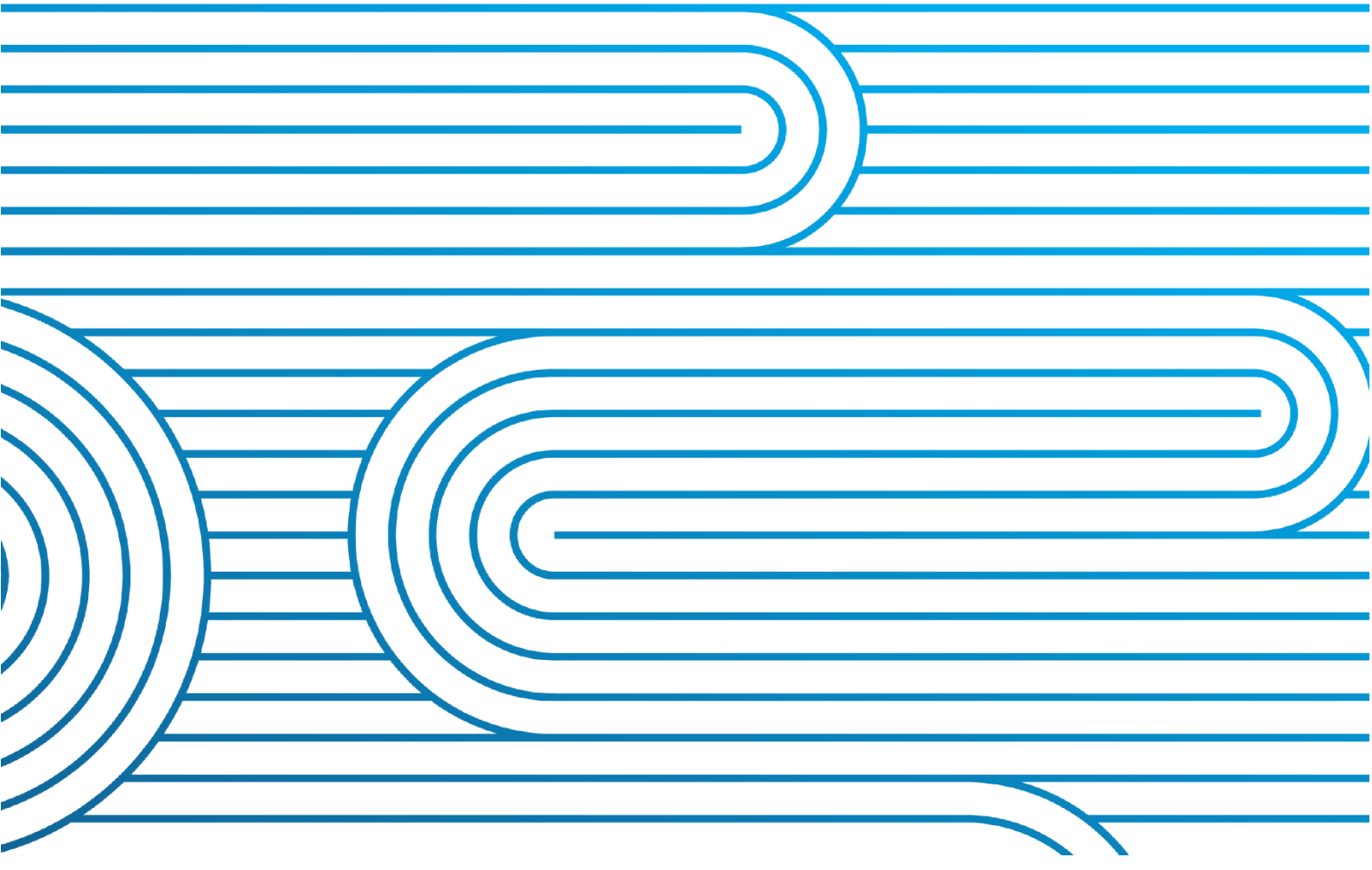
TRANSPOWER

# GL-SD-1045 Market Dispatch Integration: ICCP and Web Services Guideline

This document provides the detailed specifications of the Web Services and ICCP interfaces between the System Operator and Participants for market dispatch.

| | | | | | |
|---|---|---|---|---|---|
| **TP Ref:** | **GL-SD-1045** | **Status:** | **Issued** | **Approval Date** | **4/09/2026** |

**Transpower New Zealand Limited**



## IMPORTANT

LIMITATION OF LIABILITY AND DISCLAIMER OF WARRANTY

Transpower New Zealand Limited makes no representation or warranties with respect to the accuracy or completeness of the information contained in the document. Unless it is not lawfully permitted to do so, Transpower specifically disclaims all liability in contract, tort, equity or otherwise and any implied warranties of merchantability or fitness for any particular purpose and shall in no event be liable for, any loss of profit or any other commercial damage, including, but not limited to, special, incidental, consequential or other damages.

CONFIDENTIALITY

All information disclosed in this document that is not general public knowledge must be treated as strictly confidential and may not be used or disclosed except for the purpose of developing documentation for the benefit of Transpower.

MINIMUM REQUIREMENTS

The requirements set out in Transpower's standards are minimum requirements that must be complied with by Transpower personnel, contractors and consultants as applicable. All personnel are expected to implement any practices which may not be stated but which can reasonably be regarded as good practices relevant to the purpose of this standard. Transpower expects all personnel to improve upon these minimum requirements where possible and to integrate these improvements into their processes and quality assurance plans.



Contact Details

Address: Transpower New Zealand Ltd
22 Boulcott St
PO Box 1021
Wellington
New Zealand

Telephone: +64 4 495 7000

Email: system.operator@transpower.co.nz Website: https://www.transpower.co.nz

Version Control

| # | Date: | Comment |
|---|---|---|
| V4.0 | 4/9/2026 | Amended references to Energy dispatch group to EnergyReserve to match dispatch model label. Updated some details for new DD and DNx dispatch types. |
| V3.0 | 19/04/2023 | Major revision post GENCO decommissioning.<br>Updated Dispatch Instruction details to include new instruction Types and Groups for Dispatchable Demand (DD) and Dispatch Notified products (DNx) |
| V2.2 | 28/08/2020 | Updated ICCP datatype information for GC and IG Dispatch Instructions |
| V2.1 | 19/08/2020 | Updated EXAMPLE (ICCP BLOCK 5) for military time to NZ timezone. |
| V2.0 | 11/08/2020 | Updated Appendix 3 documenting logic specifically pertaining to ICCP and the tags VALUE, VALUE_C, FST, FST_C. |

| # | Date: | Comment |
|---|---|---|
| V1.9 | 15/07/2020 | Updated for datetime handling for ICCP and correlationid header in webservices. |
| V1.8 | 18/05/2020 | Clarified in Section 3.5.1 the specific fields required in the JWT token for Transpower to conduct validation. |
| V1.7 | 09 Apr 2020 | Added Appendix 4 to address Data Quality/Validation issue(s). |
| V1.6 | 25 Feb 2020 | Clarified in Section 3.4 that a Voltage dispatch via web-services can include a Primary Dispatch Type with a null value. |
| V1.5 | 12/12/2019 | The logic in the ICCP Adapter has been updated and released on 12-Dec-2019 - refer Appendix 3 |
| V1.4 | 21 Oct 2019 | Further updates<br>• Feedback tags for Block5 ACK, ACKA and ACKQ<br>• Additional tags for Frequency dispatch<br>Added appendix 3 with Frequency dispatch example(s) |
| V1.3 | 8 Aug 2019 | Further clarifications:<br>• Added detail about dealing with occurrence of (special) Sequence Number 1<br>• Added "Not Before" (nbf) and "Expiry" (exp) claims to JWT validation along with 300 second maximum expiration. Also removed audience claims<br>• Added comment that direct connect URLs will be confirmed during Onboarding<br>• Clarified data type for ICCP date/time<br>• Clarified use of Feedback tags for Block5 to return ACKA and ACKQ<br>Included extra examples of Block2 and Block5 configurations |
| V1.2 | 31 Jul 2019 | Updated following build completion:<br>• Web service dispatch history changed from 30 minutes to 180 days<br>• WS floating point format confirmed as 2 DP accuracy<br>• Added statement that configuration of Web Service SSE heartbeat should not be longer than 30 seconds<br>• Swagger updated to reflect as built (including addition of dispatchTime to primary values)<br>• Added clarification of the difference between Voltage and other dispatches<br>Lots of miscellaneous minor corrections |
| V1.1 | 22 Jan 2019 | Added greater flexibility to ICCP channel to allow Participants to configure a Dispatch Endpoint to utilise ICCP block 5 (controls) as an alternative option to ICCP block 2 (status). |
| V1.0 | 20 Nov 2018 | Finalised following review by Dave Waine & Holly Denton |
| V0.4 | 16 Nov 2018 | Final draft updates prior to industry workshop |
| V0.3 | 14 Nov 2018 | Updated ICCP section based on reviews |
| V0.2 | 5 Nov 2018 | Updated following first review and merged ICCP channel |
| V0.1 | 12 Oct 2018 | Initial draft |

# CONTENTS

# FIGURES



# TABLES

# 1 INTRODUCTION

## 1.1 DOCUMENT PURPOSE AND SCOPE

This document provides the detailed specifications of the (RESTful) Web Services and ICCP interfaces between the System Operator and Participants for market dispatch. Its purpose is to provide sufficient technical information to enable Participants to design and build software that integrates with the System Operator, via web services or ICCP, for the purposes of participating in the market dispatch process.

To integrate with the System Operator, as a Participant, interested parties need to establish a Data Transmission Agreement and go through the onboarding process, appropriate to the given connection type, as described in detail in the document references in Table 1.

In the remainder of this document the term Market Dispatch Service is used to refer to the new subcomponent of the Market System broadly responsible for routing and distribution of dispatch instructions and acknowledgements between the System Operator and Participants.

## 1.2 INTENDED AUDIENCE

This document is intended for the following audience:

- Participant Solution Architects, Business Analysts, Software Developers and Test Analysts;
- External Software Developers contracted by Participants to build market dispatch solutions on their behalf.

## 1.3 REFERENCES

*Table 1 – Document References*

| Document Reference | Version | Date |
|---|---|---|
| Dispatch participant transition processes | V1.0 | 12-Jul-2019 |
| ICCP Data Exchange (Transpower – Client; unique for each client) | N/A | N/A |

*Table 2 – Web Document References*

| Subject | URI |
|---|---|
| OpenAPI Specification Version 2.0 | https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md |
| Date and Time on the Internet | https://xml2rfc.tools.ietf.org/public/rfc/html/rfc3339.html#anchor14 |
| HTML Living Standard (Server-sent events) | https://html.spec.whatwg.org/multipage/server-sent-events.html#server-sent-events |
| Using server-sent events | https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events/Using_server-sent_events#Event_stream_format |
| REST API Tutorial (HTTP Status Codes) | https://restfulapi.net/http-status-codes/ |
| Introduction to JSON Web Tokens | https://jwt.io/introduction/ |

| Subject | URI |
|---|---|
| JSON Web Token (JWT) | https://tools.ietf.org/html/rfc7519 |
| The OAuth 2.0 Authorization Framework | https://tools.ietf.org/html/rfc6749 |
| The OAuth 2.0 Authorization Framework: Bearer Token Usage | https://tools.ietf.org/html/rfc6750 |

## 1.4 DEFINITIONS

In this document, formal definitions are very important and are referred to within the document using title case format to distinguish them from the looser definition that applies when used in normal prose. e.g. use of the term Participant is as defined in the table below whereas a participant is dependent on the sentence context.

*Table 3 – Definitions*

| Term | Definition |
|---|---|
| ACK | Short code/synonym for Dispatch Instruction Receipt. |
| ACKA | Short code/synonym for an acknowledgment of an accepted Dispatch Instruction. i.e. the Participant has "Acknowledged and intends to comply with dispatch". |
| Acknowledgement | A message sent, by a Participant to the System Operator, to acknowledge receipt of (ACK), acceptance of (ACKA), or querying of (ACKQ), a Dispatch Instruction. The acknowledgement process for each Dispatch Instruction typically consists of two parts: the (automatic) Dispatch Instruction Receipt (ACK) followed by a Business Acknowledgement that either the instruction is accepted or queried (ACKA or ACKQ). |
| ACKQ | Short code/synonym for an acknowledgment that queries a Dispatch Instruction. i.e. the Participant has "Acknowledged but is querying the dispatch instruction". |
| ACKR | Short code/synonym for an acknowledgment that rejects a Dispatch Instruction. i.e. the Participant has "Acknowledged but is rejecting the dispatch instruction". Applicable only to Dispatch Notification (DNx) products. |
| Block | A Block is a collection of one or more (child) Nodes that in some sense generate together. Dispatch Instructions for a Dispatch Endpoint also specify the generation expected (dispatch values) for a Block and its child Nodes. The dispatch values for the child Nodes may be mandatory or suggested values, although their collective sum is required to meet the dispatch value specified for the (parent) Block. For the purposes of this document, participants using station dispatch under clause 13.64 of the Code should read 'Block' as equivalent to 'Station' in this context. |
| Block 1, 2, 5,... | Block n (where n as an integer) refers to basic ICCP functionality referred to in the standard as Conformance Blocks. See Section 4.5 for details of the functionality of the various block levels. |
| Business Acknowledgement | Business Acknowledgement refers to a Participant acknowledgement that a Dispatch Instruction has been accepted (for actioning) or queried. The term Business Acknowledgement is also known of as manual acknowledgement though it does not necessarily have to be manual. |

