Industry Data Exchange Platform Standard

Prepared by:	AEMO
Document ref:	XX-XXXX
Version:	0.2
Effective date:	TBD
Status:	DRAFT

Approved for distribution and use by:

Approved by:	
Title:	
Date:	31/10/2025

1. Version release history

Version	Effective Date	Summary of Changes
0.1	30/05/2025	First issue to support Basic Power Quality Data foundation use case. Sections marked as placeholder where not required to support BPQD.
0.2	31/10/2025	Project checkpoint, interim release for Industry Data Exchange Platform. Renamed document from “IDX Technical Delivery Specification” to “Industry Data Exchange Platform Standard”.

2. IDX Framework Core Concepts

2.1. Introduction

The IDX platform aims to provide a standardised set of industry-agreed channels, protocols, patterns, and capabilities for the exchange of all market and B2B transactions related to the energy industry. This standardisation helps reduce complexity and ensures consistency across different data exchanges.

The IDX platform is designed specifically for any data exchange between participants and AEMO. It facilitates efficient and reliable data exchange between different systems, ensuring that all new use cases are assessed against the decision tree to determine the appropriate payload format, pattern, channel and protocol. The intent is to simplify data exchanges by standardising protocols and patterns, making them easier to govern and maintain, enhance security and compliance in line with SOCI recommendations, and improve extensibility.

Each business function within the NEM, WEM, and Gas markets will have each business use case assessed and assigned a single channel and protocol. This enables targeted industry consultations, more efficient implementations due to standardised protocols.

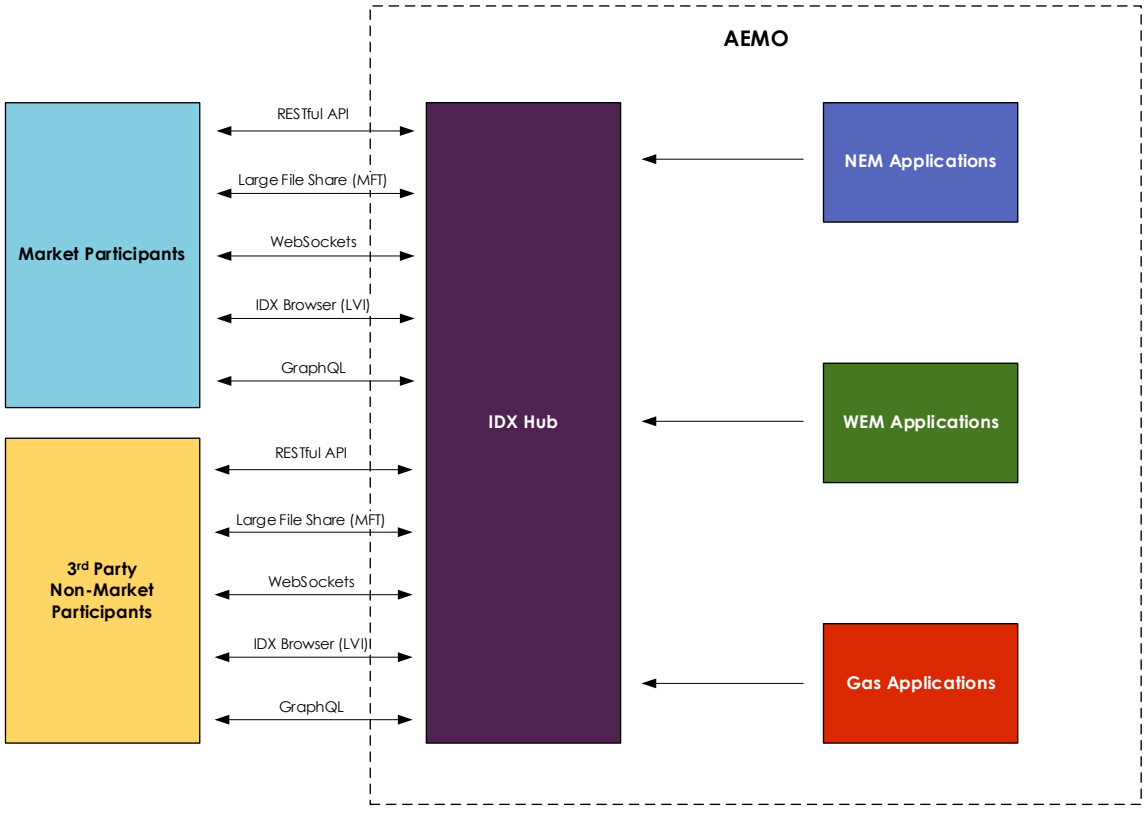


Diagram 2.1 IDX Core Concept

2.2. Solution Overview

The IDX Hub is AEMO's provided communication platform supporting NEM, WEM, and Gas transactions. The diagram below illustrates the AEMO and participant components of the solution, relating to the centralised IDX Hub:

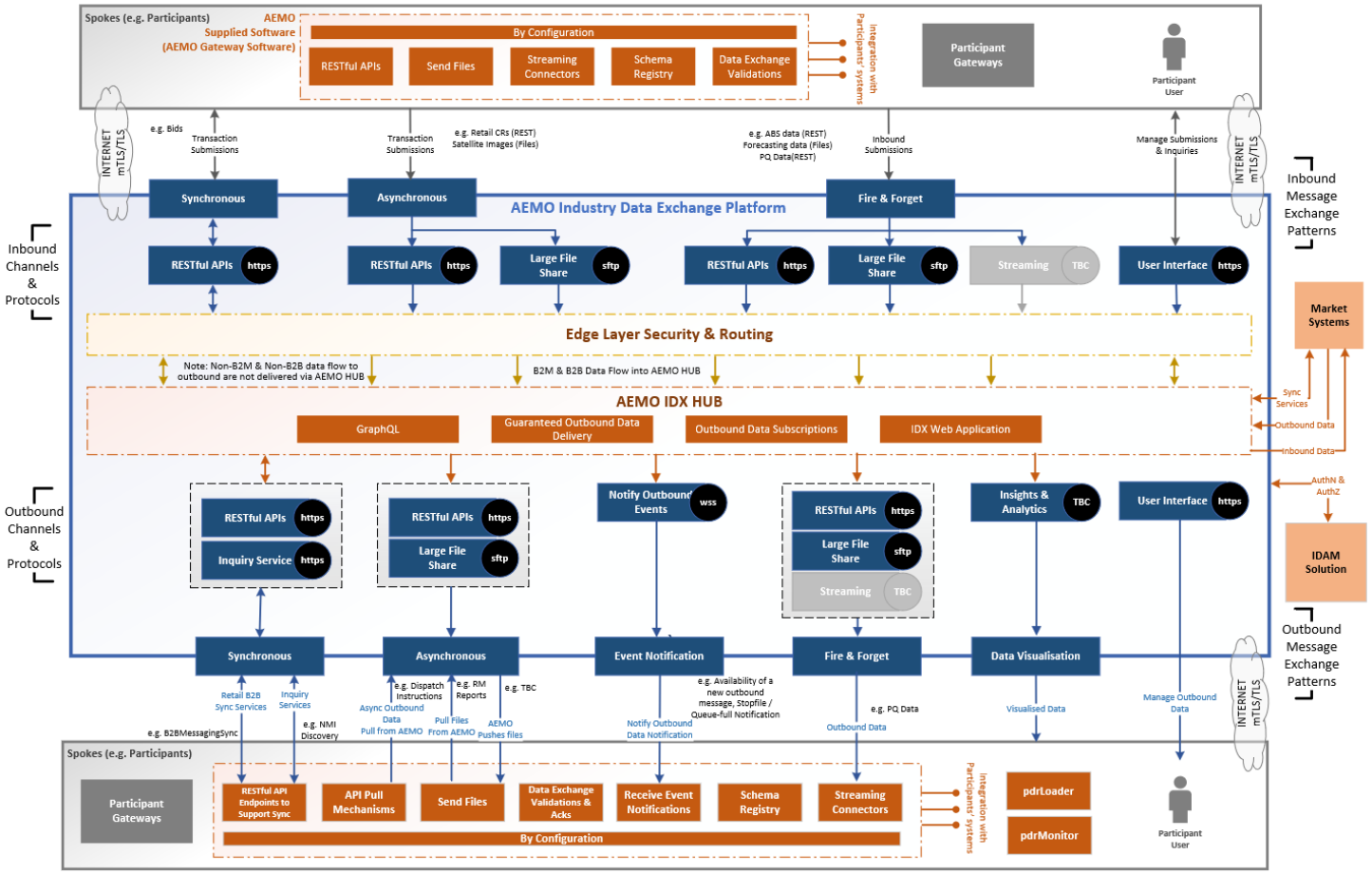


Diagram 2.2 IDX Solution overview

2.3. Business Functions and Resources

Business Functions

(a) A Business Function typically equates to a Business Service with group of logical business sub-services, for example Meter Reads (MTRD) as a Business Function would have multiple sub-services to support the MTRD business function e.g. Exchange MTR Reads, PMD, VMD.

(b) A Business Function is established as an access point for the exchange of information between a Market Participant/Non-Market Participant and AEMO.

(c) A Business Function can have any number of Business Resources underneath it.

(d) Each Business Function added to the IDX Platform will have its own Technical Specification which details the transactions applicable and relevant information for that business function.

Business Resources

(a) A Business Resource typically equates to a discrete business sub-service within a logical grouping of business services, for example Provide Meter Data would be a Business Resource of the MTRD Business Function.

(b) A Business Resource may be a collection of related logical sub-services which are established as part of the wider Business Function to operate.

(c) Each Business Resource's specific transaction requirements are detailed in either its Parent Business Function's Technical Specification or within its own Business Resource Technical Specification.

2.4. Accessing IDX Services

(a) Accessing IDX Platform can be dependent on the service access requirements itself.

(b) For transacting over the API Channel, an organisation must:

(i) Establish TLS Certificates - for those services requiring MTLS authentication

(ii) Set up IDAM credentials with AEMO

(iii) Where required for the business function, have MarketNet connectivity

(iv) Completion of the relevant Service registration, or accreditation

(v) Meet the system requirements outlined in Section 2.5

(c) For transacting over the Large File Share Channel, an organisation must:

(i) Set up IDAM credentials with AEMO

(ii) Where required for the business function, have MarketNet connectivity

(iii) Completion of the relevant Service registration, or accreditation

(iv) SFTP client

(v) Meet the system requirements outlined in Section 2.5

2.5. System Requirements to Access IDX Hub

(a) To exchange information on IDX hub, over API channels a participant is required to:

(i) Have their IP address range whitelisted by AEMO

And either:

(ii) Install AEMO Gateway Software within their client environment; or

(iii) Set up their own gateway software

This section will be elaborated on in later versions of the document.

2.6. Availability

(a) AEMO agrees to use reasonable endeavours to make that portion of the National B2B and B2M Infrastructure over which they have control and for which they are responsible available at all times. However, AEMO are not able to guarantee the provision of a continuous and fault free National B2B and B2M Infrastructure for various reasons, including: the conduct of a user of the National B2B and B2M Infrastructure;

(i) an electrical or telecommunications fault or failure;

(ii) an emergency or fault rectification procedure;

(iii) scheduled maintenance;

(iv) a fault, virus, security breach or breakdown;

(v) an event of force majeure.

(b) All obligations imposed on a Participant and/or AEMO in this document must be read subject to clause (a) above.

(c) Tier 1 IDX services and business functions will adhere to a 99.99% uptime per the agreed service level.

(d) Tier 2 IDX services and business functions will adhere to a 99.9% uptime per the agreed service level.

3. Decision Tree for Business Functions

3.1. Overview

A decision tree is a decision support hierarchical model that uses a tree-like model of decisions and their possible consequences. Each business function entered onto the IDX framework is assessed through the decision tree (and elaborated on through industry consultation), determining the best suited pattern and channel for an interface. For any given use case there will be only a single channel supported (outside of transitional arrangements).

3.2. Message Pattern Decision Tree

3.2.1. Overview

The message pattern decision tree is used to determine the appropriate message pattern for transacting between AEMO and participants. The key determinants driving the decision centre around whether an initiator (participant) is:

- Requesting information
- Whether the information requested is always the same, or dynamic based on request
- Whether a non-information request is required and if a business response is required
- Whether that business response is required immediately or not

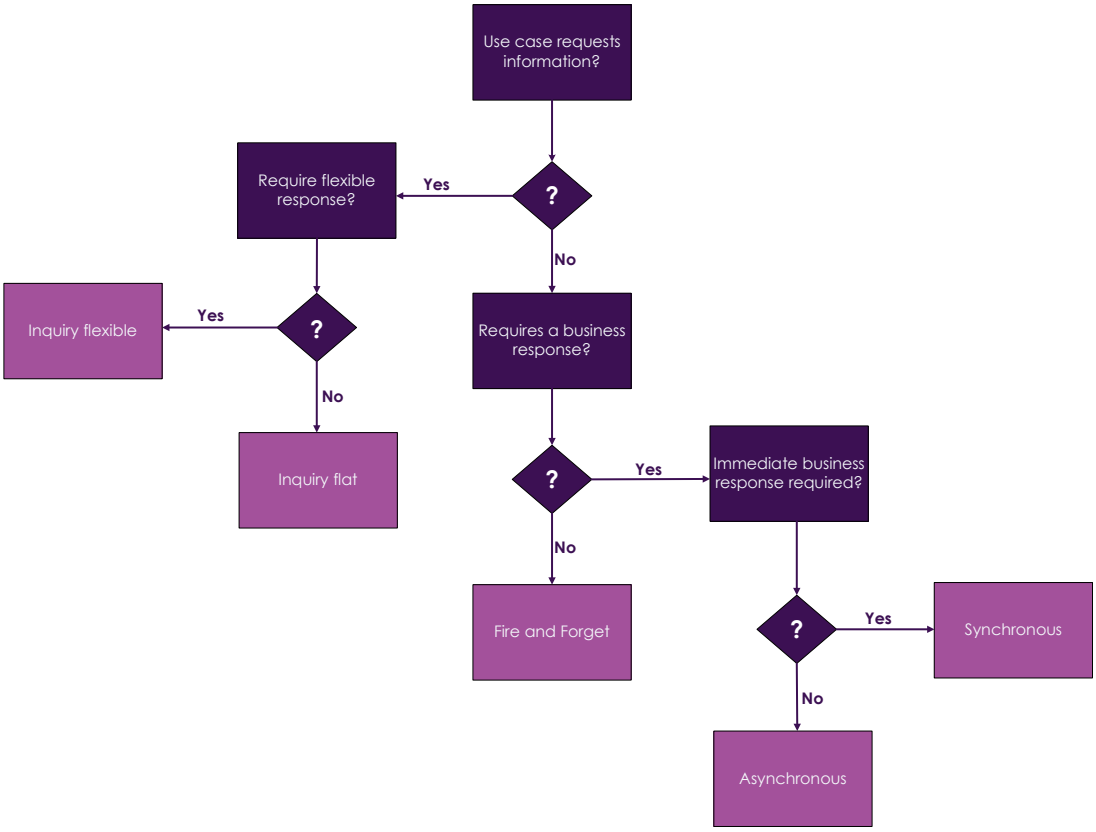


Diagram 3.2.1 Message Pattern Decision Tree

3.2.2. Decision Tree Pattern Outcomes

Following the decision tree for message pattern makes the determination of most appropriate message interaction pattern, which can be one of the following outcomes:

Pattern Type	Description
Synchronous	An immediate business response is required to confirm a request has been received and processed.
Asynchronous	Additional processes or validations are required to enable a response to the request. It follows a multi-legged approach to deliver the business response.
Fire & Forget	Request message is sent with no expectation of receiving a business or technical (e.g. MACK) response message from another market participant. Hub MACK responses may still be provided.
Inquiry-Flexible	The Interface data exchange consists of an on-demand service that allows for a dynamic subset of data elements to be returned.
Inquiry-Flat	The Interface data exchange consists of an on-demand service that allows for a fixed set of data elements to be returned.

Table 3.2.2 Decision Tree Pattern Outcomes

3.3. Message Compression and Bundling Decision Tree

3.3.1. Message Compression and Bundling Overview

Message compression and bundling are techniques used to optimise data transmission between participant and AEMO's infrastructure. These methods help reduce the size of data being transmitted and improve the efficiency of communication channels. The decision tree for message compression and bundling provides a structured approach to determine when and how to apply these techniques based on specific criteria and use cases.

Refer to *14.2 Examples – Decision Tree Worked Examples* to see example business functions evaluated by Decision Tree.

3.3.2. Message Bundling Decision Tree

As an input to message compression and bundling, the message size must be determined to apply the most appropriate options.

