/en/java/javase/21/docs/specs/man/jarsigner.html**

3. Install the pdrMonitor v1.4.0 to the current installation folder (for example, C:\Pdr\Monitor).

   The current installation is backed up to a folder alongside the current installation with a timestamp appended to the folder name.

4. If running as a Windows service, ensure the Windows service name entered into the GUI installer matches the existing service name.

5. Check the pdrMonitor.xml file changes and re-apply any customised configurations. See **configuration files updates and deletions**.

6. Run the pdrMonitor.bat or the Windows service and check the pdr.log in the Log folder for any errors.

Manual installation process

Follow the steps below to manually upgrade an existing Participant Data Replication Monitor software:

1. Backup your pdrMonitor v1.3.0 installation folder.

2. Remove the pdrMonitor v1.3.0 Windows service by running:

```
pdrServiceUninstall.bat <insert_Windows_service_name_here>.
```

3. Remove all existing jar files from the Lib folder.

4. Copy the following content from the pdrMonitor v1.4.0 installation media into your Lib folder:

   - All jar files
   - New windows service wrapper:
     - pdrServiceInstall.ini
     - prun*.exe

5. Run the following command, after unzipping the command line prompt in the Lib folder:

```
jarsigner -verify "AppPdrMonitor.jar"
```

6. Update script pdrEnvironment.* and align CLASSPATH environment variable definition to match the v1.4.0 definition.

7. Copy the following content from the pdrMonitor v1.4.0 installation media into your Website folder:

   – All folders under Website

8. If it is not already installed, install JRE 21 and update the JRE path variable in:

   – pdrEnvironment.bat

   – pdrServiceInstall.bat

9. Adjust configuration of the following elements in pdrMonitor.xml:

   – /Database/DbPassword
   Wrap the existing encrypted password as follows:

     ○ ${encrypted:XXX}

10. Apply the database update scripts applicable to your database platform in the following order:

    a. <database_platform>_alter_pdr.sql

    b. populate_parser_config_metadata_delta.sql

    c. commit

11. Install a new Windows service by running:
    pdrServiceInstall.bat <insert_Windows_service_name_here>.

12. Check the database password is correctly encrypted before running the service by running the following command to avoid any password lockout:
    pdrConnectionTest.bat

13. Run pdrMonitor.bat or the Windows service and check the pdr.log in the Log folder for any errors.

14. Test upgrade by loging to pdrMonitor Web url as Admin. If login cannot be completed using previous password, try resetting password using pdrMonitorPasswordReset.bat. Restart pdrMonitor service.

15. API keys defined within pdrMonitor source connections to pdrBatcher and pdrLoader need to match with respective keys defined within pdrBatcher and pdrLoader (pdr_key.properties). If these are changed during pdrBatcher/pdrLoader upgrades, please align the values and re-test pdrMonitor.

12 Governance Process

TBC

13 FAQs

This section is updated based on participant queries from the MSUG meetings and Industry testing phase.

14 Terms

14.1 Rules Terms

You can find the following terms defined in the **National Electricity Rules (NER)** and the **Settlements Residue Auction Rules**.

Term
AEMO
AEMO Clearing Account
AEMO Markets Portal
AEMO Website
Allocated Units
APA
Auction
Auction Participant
Auction Rules
Average cancellation price
Average purchase price
Basic Power Quality Data
Bid File
Cancelled Units
Cancelled volume
Cash Security
Confidential Information
Directional interconnector
Industry Data Exchange

Term
Linked Bid
Market Clearing Price
Market Participants
Maximum Units
NEM
Notional Interconnector
Offer Database
Offer File
Offer Period
Offer Submission
Offered Units
Offers
Product
Prudential Approved Participant
Prudential Exposure
Region
Regional reference prices
Registered Participant
Relevant Quarter

Term
Settlement residue auction
Settlement residue committee
Settlement residue distribution agreement
SRDA Units
Trading Limit
Trading Margin
Trading Position
Unit Category
Units

14.2 Glossary

You can find a full list of AEMO glossary terms in **Industry Terminology** on AEMO's website.

Abbreviation/Term	Explanation
AEST	Australian Eastern Standard Time
B2B	Business-to-business
B2M	Business-to-market
BPQD	Basic Power Quality Data
EMMS	Electricity Market Management System; software, hardware, network and related processes to implement the wholesale energy market
FCAS	frequency control ancillary services
FTP	File transfer protocol
IDX	Industry Data Exchange
MSATS	Market Settlement and Transfer Solution for retail electricity
NER	National Electricity Rules
MW	Megawatt
Release	IDX - AEMO Gateway Software - Technical Specification – June 2026
Release Dates	Pre-production: Monday 18 May 2026 Production: Wednesday 1 July 2026
TBC	To be confirmed