| Term | Definition |
|---|---|
| Channel | In this document Channel refers to one of the protocols used for sending Dispatch Instructions to Participants and receiving acknowledgements; at this time either ISD, ICCP or WS. The term Protocol is occasionally also used synonymously with Channel. |
| Channel Configuration | The Channels a Participant wishes to support for a given Dispatch Endpoint. This is agreed between a Participant and at System Operator as part of broader Dispatch Endpoint Configuration as part of onboarding, subject to the constraints of Dispatch Group Configuration. |
| CLNE | Abbreviation for Client Located Network Equipment. |
| Connected | A Dispatch Endpoint is considered Connected if at least one Channel supported for that endpoint is responding successfully to Heartbeats. |
| CORS | Abbreviation for Cross-Origin Resource Sharing. |
| Cross-Origin Resource Sharing | Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional HTTP headers to tell a browser to let a web application running at one origin (domain) have permission to access selected resources from a server at a different origin. |
| DICOM | Abbreviation of Dispatch Interface Communications. |
| Dispatch Endpoint | Previously known of as Dispatch Site when used in relation to ISD/GENCO, a Dispatch Endpoint is a collection of Blocks and/or Nodes that a Participant nominates (at onboarding/modelling time) for which to receive aggregated Dispatch Instructions for. Every Dispatch Instruction is targeted at a Dispatch Endpoint and provides, for a given Dispatch Group, a complete set of dispatch values, relevant at the stated/current time, for all the Blocks and Nodes within it. Note that a Dispatch Endpoint can be associated with one or more Channels.<br>A Participant is free to utilise one or more Dispatch Endpoints to receive Dispatch Instructions to provide distribution and Channel flexibility subject to the constraints defined in Section 2.3. |
| Dispatch Endpoint Configuration | A Dispatch Endpoint Configuration encapsulates both Channel Configuration and Product Subscription for a given Dispatch Endpoint. i.e. it is both the set of Channels a Participant wishes to support for a given Dispatch Endpoint and the Dispatch Groups the Participant wants dispatched to that endpoint. |
| Dispatch Group | A collection of Dispatch Types that are related to a defined generation capability and are dispatched together. Valid Dispatch Groups currently include EnergyReserve, Frequency, Voltage, Interruptible Load and DNx |
| Dispatch Group Configuration | Dispatch Group Configuration is the System Operator imposed constraint on the set of Channels it is prepared to dispatch a given Dispatch Group over. This may be for technical or business reasons. |
| Dispatch Instruction | An instruction issued by the System Operator to generators and ancillary service agents in accordance with the dispatch schedule. Every Dispatch Instruction is targeted at a Dispatch Endpoint. |
| Dispatch Instruction Receipt | A Dispatch Instruction Receipt is a Participant acknowledgement of receipt of a Dispatch Instruction (ACK); typically, an automatic or system level acknowledgement, it serves to inform the System Operator that a given Dispatch Instruction has been received by a Participant. It does not indicate whether the Participant has accepted the instruction or otherwise. |

| Term | Definition |
|---|---|
| Dispatch Interface Communications | General term to cover all Channels. See also Market Dispatch Service. |
| Dispatch Service | Dispatch Service refers generally to the end to end process and software system that sends dispatch instructions from the Market System to Participants, irrespective of Channel. |
| Dispatch Site | Synonym of Dispatch Endpoint. |
| Dispatch Issue Time | The date/time the dispatch instruction was issued/sent by the Energy Coordinator. See also Dispatch Time. |
| Dispatch Time | The date/time at which a dispatch instruction is valid/applies. For example, a Frequency Keeping instruction may be issued at, say, 14:45:08 (the Dispatch Issue Time) but be for a Dispatch Time of 15:00:00. |
| Dispatch Type | A descriptor for the type of dispatch values within a Dispatch Group. There can be Primary Dispatch Types and Secondary Dispatch Types associated with certain Primary Dispatch Types. Valid Primary Dispatch Type values depend on the Dispatch Group and currently include FREQ, INTF, INTS, MVAR, MW, RESF, RESS, VOLT, DD, MWN, DDN. There are also Secondary Dispatch Types associated with Dispatch Groups, but these are not listed here. |
| Endpoint | Contraction of Dispatch Endpoint. |
| Heartbeat | A term describing the system interaction between a Participant's Dispatch Endpoint, for a given Channel, and the Market System in order that the System Operator considers a given Channel, to a given Dispatch Endpoint, to be connected and therefore eligible for receiving electronic Dispatch Instructions. |
| Heartbeat Frequency | A configuration parameter settable for each Dispatch Endpoint/Channel combination, being the number of seconds between sending subsequent Heartbeat messages to maintain the connection status for the given Dispatch Endpoint/Channel combination. |
| HTTP | Abbreviation of Hypertext Transfer Protocol. |
| Hypertext Transfer Protocol | HTTP (and HTTPS) is an application protocol that runs on top of TCP/IP for transferring files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web. HTTP operations include GET, POST, PUT and DELETE for querying, creating, updating and deleting resources. |
| ICCP | Abbreviation for Inter Control Center Protocol. |
| *i*COM | Transpower abbreviation for ISD/ICCP Communications. More specifically *i*COM servers are used by the Market Dispatch Service as gateways to send Dispatch Instructions via the ICCP Channel. |
| Idempotent | Refers to the property that repeated application of an operation is equivalent to, and has no further effect than, single application. In the context of distributed systems this is typically required to ensure no message loss but necessitates systems to be able to gracefully deal with duplicate message delivery. |
| Inter Control Center Protocol | A standard under the IEC 60870-6 specifications that allows for data exchange over WANs between utility control centres and in the context of this document is one of the Channels available for sending Dispatch Instructions (for certain Dispatch Groups) to Participants. |

| Term | Definition |
|---|---|
| JavaScript Object Notation | JSON is an open-standard file format, originally derived from JavaScript, that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types. |
| JSON | Abbreviation for JavaScript Object Notation. |
| JWT | Abbreviation for JSON Web Token; a JSON-based open standard for creating access tokens. |
| MACE | Acronym for MMS Application Certificate Exchange; a protocol for exchanging X.509 certificates under the MMS standard. |
| Manual Acknowledgement | Refer to Business Acknowledgement. |
| Market Dispatch Service | A term referring to the middleware-based components of the Market System, broadly responsible for routing and distribution of dispatch instructions and acknowledgements between the System Operator and Participants. In other words, the Market Dispatch Service is responsible for Dispatch Interface Communications (DICOM). |
| Market Operator Interface | The Market Operator Interface (MOI) is the primary user interface for running the Electricity Market. Provides viewing and control of market inputs and outputs. |
| Market System | Market System refers to the suite of products used to run the Electricity market system. |
| MMS | Abbreviation for Manufacturing Message Specification; an international standard (ISO 9506) dealing with messaging systems for transferring real time process data and supervisory control information between networked devices or computer applications. |
| MOI | Abbreviation for Market Operator Interface. |
| MS | Abbreviation for Market System. |
| Node | A (dispatch) Node (aka Pnode) is the finest grained entity in the dispatch network topology and represents a physical generation unit (or collection of units where units are not offered individually), or an interruptible load node. Every Node in the dispatch network topology must be unique and either be a direct child of a single Dispatch Endpoint or a single Block. |
| Number of Delivery Attempts | A configuration parameter settable for each Dispatch Endpoint, being the number of times the Market System should attempt to deliver a Dispatch Instruction, within a fixed four-minute period, before giving up. |
| OAuth | Refers to OAuth 2.0; an open specification that defines a delegation protocol that is useful for conveying authorization decisions across a network of web-enabled applications and APIs. |
| OpenAPI | OpenAPI refers to the OpenAPI Specification, originally known as the Swagger Specification, is a specification for machine-readable interface files for describing, producing, consuming, and visualising RESTful web services. |
| OSI | Abbreviation for Open Systems Interconnection; a conceptual model that characterizes and standardizes the communication functions of a telecommunication or computing system without regard to its underlying internal structure and technology. |
| Participant | The Electricity Industry Participation Code 2010, clause 13.72, stipulates that dispatch instructions are to be issued to generators, ancillary service agents (and dispatch capable |

| Term | Definition |
|---|---|
| | load purchasers). In this document the term Participant is used to cover the generators and ancillary service agents, as well as any other future category of dispatch participant. |
| Pnode | Short for Pricing Node. In this document however, it refers to a (dispatch) Node. |
| Primary Dispatch Type | The main value types that can be dispatched as part of a Dispatch Group. For example, the EnergyReserve Dispatch Group consists of the MW, DD, RESF and RESS Primary Dispatch Types and any/all of these can be dispatched for a given Dispatch Instruction (for the Energy Dispatch Group). |
| Product Configuration | Synonym of Dispatch Group Configuration. |
| Product Group | Synonym of Dispatch Group. |
| Product Subscription | A Product Subscription embodies a Participant's "enrolment" to receive Dispatch Instructions, at a given Dispatch Endpoint, for a given Dispatch Group. A Participant can have multiple subscriptions for a given Dispatch Endpoint to receive Dispatch Instructions for multiple Dispatch Groups. Subscriptions are defined and modelled, as part of the Participant onboarding process. |
| Protocol | In this document the term is synonymous with Channel and can be one of ISD, ICCP or WS. |
| Red Shield | A web application shielding-as-a-service company/service offering that is part of the Market Dispatch Service for the Internet facing Web Service Channel to provide additional security shielding and operations. |
| Representational state transfer | Representational state transfer is an architectural style for creating web services. Web services that follow this style are known of as RESTful web services. While the definition of REST is broad it promotes an approach where networked components (identified with URIs) are resources, accessed using stateless API calls built on standard HTTP methods. |
| Resend | In this document is an explicit action, performed by an Energy Coordinator, to resend a Dispatch Instruction if, for example, the participant hasn't responded with a business acknowledgement or some other such situation. Note that a resend is treated as a new dispatch instruction and therefore acquires a new sequence number and the isUserResend flag will be set to true, even though the dispatch content is otherwise the same. A Resend should not be confused with a Retry, which is a lower level redelivery attempt. |
| REST | Acronym for Representational state transfer. |
| RESTful | A Web service architecture based on REST technology, typically utilising JSON over standard HTTP operations such as GET, POST and PUT. |
| Retry | A Retry is a Market Dispatch Service (system) level redelivery attempt of a Dispatch Instruction in the case where it has not received an ACK within a certain period. In this case the Market System may indicate that a Dispatch Instruction is a "retry", but it will otherwise be unchanged; specifically, it will retain the same Sequence Number. A Retry should not be confused with a Resend which is a user driven action to achieve a similar purpose. |
| SAD | Acronym for Stand Alone Dispatch. |
| SCADA | Acronym for Supervisory control and data acquisition; a control system architecture for high-level process supervisory management. |

| Term | Definition |
|---|---|
| Secondary Dispatch Type | Secondary value types can be associated with Primary Dispatch Types and dispatched with them. For example, the MW Primary Dispatch Type can also be accompanied by the following Secondary Dispatch Types for a given Dispatch Instruction (for the Energy Dispatch Group): RUP, RDN, RUPB, RDNB, IG, GC, FRK, SRK, ND corresponding to Ramp Up, Ramp Down, Ramp Up Broken, Ramp Down Broken, Intermittent Generation, Grid Constraint, Fast Risk and Sustained Risk, Non-Dispatchable, respectively. Similarly, the FREQ Primary Dispatch Type (of the Frequency Dispatch Group) can have the FST (Frequency Start Time) Secondary Dispatch Type associated with it, and the DNx group also has an ND Secondary Dispatch Type. |
| Sequence Number | An integer valued identifier, included in every Dispatch Instruction that, for a given Dispatch Group and Dispatch Endpoint combination, is monotonically increasing and incremented by one, for each subsequent dispatch, *as viewed from the perspective of the Market System*. This identifier must be referred to (along with the associated Dispatch Group) in the corresponding Acknowledgement so that the Market System can correlate Acknowledgements to the appropriate Dispatch Instruction, for a given Dispatch Endpoint. |
| Server-Sent Events | Server-Sent Events (SSE) is a technology enabling a server to send automatic updates to a client via an HTTP connection and has been standardised as part of HTML5. It is one of two approaches used to provide dispatch instructions via web services. |
| SSE | Abbreviation for Server-Sent Events. |
| SSL/TLS | Abbreviation for Secure Sockets Layer/Transport Layer Security; cryptographic protocols designed to provide communications security over a computer network. |
| Stand Alone Dispatch | Stand Alone Dispatch (SAD) is the system that provides dispatch continuity in the event of a breakdown of the Market System. |
| Subscription | Contraction of Product Subscription. |
| Swagger | Synonym for OpenAPI. |
| System Operator | Service provider responsible for scheduling and dispatching electricity, in a manner that avoids fluctuations in frequency or disruption of supply. The System Operator is currently Transpower. |
| TASE | Abbreviation for Telecontrol Application Service Element; a suite of protocols referred to as TASE.1, TASE.2, etc. |
| TCP/IP | Abbreviation for Transmission Control Protocol (TCP) and the Internet Protocol (IP); the communications protocols used on the Internet and similar computer networks. |
| Uniform Resource Identifier | A string of characters that unambiguously identifies a resource, and in the context of dispatch generally in the form https://transpower.co.nz/api/v1/dispatch/... |
| URI | Abbreviation for Uniform Resource Identifier. |
| VCC | Abbreviation for Virtual Control Centre; an ICCP abstraction of a real-world control centre. |
| Voltage Support | The ancillary service that injects reactive power into the system to boost voltage at the point of injection. In this document the term Voltage refers to the Dispatch Group consisting of VOLT and MVAR Dispatch Instructions. |
| WAN | Abbreviation for Wide Area Network. |

| Term | Definition |
|---|---|
| Web Services | In the context of this document Web Services is one of the Channels available for sending Dispatch Instructions (for certain Dispatch Groups) to Participants and receiving Acknowledgements. More specifically it refers to RESTful delivery of JSON formatted content over HTTPS, either as an event stream (SSE) or via the standard request/response pattern. |
| WS | Abbreviation for Web Services. |
| X.509 | A standard defining the format of public key certificates. |
| YAML | Acronym for (the self-referential definition) YAML Ain't Markup Language; originally known of as Yet Another Markup Language. A minimal syntax language for describing data. In this document YAML is the format used for the OpenAPI Specification (V2) of the Market Dispatch Web Service API. |

# 2 ARCHITECTURAL OVERVIEW

## 2.1 SOLUTION GOALS

The goals of the Market Dispatch Service are to provide Participant choice in integrating with market dispatch, reduce technical barriers to entry for new Participants and enable more flexible and timely extension of the dispatch system to support future products.