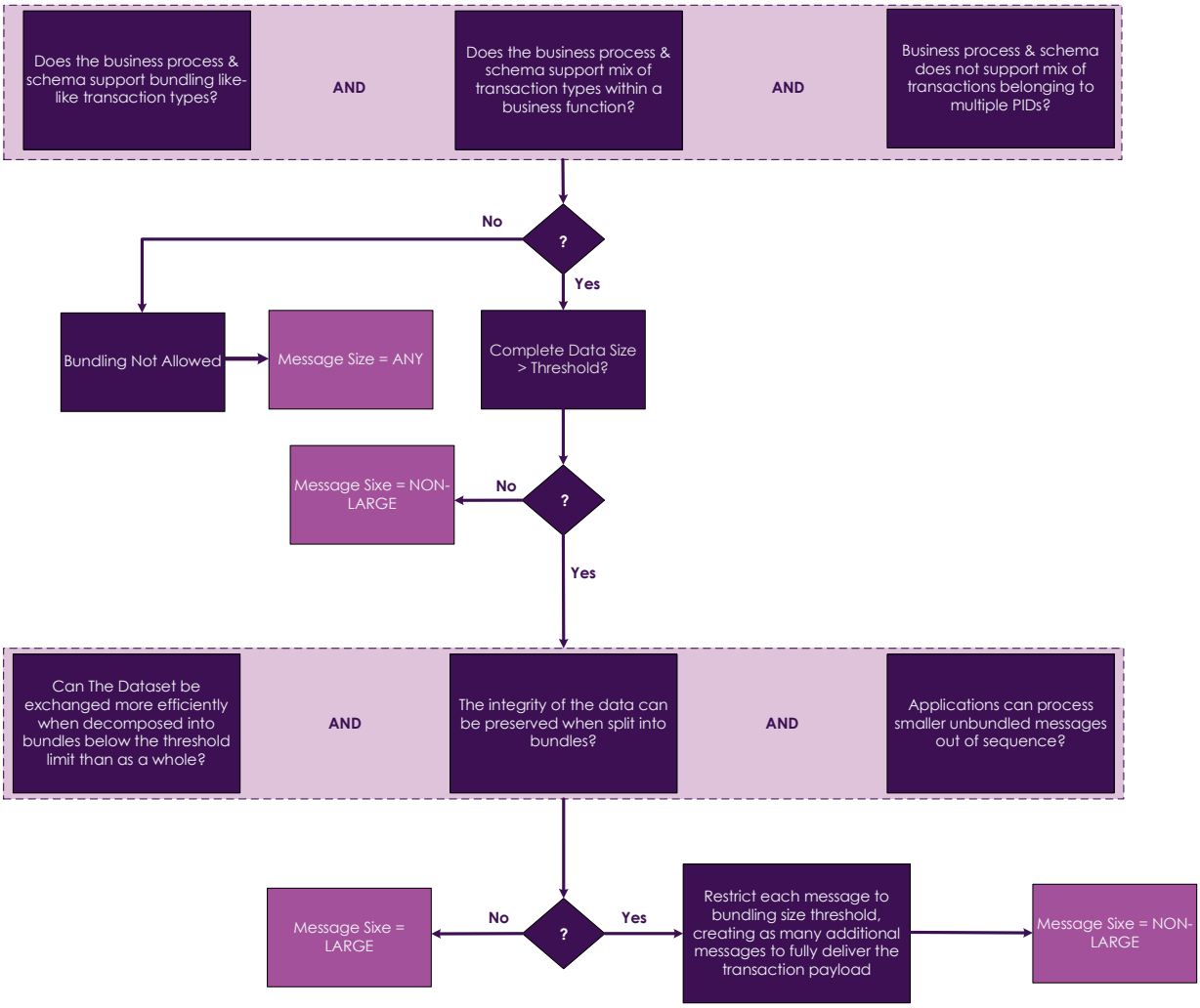


Diagram 3.3.2 Message Bundling Decision Tree

3.3.3. Pattern Support for Message Bundling

The IDX framework acknowledges that some transactions are better fulfilled as 1:1 or 1:many; certain patterns can invoke message bundling where the business contexts meet the criteria in section 3.3.2 provided the pattern supports message bundling.

Message Pattern	Supports Transaction Bundling?
Synchronous	Yes
Asynchronous	Yes
Fire & Forget	Yes
Inquiry with flexible payload formats	No

Table 3.3.3 Pattern Support for Message Bundling

3.3.4. Message Volume Classification

As an input to message compression and bundling, the message size must be determined to apply the most appropriate options.

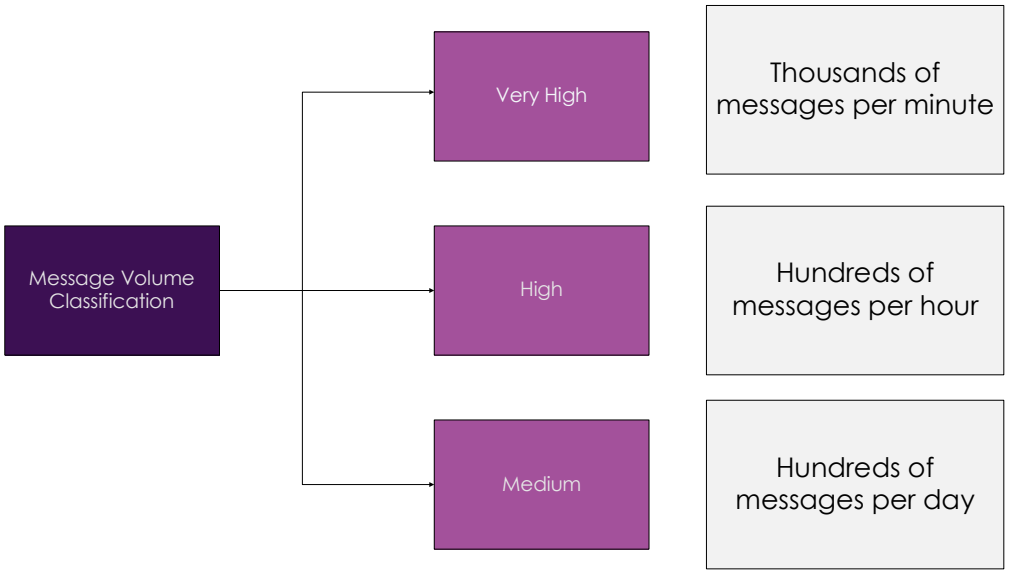


Diagram 3.3.4 Message Volume Classification

3.3.5. Decision Tree for Message Compression and Bundling Outcomes

Following the decision tree for message compression and bundling assesses the following qualifiers as input to the decision tree:

1. Message Size
2. Message Volume
3. Pattern Support
4. Business Context

Based on the determinants above in conjunction with message pattern, then bundling and compression eligibility and method is to be applied.

1. No bundling
2. Bundling within file size threshold
3. Eligible for large file share process

Message Size	Bundling Option
Any	No bundling
Non-Large	Bundled to file size limit
Large	Eligible for large file share process

Table 3.3.5 Message Size Bundling Options

3.4. Channel Decision Tree

The channel decision tree is a structured approach designed to determine the most appropriate communication channel for different types of data exchanges between Participants and AEMO. By following the decision tree, the industry will transmit information efficiently, securely, and in a manner that meets the specific requirements of each use case.

The decision tree considers various factors to select the appropriate channel, including the message patterns, message size, frequency, response requirements, and the nature of the data being transmitted. The goal is to match each use case with the channel that best supports its needs while optimizing performance and minimising costs.

3.4.1. Channel Selection for Synchronous Transactions

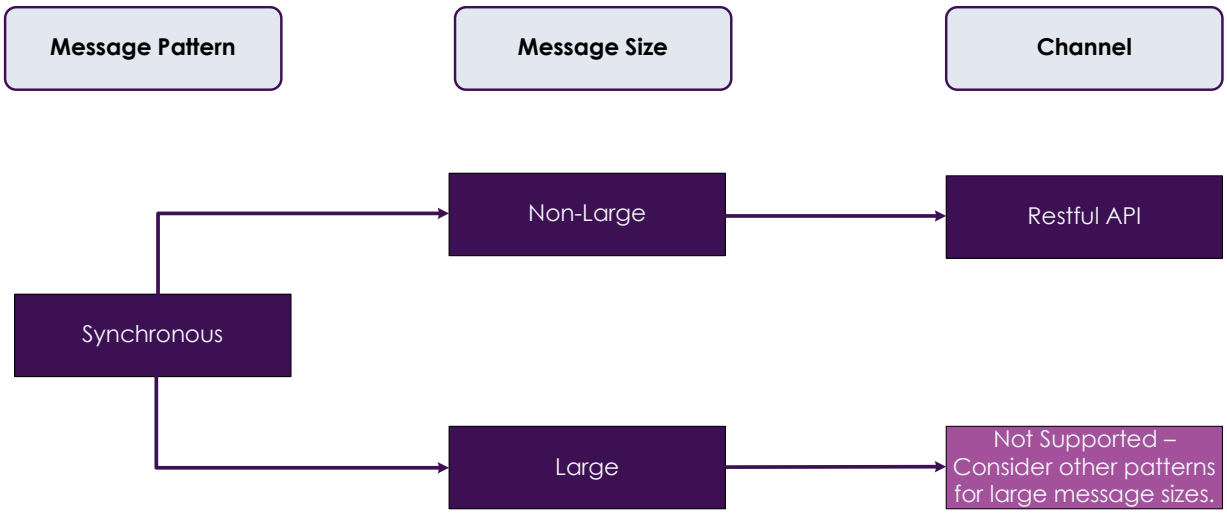


Diagram 3.4.1 Channel Selection for Synchronous Transactions

3.4.2. Channel Selection for Asynchronous Transactions

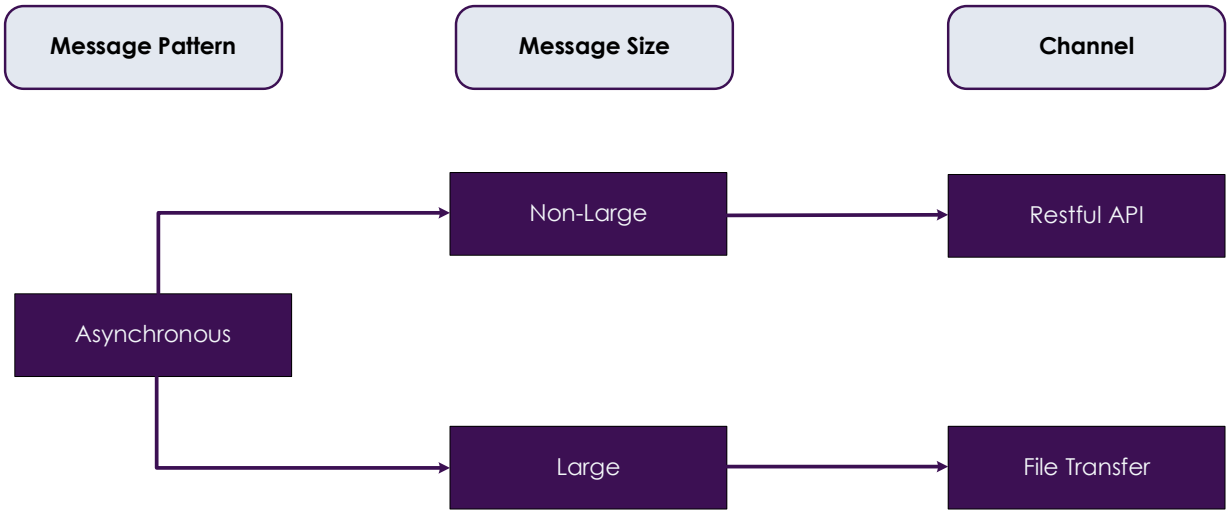


Diagram 3.4.2 Channel Selection for Asynchronous Transactions

3.4.3. Channel Selection for Fire and Forget Transactions

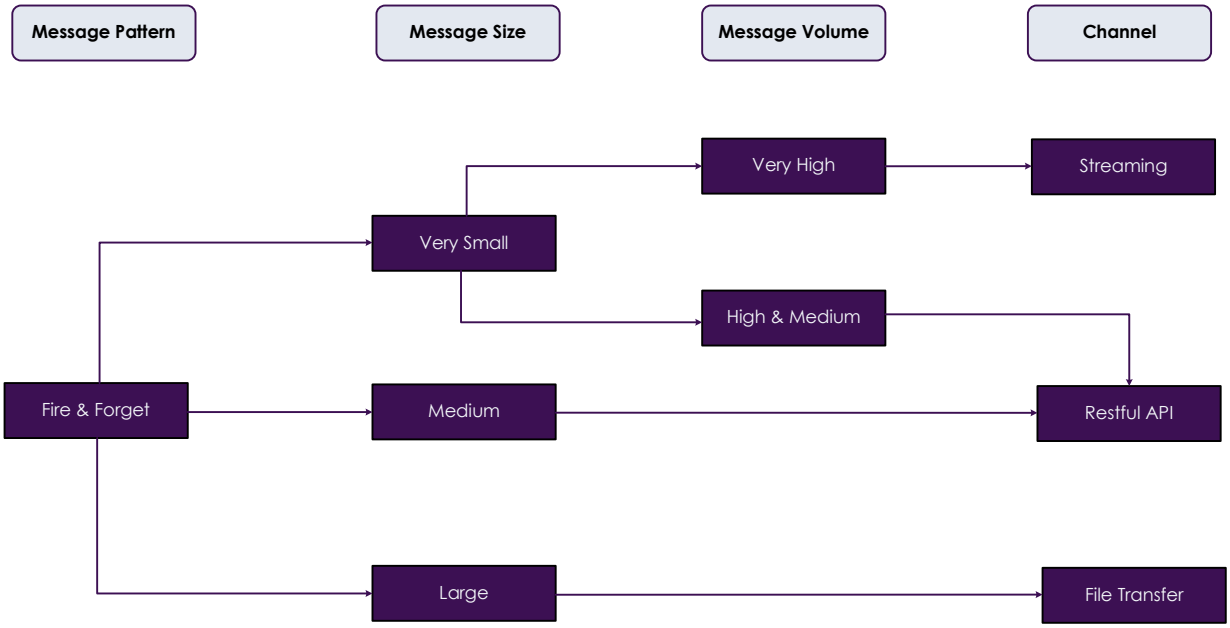


Diagram 3.4.3 Channel Selection for Fire and Forget Transactions

3.4.4. Channel Selection for Inquiry-Flexible Transactions

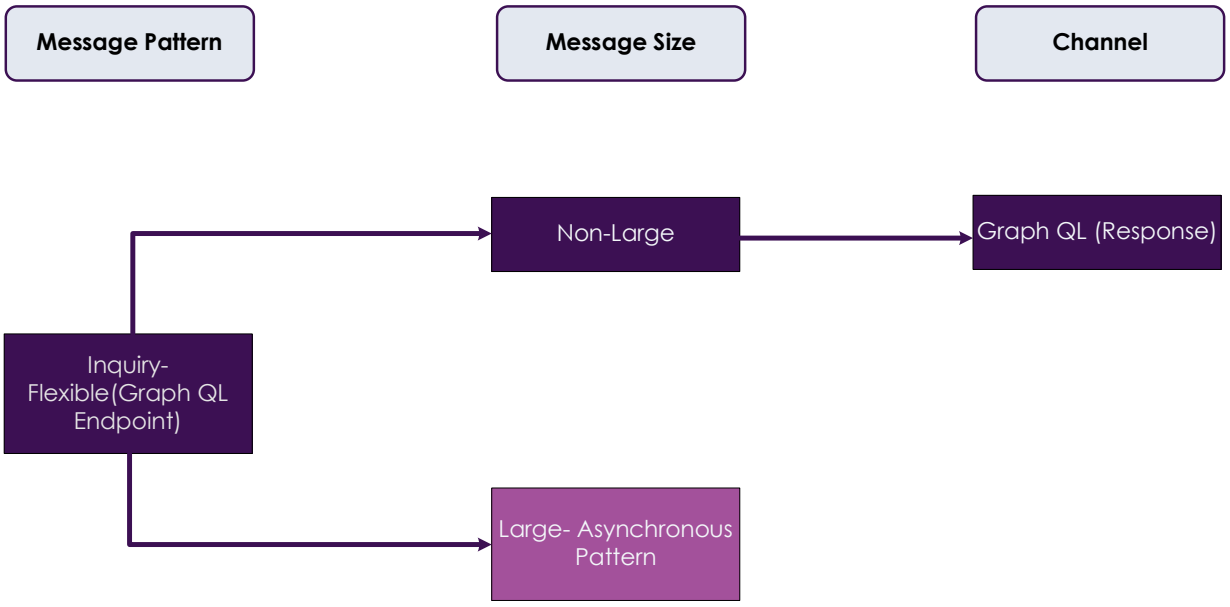


Diagram 3.4.4 Channel Selection for Inquiry-Flexible Transactions

3.4.5. Channel Selection for Inquiry-Flat Transactions

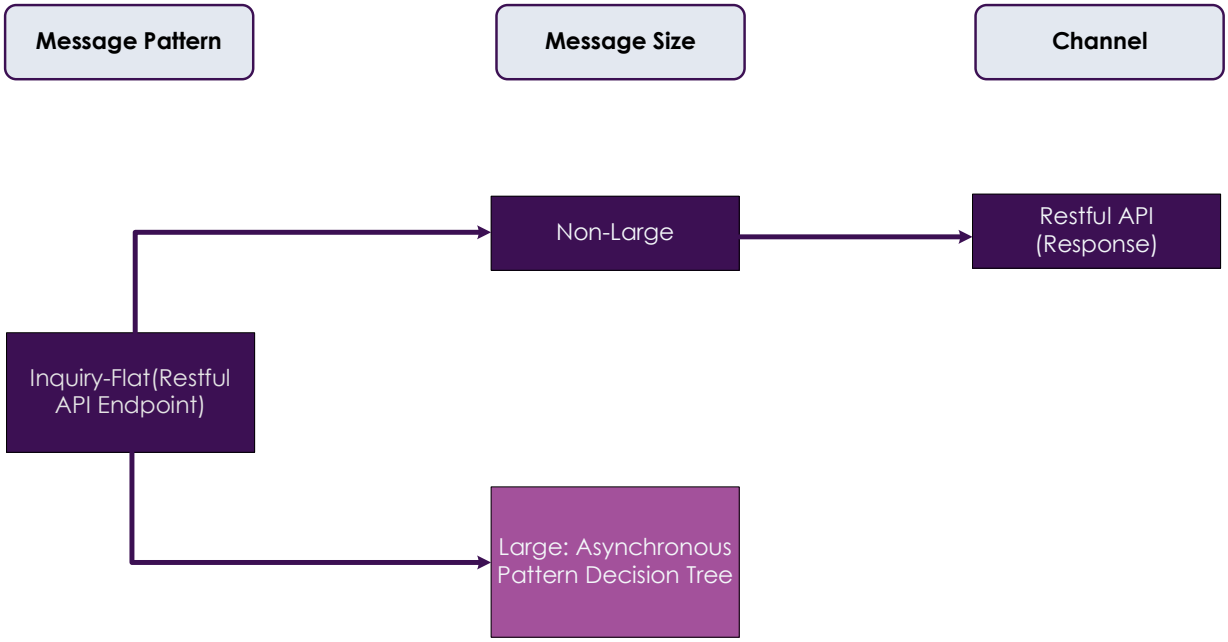


Diagram 3.4.5 Channel Selection for Inquiry-Flat Transactions

3.4.6. Chanel Selection Decision Tree Outcomes

Following the decision tree for message pattern makes the determination of most appropriate channel for participants to direct transactions, which can be one of the following outcomes:

Channel	Description
RESTful API	A web-based communication channel that conforms to REST architectural principles, utilising standard HTTP methods (GET, POST, PUT, DELETE) to interact with resources. The representation of resources, available endpoints, and methods is defined in a RESTful API schema.
Large File Share	A file-based transfer channel designed to securely and reliably transfer large files or payloads between organisations that are too large to be sent via RESTful API channel.
Inquiry Service (GraphQL)	An API-based channel that enables clients to request specific data structures using a query language.
Streaming	A streaming pattern channel designed to securely and reliably transfer near real-time continuous collection and movement of data, handling high volumes, at scale, with high throughput and low latency. Streaming will not be implemented until such time that a valid use case is adopted by the IDX Platform.

Table 3.4.6 Decision Tree Outcomes for Channel Selection

3.5. Payload Decision Tree

3.5.1. Overview

The payload decision tree is a structured approach designed to determine the most appropriate payload type for different types of data exchanges between Participants and AEMO.

Schemas will be per business function per payload format. This permits schema updates on a portion of a business function for that payload format, which allows for faster version changes and greatly minimises effects on participants.

The decision tree considers various factors to select the appropriate payload, including the data structure, patterns, message size, frequency, response requirements, and the nature of the data being transmitted. The goal is to match each use case with the channel that best supports its needs while optimising performance and minimising costs.

3.5.2. Payload Decision Tree

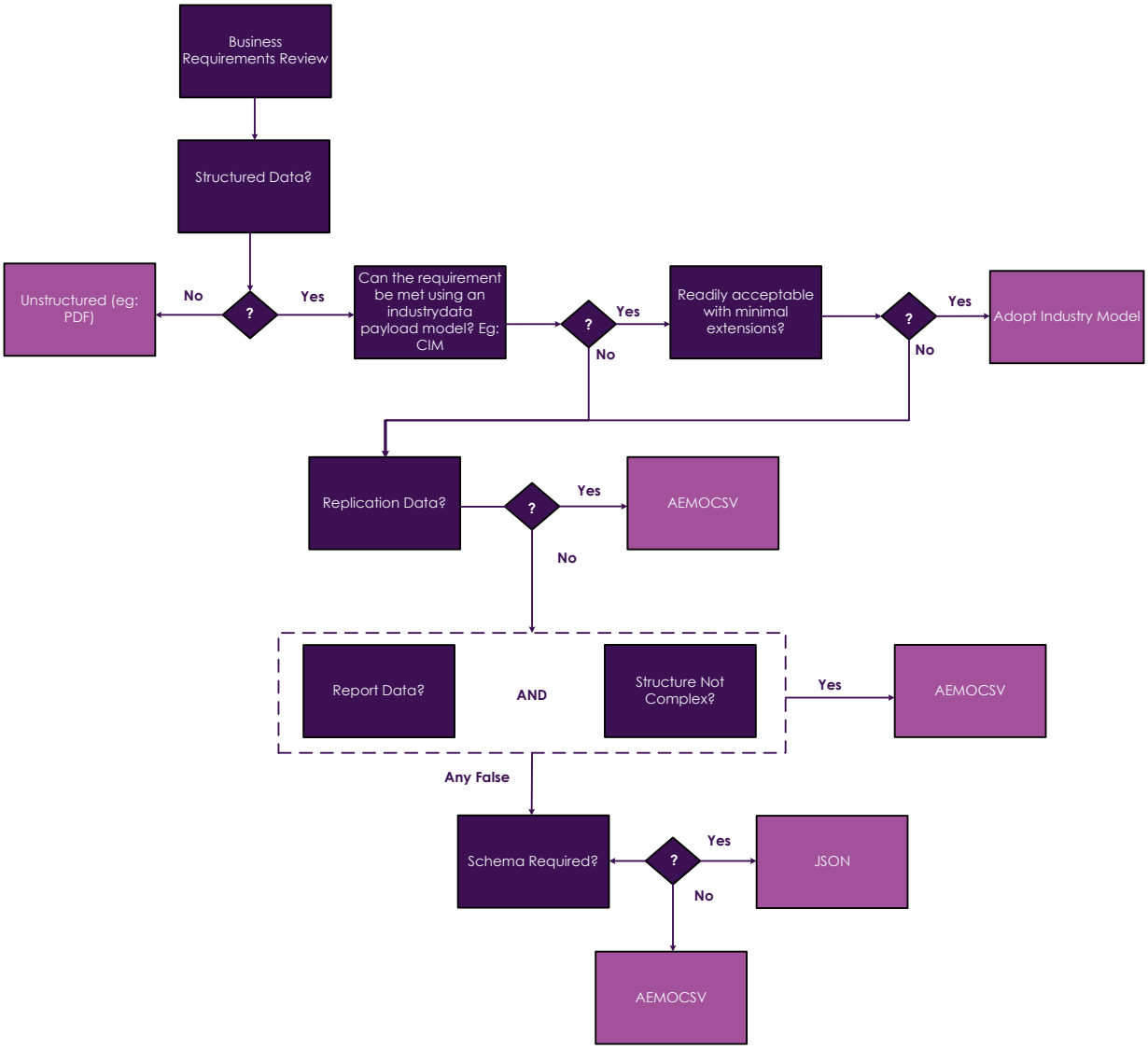


Diagram 3.5.2 Payload Decision Tree

3.5.3. Payload Decision Tree Outcomes

Following the decision tree for payload makes the determination of most appropriate payload for a use case, which can be one of the following outcomes:

Payload Type	Description
JSON	JSON is a lightweight data-interchange format that is easy to read and write for humans and easy to parse and generate for machines.
AEMO_CSV	The AEMO_CSV (Comma-Separated Values) format is a standardised data format used by the Australian Energy Market Operator (AEMO) for data interchange. This format is designed to facilitate the exchange of structured data between AEMO's systems and market participants.
Unstructured	Portable Document Format – E.g.: Settlement Statement Report Power System Simulation for Engineering files – E.g.: PSSE files Graphs – E.g.: jpg, bmp
Adopt Industry Standard	A need is identified for a business function specific to adopt an external industry standard as governed by a third party.

Diagram 3.5.3 Payload Decision Tree Outcomes

4. Communication Channels

4.1. RESTful API

4.1.1. Overview

AEMO chose RESTful (REST) architecture for participants to leverage in communications with IDX architecture because of its lightweight nature and ability to transmit data directly over HTTPS. REST is an alternative to SOAP and WSDL. The REST architecture makes it possible to start small, developing what is required with available resources, and scaling up as the number of services increase.

The REST approach uses the features of HTTP to make requests and follows these design principles:

1. Services are provided using HTTPS protocol over MarketNet and Internet.
2. Transfer of variety of payloads such as AEMO_CSV and JavaScript Object Notation (JSON).

Resources are addressed by mapping to a location within a hierarchy of URLs. For example, the root of the hierarchy might represent the web service/API application and provide a listing of the resources available. Drilling down one level then provides specific information about a particular resource, and further levels provide data from specific resource records.

AEMO's goal in implementing a RESTful IDX approach is to achieve the following:

1. Performance: quality of responsiveness.
2. Scalability: many users can simultaneously use the systems.
3. Generality: solve a wide variety of problems.
4. Simplicity: no complex interactions, easy to prove the system is doing as it is supposed to.
5. Modifiability: extensible in the face of new requirements and technologies.

4.1.2. JSON Format

The JSON payloads will primarily consist of multiple sections:

(a) data section (required for requests and successful responses), can consist of 3 sub-sections - header (optional), routing (optional) and transactions, or acknowledgments. This section contains the content of the payload.

  (i) header section (optional) - Although this is optional across APIs, it will generally be a mandated section and must be included for all B2B API requests. This will include all message level metadata, such as Message Identifier, Message Date/Time, Business Function ID, From Participant, To Participant(s), Market, Priority.

(b) error section (required for response that is an error) - contains details on the error that occurred.

(c) links section (optional) - contains a field called self that will have the fully qualified URI to the current request as a value. This is applicable to the API response only.

(d) meta section (optional) - contains the metadata of the payload.

4.1.3. API URL Standards

URL Structure	<protocol>://<domain>/<business function>/<version>/<business function resources> Example: https://api.nem.aemo.com.au/pqd/v1/bpqd
<Domain>	Common domain name per environment to all API within a market.
<Business Function>	A Business Function typically equates to a Business Capability consisting of several related services which may fall under it, for example MTRD.
<Resources>	A Business Resource typically equates to a discrete business service within a business capability, for example Provide Meter Data (PMD) would be a Business Resource of the MTRD Business Function.

Table 4.1.3 API URL Standards

4.1.4. Methods

HTTP Method	Operation
GET	Requesting a resource / retrieving data.
POST	Submitting or creating a new resource (generally requires a payload)
PUT	Create or update (replace) a resource. This can be useful for syncing data.
PATCH	Partially update data. Only passed data will be updated.
DELETE	Delete Data

Table 4.1.4 API Methods

4.1.5. HTTP Request

(a) All HTTP requests utilise the API Methods outlined in this document; however, some Business Functions may only support a subset of the methods.

(b) All HTTP requests adhere to the naming scheme outlined in this document.

(c) All HTTP requests will have a well formatted header in line with what is outlined in this document.

(d) Payloads included in the HTTP request adhere to the specifications outlined in the Business Function's or Business Resources technical specification document.

(e) Requests and headers which do not meet the industry specifications may be rejected by AEMO or other participants.

4.1.6. HTTP Response

(a) All participants must respond with a HTTP response within the technical exchange of messages.

(b) All Acknowledgements and Error Responses follow the JSON Format outlined in this document.

(c) Where possible all responses should adhere to the Australian consumer data standards.

(d) The HTTP response has:

(i) A successful request to the web services service is indicated by an appropriate 2xx response code.

(ii) Appropriate HTTP response codes for technical / payload validation failures e.g. 4xx / 5xx.

(iii) All error responses adhere to the RFC9457 Standard.

4.2. Inquiry Flexible

This section is a placeholder which will be addressed v0.4 (Q2 2026) of the document.

4.3. Large File Share

This section is a placeholder which will be addressed v0.4 (Q2 2026) of the document.

4.4. User Interface

This section is a placeholder which will be addressed v0.4 (Q2 2026) of the document.

4.5. WebSockets

WebSockets are a communication protocol that provides full-duplex communication channels over a single, long-lived connection. This means that both the client and server can send and receive messages simultaneously, without the need to repeatedly open and close connections.

AEMO uses WebSockets to push event notifications to Participants in scenarios such as:

- A new message is available for pick up from their IDX Hub Outbound Queue
- A flow control breach event is currently active
- Outage events

Note: WebSocket Event notifications are not the message/payload, but an alert for Participants to retrieve from their outbound queue.

The following steps outline the high-level steps for the WebSockets communication channel:

Connection Establishment:

- A WebSocket connection starts as an HTTP request, known as the WebSocket handshake. The client sends an HTTP request to the server, requesting an upgrade to the WebSocket protocol.
- If the server supports WebSockets, it responds with an HTTP 101 status code, indicating that the protocol is switching to WebSocket.

Full-Duplex Communication:

- Once the connection is established, both the client and server can send and receive messages independently of each other. This is known as full-duplex communication.
- Messages are sent in frames, which can be either text or binary data.

Persistent Connection:

- The WebSocket connection remains open as long as both the client and server agree to keep it open. This reduces the overhead associated with repeatedly opening and closing connections.

Closing the Connection:

- Either the client or server can close the WebSocket connection by sending a close frame. The other party responds with a close frame, and the connection is terminated.

By maintaining a single, long-lived connection, WebSockets reduce the overhead associated with repeatedly opening and closing connections making them ideal low latency bi-directional

communication. With his concept in mind, WebSockets are used in the AEMO Gateway Software to establish an event notification channel for participant messages and transactions, enabling real-time, efficient, and bi-directional communication between participants' systems and the IDX Hub.

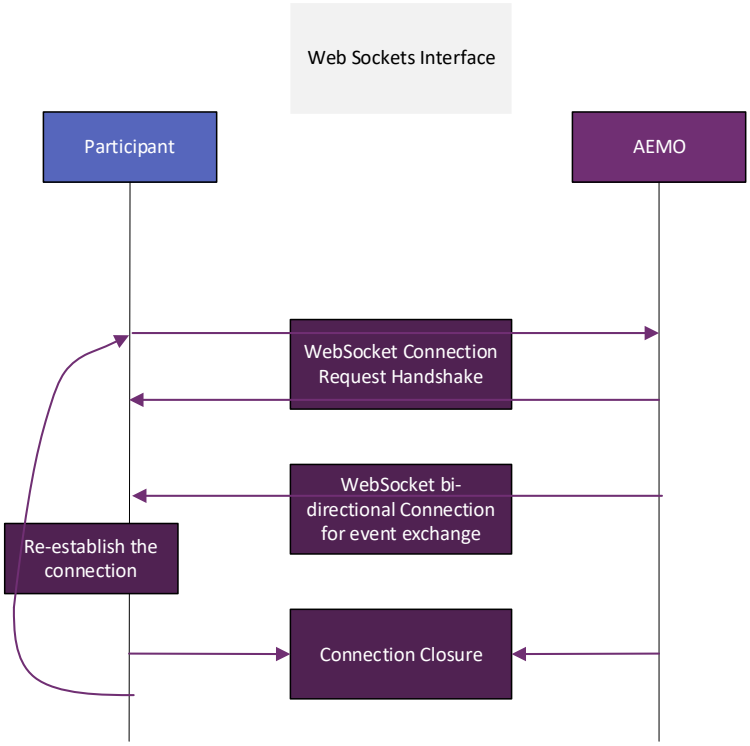


Diagram 4.5 WebSockets High-level interface

This section will be elaborated on in later versions of the document.

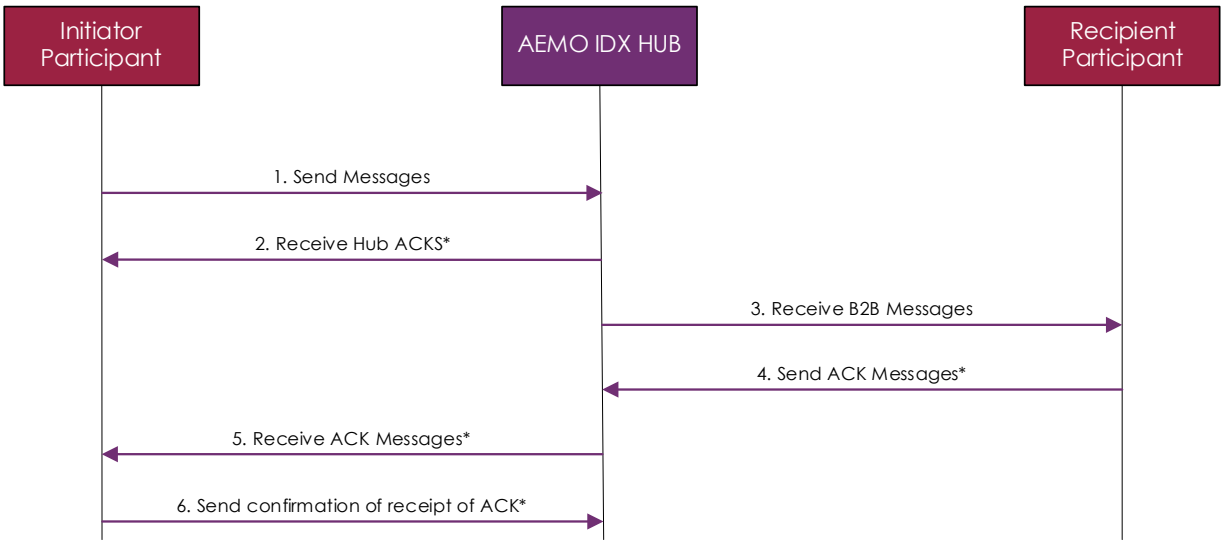
5. IDX Hub Message Flow

5.1. Overview

The IDX platform utilises multiple patterns to ensure efficient and reliable data exchange between participants and AEMO across a wide range of business functions and scenarios. By leveraging multiple patterns, AEMO and participants can select the optimal way (via decision tree) to transfer messages between each other for a wide range of business functions.

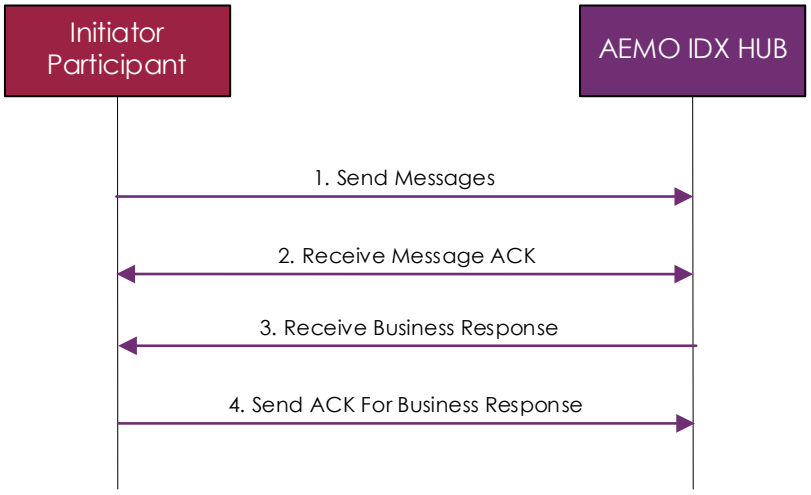
The diagram below illustrates the high-level message communications when interfacing with the IDX Hub.

B2B:



*Optional messaged based on pattern

B2M:



5.2. IDX Message Patterns

5.2.1. Synchronous

Synchronous message patterns involve sending a message and waiting for an immediate response before proceeding. This pattern is particularly useful for scenarios where real-time interaction is required, data reconciliation is necessary, or there are regulatory requirements to provide and confirm data. It ensures that the sender receives a response before continuing with other tasks, making it ideal for critical operations that depend on immediate feedback.

This section will be elaborated on in later versions of the document.

5.2.2. Asynchronous

Asynchronous are designed to handle scenarios where additional processes or validations are required to enable a response to the request. Unlike synchronous patterns, asynchronous patterns do not require an immediate response. Instead, they follow a multi-legged approach to deliver the business response.

Asynchronous message patterns are particularly useful for situations where the response may take some time due to the need for further processing or validation. This pattern ensures that the sender can continue with other tasks while waiting for the response, making it ideal for operations that do not require real-time interaction.