## 2.2 SOLUTION OUTLINE

The enhancements to the Market System embodied by the Market Dispatch Service address the goals above primarily by enabling additional channels for sending dispatch instructions to, and receiving acknowledgements from, Participants. New channels include ICCP and Web Services; Participants wishing to utilise the ICCP channel must utilise Block 2 ("status") conformance. Participants wishing to utilise the Web Services channel have the choice of a dedicated network connection and/or connecting via the Internet.

The solution also introduces the concept of a Dispatch Endpoint, discussed in detail in the following section, that is a logical aggregation of Blocks and Nodes, that a Participant can define in conjunction with the System Operator, to which dispatch instructions will be targeted over one or more channels of the Participant's choosing.

Finally, the concept of Dispatch Group is also made more explicit in the solution, to enable products to be introduced more easily in the future. Participants can "subscribe"/opt in to receive dispatches of certain Dispatch Groups to certain Dispatch Endpoints.

Figure 1 contains a very high-level diagram of the Market System in relation to Participants and the Market Dispatch Service for which the details are elaborated in the remainder of this document.

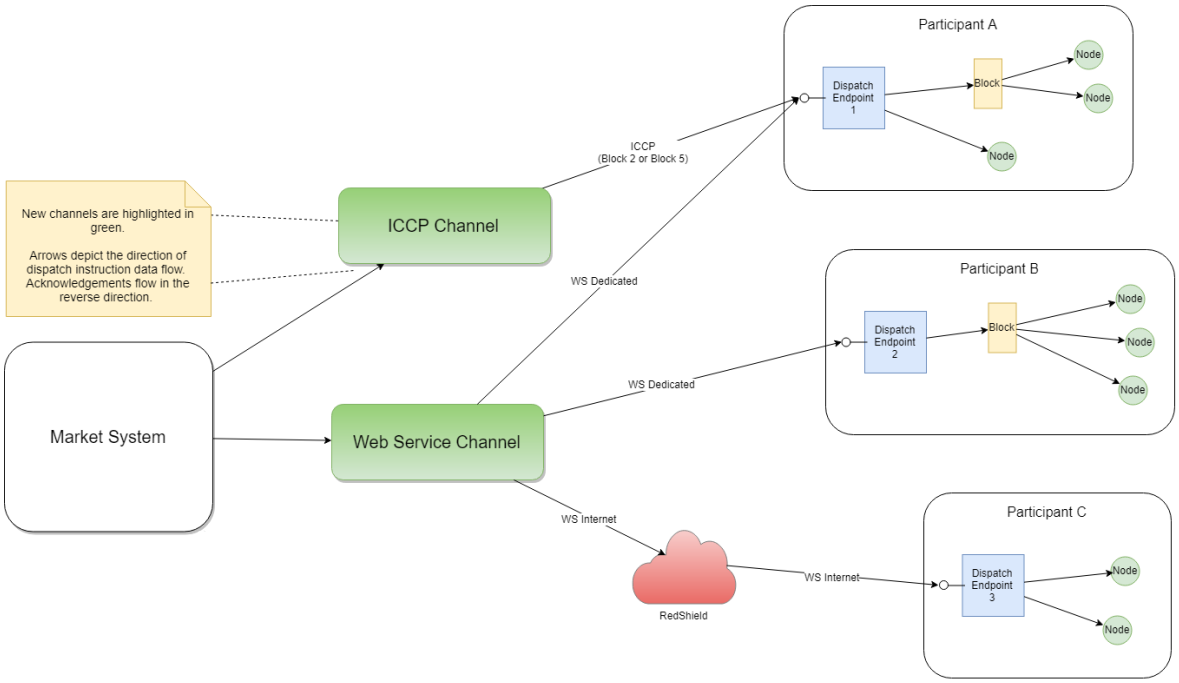



*Figure 1 – Market Dispatch Solution Outline*

## 2.3 MARKET DISPATCH MODEL

This section introduces the overall market dispatch model including the network topology and the "product subscription" model. Figure 2 provides a conceptual view of the model that should be viewed in conjunction with reading the following sections. See also Section 1.4 for formal definitions of various terms and entities.

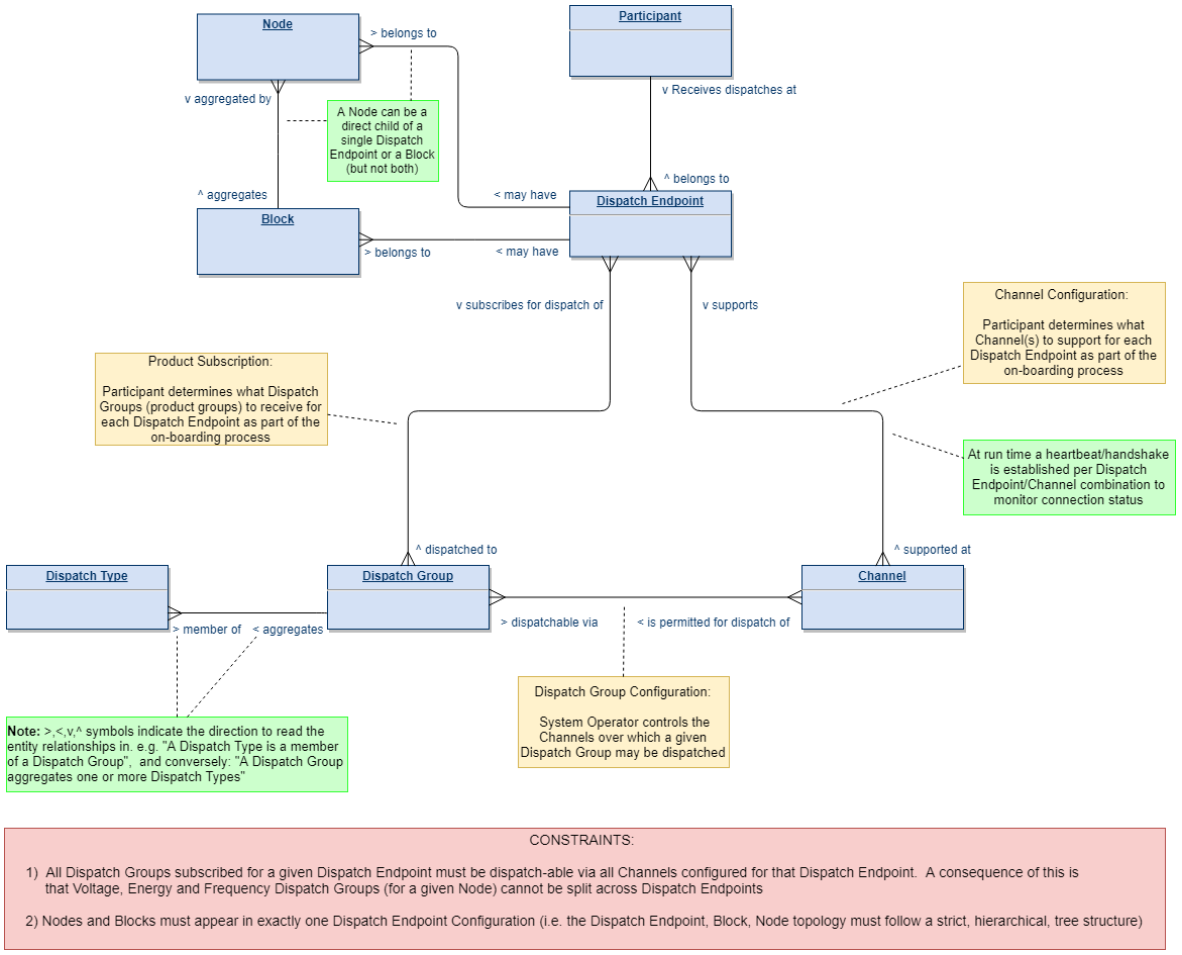



*Figure 2 – Conceptual View of Market Dispatch Model*

### 2.3.1 Dispatch Network Topology – Dispatch Endpoints, Blocks & Nodes

The dispatch network topology consists, fundamentally, of a collection of Dispatch Endpoints. A Participant may utilise one or more Dispatch Endpoints, each of which serves as a logical receipt point for Dispatch Instructions for a hierarchical arrangement (i.e. tree structure) of Blocks and Nodes. A Node may be a child to either a single Dispatch Endpoint or to a single Block which, in turn, may be a child to a single Dispatch Endpoint[1]. Note that this topology can also vary for a given endpoint, depending on the Dispatch Group being dispatched, for example to exclude Nodes/Blocks that do not participate in the dispatch of a given Dispatch Group[2].

When a Dispatch Instruction, for a given Dispatch Group, is required to be sent to a Participant, all information related to the Dispatch Group will be included. i.e. Instead of sending only Nodes/Blocks with changed values, values for all Nodes/Blocks

[1] i.e. a Block cannot be a child to a Node nor can a Node or Block be a child of more than one Dispatch Endpoint (i.e. they cannot be "duplicated").

[2] Voltage is an example where the dispatch is to a Block or a Node and never to a Block and its subordinate Node, so the hierarchy is not represented in Voltage dispatch.

for the Dispatch Endpoint configuration will be included.

### 2.3.2 Dispatch Endpoint Configuration

As well as the definition of Nodes and Blocks that fall within the scope of a Dispatch Endpoint, discussed in the previous section, the configuration of a Dispatch Endpoint consists of defining further configuration parameters:

- **Channel Configuration:** The Channels the Participant wishes to support at the Dispatch Endpoint and can include one or more of the ICCP[3] and/or Web Service (WS) Channel, subject to the constraints discussed below. Each Dispatch Instruction for a given Dispatch Endpoint will be broadcast via all the configured channels for that endpoint. For example, if a given endpoint was configured for both ICCP and WS, Dispatch Instructions will be sent via both the ICCP and web service channel. Note, however, as discussed further in Section 2.4.3, acknowledgements can be sent via either channel, or both channels and only the first acknowledgement (of a given type) counts;
- **Product Subscription:** The Dispatch Groups the Participant wishes to receive at the given Dispatch Endpoint. The various Dispatch Groups that can be subscribed to are as per Table 4;
- **Number of Delivery Attempts:** The number of times the Market System should attempt to deliver a Dispatch Instruction, within a fixed two-minute period, before giving up. The default value is 4;
- **Heartbeat Frequency:** The number of seconds between sending subsequent heartbeat messages to maintain the connection status for each Channel configured (different Channels can have different Heartbeat Frequency values). Endpoints configured for Web Services should define a Heartbeat Frequency not greater than 30 seconds. For endpoints configured for ICCP, the ICCP tag/fieldname that will be used for the heartbeat must also be defined[4];
- **Cross-Origin Resource Sharing (CORS) Filter:** This is only applicable to the Web Services Channel and discussion is deferred until Section 3.6.

Table 4 lists the current Dispatch Groups that can be subscribed to as well as the corresponding Dispatch Types that can be dispatched for a given Dispatch Group.

*Table 4 – Dispatch Groups and their Dispatch Types*

| Dispatch Group | Primary Dispatch Types | Secondary Dispatch Types |
|---|---|---|
| EnergyReserve | MW | RUP (Ramp Up)<br>RDN (Ramp Down)<br>RUPB (Ramp Up Broken)<br>RDNB (Ramp Down Broken)<br>IG (Intermittent Generation)<br>GC (Grid Constraint)<br>FRK (Fast Risk)<br>SRK (Sustained Risk) |
| | RESF (Fast Reserve) | |
| | RESS (Sustained Reserve) | |
| | DD | ND (Non-Dispatchable) |
| Frequency | FREQ | FST (Frequency Start Time) |
| Interruptible Load | INTF (Fast Interruptible Load) | |
| | INTS (Sustained Interruptible Load) | |

[3] Configuration of an ICCP channel for a Dispatch Endpoint requires additional agreement of the tag names that will be used for the various data fields and whether to utilise Block 2 (status) or Block 5 (controls). This is covered in detail in Section 4.

[4] Note that Transpower also exposes an ICCP heartbeat (TP_HEARTBEAT) to participants as a status as well as a control tag.

| Dispatch Group | Primary Dispatch Types | Secondary Dispatch Types |
|---|---|---|
| Voltage | VOLT | |
| | MVAR | |
| DNx | MWN (Dispatch Notified Generation)<br>DDN (Dispatch Notified Load) | ND (Non-Dispatchable |

Note that dispatches of the MW Dispatch type may not necessarily include any or all the associated Secondary dispatch types. The agreement of the configuration topology, parameter values, Dispatch Groups and Dispatch Types for a Participant's Dispatch Endpoint(s) is part of the onboarding process.

### 2.3.3 Dispatch Group Configuration

Figure 2 also shows Dispatch Group Configuration which acts as a constraint on the Channels allowed for dispatching a given Product. Constraints may be imposed, for example, for reasons such as the business or technical suitability of a given Channel for a certain product. These constraints are defined as part of Transpower interconnection policy and not laid out in this document.

## 2.4 INTERACTION OVERVIEW

At a conceptual level, irrespective of the specifics of a given Channel, there are three essential interactions between a Participant and the Market System for dispatching instructions:

- **Connection management/Heartbeats:** concerned with the sending and receiving of heartbeats so that the connection status of each Dispatch Endpoint/Channel combination is always known, by the Market System. This is used to determine whether a Dispatch Instruction can be electronically dispatched to a given Dispatch Endpoint or not;
- **Sending/Receiving Dispatch Instructions:** the sending of Dispatch Instructions to Dispatch Endpoints, by the System Operator, and receiving of them, by Participants;
- **Acknowledgement:** concerned with Participant acknowledging receipt of Dispatch Instructions including the automatic, Dispatch Instruction Receipt (ACK) and the (possibly manual) Business Acknowledgement that the Participant accepts and will action (ACKA), query (ACKQ), or if applicable reject (ACKR) the instruction (DNx only).

These are described further, in a Channel agnostic sense, in the following section, while the Web Service and ICCP specific details can be found in Section 3 and 4, respectively.