The diagram below illustrates the high-level communication flow when interfacing with the IDX Hub with an Asynchronous message pattern:

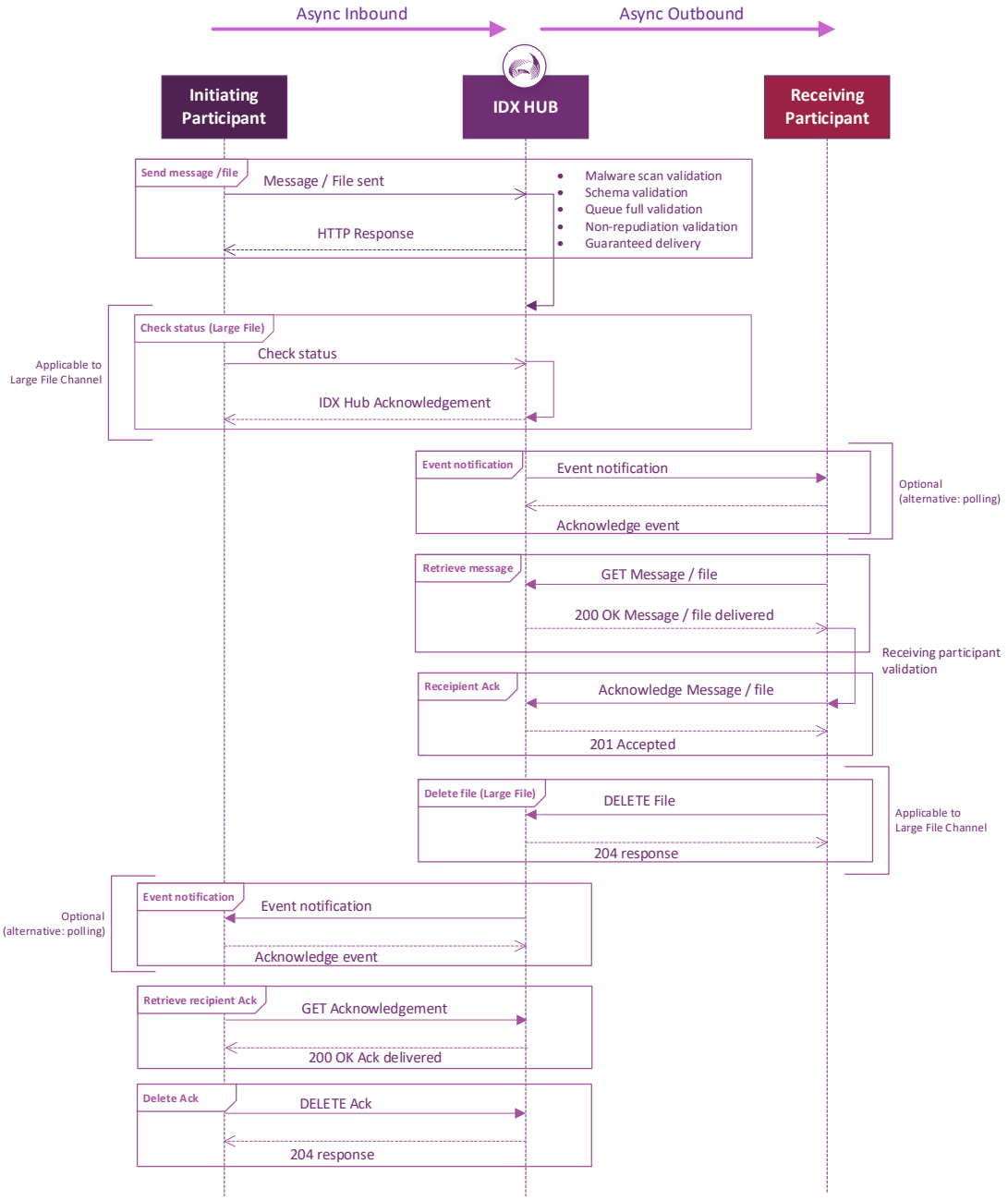


Diagram 5.2.2 Asynchronous message pattern - happy path.

This section will be elaborated on in later versions of the document.

5.2.3. Fire and Forget

Fire and forget is a messaging pattern where a message is sent to a recipient without requiring a business response or message acknowledgment. Hub Message Acknowledgement may optionally be returned to the initiator. It is ideal for situations where the sender does not need to wait for a reply and can continue processing other tasks.

Given the lower governance of the fire-and-forget messaging pattern, which lacks dependent regulatory message dependencies and reconciliation, participants are not required to send a transaction

acknowledgment (TACK) or business response to indicate the acceptance or rejection of the market message. This approach keeps the fire-and-forget messaging pattern lean and streamlined, making it ideal for low validation and lower stakes use cases.

As there are no returned acknowledgements from the recipient, the onus is on the recipient to issue a DELETE message to clear their queue after consuming the message. If the message is not deleted by the recipient, the time to live (TTL) protocol will remove the message from the IDX hub queue after a configured (per business function) amount of time.

Fire and Forget message pattern supports the following communication channels, payload types and message types:

Communication Channels:

Communication Channel	Support for Fire and Forget
RESTful API	✓
Large File Share	✓
IDX Web Application	✓
Inquiry Service	✗

Table 5.2.3.1 Fire & Forget Communication Channels

Payload Types:

Payload Type	Support for Fire and Forget
JSON	✓
AEMO .csv	✓
Unstructured	✓
Industry Format	✓

Table 5.2.3.2 Fire & Forget Payload Types

Message Types:

Message Type	Support for Fire and Forget
IDX HUB Acknowledgement	✓
Event Notification	✓
Message Acknowledgement	✗
Transaction Acknowledgement	✗

Table 5.2.3.3 Fire & Forget Message Types

The diagram below illustrates the high-level communication flow when interfacing with the IDX Hub with a Fire & Forget message pattern.

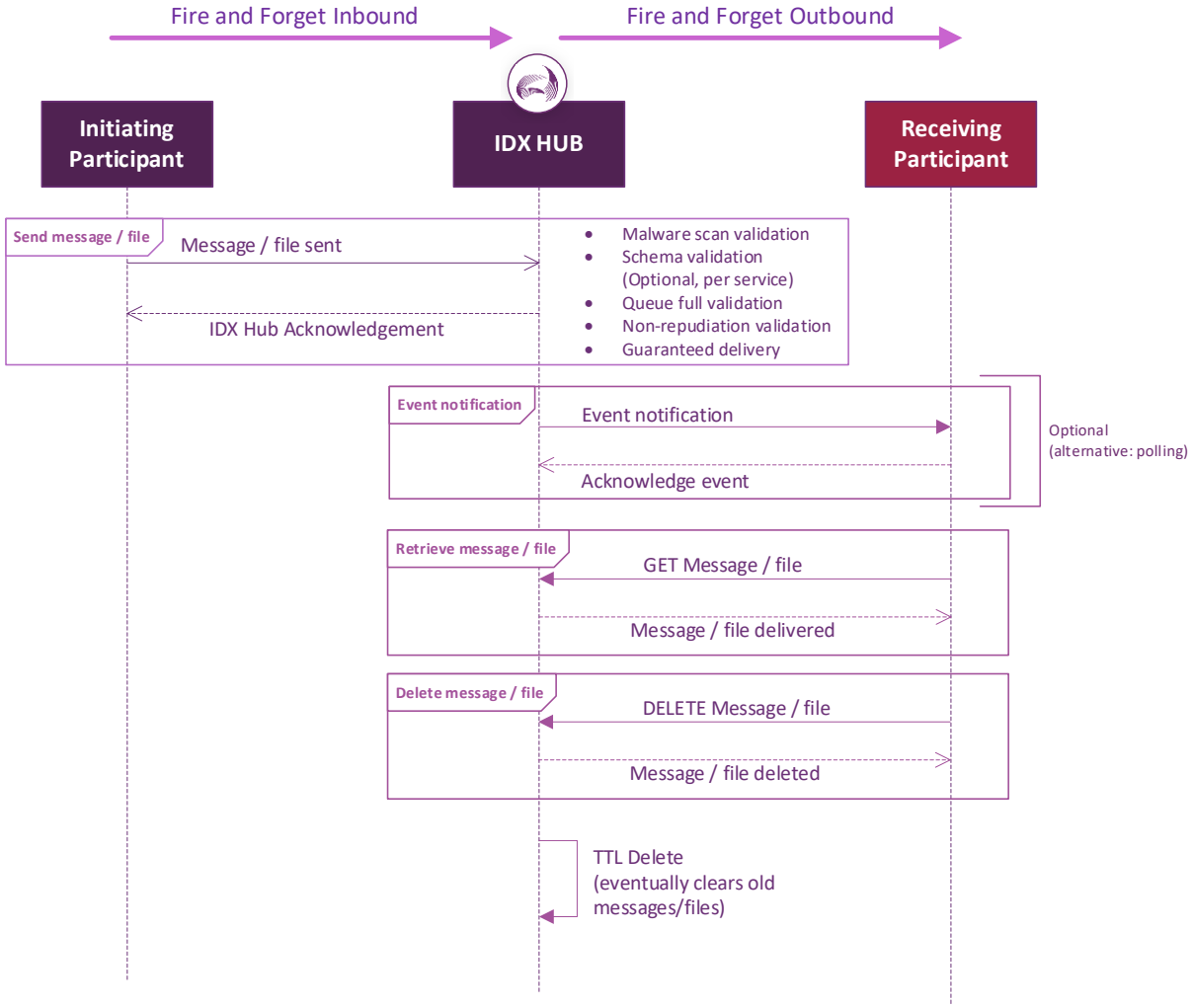


Diagram 5.2.3 Fire & Forget Message Pattern - Happy path

1. An initiating Participant sends a message or file to IDX Hub.
    1. AEMO performs a series of technical validation to ensure the integrity of the message/file and stores the message/file in the Participants inbox. The HTTP response or IDX Hub Acknowledgement (Business Function/Resource dependant) is sent back to the participant indicating the successful submission of the message and its guaranteed delivery to the recipient participant.
    2. AEMO re-signs the message/file and places it in the recipient's outbox.
    3. (option 1) Where the recipient is subscribed to WebSockets event notification, AEMO will trigger an event notification alerting the recipient of the pending message/file.
    4. (option 1) Where the recipient is subscribed to WebSockets event notification and has been alerted by AEMO about a new pending message, the recipient will respond to the notification with an acknowledgement. Failure to acknowledge the message will result in

additional notifications being sent by AEMO detailing the same information at the first notification.

5. (option 2) Alternatively a participant polls their Business Function End Point.
6. (option 2) Where a participant has polled their Business Function endpoint, IDX Hub responds back with a list of pending message's metadata for that endpoint.
7. Using the messageContextID provided by either option 1 or option 2, a recipient will send a GET request to retrieve the message from their Business Function endpoint.
8. AEMO provides the resigned message back to the recipient participant.
9. The recipient sends a DELETE request using the same messageContextID to IDX hub to remove the pending message from the endpoint after consuming the message.
10. AEMO responds back to the delete request indicating the message has been removed from their Business Function endpoint.
11. On successful DELETE request, the message is moved and compressed to the participant archive folder.
12. Where no DELETE request is processed, once enough time has passed the time to live (TTL) process will remove the pending message from the recipients Business Function endpoint and archive the message on behalf of the recipient.

5.2.4. Fan-in

This section is a placeholder which will be addressed in later versions of the document.

5.2.5. Fan-out

This section is a placeholder which will be addressed in later versions of the document.

5.2.6. Inquiry-Flexible

This section is a placeholder which will be addressed in later versions of the document.

5.2.7. Inquiry-Flat

This section is a placeholder which will be addressed in later versions of the document.

5.3. WebSockets

5.3.1. Overview

As stated in section 4.1.5. WebSockets are a communication protocol that provides full-duplex communication channels over a single, long-lived connection. This means that both the client and server can send and receive messages simultaneously, without the need to repeatedly open and close connections.

The WebSockets connection is used to open a long-lived channel of communicating between the IDX hub and a recipient to allow seamless communication over the channel. The channel remains open by allowing IDX hub and participants to send periodic ping messages, referred to as a WebSocket heartbeat, to ensure the connection is still alive. The recipient of the ping responds with a pong message, confirming that the connection is active. This process helps detect and handle connection issues, such as network interruptions or server failures.

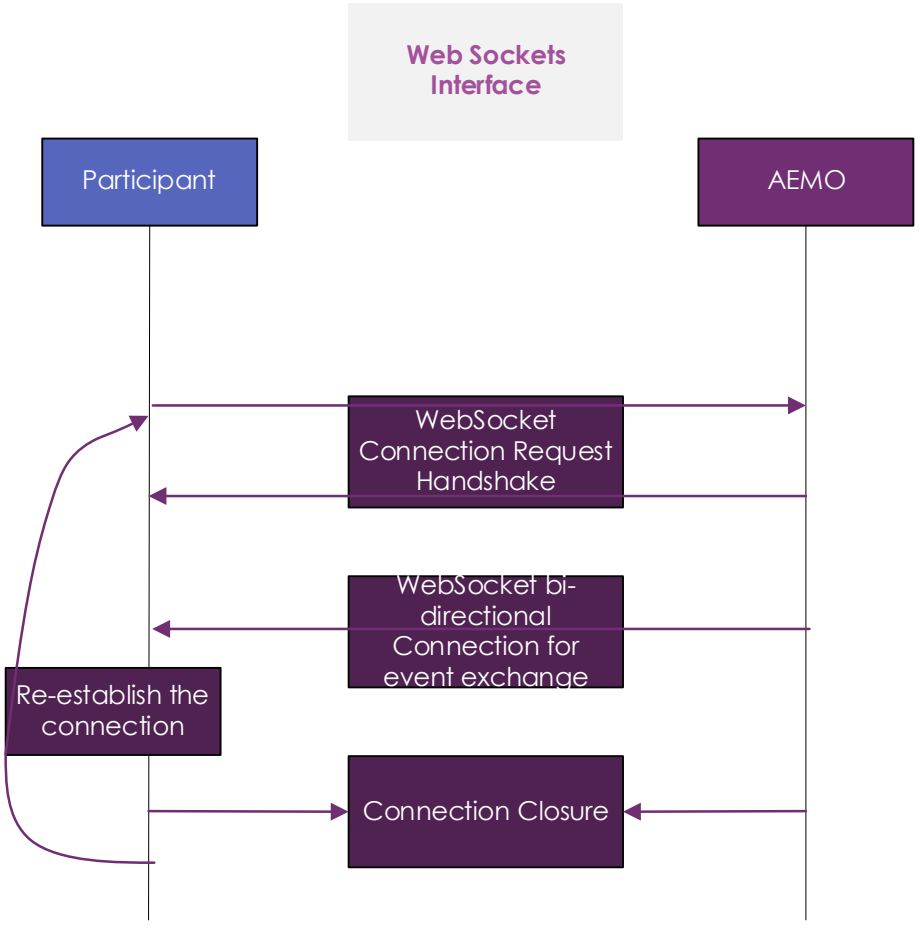


Diagram 5.3.1 High Level WebSockets interface

This section will be elaborated on in later versions of the document.

5.3.2. Subscription to WebSockets

Through the IDX Web Application, a participant can manage their IDX HUB queue, including whether they will be notified of new messages via the WebSockets event notification process. The alternative is to poll their IDX message queue periodically, and either or both options can be used at once.

This section will be elaborated on in later versions of the document.

5.3.3. Event Notification Messages

In relation to retrieval of messages from IDX hub, WebSockets are utilised to allow a persisting connection over a long period of time to notify participants of the availability of a message to be picked up. The event notification which is sent to the participant includes meta data about what the message is, so that the participant can determine its next steps for retrieval.

This section will be elaborated on in later versions of the document.

5.3.4. Type of Events

(a) All events should have an event acknowledgement or HTTP response.

WebSocket Event Type	B2B/B2M	Initiating Party	Event Acknowledgement Required?	Description
Event Notification	Both	AEMO	✓	This notification informs recipients that a message is available for retrieval. It includes metadata about the message in the IDX Hub queue to provide recipients with information regarding the type and priority of the message.
Flow Control Breaches	Both	AEMO	✗	This event is used for informing a participant of notification of flow control events – Participant Outbox is full – Insufficient delete rate
Flow Control Breaches	B2B only	AEMO	✗	This event is used for Informing the industry participants that a recipient has a full IDX Hub queue and that messages will not be passed through for the impacted participant.
Outage Events	Both	AEMO	✗	This event is used to broadcast whether business function end points are alive or experiencing an outage.
Ping/Pong Events	Both	Market Participants	✓	Handshaking framework to check the WebSocket connection is alive.

Table 5.3.4 Types of Events

This section will be elaborated on in later versions of the document.

5.3.5. Retry Mechanism

In the scenario a WebSockets connection goes down there are retry mechanisms in place to send messages left in IDX queue to the recipient.

Event draining

(a) While draining of pending outbound events after the reestablishment of a WebSocket connection in the event of the client gateway crash/outage or something else, a default priority will be established for each Business functions and draining will happen based on this priority.

(b) Draining of pending events within a business function will be done using the default business function sort order First in first out (FIFO) or Last in first out (LIFO).

(i) For FIFO business functions, events stored in IDX queue are sent to participants in order of being received, oldest to newest.

(ii) For LIFO business functions, events stored in IDX queue are sent to participants in order of newest to oldest.

(iii) Event draining has a limit of configurable number retries, after which events are not retried further.

This section will be elaborated on in later versions of the document.

5.3.6. Length of time for Authorisation

This section is a placeholder which will be addressed in later versions of the document.

5.4. Polling

As an alternative to WebSockets, IDX Hub supports polling. Polling involves a participant routinely pinging their IDX Hub Queue, which will return a response detailing messages (and associated meta data) which are waiting to be retrieved.

Exact data attributes are defined on a business function level; however some common attributes are tabled below:

Metadata attribute samples	Attribute value samples	Description
itemCount	Number	The count of messages in the returning call.
messageContextId	pqd_bpqd_l_retailer1_abcd1234	The unique reference number used for each message.
businessFunction	PQD	The business function for the message.
marketSegment	B2M/B2B	For NEM only, which segment of the market the business function exists in.
messageType	Message, ACK	The type of message.
messageDate	YYYY-MM-DDTHH:mm:ss.SSSZ	The date / time the message was received by IDX.
priority	H, M, L	The priority within the payload.

Table 5.4 Polling Sample Meta Data

A participant can then retrieve their messages based on a priority fitting their business functions.

This section will be elaborated on in later versions of the document.

5.5. Flow Control

5.5.1. Overview

The primary purpose of IDX is to function as an Industry Data Exchange, not to act as a storage medium for market data. To ensure efficient and cost-effective operation of the Industry Data Exchange, Participants using the exchange will be required to submit and consume messages from queues in the IDX framework to avoid the IDX Hub from being overloaded.

AEMO's IDX Hub utilises multiple flow control mechanisms to ensure that the platform is resilient and robust under all market conditions:

Flow Control Method	Description
Throttling	Throttling is the process of controlling the rate of requests that consumer (Participant) can make to a resource, it is implemented to protect the performance and quality of service of the resource.
Inbound quota limits	Business function based specific Inbound limits, i.e. limit on number of inbound messages that are allowed for a specific business function.
Outbound flow control	Process to manage the number of pending outbound messages in the Participant's queue for a Participant to be picked up and acknowledged.
Time to live (TTL)	Specific Expiry times set for the messages in the outbound store to be moved into archive upon non –acknowledgement of being processed.
Management of uncleared ACKS	If ACK messages are not routinely deleted by participants, then a message limit is implemented whereby AEMO will no longer accept inbound messages for that participant.