### 2.4.1 Connection Management/Heartbeats

A heartbeat mechanism exists for every Dispatch Endpoint/Channel combination so that the Market System can monitor the connection status of every Channel to every Dispatch Endpoint and the Participant can monitor its connectivity to the System Operator. It generally consists of a simple poll to check liveness and is periodically initiated by the Market System to the Participant system, at a rate governed by the Heartbeat Frequency configuration parameter corresponding to that Dispatch Endpoint/Channel.

The Market System will only send an (electronic) Dispatch Instruction to a Dispatch Endpoint if at least one configured Channel is live. The concept of a Dispatch Endpoint being Connected is equivalent to there being at least one "live" channel (i.e. responding successfully to heartbeats) for that endpoint.

### 2.4.2 Sending/Receiving Dispatch Instructions

Given the circumstance where a Dispatch Endpoint is considered by the Market System to be Connected, it can electronically send Dispatch Instructions to a Dispatch Endpoint. The sending of Dispatch Instructions is essentially a one-way interaction so that when the Market System sends a Dispatch Instruction the Participant system should consume the instruction, persist it and move on to the processing and acknowledgement phase. i.e. Acknowledgement is a separate operation explicitly initiated by the Participant.

There are several important aspects relating to receiving Dispatch Instructions that need to be understood to ensure the correct behaviour of the Market System:

- **Sequence Numbering:** An integer Sequence Number with maximum value of 9999 is maintained for each Dispatch Group, Dispatch Endpoint combination and included in each Dispatch Instruction. The Sequence Number is generally[5] monotonically increasing and incremented by one, for each dispatch, *as viewed from the perspective of the Market System*[6]. Because, however, the overall Market System is a distributed system, there is some potential for a Participant to occasionally not receive a Dispatch Instruction, receive one out of order, or receive a duplicate Dispatch Instruction and the Participant system must be able to handle these situations (discussed further below);
- **Special Significance of Sequence Number 1:** While the Sequence Number is generally increasing, there may be circumstances where it needs to be reset back to "1", for example, if the sequence number reaches the maximum value or a Participant system gets irrevocably out of sync with the Market System Sequence Number. In order that such circumstances do not cause significant issues, Participant systems must be coded for this special case so that when the Sequence Number "1" is detected, for a given Dispatch Instruction/Group, the Participant system resets its Sequence Number count to "1" and does NOT discard the instruction as if it is "out of order" as discussed below;
- **Out of order delivery and missing Dispatch Instructions:** Given there is a small chance of receiving an out of order Dispatch Instruction the Participant needs to, for each Dispatch Endpoint, keep track of the most recent Sequence Number received[7] and for each new Dispatch Instruction check the Sequence Number has strictly increased. Any Dispatch Instruction with a Sequence Number less than the most recently received Dispatch Instruction[8] should be discarded without acknowledgement. If the Sequence Number "jumps", indicating a missing Dispatch Instruction the Participant should act on the received instruction and discard the missing instruction if it arises out of order later. It is advisable that the Participant system log the event of a missing Dispatch Instruction, for statistical purposes, and monitor/report the rate of message loss to ensure it is kept to an acceptably low level;
- **Delivery Retries:** If the Market System gets an error attempting to send a Dispatch Instruction to a Dispatch Endpoint or does not receive an ACK within an appropriate timeframe[9] it will automatically reattempt delivery several times, governed by the Number of Delivery Attempts configuration parameter. In this case the Market System may indicate that a Dispatch Instruction is a "retry", but it will otherwise be unchanged; specifically, it will retain the same Sequence Number. Note that this circumstance is different from the case when an Energy Coordinator explicitly resends a Dispatch Instruction. In this case a "new" Dispatch Instruction will be issued with a different Sequence Number and the Dispatch Instruction will indicate that it is a "resend";
- **Idempotency:** Participants must ensure their system design is Idempotent to receiving duplicate Dispatch Instructions. This can be achieved by keeping track of the Sequence Numbers received for a given Dispatch Group and ignoring duplicates. It is advisable for Participant systems to log the event of duplicate messages and note the value of the retry indicator, if applicable, for statistical purposes;
- **Dispatch Instruction Completeness:** In general, except for Voltage dispatches, every Dispatch Instruction provides a complete set of the necessary target values for a given Dispatch Group and Dispatch Endpoint (i.e. including its child

---

[5] Except when reverting back to Sequence Number 1 as explained below.

[6] For example, if endpoint EP_1 is subscribed for Energy and Voltage then dispatches of Energy could contain Sequence Numbers 1, 2, 3, etc. Similarly, (separate) dispatches of Voltage could contain Sequence Numbers 42, 43, 44, etc. Note that the numbers are incremental within a Dispatch Group but not necessarily in step across Dispatch Groups.

[7] Note this needs to be keyed by Dispatch Group since each Dispatch Group has its own Sequence Number, for a given Dispatch Endpoint.

[8] Except for the special case of receiving Sequence Number 1.

[9] This timeframe is governed by the Number of Delivery Attempts configuration parameter. Since there is a requirement to receive an ACK within 2 minutes (120 secs), the timeframe is 120 /Number of Delivery Attempts. e.g. If Number of Delivery Attempts is 4 then it will retry every 30 seconds.

Blocks and Nodes) at a given point in time. Note this effectively means that, except for Voltage dispatches, any previous Dispatch Instruction (i.e. with lower Sequence Number) for the same Dispatch Group, Dispatch Endpoint combination is superseded by the latest instruction. Voltage instructions, because they are dispatched ad-hoc and not as a continuous set of set-point changes, are sent in isolation. In all cases the Dispatch Time is included with each primary dispatch value to identify its applicability/validity.

### 2.4.3 Acknowledgements

Following the sending of a Dispatch Instruction to a Dispatch Endpoint, for a given Dispatch Group, the Participant must[10] respond by acknowledging the dispatch instruction, typically in two phases[11]:

- Sending a Dispatch Instruction Receipt (ACK). The Market System raises an alert if an ACK has not been received within two minutes of sending an Dispatch Instruction (or unless a Business Acknowledgement has already been received);
- Sending a Business Acknowledgement (ACKA, ACKQ or ACKR). The Market System raises an alert if no Business Acknowledgement is received within four minutes of receiving an ACK. A second escalation alert is raised if neither has been received with a further four minutes.

An Acknowledgement message must include an acknowledgement type to indicate whether it is an ACK, ACKA, ACKQ or ACKR along with the Dispatch Group and Sequence Number corresponding to the originating Dispatch Instruction[12].

Given that, for a Dispatch Group other than Voltage, a Dispatch Instruction supersedes previous (i.e. those with lower Sequence Numbers) Dispatch Instructions for a given Dispatch Group, acknowledgment of a Dispatch Instruction effectively also acknowledges all previous Dispatch Instructions for the given Dispatch Group for the Dispatch Endpoint. For Voltage dispatches the nodes applicable for the dispatch are included in the Dispatch Instruction so each Voltage Dispatch Instruction needs to be acknowledged individually.

Note that an Acknowledgement is for a given Dispatch Group/Dispatch Endpoint combination; it does NOT need to identify which Channel it is coming from because the Market System only requires (at least) one Dispatch Instruction Receipt and/or Business Acknowledgement irrespective of whether a given Dispatch Endpoint is configured for multiple Channels. This means the Participant can Acknowledge via any one or more of the supported Channels. The Market System handling of Acknowledgements is Idempotent, so the first Acknowledgement received of each type is taken as read and any subsequent receipt of the same acknowledgement type is essentially ignored[13]. Note, however, that acknowledging via a given Channel may not necessarily be reflected in the view of another Channel.

Participants utilising multiple Channels therefore need to think through the contractual implications of this and consider only acknowledging via a Channel that is associated with its dispatch control system, if the alternative Channel(s) are essentially only serving as "listeners" to the flow of dispatch instructions.

Figure 3 summarises the System Operator view of the Acknowledgement state and valid state transitions for a given Dispatch Instruction/Dispatch Group combination (and is independent of Channel). Receipt of any acknowledgment that falls outside the state transitions in this diagram will be logged but ignored. For example, if an ACKQ is received at any point this is

[10] The system is designed to allow the configuration of future Dispatch Groups that do not require acknowledgement; currently, however, it is a Code requirement under 13.79 that all Dispatch Groups are acknowledged.

[11] Note that Endpoints configured for ICCP Block 5 (controls) will not need to explicitly send an ACK since the successful setting of a control, by the Market Dispatch Service, can be taken as equivalent.

[12] For Endpoints configured for ICCP Block 5 (controls) the participant can use feedback to communicate ACKA/ACKQ and differentiate ACKA from ACKQ by setting quality to poor on the latter. No explicit reference to Sequence Numbers is required. See Section 4 for more details.

[13] While the dispatch system to be able to handle multiples of the same ACK message, this is intended for a case where there are acknowledgements to the same dispatch instruction via two different channels. It is not intended to handle many multiples of system-generated acknowledgement messages. There is a risk of unintended consequences arising from participants systems sending high volumes of repeat messages. To mitigate this risk, we strongly recommend participants' interfaces include, as far as practical, safeguards to prevent high volumes of acknowledgement message repeats. If participants already have these mechanisms in place as part of their GENCO interface implementation, then these mechanisms could be ported over to the new communications channels.

considered a terminal state and a subsequent ACKA will be discarded. Note that if an ACKQ is received, once the query is resolved, the subsequent Dispatch Instruction will contain, unchanged, the other parameters that were not queried. Because this is a "new" dispatch instruction, however, it will be subject to its own Acknowledgment state transitions as per the figure below.

Finally note that an ACKA or ACKR is generally considered a terminal state in the business sense, but an ACKQ arriving after an ACKA or ACKR will result in a transition to ACKQ and generate an alarm within the Market System.

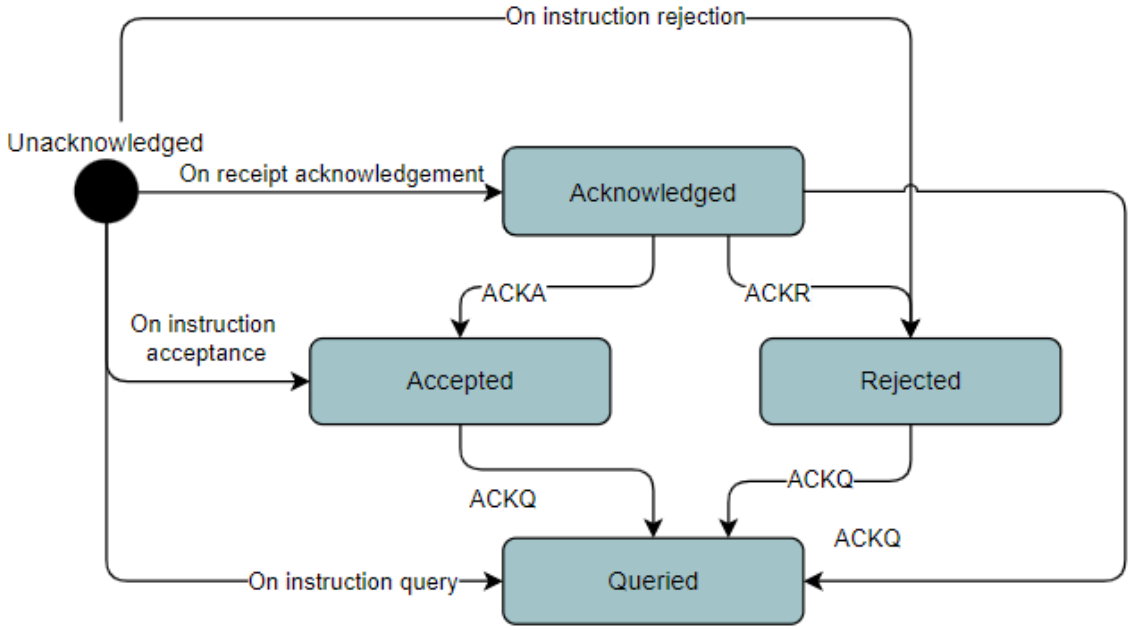



*Figure 3 – Acknowledgement State Diagram following sending of a given Dispatch Instruction*

# 3 WEB SERVICES CHANNEL SPECIFIC DETAILS

Section 2 provides a comprehensive description of the interactions of the Market Dispatch system, largely independent of the Channel. This section along with the appendices provides additional details necessary to understand the specific semantics of the Web Services Channel.

The Market Dispatch Web Services API follows a RESTful architecture using JSON message formats. The OpenAPI specification for the Web Services Channel can be found in *Appendix 1: Dispatch Web Service Specification (**OpenAPI** 2.0)* and contains structural details of the Web Service operations and formats of the request and response objects. These formats closely reflect the network topology discussed in Section 2 and are best viewed within the "Swagger editor" as described in Appendix 1. Consequently, they need not be repeated here.

The remainder of this section provides additional semantic details not explicitly covered by the OpenAPI specification and is best read in conjunction with viewing the OpenAPI specification within the Swagger editor.

Two styles of web service interaction are supported for receiving dispatch instructions via the Web Services Channel:

- Server-sent Events approach, in which the Participant client[14] effectively opens a long running connection to the Market Dispatch Web Service (server) down which the server **pushes** Dispatch Instructions as they are dispatched from the Market System and sends out periodic Heartbeats to ensure the connection is live;
- Standard HTTP request/response approach, in which the client regularly polls the Market Dispatch Web Service (server) to get (**pull)** the latest Dispatch Instructions.

These two alternative styles of web service interaction are discussed in Section 3.3 and 3.4.

There are also two alternative network interconnection configurations for the Web Services Channel: a dedicated network connection or connecting via the Internet. Further information on the network connections can be found in Section 3.10.

## 3.1 TRANSPOWER WEB SERVICE BASE URL

There are four base URLs that Participant Web Services clients can connect to; one for each Data Centre whether connecting via the Internet or via dedicated network. Further information on this is contained in the following sections.

*Table 5 – Dispatch Web Service URLs*

| Logical Location | Web Service URL |
|---|---|
| Internet - Northern Data Centre (NDC) | **Error! Hyperlink reference not valid.**https://dispatch-north.transpower.co.nz |
| Internet - Southern Data Centre (SDC) | https://dispatch-south.transpower.co.nz |
| Direct Connect - Northern Data Centre (NDC) | To be confirmed as part of Participant onboarding |
| Direct Connect - Southern Data Centre (SDC) | To be confirmed as part of Participant onboarding |

[14] From here on the term "client" is used in the usual client/server sense. In general, the "client" can be thought of as a Participant or more specifically a Participant system that is acting as a client of the Market Dispatch Web Service (which will also be referred to as the "server"). Note that a Participant can have multiple clients connected to a single Dispatch Endpoint.

## 3.2 MESSAGE FORMATS

All messages sent between the Market System and Participants, via the Web Services Channel, are in JSON format as specified in Appendix 1 and the associated YAML format definition (swagger.yml).

There are two data formats of note:

1. Floating point values will generally be set to 2 decimal places, consistent with Table 7;

2. Date/times: These will always be sent as ISO 8601 datetime formatted strings with a time zone designator, so they can be unambiguously interpreted. They will generally be sent as UTC date/times but Participant systems must be prepared to handle this date/time data format whether specified in UTC (i.e. with a "Z" zone designator) or as an offset from UTC time.

Further information on the OpenAPI Specification itself can be found here: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md and details of the ISO 8601 date format can be found here: https://xml2rfc.tools.ietf.org/public/rfc/html/rfc3339.html#anchor14.

## 3.3 SERVER-SENT EVENTS APPROACH

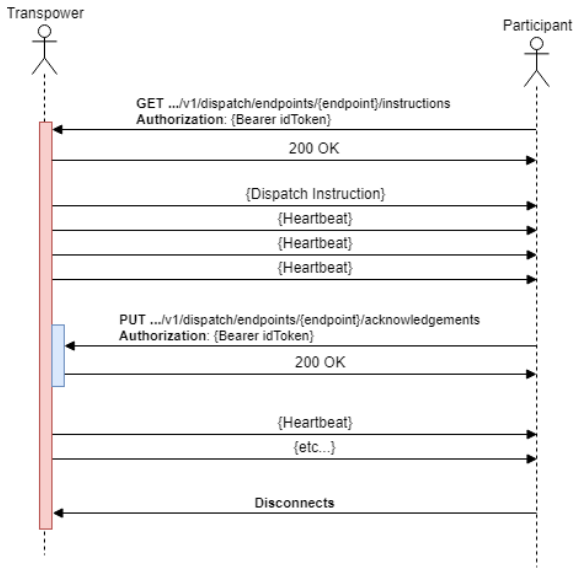
Server-sent Events (SSE) allows a client to establish a "long running" connection to the Market Dispatch Web Service down which the server can push Dispatch Instructions. Note that the term "long running" is only relative to the usual request/response pattern where a connection only exists for the relatively short duration of the request/response interaction; it does not imply a connection that provides logical continuation following connection failure i.e. every re-connection will be treated as a new connection.

The SSE approach has the advantage, over the request/response approach discussed in Section 3.4, that Dispatch Instructions will be sent to the client as soon as they become available on the server, but requires a connection to be held open for an extended period of time.

### 3.3.1 Dispatch Instruction (and Heartbeat) streaming

Dispatch Instructions and Heartbeats arrive as a stream once the client has initiated a connection using the SSE approach. On connection, all the "latest" (as defined below) instructions for the Endpoint are sent and following that instructions are sent as they are issued. Figure 4 provides an illustrative sequence diagram for the examples that follow.

*Figure 4 – Server-sent Events Approach*




An example of an SSE connection, for a fictitious Dispatch Endpoint named `acme-endpoint-1`, starts with a client initiating the following request and then subscribing to receive `dispatch` events[15]:

*Example Request*

```
GET /api/v1/dispatch/endpoints/acme-endpoint-1/instructions HTTP/1.1
Host: www.acme.com
Accept: text/event-stream
Authorization: Bearer eyJhbGciOiJIUzI1NiIXVCJ9...TJVA95OrM7DcEfxjoYZgeFONFh7HgQ
```

The presence of `Correlation-Id` and `Authorization` headers in the various requests and responses is discussed later, in Section's 3.7 and 3.5.

The event stream "response" to the request, corresponding to the above sequence diagram is as follows[16]:

*Example Response ("SSE event stream")*

```
HTTP/1.1 200 OK
Content-Type: text/event-stream

event: dispatch
id: c1b3fba1-02ca-45cb-afcd-c97324d28e1c
data: [
  {
    "dispatchEndpointName": "acme-endpoint-1",
    "dispatchEndpointOwner": "ACME",
    "correlationId": "c188cb0e-1a09-4184-890c-0f0bb2255425",
    "dispatchGroupName": "voltage",
    "sequenceNumber": 299792458,
    "traderCode": "ACME",
    "isUserResend": false,
    "messageSentTime": "2019-06-20T15:49:00.123Z",
    "voltageDispatch": {
      "blocks": [],
      "nodes": [
        {
```

[15] For details on how to subscribe to named events look here: https://html.spec.whatwg.org/multipage/server-sent-events.html#server-sent-events and https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events/Using_server-sent_events#Event_stream_format.

[16] Note the depiction of the dispatch instruction data being split across lines and coloured is for document readability purposes only. In practice it will most probably appear on the same line as the data tag.

```
          "name": "ACME_1",
          "traderCode": "ACME",
          "primaryValues": [
            {
              "dispatchType": "VOLT",
              "dispatchValue": 230.1,
              "dispatchTime": "2018-10-10T19:57:08.827Z",
              "dispatchIssueTime": "2018-10-10T19:57:08.827Z"
            }
          ]
        }
      ]
    }
  }
]

event: heartbeat
id: a90dd4f5-471a-4b7b-8b9c-087cfbb31709

event: heartbeat
id: 2b3e4122-0e25-4290-a7c3-dbd76fae60a0

event: heartbeat
id: bb442fbe-2d27-4608-969a-02813463a732

: etc…
```

In the example above, the dispatch is for the Voltage Dispatch Group to a single node configuration. Refer to Appendix 1 for the OpenAPI specification detailing the format of Dispatch Instructions, in general, as well as a more realistic example.

Note that Heartbeats and Dispatch Instructions arrive as individually named events and a `dispatch` event consists of an array of Dispatch Instructions, typically one Dispatch Instruction for each Dispatch Group configured for that endpoint. In practice this means that on first connection, for a given Dispatch Endpoint, the "latest" instructions for each Dispatch Group will be dispatched by the Market System "together" as a list, but after that will generally arrive as a singleton array as each Dispatch Instruction is issued.

Heartbeat events don't need to be subscribed for and do not require any specific response by the client as they are only sent by the server as "keep-alive" messages; if the client prematurely disconnects, or the connection fails for some other reason, the server will receive an error message and will close/cleanup the connection. At that point, if the Participant does not have any other WS clients running (with live connections), for *the same Dispatch Endpoint*, the Market System will consider **the Webservice Channel** for this Dispatch Endpoint disconnected. Note, however, that if the Dispatch Endpoint is configured for other Channels (that are connected), the Market System will still treat the endpoint as Connected. This point is discussed again at the end of this section.

### 3.3.2 Acknowledgements

Acknowledgements are not included in the event stream, as they occur "out of band", as an independent request/response flow to the event stream. An example of a Dispatch Instruction Receipt is shown below and consists of putting an Acknowledgement object (populated with an ACK plus Dispatch Group and Sequence Number values corresponding to those in the Dispatch Instruction) to the acknowledgements resource (Note also the Correlation-Id header value must correspond to the correlationId specified in the dispatch instruction body):

*Example Request*

```
PUT /api/v1/dispatch/endpoints/acme-endpoint-1/acknowledgements HTTP/1.1
Host: www.acme.com
Correlation-Id: c188cb0e-1a09-4184-890c-0f0bb2255425
Authorization: Bearer eyJhbGciOiJIUzI1NiIXVCJ9...TJVA95OrM7DcEfxjoYZgeFONFh7HgQ
Content-Type: application/json
```

```
{
  "ackType": "ACK",
  "dispatchGroupName": "voltage",
  "sequenceNumber": 299792458
}
```

The response from the server would be as follows:

*Example Response*

```
HTTP/1.1 200 OK
Correlation-Id: c188cb0e-1a09-4184-890c-0f0bb2255425
```

Note that each acknowledgement operation only allows a single Dispatch Instruction to be acknowledged so it is quite possible for an array with multiple Dispatch Instructions to arrive at once, necessitating several acknowledgement operations to be issued to acknowledge the "batch".

### 3.3.3 Server-sent Event considerations

There are a few important things Participants need to be aware of if contemplating implementing systems that use the SSE approach:

- A Participant may have multiple clients connected to a single Dispatch Endpoint and they will each get all the Dispatch Instructions targeted at that Dispatch Endpoint, for all Dispatch Groups that the Participant has subscribed to (at onboarding/modelling time). Note that there is a configuration limit to the number of concurrent client connections allowed per Dispatch Endpoint anticipated to be in the order of 12 or so;
- The Market Dispatch Web Service (server) retains only the most recent Dispatch Instruction, within a 180-day window, for a given Dispatch Endpoint and Dispatch Group, except for Voltage, which can be dispatched on a per node basis. This means that any new client connections established will typically receive at most one 'historical'[17] Dispatch Instruction (with more than one possible for Voltage), for each Dispatch Group, before waiting for new dispatches. The client will then receive every new Dispatch Instruction that arrives until it is disconnected. Note also that if there are no current Dispatch Instructions when the client connects, an empty array will be returned rather than a null value;
- Each client will also receive Heartbeats at a rate dictated by the Heartbeat Frequency agreed at onboarding/modelling time but need not take any action or reply as discussed in Section 3.3.1;
- If a Participant has, for a given Dispatch Endpoint, at least one WS client connected and receiving Heartbeats (or otherwise connected via request/response polling as discussed in the next section), the Market System will consider the Webservice Channel connected for the Dispatch Endpoint (and all its subordinate Blocks and Nodes). Participants considering utilising multiple clients per Dispatch Endpoint, therefore, need to carefully think through the possible connection/disconnection scenarios to ensure the appropriate overall dispatch behaviour is obtained;
- Any Participant contemplating developing a browser-based application that directly connects to the Market Dispatch Web Service should note that while most browsers support Server-sent Events Microsoft Internet Explorer and Microsoft Edge do not. The current list of browsers that support Server-sent Events as well as further general information on Server-sent events can be found here:
  - https://html.spec.whatwg.org/multipage/server-sent-events.html#server-sent-events
  - https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events/Using_server-sent_events#Event_stream_format

## 3.4 Request/Response Approach

The standard HTTP request/response pattern, requires the client to regularly poll the Market Dispatch Web Service (server) to get the latest Dispatch Instructions. This approach has the advantage of simplicity as it does not require the establishment of

[17] i.e. up to 180 days old, irrespective of whether the instruction has previously been acknowledged or not.

a long running connection and can be implemented as a single threaded client but also has a few disadvantages, discussed later.

The definition of what "latest" means needs re-emphasising:

- for Dispatch Groups, other than Voltage, the "latest" dispatch instruction is the most recently issued instruction for the Dispatch Endpoint, Group combination, **irrespective of whether it has been acknowledged or not**. There will be at most one dispatch instruction for each possible Dispatch Group, since the whole Endpoint can be dispatched in a single instruction, and, noting that each primary dispatch value includes the Dispatch Time, the instruction can potentially be quite some time in the past (in fact up to 180-days);
- For Voltage the number of "latest" dispatch instructions for a given Endpoint can be greater than one and in fact can be up to one for each node (or block) configured for the Endpoint, since Voltage can be dispatched on a per-node basis. As for other dispatch groups, this is irrespective of whether they have been acknowledged or not. Similarly, the Dispatch Time is included in a Voltage dispatch and can be a historical value. Finally, since Voltage dispatches are sent for either a VOLT or MVAR Dispatch Type (not both), one of the Dispatch types will be dispatched with a null value and should be ignored.

To summarise the above, the "latest" dispatch instructions for an Endpoint represent the most recently issued instructions that cover all Blocks and Nodes in the Endpoint topology, whether they are currently applicable or historical, irrespective of whether they have been acknowledged or not.

### 3.4.1 Getting the latest Dispatch Instructions

A client gets the most recent Dispatch Instructions (as defined above) for a given Dispatch Endpoint by issuing a standard HTTP GET as illustrated by the following example, for a Dispatch Endpoint named `acme-endpoint-1`:

*Example Request*

```
GET /api/v1/dispatch/endpoints/acme-endpoint-1/instructions/latest HTTP/1.1
Host: www.acme.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIXVCJ9...TJVA95OrM7DcEfxjoYZgeFONFh7HgQ
```

The presence of `Correlation-Id` and `Authorization` headers in the various requests and responses is discussed later, in Section's 3.7 and 3.5. The response in this case is a standard application/json message:

*Example Response*

```
HTTP/1.1 200 OK
Correlation-Id: ddf09d98-55a6-4712-bd64-c509f941333e
Content-Type: application/json

[
  {
    "dispatchEndpointName": "acme-endpoint-1",
    "dispatchEndpointOwner": "ACME",
    "correlationId": "ddf09d98-55a6-4712-bd64-c509f941333e",
    "dispatchGroupName": "frequency",
    "sequenceNumber": 885418782,
    "traderCode": "ACME",
    "isUserResend": false,
    "messageSentTime": "2019-06-20T15:49:00.123Z",
    "frequencyDispatch": {
      "blocks": [],
      "nodes": [
        {
          "name": "ACME_1",
          "primaryValues": [
            {
              "dispatchType": "FREQ",
              "dispatchValue": 15,
              "secondaryDateTimeValues": [
```

```
            {
              "dispatchType": "FST",
              "dispatchValue": "2018-10-12T01:00:00.000Z"
            }
          ]
        }
      ]
    }
  ]
}
]
```

Refer to Appendix 1 and 2 for the OpenAPI specification detailing the format of Dispatch Instructions, in general, as well as some more realistic examples.