Table 5.5.1 Flow Control Methods

5.5.2. Flow Control Watermarks

(a) Watermark levels are a configured attribute per business function (agreed with Industry at the time of onboarding) which indicates how full a participant's queue is and triggers flow control notifications and flow control restrictions accordingly.

(b) The assigned Watermark values are based on the number of pending messages in a Participant IDX Hub Queue.

(c) A separate set of Watermark threshold limits is applied to pending standard messages and to pending message ACKs in a Participants IDX Hub Queue.

(d) The count of messages which underpin the Watermark counts is based on pending messages ready to be consumed at a participant's business function endpoint.

(e) The count of message ACKs which underpin the Watermark ACK counts is based on pending message ACKs ready to be consumed at a participant's business function endpoint.

(f) A pending message is defined as a message which has not been ACK'd or Deleted (whichever is appropriate for the message pattern). Using GET Message successfully, does not change the pending status of a message.

(g) Flow Control watermark levels are capable of triggering notifications to:

(i) The Impacted Participant (where subscribed to WebSockets for that Business Function)

(ii) All Participants for B2B (any participant subscribed to WebSockets for that Business Function).

(h) Flow Control watermark levels are also available via the Flow Control API, which can be polled for active breach events and watermark information.

(i) The Flow Control Watermarks are as follows:

Watermark Level	Definition
Low	• A low watermark indicates normal operation volumes of pending messages in a Participant's outbox. • In scenarios where a Participant has reached the High Watermark, the Participant must return their pending message count down to the low water mark to resume normal operations and have flow control restrictions removed. • When a low watermark has been reached after a flow control breach has occurred, a notification will be triggered to all Participants (any Participant subscribed to WebSockets for that Business Function) that the impacted Participant has resumed normal operations and messages will no longer be rejected by AEMO.
Warn	• A warn watermark indicates that a Participant has reached a pending message count above the Warn watermark threshold and is experiencing issues clearing their message from their outbox. • Exceeding the warn watermark creates a Warn breach. • Reaching the warn watermark status will trigger a Flow Control Breach, which will notify all Participants (any participant subscribed to WebSockets for that Business Function) of the Flow Control Breach. • A warn watermark does not trigger flow control restrictions, nor does it remove flow control restrictions.
High	• A high watermark indicates that a Participant has reached a pending message count above the High watermark threshold and is experiencing significant issues clearing messages from their outbox. • Exceeding the high watermark creates a queue full breach. • Reaching the high watermark status will trigger a Flow Control Breach, which will notify other participants (via WebSockets where subscribed) of the Flow Control Breach. • While the queue full breach is active, all messages to the impacted Participant will be rejected by AEMO. • To remove the breach, the impacted participant must reduce their pending message count below the low watermark threshold.

Table 5.5.2 Flow Control Watermark Definition

5.5.3. Flow Control Breach Events

(a) Flow Control Breach Events are generated by:

(i) Increasing or decreasing the pending message count above or below the configured Flow Control Watermarks for a given business function or;

(ii) Triggered by Participant who enables/disables a Manual Stop for their queue.

(b) Flow Control Breach Events drive the Outbound Flow Control and Management of Uncleared ACKs Flow Control restrictions which can be put in place during periods of active breaches.

Breach Event Type	Definition
Queue Full	• Occurs when a Participant has exceeded the high watermark threshold. • The impacted participant can no longer receive new messages for this business function. • An initiating Participant attempting to send messages to the impacted participant will be rejected by AEMO on the recipient's behalf. • Queue full breach can only be removed by reducing the pending message count to below the low watermark threshold.
Manual Stop	• A Participant may request a manual stop for their own queue. This can be requested using the Flow Control End Point or via the IDX Web Application. • A manual stop operates similar to queue full breaches, whereby a notification is sent to Participants (where subscribed to WebSockets) and all messages sent to the stopped participant are rejected by AEMO. • It is expected that participants will only request a Manual Stop during exceptional circumstances such as an extended system outage. • A Manual Stop can only be removed by the Participant who requested it.
Queue Active	• This notification can only be created following either a Queue Full, Manual Stop or Warn Breach. The recipient participant has either reduced their pending message count within the expected flow control limits (below the low watermark threshold), or has requested the Manual Stop to reactivate their Business Function End Point. • Any flow control restrictions which may have been in place due to the prior breaches are removed.
Warn Breach	• Occurs when a Participant has exceeded the warn watermark threshold. • The impacted participant will not have Flow Control restrictions enforced, however all Participants are notified of the breach (where Participants are subscribed to WebSockets).
ACK Queue Full	• The participant has exceeded their outbox ACK queue limit. • The impacted participant can no longer send new messages to other Participants for the impacted business function. • Notifications are triggered for ACK Queue Full breaches only to the impacted Participant (where subscribed to WebSockets).
ACK Queue Warn	• The participant is close to exceeding their outbox ACK queue limit. • The impacted participant is still able to send/receive new messages but is close to reaching the ACK High threshold and the associated Flow Control restrictions being enforced. • Notifications are triggered for ACK Queue Warn breaches only to the impacted Participant (where subscribed to WebSockets).
ACK Queue Cleared	• This event can only be created following an ACK Queue Full breach. • The impacted participant has reduced their message ACK count to below the low threshold. • Where being enforced, the participant will no longer have flow control restrictions enforced for their messages sent to IDX/Other Participants.

Table 5.5.3 Flow Control Breach Events

5.5.4. Rate Limit

(a) Rate Limiting is the process of controlling the rate of requests that consumer (Participant) can make to a resource, it is implemented to protect the performance and quality of service of the resource.

(b) The specific quote limits will be defined per business function/resource in a dedicated technical specification.

Channel	Throttling Options
RESTful API	Rate limiting can occur based on any combination the following: • Message Count • Time duration

	• Participant ID • Business Function • Business Resource • API Method
Large File Share	• ParticipantID based quotas based on the business function

Table 5.5.4 Channel Rate Limit Options

5.5.5. Inbound Quota Limits

(a) AEMO application set specific limits on the number of inbound messages that can be processed within a given timeframe to ensure system stability and performance.

(b) These limits help prevent system overloads and ensure that all participants' messages are processed efficiently.

(c) This is enforced as a policy all business functions applied per Participant.

5.5.6. Outbound Flow Control

There is a limited number of messages per business function for recipient participants. If the participant does not retrieve messages frequently enough, the limit can be reached. When reached, initiating participants who send messages to that participant with a full pending messages queue will receive a Negative Acknowledgement (NACK) from IDX hub, indicating that the participant is not able to receive new messages at this time due to a full pending message queue.

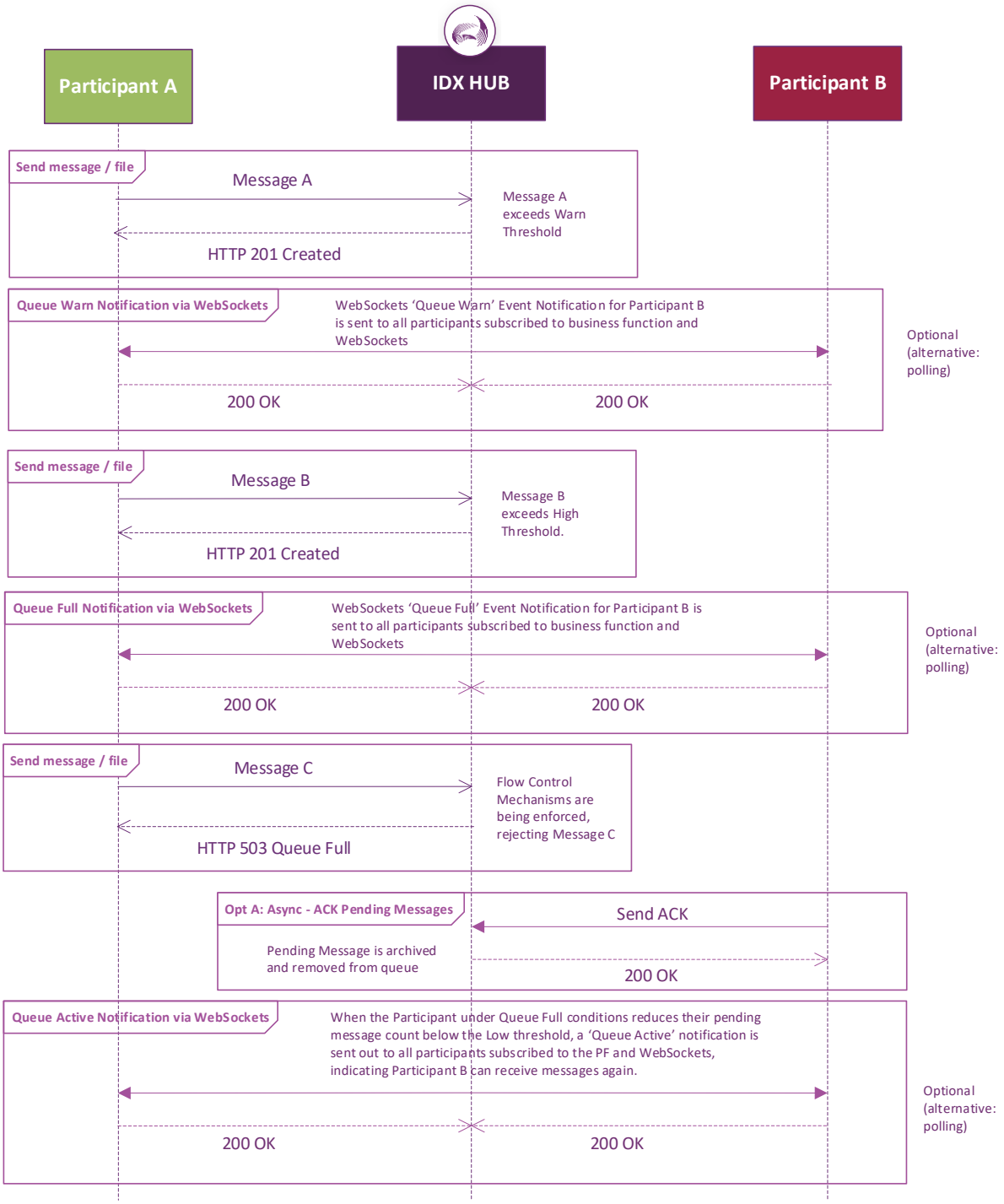


Figure 5.5.6 Outbound Flow Control – End to end

5.5.7. Time to Live (TTL)

(a) Each business function will have a configured Time to Live (TTL). These are defined on the Business Function's dedicated technical specification.

(b) Messages are archived from the participant's Business Function Endpoint once a message which exceeds the configured TTL time threshold.

(c) If the Business Function's High Watermark is reached before TTL window removes messages, then other flow control mechanisms such as stopping new messages from being added to the queue will be triggered.

5.5.8. Management of Uncleared ACKs

(a) Participants must delete ACK messages from their IDX Hub Queue after consumption of the ACK.

(b) A high count of pending ACKs maybe eventually result in the ACK High Threshold being met if they are not removed.

(c) Where the ACK High Threshold is met or exceeded, IDX will reject inbound messages from the participant with the High ACK queue, for the impacted business function.

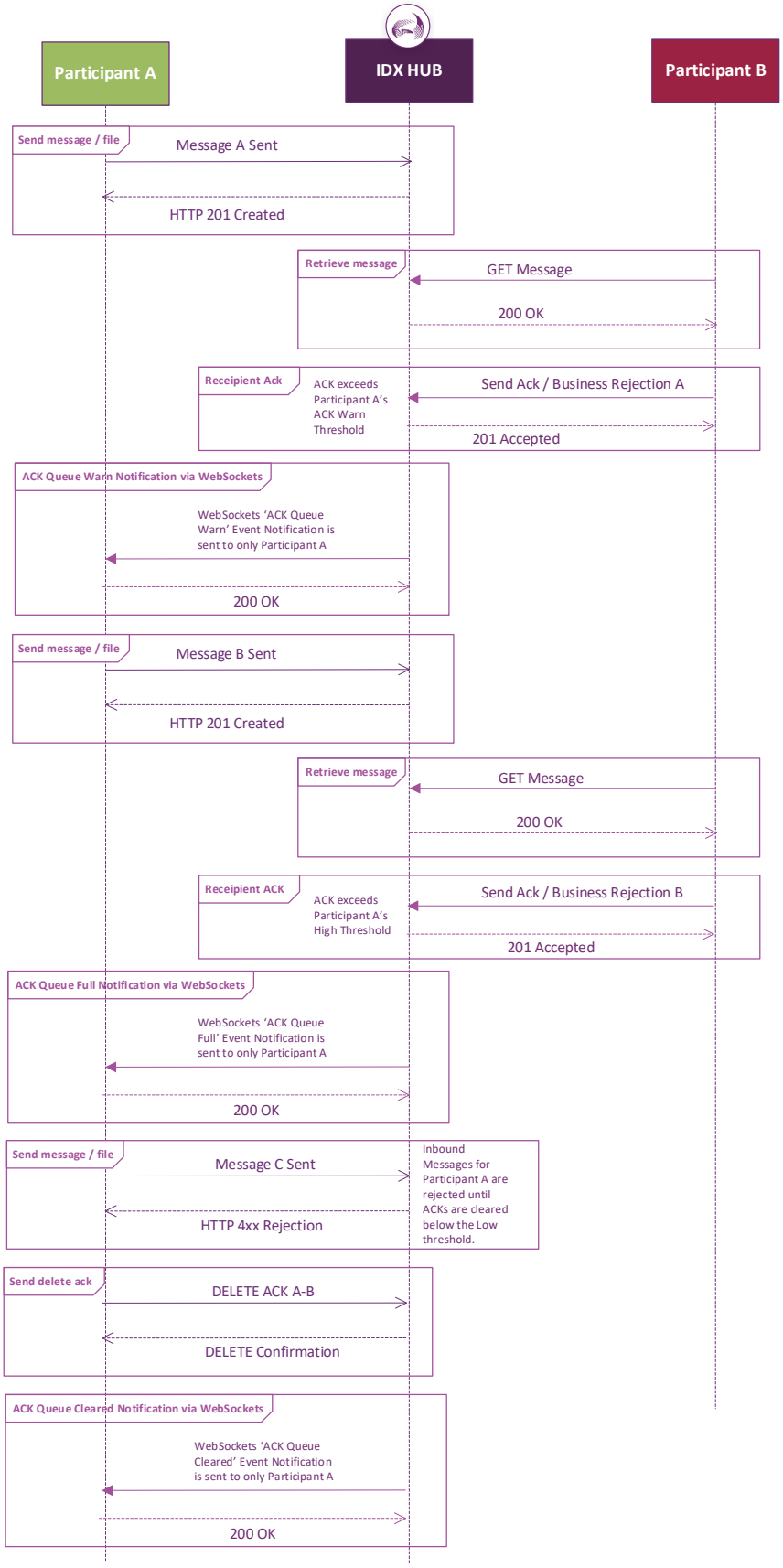


Figure 5.5.6 Management of Uncleared ACKs – End to end

5.5.9. Flow Control APIs

The Flow Control APIs are used as following:

Call Type	HTTP METHOD	URL	Description	API Specification Link
GET Flow Control Status	GET	/flowControl/status/	Retrieves the following information: • Pending Message Count • Pending ACK Count • High Watermark • Warn Watermark • Low Watermark • ACK High Watermark • ACK Warn Watermark • ACK Low Watermark • Active Breach Events	TBD
GET callers list of active breaches	GET	/flowControl/businessFunctions	Gets a summary of the calling participant's breaches for all business function.	TBD
GET callers list of active breaches	GET	/flowControl/businessFunctions/{businessFunctionId}	Gets a summary of the calling participant's breaches for a particular business function.	TBD
GET PID breaches	GET	/flowControl/participants	Get a list of Participants who have active flow control breach events.	TBD
GET PID breaches	GET	/flowControl/participants/{participantId}	Get active breach events for a Participant.	TBD
GET PID breaches by BF	GET	/flowControl/participants/{participantId}/businessFunctions	Get list of business functions with active breach events for a Participant.	TBD
GET PID breaches by BF	GET	/flowControl/participants/{participantId}/businessFunctions/{businessFunctionId	Get a business function's active breach events for a Participant.	TBD

Table 5.5.9 Flow Control APIs and Open API Specifications

6. IDX Web Application

6.1. Overview

The IDX Web Application is a section within AEMOs Markets Portal to manage Transactions, Participant Accreditation, Participant Controls, Service Dashboard, Guides and references for participants.

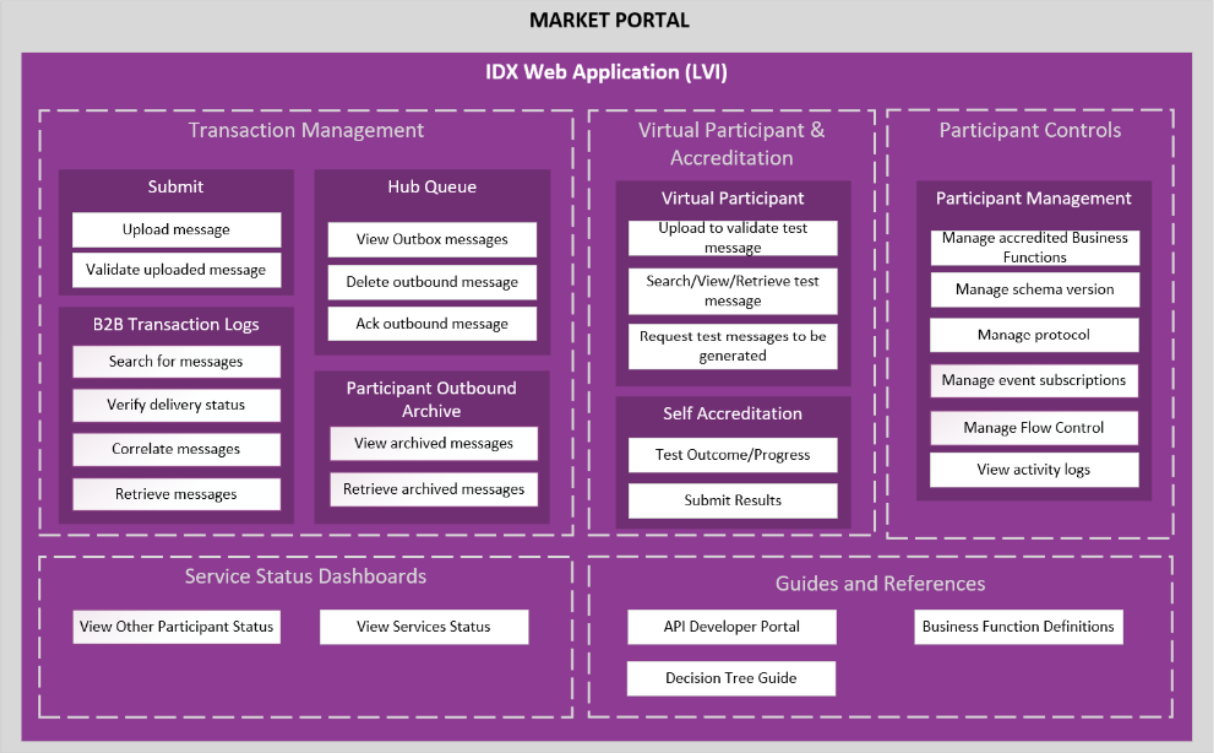


Diagram 6.1 IDX Web Application Functions

This section will be elaborated on in later versions of the document.

6.2. Web Interfaces

This section is a placeholder which will be addressed in later versions of the document.

7. Message Format Requirements

7.1. Overview

This section covers the common Industry Data Exchange conventions and message exchange protocols.

7.2. General IDX Conventions

(a) Participants must ensure that all IDX Interactions comply with the requirements for the patterns and protocols as defined in within this of this Procedure.

(b) A Participant must ensure that their Gateway and market facing systems involved in the market message handling service implements the Acknowledgement model as defined within this Procedure.

7.3. Standard IDX Message Exchange API Calls

Participants can use the following API calls to send, receive or remove messages on the Information Data Exchange platform. The API specification for each will be published on the Business Function/Business Resource's Technical Specification.

Call Type	HTTP METHOD	URL	Description
POST Request	POST	/{businessFunctionId/v1/{businessResourceId}	Posts a message via IDX.
GET Message Metadata	GET	/{businessFunctionId/v1/{businessResourceId}	Retrieves a list of pending message metadata for the Participant.
GET Message	GET	/{businessFunctionId/v1/{businessResourceId}/{messageContextId}	Retrieves the message for the provided messageContextId
		/{businessFunctionId/v1/{businessResourceId}/first	Retrieves a pending message based on the business function's sort order (FIFO/LIFO).
		/{businessFunctionId/v1/{businessResourceId}/first?priority=HIGH	Retrieves a pending message based on the business function's sort order (FIFO/LIFO) and priority provided.
DELETE Message	DELETE	/{businessFunctionId/v1/{businessResourceId}/{messageContextId}	Deletes the pending message for the provided messageContextId from the Participants outbox and triggering the archival process.

Table 7.3 Message Exchange API Calls

7.4. Common Header Parameters

Common header parameters are included on all messages irrespective of Business Function. The headers expected by IDX are:

HTTP Header	Optionality		
	POST	GET	DELETE
x-initiatingParticipantId	M	M	M
x-messageContextId	M	O	M
x-signature	M	N	N
Content-Encoding	R	N	N
Content-Type	M	N	N
Authorization	M	M	M

Key

M = Mandatory (must be provided in all situations).

R = Required (must be provided for some business functions).

O = Optional (may be provided and should be used by the participant).

N = Not required

7.4.1. x-initiatingParticipantId

The initiatingParticipantId is used in the following ways:

(a) To identify the participant responsible for initiating market activities

(b) To track and manage participant-specific actions and responsibilities.

(c) Ensures secure and accurate communication between market participants.

(d) The initiatingParticipantId must align with the participantId assigned by AEMO to participants i.e. aligns to participantId in MSATS.

7.4.2. x-messageContextId

The messageContextID is used in the following ways:

(a) To provide a contextID for the message exchange. The participant/IDX Hub uses the messageContextId of the original request when delivering its corresponding ACK(s).

(b) The message or file name is set to <messageContextID>.

(c) The name of the archive files is set to <messageContextId>_<versionX_Y_Z>_<duplicate count>.

(d) The format is:

```
[BusinessFunctionID 0-9_a-z]{1,4} + “_” + [ResourceID 0-9_a-z]{1,4} + “_” + [Priority
h|m|l] + "_" + [InitiatingParticipantID]{1,10} + "_" + [MessageID, 0-9_a-z_-]{1,36}
```

(e) An example:

```
pqd_bpqd_1_retailer1_abcd1234
```

(f) messageContextId is case sensitive and required in lower case.

(g) IDX does not validate the uniqueness of messageContextId. Participants are required to ensure messageContextId is unique when a new request is sent to IDX Hub.

7.4.3. x-signature

The signature is used in the following ways:

(a) A unique and unforgeable cryptographic signature that verifies the integrity of the message. The digital signature also provides proof of the sender of the message is who they claim to be.

(b) The digital signature of the header must match that of the payload.

(c) For more information on how to populate the x-signature, see the Non-Repudiation section.

7.4.4. content-Encoding

(a) Content encoding refers to the method used to compress or format the data being sent to the API.

(b) Where content encoding is required by the business function or resource, this field is treated as mandatory. This will be defined in the Business Function's specification.

(c) The allowed values in content-Encoding are:

(i) gzip

(ii) <blank>

7.4.5. content-Type

(a) Content type specifies the format of the data being sent to the API, allowing the receiving system to correctly parse and process the incoming request.

(b) The allowed groups of content-type are supported:

(i) Text and Data

(ii) File Uploads

(iii) Specialized Formats (for GraphQL requests only)

(c) Content types associated to Media Type are not supported.

(d) The follow values are supported in the content-Type field;

(i) Text and Data Formats:

(A) application/json

(B) application/xml

(C) text/csv

(ii) For File Uploads:

(A) application/pdf

(B) multipart/form-data

(C) image/jpeg

(D) imagine/png

(E) application/zip

(iii) For Specialised Formats

(A) application/graphql

(e) The accepted values will differ based on the agreed standards of the business function/resource.

7.4.6. authorization

(a) Contains the participant's opaque client token details.

8. Payload Format Conventions

8.1. Overview

This section outlines some of the common format conventions employed by the IDX Platform. The intent is to cover the commonalities across a broad range of Business Functions, while specifics including the Business Function payload will be covered in associated technical specifications.

8.2. Schema Validation

Include schema version validation and that schema version is optional

Schema validation confirms a message in its entirety complies with a structural definition and can be read. Schema validation failure results in the entire message being rejected at the message level (via a message acknowledgement, MACK).

The IDX Hub (including the AEMO Gateway Software) supports schema validation on JSON Payloads. There will not be any schema validation for AEMO_CSV or unstructured payload. However, for AEMO_CSV there will be some basic validation pertaining to the basic structure of the file, such as the presence of a C (comment), D (data) and I (information) rows.

Payload Type	Schema Validations
JSON	Yes
AEMO_CSV	Basic validations only on the presence of C, D and I row; A-C row exists at the start and the end of the AEMO_CSV, and the I row exists as the second row.
Unstructured Data	No

Table 8.2 Schema Validation

8.3. Business Validation

Business validation occurs at the transaction level and results in individual acceptance or rejection of each transaction (via a transaction acknowledgement).

The IDX Hub does not perform business validation on behalf of participants, irrespective of expected values outlined in other procedures. The onus is on participants to perform business validations and respond with a transaction acknowledgment indicating the outcome of the business validation.

8.4. Payload Types

(a) Despite the originating submitted message type, MACKs and TACKs will always be in a JSON format.

Payload Type	Channel	Format
Message / Transaction Request and Response	API	JSON AEMO_CSV Unstructured Industry Standard

Payload Type	Channel	Format
	Large File Share	JSON AEMO_CSV Unstructured Industry Standard
HTTP Error Response	API	JSON
	Large File Share	JSON
IDX Hub Acknowledgement (Hub MACK)	API	JSON
	Large File Share	JSON
Recipient Acknowledgement (MACK)	API	JSON
	Large File Share	JSON
Transaction Acknowledgement (TACK)	API	JSON
	Large File Share	JSON
Event Notification	API (WebSocket)	JSON
Event Acknowledgement	API	JSON

Table 8.4 Payload Types

8.5. JSON Conventions

8.5.1. JSON Schema

(a) To ensure consistency, interoperability, and future-proofing our data validation and exchange mechanisms, all JSON schemas in IDX will be authored using the latest JSON Schema specification, as defined by the JSON Schema organisation (link).

(b) The JSON payload will primarily consist of four sections - Data, Links, Error and Meta;

(i) data section (required) - contains the content of the payload

(ii) meta section (optional) - contains the metadata of the payload

(iii) links section (required for response) - contains a field called self that will have the fully qualified URI to the current request as a value. This is applicable to the API response only.

(iv) error section (required for response that is an error) - contains details on the error that occurred.

(c) The Data section will consist of the following sections - Headers / Transactions / Acknowledgements.

(i) Headers section - This will include all message level metadata, such as Message Identifier, Message Date, Business Function, From Participant, To Participant, CC Participant, Version, Priority.

(ii) Transactions section - This will contain transaction level metadata, such as Transaction Identifier, Transaction Type, Initiating Transaction Identifier, transaction time etc. It will also contain the actual data, structured as an array of objects.

(iii) Acknowledgments section – This will contain acknowledgement data such as whether a business request has been successfully received by a recipient or provide the outcome of a business validation and subsequent business accept/reject of a request.

(d) The available combinations of Data sections are

(i) Header and transactions

(ii) Header and acknowledgements

(e) Payloads can be attached as part of data section and/or as a file upload as specified on a Business Function/Resource's technical specification.

8.5.2. JSON Samples

Request Sample

```
{
  "data": {
      "headers": {
        "messageId": "<<unique-message-id>> e.g.OR-MSG-1234",
        "initiatingParticipantId": "<<PID>> AGLE",
        "receivingParticipantId": "<<PID>> ACTGW",
        "messageDate": "<<DateTime Stamp>> e.g. 2002-01-01T12:00:00+10:00",
        "transactionGroup": "<<Transaction Group>> e.g. MGMT",
        "market": "<<Market>> e.g. VICGAS"
      },
      "transactions": [
        {
          "transactionId":"ABCDSOLD663005143f170111111",
          "transactionDate":"2005-10-11T09:30:40+10:00",
          "initiatingTransactionId":"SORD-123456XX",
          "transactionDetails": {
            "transactionType":"ServiceOrderResponse"
          }
        },
        {
          "transactionId":"",
          "transactionDate":"",
          "initiatingTransactionId":"",
          "transactionDetails":{}
        }
      ]
   },
  "meta": {
   }
}
```

Response Sample


```
{
  "data": {
    "headers": {
        "messageId": "<<unique-message-id>> e.g.OR-MSG-1234",
        "initiatingParticipantId": "<<PID>> AGLE",
        "receivingParticipantId": "<<PID>> ACTGW",
        "messageDate": "<<DateTime Stamp>> e.g. 2002-01-01T12:00:00+10:00",
        "transactionGroup": "<<Transaction Group>> e.g. MGMT",
        "market": "<<Market>> e.g. VICGAS"
    },
    "transactions": [
        {
            "transactionId": "ABCDSOLD663005143f170111111",
            "transactionDate": "2005-10-11T09:30:40+10:00",
            "initiatingTransactionId": "SORD-123456XX",
            "transactionDetails": {
                "transactionType": "ServiceOrderResponse"
            }
        },
        {
            "transactionId": "",
            "transactionDate": "",
            "initiatingTransactionId": "",
            "transactionDetails": {}
        }
    ]
  },
  "links": {
    "self": "..."
  },
  "meta": {
  }
}
```

8.5.3. JSON Acknowledgements

(a) An initiating Participant must ensure they are able to receive the following acknowledgement messages for every JSON Message sent.

Key

M = Mandatory (must be provided in all situations).

R = Required (must be provided if this information is available or has changed).

O = Optional (may be provided and should be used by the Recipient if provided, as per bilateral agreements).

N = Not required (not required and may be ignored by the Recipient if provided).

Message Pattern	B2B/B2M	HTTP Response	Hub Ack Response	Message Ack Response (MACK)	Transaction Ack Response (TACK)
Synchronous	B2B	TBD	TBD	TBD	TBD
	B2M	TBD	TBD	TBD	TBD
Asynchronous	B2B	TBD	TBD	TBD	TBD
	B2M	TBD	TBD	TBD	TBD
Fire & Forget	B2B	M	O	N	N
	B2M	M	O	N	N
Enquiry-Flat	B2B	TBD	TBD	TBD	TBD
	B2M	TBD	TBD	TBD	TBD
Enquiry-Dynamic	B2B	TBD	TBD	TBD	TBD
	B2M	TBD	TBD	TBD	TBD

Table 8.5.2.1 Initiator JSON Acknowledgements

(b) A recipient Participant must ensure they are able to send the following acknowledgement messages for every JSON Message received.

Message Pattern	GET Request	POST Message Ack (MACK)	POST Transaction Ack (TACK)	DELETE Request
Synchronous	TBD	TBD	TBD	TBD
	TBD	TBD	TBD	TBD
Asynchronous	TBD	TBD	TBD	TBD
	TBD	TBD	TBD	TBD
Fire & Forget	M	N	N	M
	M	N	N	M
Enquiry-Flat	TBD	TBD	TBD	TBD
	TBD	TBD	TBD	TBD
Enquiry-Dynamic	TBD	TBD	TBD	TBD
	TBD	TBD	TBD	TBD

Table 8.5.2.2 Recipient JSON Acknowledgements

8.5.4. Schema Errors

(a) Schema errors can be detected based on sending IDX Hub:

(i) Non-current or unsupported schema versions

(ii) Incorrect schema payloads

(b) Where detected by IDX hub, they are rejected by IDX hub.

Synchronous Schema Error Response

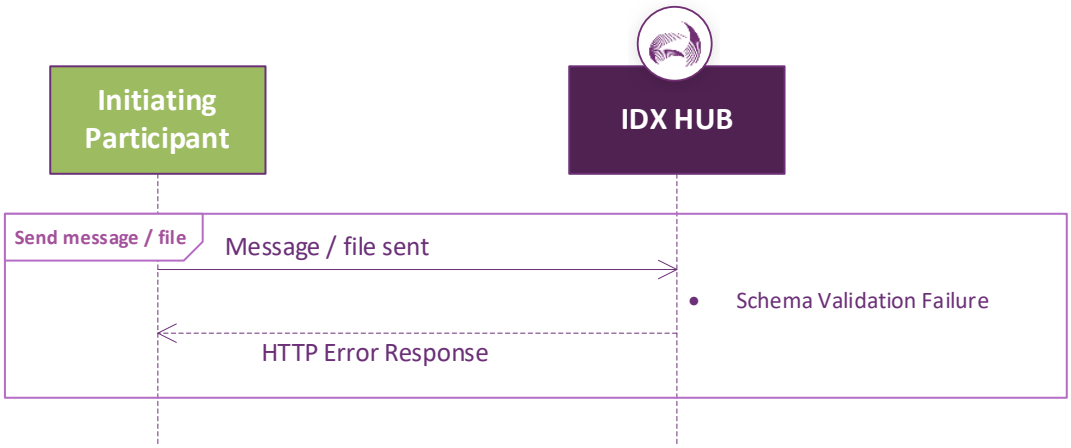


Diagram 8.5.4.1 Synchronous Schema Error Response

Asynchronous Schema Error Response

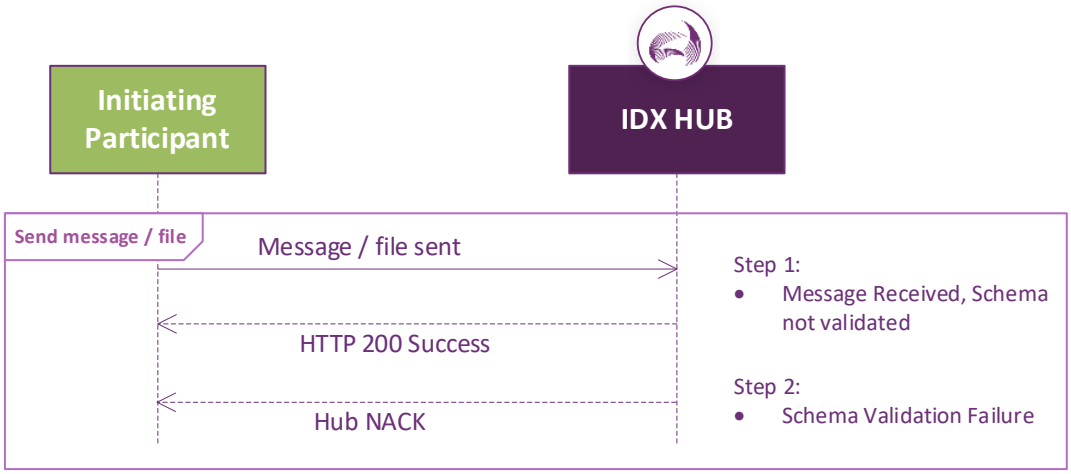


Diagram 8.5.4.2 Asynchronous Schema Error Response

8.6. AEMO_CSV Conventions

8.6.1. Overview

(a) The AEMO_CSV (or AEMOCSV) format will follow the existing standards set-out in our CSV Data Format Standard.

(b) A specific type of comma separated file format, currently used exclusively by AEMO for sending wholesale reports to the participants, will be extended for use in retail and gas markets wherever feasible.

(c) The structure of an AEMO_CSV is as follows;

(i) The first row (starts with **C**) of the file contains the information about message (e.g. Market, transaction group, file generation date etc). This row can be present only once in the file.

(ii) Next row (starts with **I**) contains the transaction level information (e.g. transaction type, transaction id etc) and column header of data. This row can be present multiple times if there are multiple transactions in the same file.

(iii) Next rows (start with **D**) contain the actual data.

(iv) Last row (starts with **C**) contains an End-Of-Report indicator and the total number of lines in file. This row can be present only once in the file.

(v) C row is static.