Note that the body of the HTTP message is the same as the content of the dispatch event for the SSE approach; an array of Dispatch Instructions, one for each subscribed Dispatch Group. In this example, the dispatch is for the Frequency Dispatch Group to a single node configuration.

There is also a request/response style operation for getting the most recent Dispatch Instruction for a given Dispatch Endpoint and Dispatch Group plus an operation to confirm which Dispatch Groups have been subscribed to for a given Dispatch Endpoint. Refer to the Open API specification for more details.

There is also no explicit Heartbeat operation when using the request/response polling approach to receive Dispatch Instructions. Instead each poll for the latest Dispatch Instructions also serves as a Heartbeat as far as the Market System is concerned.

### 3.4.2 Acknowledgements

The process of sending Acknowledgements to the Market System is the same as for the SSE approach since SSE only applies to the Dispatch Instruction/Heartbeats event stream. To supplement the receipt Acknowledgement example in Section 3.3.2, which equally applies to the request/response approach, an additional example of a Business Acknowledgement (acceptance) request/response follows (as for SSE the Correlation-Id header value must correspond to the correlationId specified in the dispatch instruction body):

*Example Request*

```
PUT /api/v1/dispatch/endpoints/acme-endpoint-1/acknowledgements HTTP/1.1
Host: www.acme.com
Correlation-Id: ddf09d98-55a6-4712-bd64-c509f941333e
Authorization: Bearer eyJhbGciOiJIUzI1NiIXVCJ9...TJVA95OrM7DcEfxjoYZgeFONFh7HgQ
Content-Type: application/json

{
  "ackType": "ACKA",
  "dispatchGroupName ": "frequency",
  "sequenceNumber": 885418782
}
```

The response from the server would be as follows:

*Example Response*

```
HTTP/1.1 200 OK
Correlation-Id: ddf09d98-55a6-4712-bd64-c509f941333e
```

Note that acknowledgements need only be sent for Dispatch Instructions not already acknowledged so if the client detects a Dispatch Instruction is a duplicate then acknowledgement is not necessary although it is not harmful to the Market System if it does receive one since it manages its acknowledgement state according to the state transition diagram in Figure 3 of Section 2.4.3.

### 3.4.3 Request/Response considerations

There are a number behavioral aspects Participants need to consider if contemplating implementing a client that uses the request/response approach:

- This approach does not utilise an explicit Heartbeat request/response; instead, each poll for the latest dispatch instructions also serves as a Heartbeat as far as the Market System is concerned. This (potentially[18]) introduces a lower bound on the frequency that the client needs to poll for dispatch instructions, given the Heartbeat Frequency agreed at onboarding/modelling time for the given Dispatch Endpoint. i.e. In the absence of other clients using the SSE approach, for a given Dispatch Endpoint, the Participant needs to ensure the request/response client is polling at a rate no less than the (WS Channel) Heartbeat Frequency to ensure the Market System considers the Webservice Channel connected, for the Dispatch Endpoint;
- A Participant may have multiple clients that poll a single Dispatch Endpoint for Dispatch Instructions and they will each get all the Dispatch Instructions targeted at that Dispatch Endpoint, for all Dispatch Groups that the Participant has subscribed to (at onboarding/modelling time). This is the same as for the SSE approach;
- The Market Dispatch Web Service (server) retains only the most recent Dispatch Instruction, within a 180-day window, for a given Dispatch Endpoint and Dispatch Group. This means that every client poll for the latest dispatch instruction will receive at most one 'historical'[19] Dispatch Instruction, for each Dispatch Group[20]. How this differs from the SSE approach is that, depending on the polling rate, the client will receive the same Dispatch Instructions multiple times and will therefore need to deal with duplicates;
- If a Participant has, for a given Dispatch Endpoint, at least one client connected and polling at a rate greater than the WS Heartbeat Frequency (or otherwise connected via SSE), the Market System will consider the Webservice Channel connected for the given Dispatch Endpoint (and all its subordinate Blocks and Nodes). Participants considering utilising multiple clients per Dispatch Endpoint, therefore, need to carefully think through the possible connection/disconnection scenarios to ensure the appropriate overall dispatch behaviour is obtained.

## 3.5 Client Authentication and Security

### 3.5.1 Authentication/Access Control

The Dispatch Web Service uses JSON Web Tokens (JWT) for stateless authentication. JWT provides ease of usage across multiple platforms and is very commonly used at internet scale enabling OAuth 2.0 Authorization.

JWT tokens are generated by the Participant using their private key from a RSA public/private key pair for signing. The on-boarding process will include steps for Participants to provide their public key to Transpower so the validity of the tokens can be established.

Transpower will not have any access to a Participant's Private key and the Participant is responsible for managing the security of their private key i.e. Participants should NEVER provide their private key to Transpower or any other party; if a Participant accidentally releases, or suspects loss of, their private key, they should immediately take steps to revoke their X.509 certificate and advise Transpower of the loss. Transpower will maintain processes to enable Participant's to cancel and update their X.509 certificates in the Dispatch System upon request.

---

[18] The reason the lower bound on polling rate is not necessarily mandatory is that, if there is at least one other client that utilises the SSE approach, for the given Dispatch Endpoint, then that client will satisfy Market System's Connectedness criteria. i.e. recall that the Market System considers the Webservice Channel, for a given Dispatch Endpoint, connected if at least one client is responding to Heartbeats/alive. i.e. Connectedness is a Dispatch Endpoint/Channel concept not a "client connectedness" concept.

[19] i.e. up to 180 days old, irrespective of whether the instruction has previously been acknowledged or not.

[20] Note also that if there are no current Dispatch Instructions, an empty array will be returned rather than a null value; the same as the behaviour for the SEE approach on initial connection establishment.

An example `Authorization` header is as follows:

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIXVCJ9...TJVA95OrM7DcEfxjoYZgeFONFh7HgQ
```

Transpower will verify the RS256 signed JWT (as specified in rfc7519) before providing the resources. JWT verification will include:

- Verifying the signature: using the Participant certificate and RSA (RS256) as the algorithm;
- Validating the following claims fields:
  - Token currency: The token must include an "nbf" ("Not Before") claim so that the current date/time is not before the given date/time. If not, the request will be rejected;
  - Token expiration: The current date/time must be before the expiration date/time listed in the "exp" claim (which is a Unix timestamp). The expiration must also not exceed 300 seconds into the future. If not, the request will be rejected;
  - Token issuer: The "iss" claim denotes the issuer of the JWT. The value must match the one configured in the API.

Any validation failure will result in an HTTP 401, Unauthorized error response.

For further information on JWT and OAuth 2.0 see: https://jwt.io/introduction/, https://tools.ietf.org/html/rfc7519, https://tools.ietf.org/html/rfc6749 and https://oauth.net/articles/authentication.

### 3.5.2 Transport Security

To protect against man-in-the-middle attacks, only HTTPS (TLS 1.0+) connections are available for accessing Transpower resources via RESTful APIs.

One-way (Server) SSL will be used in which the Web Service will present the Transpower certificate authority signed X.509 certificate to the Participant. The Participant will not be required to present their certificate during HTTPS session establishment.

## 3.6 Cross-Origin Resource Sharing (CORS) HTTP headers

Cross-Origin Resource Sharing (CORS) is a mechanism that uses additional HTTP headers to tell a browser to let a web application running at one origin (domain) have permission to access selected resources from a server at a different origin.

In order for Participants to develop browser-based applications that directly call the Market Dispatch Web Service, the web service will populate the following HTTP header:

```
Access-Control-Allow-Origin
```

The value of this header will be a list of (possibly wildcarded) URIs that should include the Participant's own domain. By making it only the Participant's domain inhibits any call to the web service being made from any other domain which enhances the security of the service from the Participant's perspective. The value the Market Dispatch Web Service uses to populate this header comes from configuration, agreed with the Participant as part of the onboarding process.

More details on CORS can be found here: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS.

## 3.7 HTTP header - Correlation Id

To support the tracing of requests throughout the Market System the Web Service supports, and requires the client to support, the following custom HTTP header:

`Correlation-Id`

To support low-level, end-to-end traceability from Dispatch Instructions through to corresponding Acknowledgements, the rules for populating this header value are as follows:

- Each Dispatch Instruction contains a `correlationId` (within the Dispatch Instruction object) that is also populated as a `Correlation-Id` HTTP header in polling request responses;
- The client must use the provided correlation id to populate the `Correlation-Id` HTTP header in subsequent acknowledgements;
- `Correlation-Id` should be generated by client for initial GET /instructions/latest
- The client should not use a `Correlation-Id` header for any other requests. If it is provided it will be ignored.

Participants should also retain/log the values of the correlation ids to simplify fault finding between themselves and the System Operator.

The following example shows the correct use of the correlation id in a fragment using the request/response approach to get the latest Dispatch Instruction and the subsequent acknowledgement (the relevant id values are highlighted and annotated in green):

*Example Dispatch Instructions Request*

```
GET /api/v1/dispatch/endpoints/acme-endpoint-1/instructions/latest HTTP/1.1
Host: www.acme.com
Authorization: Bearer eyJhbGciOiJIUzI1NiIXVCJ9...TJVA95OrM7DcEfxjoYZgeFONFh7HgQ
```

*Example Response*

```
HTTP/1.1 200 OK
Correlation-Id: 889ffe5f-ab14-421a-a83f-38a9598ff61d          # matches correlationId in payload
Content-Type: application/json

[
  {
    "dispatchEndpointName": "acme-endpoint-1",
    "correlationId": "889ffe5f-ab14-421a-a83f-38a9598ff61d",  # also unique
    "dispatchGroupName": "frequency",
    "sequenceNumber": 885418782,
    … etc
  }
]
```

*Subsequent Acknowledgement Request*

```
PUT /api/v1/dispatch/endpoints/acme-endpoint-1/acknowledgements HTTP/1.1
Host: www.acme.com
Correlation-Id: 889ffe5f-ab14-421a-a83f-38a9598ff61d          # matches correlationId in above
Authorization: Bearer eyJhbGciOiJIUzI1NiIXVCJ9...TJVA95OrM7DcEfxjoYZgeFONFh7HgQ
Content-Type: application/json

{
  "ackType": "ACKA",
  "dispatchGroupName": "frequency",
  "sequenceNumber": 885418782
}
```

*Subsequent Acknowledgement Response*

```
HTTP/1.1 200 OK
Correlation-Id: 889ffe5f-ab14-421a-a83f-38a9598ff61d          # matches correlationId above
```

Note that while the example above only includes one acknowledgment, all acknowledgements (i.e. ACK and ACKA/ACKQ/ACKR) for a given dispatch instruction must use as a correlation id matching the event id of the originating dispatch instruction. Also note that for SSE pushed Dispatch Instructions there is no correlation header value, but the correlation id is also in the Dispatch Instruction and that can be used for subsequent acknowledgments.

Note also that while it may appear that use of the correlation id is superfluous, given the sequence number serves a similar "correlating" purpose these mechanisms serve different purposes; the sequence number is a business level value, unique (and incrementing) for a given dispatch group/dispatch instruction/end point combination and meaningful only within the context of the Market System while the correlation id is globally unique for a given dispatch instruction and meaningful within the underlying logging system for correlating all messages throughout the infrastructure relating to a given "business transaction".

## 3.8 HTTP Status Codes

Standard HTTP status codes are returned to indicate success, client and server errors. The specific status codes used can be found in the OpenAPI specification. Further information on HTTP status codes, in general, can be found here: https://restfulapi.net/http-status-codes/.

## 3.9 Web Service Versioning

The Market Dispatch Web Services are versioned by including a version number embedded in the URI immediately after the authority component and before the resource path, in the following form:

```
/v{version}/
```

*Example*

```
/transpower.co.nz/api/v1/dispatch/…
```

Version numbers represent "major" versions and will change only when there are breaking changes in the API. Examples of breaking changes include:

- The removal of any property from the request or response representation;
- A change of datatype for an existing property, or a change from optional to required;
- The removal of any resource, or HTTP verb support;
- A change to the way errors are handled;
- Any change to existing resource URI's.

A non-breaking change is any change such that any messages that could be processed by the previous version can also be processed by the latest version. Examples of non-breaking changes include:

- The addition of new (optional) properties to the JSON representation;
- The addition of resources and associated URI's;
- The addition of support for new HTTP verbs (new operations) on existing resources;
- Support for new custom headers.

Any time a new major version of an API is released the System Operator will provide a reasonable notice period where the superseded version will continue to be supported in operation.

## 3.10 Differences between WS over direct circuit versus Internet

From a functional perspective there are no differences between Web Services over direct circuit versus Internet. There may, however, be some non-functional differences in response times, performance variability and availability using Internet based connections.

From a web service client connectivity perspective there are the following differences:

- Participants using the Internet, only, will connect to a single endpoint, and Transpower recommends maintaining more than one connection to this single endpoint;
- Participants using a dedicated network must maintain at least one (recommended to maintain more than one) connection to both SDC and NDC. Note, however, that only one datacentre will be active at a point in time.

The WS over the Internet solution will also utilise RedShield to provide additional "as-a-service" (cloud based) security controls and management processes to protect Transpower from attacks originating from the Internet but these will be transparent to the Web Service applications themselves.

Participants wishing to use Internet based connections will need to supply their IP address(es) as part the onboarding process so they can be added to the Transpower firewall white-list.

Figure 5 illustrates the Internet based Web Services environment; Direct circuit connections are similar except that they do not pass through the Internet and RedShield "front end".

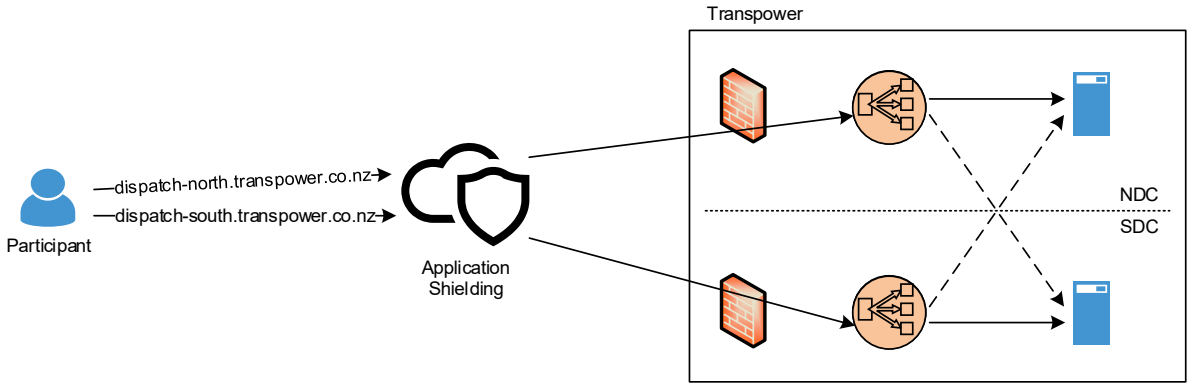



*Figure 5 – Internet access to Dispatch Web Services*

## 3.11 Software Testing Environment (Dispatch Simulator)

Transpower hosts an Internet facing web service Dispatch Simulator to enable pre-integration testing for the participant's solution[21]. This simulator:

- Has authenticated access for the users, provisioned by Transpower. User account lifecycle will be managed by Transpower, as outlined below;
- Implements the same web service specification as the Dispatch Web Service (i.e. as per Appendix 1: Dispatch Web Service Specification (**OpenAPI** 2.0));
- Implements the same authentication policies and security controls as the Dispatch Web Service;
- Presents a web interface to visualize heartbeats, acknowledgements and dispatch instructions;
- Provides an interface for Participants to view logs to aid diagnostics;
- Does not reflect end-to-end market data but uses (semi) random data instead.

Transpower has an authentication process for Web Service Dispatch Simulator users and will manage each Participant/user's account lifecycle. Participants can apply to Transpower for access and will be provided with log-in credentials (username/password) and web URL to access the application once Market Operations confirm that their Endpoint configuration has been defined and user account established. Once logged-in, users can produce and receive dispatch-like instructions to their configured Endpoints. Password reset processes and user account expiry policy remains TBA.

Users will need to contact their Transpower Market Operations representative (or email *market.operations@transpower.co.nz*) to confirm their Dispatch Endpoint configuration and obtain login-in credentials.

[21] Participants will be able to "sign up" for access to this once they have signed the Data Transmission Agreement as part of the onboarding process.

# 4 ICCP CHANNEL SPECIFIC DETAILS

## 4.1 INTRODUCTION TO ICCP

The intention of Transpower's ICCP system is that it meets the data transmission requirements of the Electricity Industry Participation Code Schedule 8.3, Technical Code C and provides both the primary and backup means of data communication.

Transpower operates an ICCP gateway service, connected to National SCADA, that provides a redundant, real time data exchange with the Participant. The ICCP gateway communications are based on the TASE.2 (Telecontrol Application Service Element-2) protocol which leverages the Manufacturing Message Specification (MMS) and the IEC 60870-6 standard which defines the objects that are used to convey the data (over TCP/IP).

To implement ICCP Interconnection, the Participant must implement an appropriate ICCP gateway service to intercommunicate with the Transpower ICCP gateway. This may consist of two or more dedicated application servers, or a set of simple ICCP protocol converters (for example, ICCP to serial DNP3).

ICCP has previously been implemented at Transpower for internal and external (generators and distributors) integration between SCADA systems and is now one of the Channels for the Market Dispatch Service.

## 4.2 ICCP SYSTEM CONFIGURATION

A typical ICCP connection for SCADA and Real Time Dispatch is illustrated in Figure 6. Note that the Participant's ICCP gateway may consist of single or redundant servers or may be an application that resides on the SCADA servers. The ICCP link will be over TCP/IP. The WAN connection for Dispatch Service between the Transpower and Participant's system will be provided and managed by Transpower and will terminate in an edge router (or routers) located at the Participant's premises. Note that the physical design of the network, location and configuration of firewalls, etc. is not included in the scope of this document.

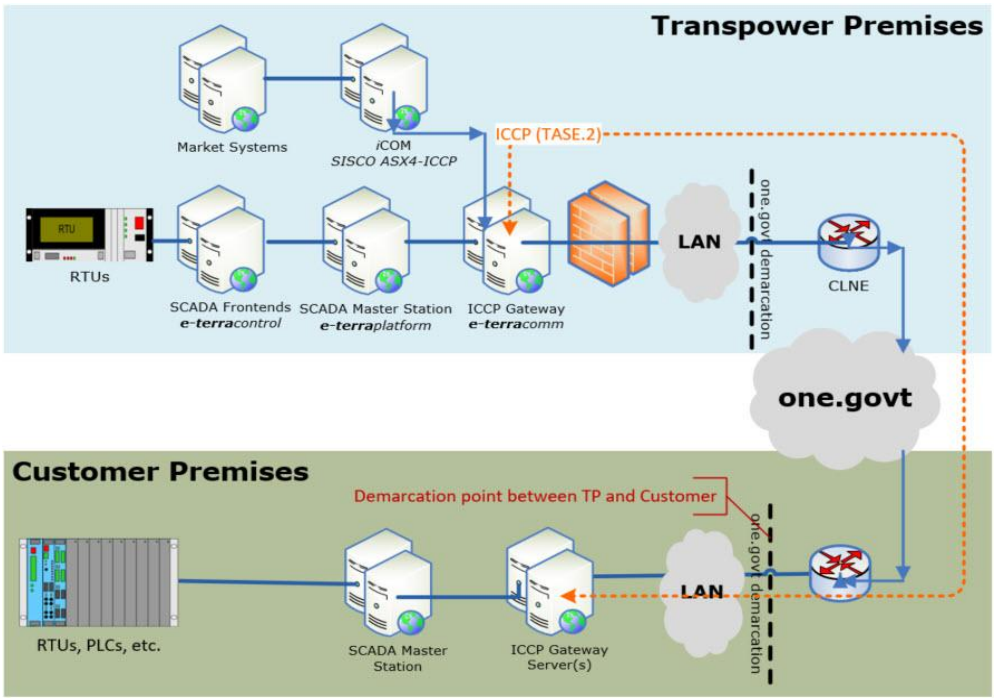



*Figure 6 – Typical ICCP Configuration*

## 4.3 System Availability and Redundancy

Transpower's ICCP service is designed to be highly resilient, offering n-1 site diversity, n-3 server infrastructure, and automatic application-level failover. All tiers of the solution are covered by 24/7 on-call support services.

Under the Code (Schedule 8.3, Technical Code C asset owners are required to provide both a primary and a backup means of providing data to the System Operator - the ICCP service is sufficiently resilient to be considered as both the primary and backup means of communication with asset owners as long as the specified minimum requirements are met. The target service availability, from the Transpower SCADA/Market Systems (Dispatch) to the demarcation point, is 99.9% or better.

### 4.3.1 ICCP links and Path redundancy

Minimum and recommended levels of ICCP resiliency are defined in the following sections. To ensure compliance with the code Participants must meet at least the minimum level.

#### Minimum requirement (n-1)

Minimum requirements are a single site with:

- A minimum of two independent network paths terminated at the Participant site. It is the responsibility of the Participant to ensure that the provided CLNE routers are hosted in a manner that ensures no single points of failure are created;
- A minimum of two independent devices, with no single points of failure and with automatic failover, providing the ICCP communications endpoint.

#### Recommended requirement (n-x)

Recommended requirements are two geographically diverse sites with:

- A minimum of two independent network paths terminated at each Participant site. It is the responsibility of the Participant to ensure that the provided CLNE routers are hosted in a manner that ensures no single points of failure are created;
- A minimum of two independent devices, with no single points of failure and with automatic failover, providing the ICCP communications endpoint. This may be two devices per site, or one device per site;
- In certain circumstances the use of single network path per site may be considered if the Participant solution can provide automated n-1 service resilience during the loss of a network link. Such redundancy would require the Participant to either:
    - Trigger an immediate and automatic device/application-level failover from the failed site to the alternate site; or
    - Provide a completely independent and diverse means of routing traffic from the alternate site link to the ICCP device at the failed site (e.g. via the Participant's own network infrastructure).