8.6.2. AEMO_CSV Sample

```
C,NEMMCO-MSG-69361375,2024-12-
16T17:48:08.000+10:00,NEMMCO,NEMMCO,PQD,Low,NEM,NEMMCOBATCH,,
I,transactionID,transType,transVer,NMI,IntervalDate,MeterSerial,IntervalD
ateTime,Volts,Amps,PA
D,NEMMCO-TNS-
0000000000001,PQD,r01,9R206P8A1Q,12/16/2024,MS174897959,12/16/2024
0:00,237,72.53,-68.852
D,NEMMCO-TNS-
0000000000001,PQD,r01,9R206P8A1Q,12/16/2024,MS174897959,12/16/2024
0:05,236,74.84,62.296
D,NEMMCO-TNS-
0000000000001,PQD,r01,9R206P8A1Q,12/16/2024,MS174897959,12/16/2024
0:10,236,73.74,38.031
D,NEMMCO-TNS-
0000000000001,PQD,r01,9R206P8A1Q,12/16/2024,MS174897959,12/16/2024
0:15,232,72.33,35.224
D,NEMMCO-TNS-
0000000000001,PQD,r01,9R206P8A1Q,12/16/2024,MS174897959,12/16/2024
0:20,236,76.47,-28.42
D,NEMMCO-TNS-
0000000000001,PQD,r01,9R206P8A1Q,12/16/2024,MS174897959,12/16/2024
0:25,237,72.12,-5.225
C,End of Report,{10020},,,,,,,,
```

8.6.3. Acknowledgements

This section is a placeholder which will be addressed in later versions of the document.

8.7. Unstructured/PDF Conventions

This section is a placeholder which will be addressed in later versions of the document.

8.8. Message/File Naming

(a) The message/filename used by participants will be the **messageContextId.**

8.9. Message Sort Order

(a) Messages storied in a Participants IDX Hub Queue are made available to Participants to retrieve in either:

(i) First in first out (provides oldest message first by default) or;

(ii) Last in first out (provides the most recent message first by default)

(b) The message sort order is used when providing a list of available messages, or when using the messages/first call.

(c) The sort order is Business Function dependant and will be specified in the Business Function's technical specification.

8.10. Message Size Compression and Bundling

8.10.1. Overview

(a) The IDX Platform shall neither compress payloads for outbound data delivery nor validate compressed payloads for inbound data delivery over the API channel. However, Transport level compression will be implemented on the API channel to reduce the size of data transmitted over the network. This will improve performance by reducing bandwidth usage and transmission time. In the context of an API, compression parameters are specified in the header.

(b) When participants POST data to AEMO, the API gateway will support inbound delivery if Content-Encoding: gzip is specified in header.

(c) When participants get data from AEMO, the API gateway will always specify an outbound compression type using header type: Accept-Encoding: gzip.

(d) The IDX Platform shall reject requests with all other compression types like deflate, both for Inbound and Outbound.

(e) When AEMO will receive an inbound request, AEMO will verify digital signature and message payload.

(f) When AEMO processes an outbound request, AEMO will generate a digital signature along with the outbound payload and make it available for participants.

(g) All large files (Greater than 10mb) will be compressed in zip as default for both inbound and outbound. In case of inbound delivery, sender participant will generate digital signature file and message file, then compress them into a single zip file before sending it to AEMO. Similarly, for outbound delivery, AEMO will generate digital signature file and message file, compress them into a single zip file, and send it to the recipient.

8.10.2. Message Size

Message Type	Applicable Channel	Message Size Threshold Value
ZIP file	Large File Share	>10MB
JSON	RESTful API	<10MB
JSON file	Large File Share	>10MB
Signature file	Large File Share	-
AEMO_CSV file	RESTful API	<10MB
AEMO_CSV file	Large File Share	>10MB
Unstructured file	RESTful API	<10MB
Unstructured file	Large File Share	>10MB

Table 8.10.2 Message Size

This section will be elaborated on in later versions of the document.

Message sizes are point in time and will be refined and tuned based on industry performance testing.

8.10.3. Compression Methods

Channel	Compression
RESTful API	gzip (for transport only)
Large file share	.zip file
Inquiry Service	gzip (for transport only)
WebSockets	No Compression
User Interface	Not Applicable

Table 8.10.3 Compression Methods

8.10.4. Message bundling conventions

This section is a placeholder which will be addressed in later versions of the document.

8.11. Anti-Virus/Malware Validation

(a) IDX HUB will always scan messages/files for malware and viruses.

(b) Where threats are detected within the message payload, the message will be rejected by AEMO (See Appendix D for Error Messages).

8.12. Non-Repudiation

Overview

(a) Non-repudiation is where the sender of a message cannot deny the sending of a particular message by virtue of the message integrity (is intact) and the message being authenticated by the sender.

(b) For a message to fit the definition of non-repudiation it must fulfill the following core principles:

(i) **Integrity**: The message has not been altered or tampered with in any way.

(ii) **Digital Signature**: A unique and unforgeable cryptographic signature that verifies the integrity of the message. The digital signature also provides proof of the sender of the message is who they claim to be.

(iii) **Audit Trail**: A record of the transactions and communication path of the message so it's delivery can be traced end-to-end.

(iv) **Timestamping**: A record of the exact time the message was sent or received.

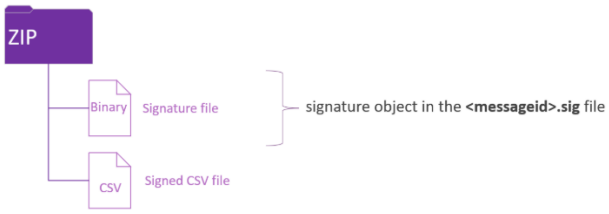
(c) Participants will sign all payloads sent into the IDX Hub, apart from non-market payloads (e.g. WebSocket Heartbeats or Event acknowledgements).

(d) AEMO will sign all payloads outbound from the IDX Hub, apart from Message Acknowledgements, and other non-market payloads (e.g. Event Notifications).

Signing Type	Detached signature
Signing keys	Private Key - participant created.

	Public Key - CSR provided by Participant and keyed from certificate chain of AEMO-RCA G2 for participant-signing certificates. To facilitate non-repudiation; Re-use the existing AEMO-RCA-G2 root (and AEMO-ICA-TEST G1 for pre-production intermediary and AEMO-ICA-MARKET G1 for production intermediary), which will continue to be used to create participant signing certificates from.
Algorithm	RSA256

API Payloads	X-Signature HTTP Header in API Requests and API Responses
	Algorithm and signature to be included in the Header, separated by a semi-colon (;).
	example:
	POST /api/v1/transactions HTTP/1.1
	Host: example.com
	Content-Type: application/json
	Authorization: Bearer <access_token>
	X-Signature: RSA-SHA256;BtSqbj4KuQX7Y/lz64OuSbMy8=
	{
	"Transactions": {
	"Transaction": {
	"CATSChangeRequest": {
	"ChangeReasonCode": "5054",
	"ProposedDate": "2009-03-09",
	}
	}
	}
	}

Large File Payloads	Large file transfer will always be transferred as a ZIP file. AEMO proposes the signature to be added as a <messageId>.sig file to the ZIP, accompanying the signed payload file. Participants must be told via Tech Spec what algorithm (RSA-256) we will use, as this won't be shown in the payload. Example:  Large File ACKs must have a renamed file extension as .ACK, not .ZIP

API Payloads

The following diagram provides B2B context of a participant sending an API payload to another participant via the IDX Hub and how the signing of the payload is treated throughout the journey.

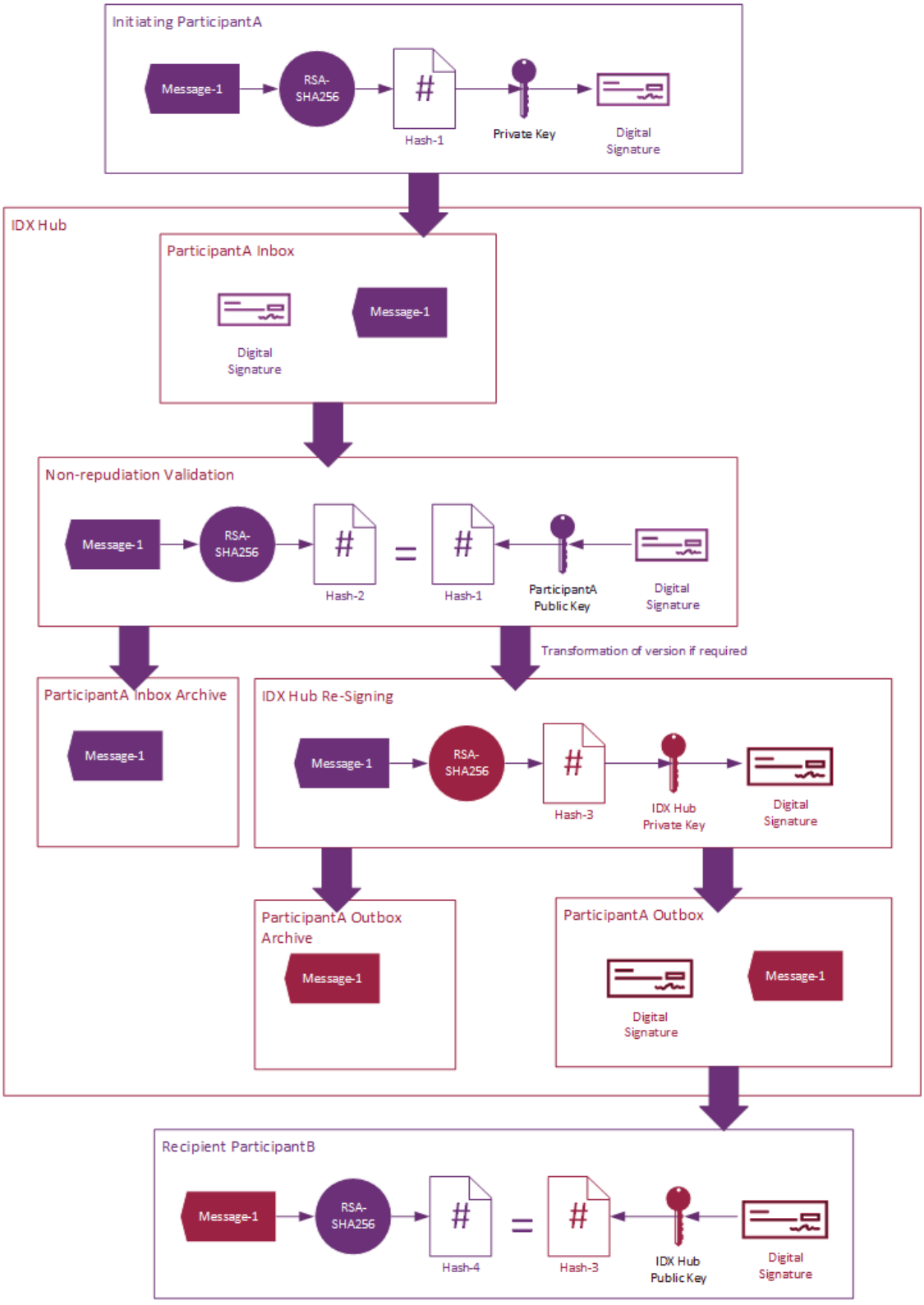


Diagram 8.12 Non-Repudiation Overview

Signing Steps for RESTful API Channel

1. The initiating participant (ParticipantA) will follow these steps:
    a. Apply the RSA algorithm to the message to generate hash value.
    b. Encrypt the hash value using the private key (that matches the AEMO-RCA-G2 certificate) to generate digital signature.
    c. Put the digital signature and the RSA algorithm to the API header of the request (x-signature).
    d. Put the message at the API body of the request.
    e. Send the request to the IDX Hub business function API endpoint (POST).
2. AEMO will follow these steps:
    a. Apply the RSA algorithm (present in API header) to the message to generate hash value (hash-2).
    b. Decrypt the digital signature using the ParticipantA's public key (PRIMARY or SECONDARY) which generates hash-1.
    c. Compare hash-2 with hash-1. If the values are matched, then the verification is successful.
    d. Put message-1 in ParticipantA's Inbox archive.
    e. (OPTIONAL) Transform the message (if required).
    f. Put message-1 in the outbound queue Participant.
    g. (OPTIONAL) ParticipantB receives the outbound event notification for message-1.
    h. ParticipantB pulls message-1 from the outbound queue.
    i. Encrypt the hash-2 value using the IDX Hub private key to generate digital signature (DS-2).
    j. IDX Hub puts DS-2 and the RSA algorithm into the API header x-signature and the content of message-1 in the body of the API response.
3. When the recipient participant (ParticipantB) will receive the API response, they will follow these steps.
    a. Apply the RSA algorithm (present in API header) to the message to generate hash value (hash-4).
    b. Decrypt the digital signature using the AEMO's public key which generates hash-3.
    c. Compare hash-4 with hash-3. If the values are matched, then the verification is successful.
    d. If the hash value mismatches, then
        1. For a-synchronous patterns; the participant must return a MACK indicating the failure of non-repudiation validation.
        2. For Synchronous and Fire and forget patterns, the participant must raise a help ticket.

Archiving of digital signature

The archiving of the digital signature is not required as there is no guarantee that the signature can be checked. As an archived payload ages, there is an increasing likelihood that the participant's original public signing certificate will expire, and therefore the signature could not be verified with expired, or revoked certificates.

For API payloads, just the payload is archived without the digital signature.

9. Field Format Conventions

9.1. Use of standardised format conventions for fields

This section is a placeholder which will be addressed in later versions of the document.

9.2. Basic field formats

This section is a placeholder which will be addressed in later versions of the document.

9.3. User-defined field formats

This section is a placeholder which will be addressed in later versions of the document.

9.4. Fields that contain codes or enumerated lists

This section is a placeholder which will be addressed in later versions of the document.

9.5. Transaction Delivery Requirements

9.5.1. Transaction Timing within the technical exchange

This section is a placeholder which will be addressed in later versions of the document.

10. Message/File Archiving

10.1. Inbound Archiving

(a) A subset of business functions on IDX will have message archiving enabled.

(b) Only messages which have passed IDX Hub Validations are stored after processing.

(c) Acknowledgements are not archived by IDX.

(d) Wherever possible, AEMO will compress the message/file during the archival process.

(e) Inbound messages to IDX are also stored after being consumed by the recipient. Inbound messages are made available to Participants via AEMO ticket only.

10.2. Outbound Archiving

(a) Archiving as defined by AEMO will move the message or file being archived from a participant's IDX Hub Queue to the Participant's archive.

(b) Wherever possible, AEMO will compress the message/file during the archival process.

(c) Archived messages are made available to Participants online through the IDX Web application or via the designated Archive Retrieval API calls, or through the Archive Replay mechanism. Further details of this process are included within the Industry Data Exchange Platform Standard.

(d) The duration that messages or files are stored in the archive folders will vary according to the business function archival requirements and will be specified in a business function's technical specification.

This section will be elaborated on in later versions of the document.

11. Contingency

This section is a placeholder which will be addressed in later versions of the document.

12. Developer Portal

For API documentation and Open API Specifications on all AEMO's APIs please see our developer portal.

This section will be elaborated on in later versions of the document.

13. Client Application Software

(a) AEMO Gateway software is an optional client-side application which connects Participants' systems and AEMO Data Exchange Environment on the agreed protocols, channels, and patterns that are specific to each of the markets.

(b) AEMO Gateway Software connects to AEMO Data Exchange Environment on the agreed AuthN & AuthZ patterns and security mechanisms (e.g. OAuth, https, mTLS connections etc).

(c) AEMO Gateway Software implements validations (such as schema validations).

(d) Gateway manages the generation and delivery of message-level acknowledgements.

(e) All Participants have the option on whether they implement their own gateway or AEMO-supplied Gateway software.

(f) The AEMO Gateway Software can be extended by Participants, for example, to add personalised validations and Transformations.

(g) AEMO Gateway Software is stateless software to manage the message data exchange. It does not leverage intrinsic business intelligence/logic or standing data reconciliation processes.

(h) AEMO Gateway Software has the capability to integrate with Participants backend systems using a variety of protocols and channels.

14. Security and Access

14.1. API Channel

14.1.1. Authorisation

AEMO has selected OAuth 2.x as an authorisation framework to authorise communications with IDX Hub. OAuth 2.x, is a widely adopted authorisation framework that allows applications to obtain limited access to resources available over an HTTP service.

14.1.2. Credential Type

The client credentials grant type (Client ID and Secret) is what AEMO will implement for communications with IDX Hub. This approach strikes an optimal balance between security, cost, and simplicity, ensuring secure authentication for applications while maintaining manageable implementation complexity.

For participants to access IDX services and business functions they will need;

(a) a service account with valid client credentials (a client ID and client secret),

(b) a valid set of entitlements and permissions for that IDX service or business function added to the service account,

(c) a valid AEMO-provided TLS certificates to establish MTLS authentication (where required).

14.1.3. Client Authorisation

Using the client credentials, a client application (e.g., a Participant System) will need to retrieve an OAuth2 token from the Authorisation server (the token endpoint on AEMO's Identity and Access Management (IDAM) system). A list of required scopes may also need to be provided as part of the token request that will bind the token to a set of entitlements for accessing IDX services and business functions. Once a valid token has been received by the client application, this token is then used in the Authorization header as a Bearer token to access the IDX service or business function.

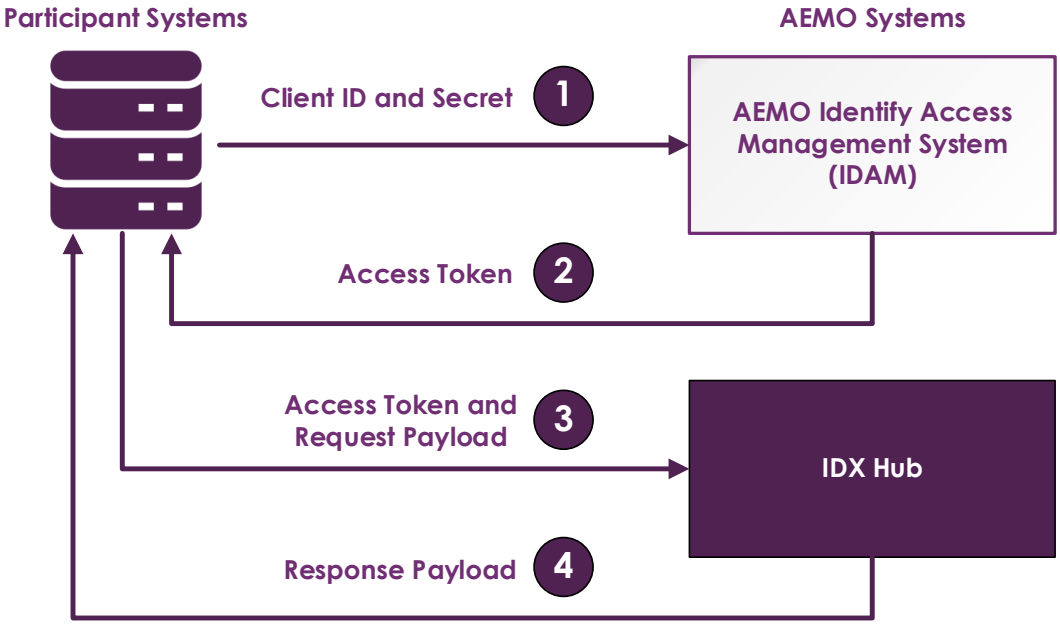


Diagram 14.2 Authorisation flow for OAuth2 client-credentials grant-type

14.2. Large File Share Channel

This section will be elaborated on in later versions of the document.

15. Appendices

15.1. Appendix A: Glossary

Term	Definition
Accept	As a general term, this means the Recipient of the Message or Transaction has agreed to process the Message or Transaction further. When used in the context of a Transaction, indicates that the Recipient of the Transaction has accepted the Transaction using a BusinessAcceptance/Rejection with a Status of “Accept”.
ACK	A shortening of the word Acknowledgement, usually referring to a Message Acknowledgement, Hub Acknowledgement or Transaction Acknowledgement.
AEMO_CSV	A payload format used in data exchange processes, typically for analysis or reporting purposes.
Asynchronous	Referring to a message pattern over IDX whereby the nature of the message exchange and tasks does not occur at the same time.
B2B	Business-to-Business. Generic term used to refer to defined business-to-business interactions between Participants; excludes interactions between a Participant and market systems such as MSATS.
Business Acceptance	Specific instance of a Business Acceptance/Rejection Business Signal indicating acceptance.
Business Acceptance/Rejection	A Business Signal indicating whether a Business Document has been accepted or rejected based on the application of business rules. Refer to each B2B Procedure for further details regarding the use of this Transaction.
Business Service	An activity performed by a company with a discrete repeatable process, typically offering value to another company or business department.
Channel	A channel is a medium through which messages or data are transmitted between participants or systems.
IDX Hub	A centralised information data exchange (IDX) used for message and files exchanges across NEM, WEM and gas markets.
IDX Hub Queue	A store for pending messages to be retrieved by a participant the messages are addressed to.
Inquiry	The inquiry pattern involves tasks that request information or data
Fire & Forget	This message pattern involves initiating a task without waiting for its completion or response.
FTP	File Transfer Protocol.
Hub Acknowledgement	A Message Acknowledgement generated by the IDX Hub.
HTTP Response	A response to a message/file sent over RESTful API indicating the success or non-success of the message/file exchange.
Initiator participant	A participant who sends the first Message in a series of related Messages, usually with another recipient participant who receives the messages. .
JSON	JSON (JavaScript Object Notation) is a lightweight data-interchange format.
Message Acknowledgement (MACK)	The Business Receipt of a received message.
Reject	When used in the context of a Transaction, indicates that the Recipient of the Transaction has rejected the Transaction using a BusinessAcceptance/Rejection with a Status of “Reject”.
Synchronous	The synchronous pattern involves a single legged data exchange, with responses provided on a blocking thread.
Time to live (TTL)	Time to Live (TTL) is a mechanism that limits the duration of data stored within a participant inbox/outbox, ensuring that outdated messages/files are discarded after a specified period.
Transaction Acknowledgement (TACK)	The indication of Business Acceptance/Rejection of a transaction, responded back to an initiator.

This section will be elaborated on in later versions of the document.

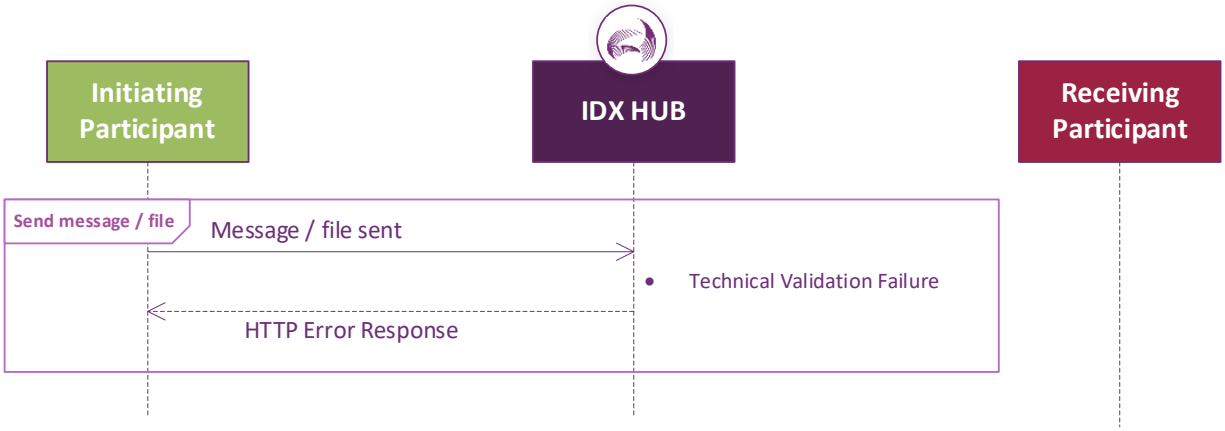
15.2. Appendix B: Examples – Decision Tree Worked Examples

This section is a placeholder which will be addressed in later versions of the document.

15.3. Appendix C: Examples – Errors worked examples Scenarios

15.3.1. B2B/B2M Asynchronous/Synchronous/F&F IDX Technical Validation Error

Used in scenarios where the message fails to pass technical validations. For a subset of validations for some business functions, the technical validations can be performed upfront and rejected as part of the POST call.



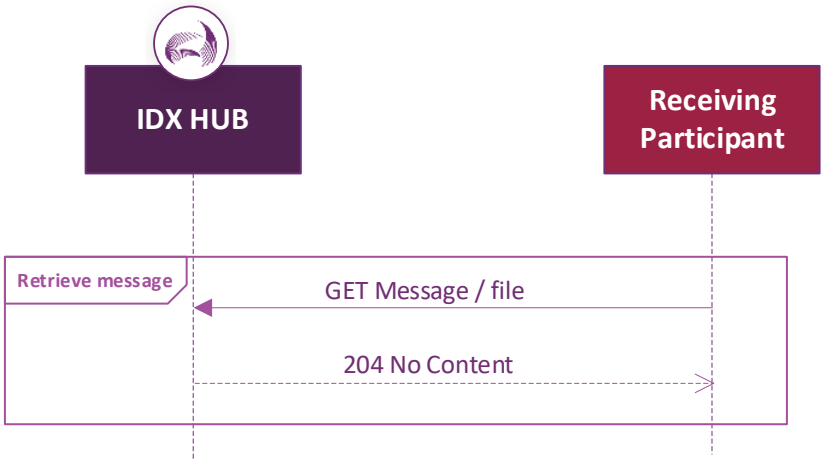
15.3.2. Participant GET Request but no content available

Scenarios:

(a) Recipient attempts to receive a list of messages from their IDX Hub Queue, but there are no pending messages.

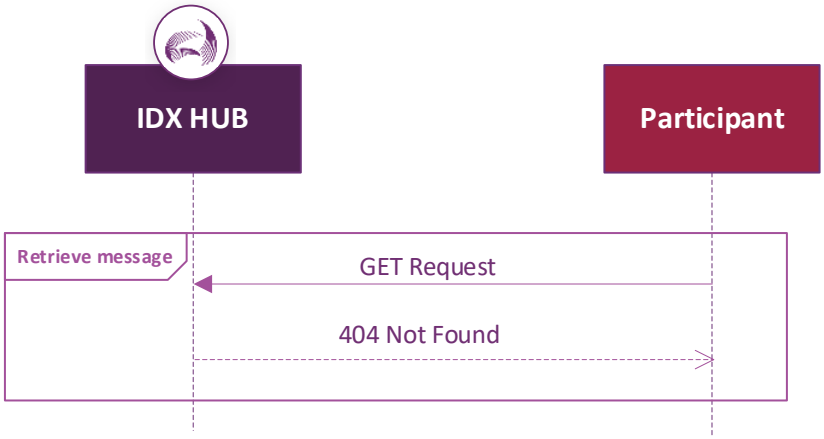
(b) Recipient attempts to retrieve a message from their IDX Hub Queue using the messages/first call or no messageContextID provided, and no messages are pending in the IDX Hub Queue for the Participant.

(c) GET Flow Control Breach Events, where no breaches are currently active.



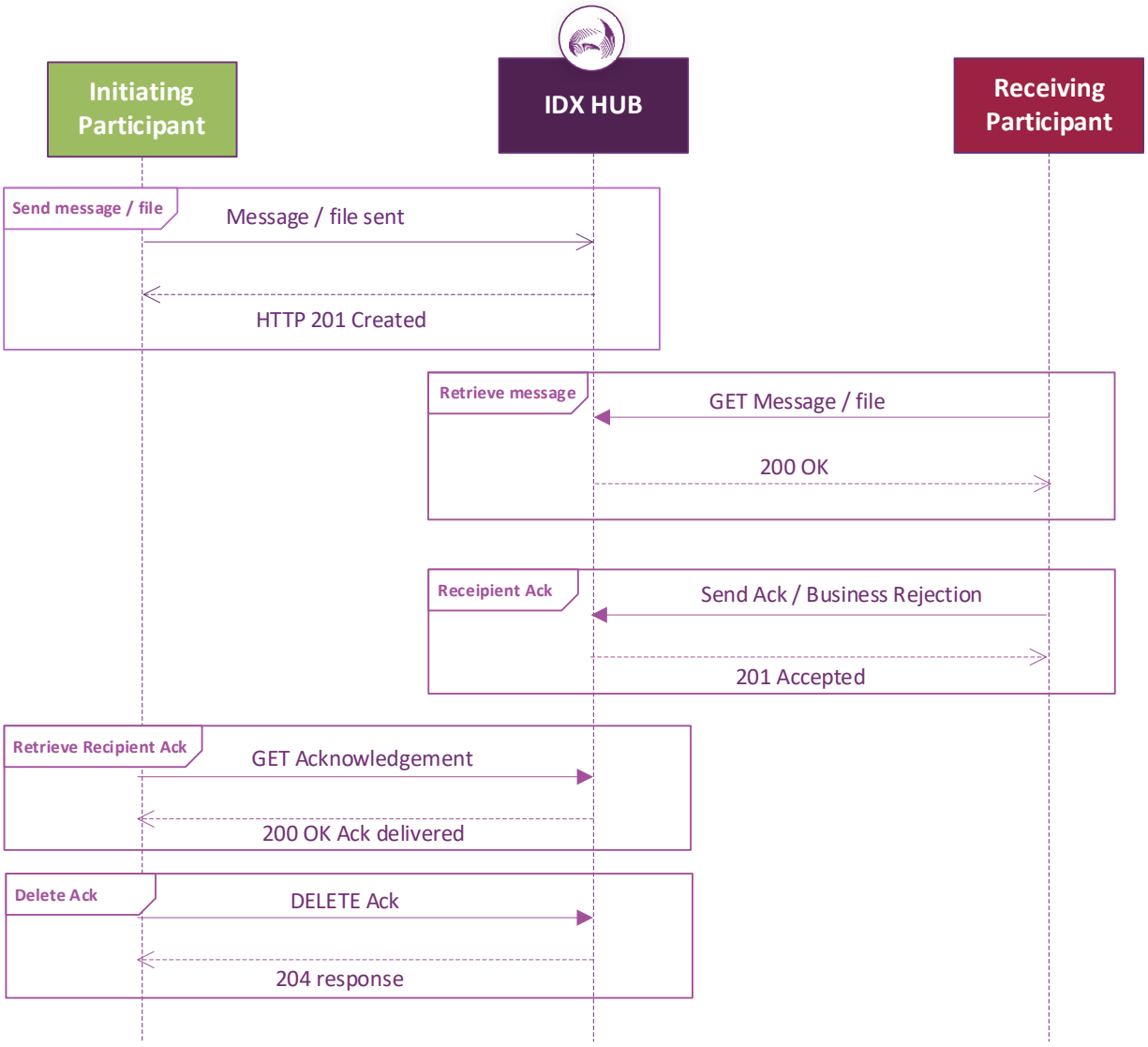
15.3.3. Receiving Participant GET message with bad messageContextID

Used in instances where a messageContextID is used for a specific function such as GET Message, but the messageContextID provided could not be found.



15.3.4. Async B2B Recipient Rejection

Used in scenarios where an initiator is sending a request to a recipient participant but has grounds to reject the message under the existing B2B rules for the Business Function.



15.4. Appendix D: Error Table

Rule Number/ Error Code	Validation	Description	Validation Type	Title	Status	Errors Detail
IDX.001	Spike Arrest	Limit the rate at which requests are allowed through the API Gateway.	Traffic Control	Spike Arrest Violation	429	There are too many requests to this resource in the given time window. Please wait before making further requests.
IDX.002a	Authorisation	Authorisation: clientJWT is not found	Authorisation	Unauthorised Access	401	Authentication credentials were missing or invalid
IDX.002b	Authorisation	Authorisation: clientJWT is not signed	Authorisation	Unauthorised Access	401	Authentication credentials were missing or invalid.
IDX.002c	Authorisation	Authorisation: clientJWT has expired	Authorisation	Expired Access Token	401	The access token provided has expired.
IDX.002d	Authorisation	Authorisation: Insufficient Privileges	Authorisation	Forbidden	403	Authentication credentials provided do not hold the required privileges to access this resource.
IDX.005a	Anti-Virus/Malware	Inspection and blocking of payloads containing malicious content	Security	Unprocessable Content	422	The request payload was rejected because it contains content that failed antivirus or malware scanning.
IDX.008	API Resource	Resource name is invalid/undefined	Traffic Control	Invalid Resource	400	The resource path used in the request is not supported by AEMO. Please use a valid path.
IDX.009	API Method	Method is invalid/undefined	Traffic Control	Method Not Allowed	405	The HTTP method <method used by participant> is not supported for this resource.
IDX.010	Rate Limit	Throttling at the API Gateway	Traffic Control	Too Many Requests	429	You have exceeded your allowed quota of requests to this resource. Please wait before making further requests.
IDX.015	Payload Type	Used when the payload is not one of the approved payload types. application/json application/xml application/pdf text/csv	Payload	Unsupported Media Type	415	The payload type '<payload type>' is not supported.
IDX.016	Payload Size Limit Validation	Ensures incoming requests do not exceed a predefined maximum size	Payload	Payload Size Limit Exceeded	413	The request payload exceeds the maximum size limit for the business resource.

IDX.017	Non-Repudiation- Verify Digital Signature of Payload	Ensures the payload digital signature is verified	Payload	Digital Signature Validation Failed	403	The digital signature provided in the request could not be validated or did not match the expected hash.
IDX.018	Corrupt Zip	Corrupt Zip file, Unable to unzip	Payload	Bad Request	400	The request payload was rejected because the payload was unable to be decompressed or was unreadable after decompressing.
IDX.021	Missing Header Field	Validates that the x-market HTTP header exists	Request	Missing Mandatory Header	400	The request is missing one or more required HTTP headers: 'Missing Header 1', 'Missing Header 2', 'Missing Header 3'.
IDX.022	Header contains unacceptable values or exceeds character limit	Header Validation Failed	Request	Invalid HTTP Header Format	400	The '<headername>' HTTP header value is incorrectly formatted or exceeds the allowed character limit.
IDX.023	Technical Error	Used in the event of AEMO side technical issues resulting in the message or file not being able to be processed.	Request	Internal Server Error	500	An unexpected error occurred while processing your request. Please try again later.
IDX.031	Recipient Queue Full	Checks if target participant is receiving requests. Recipient has a stop file i.e. this validation will fail only when the # of unacknowledged files/messages for a Participant (Recipient) > high water mark limits.	Traffic Control	Service Unavailable	503	The receiving participant is currently unable to handle the request due to a full queue or scheduled maintenance.
IDX.032a	Schema Version	The version is validated and ensures it is one of the current versions.	Request	Bad Request	400	The provided schema version is not a supported schema version for that business function.
IDX.032b	Schema Validation	Payload must be well formed and valid according to relevant version of the schema	Payload	Unprocessable Entity	422	The provided payload was not in a valid schema format for that business function.
IDX.033	Message Context in Payload	Validates the payload fields match the values in x-messageContextId	Request	Invalid HTTP Header Value	400	The message context ID in the HTTP header does not match the values in the payload. Please ensure both values are aligned and retry the request.
IDX.034	To Participant	Checks if To Participant is a valid participant.	Payload	Bad Request	400	The receiving ParticipantId [Participant ID] does not exist or is not registered.

IDX.035	Technical Error	Used in the event of AEMO side technical issues resulting in the message or file not being able to be processed.	Request	Internal Server Error	500	An unexpected error occurred while processing your request. Please try again later.
IDX.045	GET Message Request has a messageContextID which could not be found	Used in the event outbound framework cannot find a message for the provided messageContextID	Request	Not Found	404	The MessageContextID [MessageContextID] could not be found.
IDX.046	GET Message Metadata validations failed	Validations failed, mandatory fields not populated or otherwise not meeting the minimum requirements for the request	Request	Bad Request	400	The request is missing one or more mandatory fields or does not conform to the minimum required request structure.
IDX.047	DELETE message request could not find messageContextID	Used in the event outbound framework cannot find a message for the provided messageContextID	Request	Not Found	404	The MessageContextID [MessageContextID] could not be found.
IDX.048	GET Archived Message	Used in the event outbound framework cannot find a message for the provided messageContextID	Request	Not Found	404	The MessageContextID [MessageContextID] could not be found.
IDX.049	GET Archived Message List	Mandatory fields not populated or otherwise not meeting the minimum requirements for the request	Request	Bad Request	400	The request is missing one or more mandatory fields or does not conform to the minimum required request structure.
IDX.050	Technical Error	Used in the event of AEMO side technical issues resulting in the message or file not being able to be processed.	Request	Internal Server Error	500	An unexpected error occurred while processing your request. Please try again later.
IDX.051	Startdate format check	Invalid date format provided	Request	Bad Request	500	startDateTime format is invalid. Correct format is YYYY-MM-DDTHH:mm:ss.SSSZ
IDX.052	End date format check	Invalid date format provided	Request	Bad Request	400	endDateTime format is invalid. Correct format is YYYY-MM-DDTHH:mm:ss.SSSZ
IDX.053	Start date is greater than end date	Start date is greater than end date	Request	Bad Request	400	startDateTime cannot be greater than endDateTime
IDX.061	Validate if existing flow control restrictions are in place when participant wants to manually toggle stopping or activating the queue.	Used if the participant toggles flow control when restrictions have already been put in place due to high pending message count.	Request	Conflict	400	Flow Control Restrictions are in place and cannot be manually enabled or disabled at this time.