Note: Existing ICCP Participant sites for SCADA connection are on primary and secondary links with dual path connectivity. Each ICCP server both at Participants and Transpower end have dual paths i.e.:

- Participant Site: 2 servers x 2 paths
- Transpower end: 4 servers x 2 paths

## 4.4 ICCP SECURITY

The ICCP implementation between Transpower and its Participants is on Secure ICCP. Transpower's MMS and OSI stack supports secure ICCP via the MMS Security Toolkit. This provides two distinct and independent facilities:

- The MMS Application Certificate Exchange (MACE) protocol for application authentication. This provides positive identification of the application as well as anti-replay for non-SSL/TLS connections.
- SSL/TLS as secure transport layer. This provides positive identification of the remote node (bi-directional certificate exchange) and SSL/TLS data encryption.

The MMS Security Toolkit makes use of X.509 Certificates at both the SSL/TLS and MACE levels for authenticating remote nodes and passing public keys. Operating system specific tools handle the management of the certificates and keys. The security certificates will be generated and managed by Transpower and issued to the Participant for implementation of secure ICCP.

## 4.5 ICCP BLOCKS

ICCP has been designed to be modular. Its functionality is specified by a series of "*Conformance Blocks*", where each block represents a specific function or set of functions that can be implemented.

Block 1 is mandatory in all ICCP implementations. The other blocks are optional, depending on the functionality required. The ICCP implementation between Transpower and its Participants shall use the following block types (additional block types may be used by mutual agreement).

### 4.5.1 Block 1 (Basic Services, Periodic Power System Data)

This block contains the following:

- Association Establishment Services (initiate, conclude and abort), for the overall operation of the link.
- Data Value Objects, which provide for the periodic transfer of power system and Market Systems Dispatch data, specifically:
    - Status Points (e.g. IG constraint, BSC – Block Constraint)
    - Analog Points (includes SCADA Analogs and Dispatch points such as MW, FREQ, RESF, RESS, MVAR, VOLT, INTF, INTS, DD, MWN and DDN)
- Data Set Objects
- Data Set Transfer Set Object (interval time-out and operator request conditions only)

Note that block 1 uses a strict interval and does not support report by exception.

### 4.5.2 Block 2 (Extended Data Set Condition Monitoring)

Its extension of services provided by Block 1 and Reporting by exception, i.e. any change in the data values are reported by exception.

### 4.5.3 Block 5 (Device Control)

This allows select, operate, get tags, and set tags on devices that a server controls. This block provides a mechanism for transferring a 'request to operate a device' from one ICCP implementation to another. ICCP does not directly control the device; rather it communicates a Transpower's request to operate a device to the server.

On the Transpower ICCP link, this block is used where controls and/or setpoints are required (e.g. MFK setpoint controls, also, Transpower has delegated control of a circuit breaker to a Participant).

Note Block 5 conformance is not currently supported for dispatch communications.

## 4.6 DATA ITEM REPRESENTATION

ICCP defines the Virtual Control Centre concept to represent an abstraction of a real-world control centre that acts like an ICCP wrapper to the Control Centre. Information may be defined to have global scope (VCC scope) or a domain scope (ICC Scope). For the implementation of ICCP between Transpower and its Participants the ICCP scope will be as follows:

- Data Items: Transpower uses global data scope (VCC scope) for Block2,
- Data Sets: Data Sets are specific to a remote and must have ICC scope. Note that Transpower's ICCP system does not support VCC scope for Data Sets. For Transpower to receive data from a Participant, the Participant's ICCP solution MUST support ICC scope in their ICCP implementation.

Data sets are groups of data items (status points, analogs, controls and setpoints), used to split the list of data items into manageable pieces. It is not necessary to co-ordinate the data sets between the two ends of an ICCP link, as this is carried out automatically by the ICCP protocol.

Every Site (or node) will be configured as Virtual Control Centre, hiding its data items, global data scope (VCC scope) to be exposed from other participants.

A typical configuration on *i*COM server would look like Figure 7; this will provide encapsulation and logical separation to data items. Note if the relevant dispatch group is 'DNx', the From Remote data sets would also include 'ACKR_SEQ_NUM'.

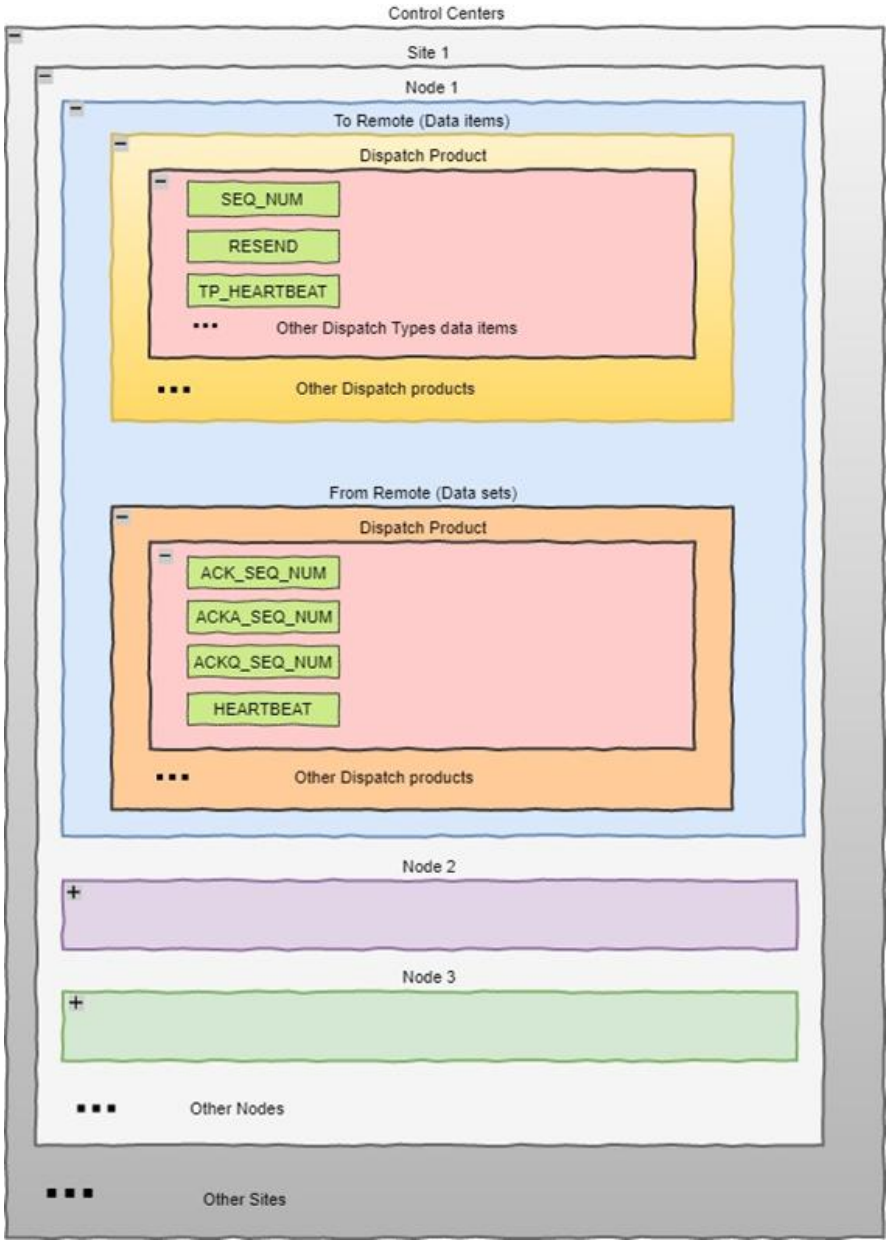



*Figure 7 – Logical iCOM server Configuration*

### 4.6.1 Acknowledgement (for *i*COM ICCP)

ICCP Acknowledgement mechanism for Dispatch Service would be used to ensure that the Participant has received a request instruction; when a dispatch is received by the Participant, the Participant's system would then set an Acknowledgement Response Data item equal to the last Dispatch Sequence it received as a form of acknowledgement.

Every dispatch type will have the following data items specific to acknowledgement mechanism.

*Table 6 – Acknowledgement Data Items*

| ICCP Data Item | Updated by | Description |
|---|---|---|
| Block 2 | | |
| SEQ_NUM<br>e.g. ACME_ENERGYRESERVE_SEQNUM | Transpower | Sequence number |
| ACK_SEQ_NUM<br>e.g. ACME_ENERGYRESERVE_SEQNUM_ACK | Participant | Automatic acknowledgement sequence number |
| ACKA_SEQ_NUM<br>e.g. ACME_ENERGYRESERVE_SEQNUM_ACKA | Participant | Accepted acknowledgement sequence number |
| ACKQ_SEQ_NUM<br>e.g. ACME_ENERGYRESERVE_SEQNUM_ACKQ | Participant | Acknowledgement query sequence number |
| ACKR_SEQ_NUM<br>e.g. ACME_DNX_SEQNUM_ACKR | Participant | Acknowledgement reject sequence number |

Data flow for Block 5 and Block 2 Acknowledgement will be as per

Figure 9 below. Note that any bad quality attribute except badCommFailure (for example "bad") will be treated as an ACKQ. Communication failure will result in retry.

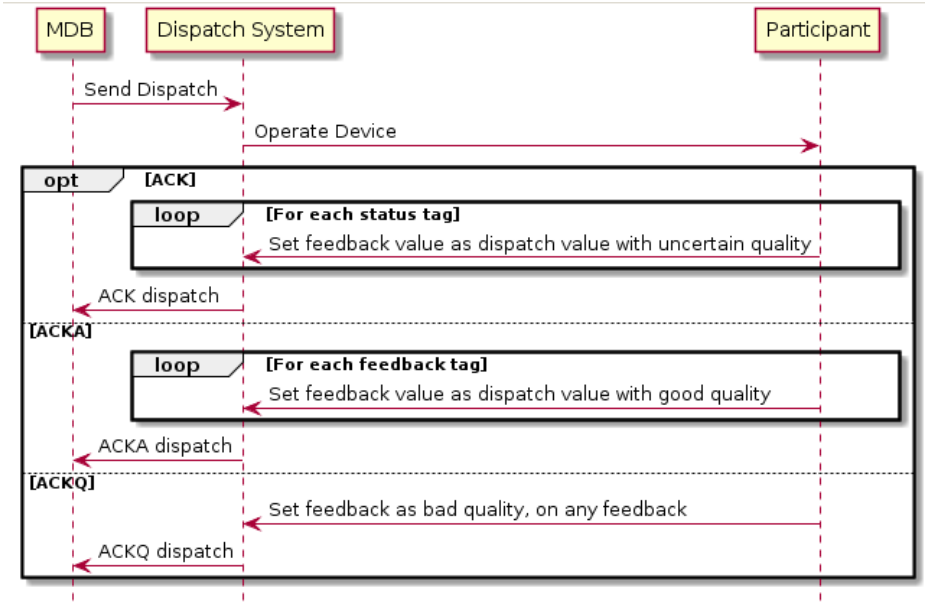



*Figure 8 – Block 5 (Setpoint equipment) Acknowledgement Process*

**Block 5 Acknowledgement Process**

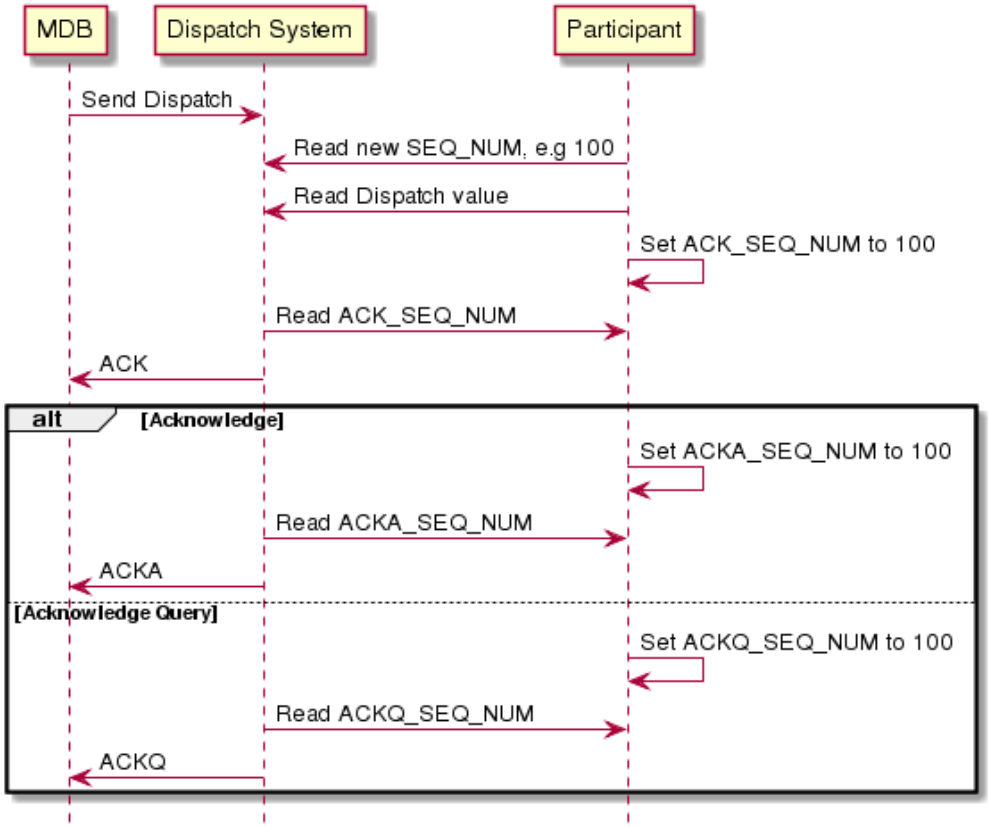
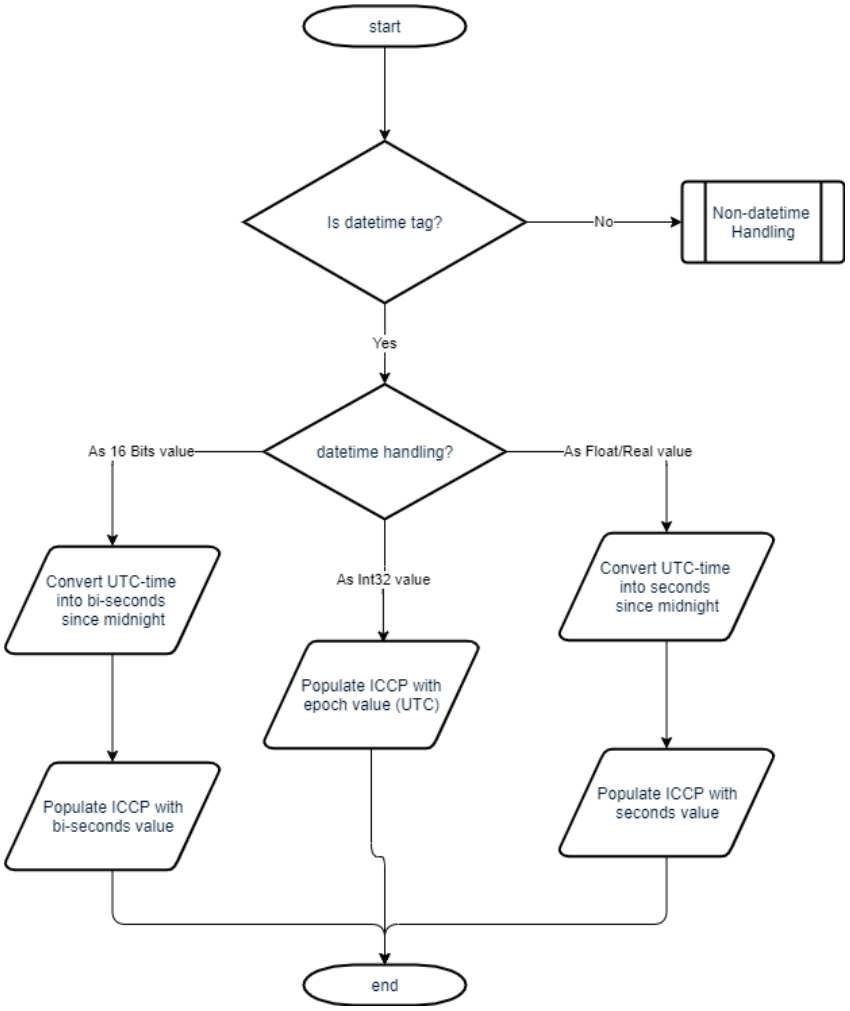



*Figure 9 – Block 2 Acknowledgement Process*

### 4.6.2 Datetime handling

There will be different configuration required based on handling of datetime tags in participant's system.

- Participant supporting Int32, will be send epoch time.
- Participant supporting Float32, will be send seconds since midnight. Field time on the tag will be used as time associated with the field in Participant's system.
- Participant who support 16 bits will be send bi-seconds since midnight i.e. seconds value at the granularity of 2 seconds. Again, field time on the tag will be used as time associated with the field in Participant's system.




### 4.6.3 Analog Value Scaling

ICCP transmits analog values as 32-bit IEEE floating point values (IEEE 754). Therefore, the actual engineering value can be sent over ICCP, and there is no need for scaling. The values sent and received over the Transpower ICCP link shall be the engineering values, sent as 32-bit IEEE floating point values. The values sent and received shall be as per the units defined in the following table:

*Table 7 – Data points Value Scaling*

| Measurement | Type | Units | Data Type | Value of 1.0 equals | Normal Resolution |
|---|---|---|---|---|---|
| | ***i*COM ICCP for Dispatch Service** | | | | |
| Energy | MW | MW | REAL32-Q-T (RealQTimeTag) | 1.0 MW | 0.01 MW (i.e. two decimal places) |
| RESF (Fast Reserve) | RESF | MW | REAL32-Q-T | 1.0 MW | 0.01 MW (i.e. two decimal places) |
| RESS (Sustained Reserve) | RESS | MW | REAL32-Q-T | 1.0 MW | 0.01 MW (i.e. two decimal places) |
| IL - Fast Interruptible Load | INTF | MW | REAL32-Q-T | 1.0 MW | 0.01 MW (i.e. two decimal places) |

| Measurement | Type | Units | Data Type | Value of 1.0 equals | Normal Resolution |
|---|---|---|---|---|---|
| IL - Sustained Interruptible Load | INTS | MW | REAL32-Q-T | 1.0 MW | 0.01 MW (i.e. two decimal places) |
| Voltage | VOLT | kV | REAL32-Q-T | 1.0 kV | 0.01 kV (i.e. two decimal places) |
| Frequency (Band +/-) | FREQ | MW | REAL32-Q-T | 1 MW | (i.e. no decimal places) |
| MVAR | MVAR | Mvar | REAL32-Q-T | 1 Mvar / -1 Mvar | 1 Mvar / -1 Mvar (i.e. no decimal places) |
| Dispatchable Demand | DD | MW | REAL32-Q-T | 1.0 MW (to consume) | 0.01 MW (i.e. two decimal places) |
| Energy (Dispatch Notified) | MWN | MW | REAL32-Q-T | 1.0 MW | 0.01 MW (i.e. two decimal places) |
| Dispatchable Demand (Dispatch Notified) | DDN | MW | REAL32-Q-T | 1.0 MW (to consume) | 0.01 MW (i.e. two decimal places) |
| Sequence Number | SEQNUM | | INT32-Q-T | 1 | 1 (no decimal places) |
| ACK_SEQNUM | ACK | | INT32-Q-T | 1 | 1 (no decimal places) |
| ACKA_SEQNUM | ACKA | | INT32-Q-T | 1 | 1 (no decimal places) |
| ACKQ_SEQNUM | ACKQ | | INT32-Q-T | 1 | 1 (no decimal places) |
| ACKR_SEQNUM | ACKR | | INT32-Q-T | 1 | 1 (no decimal places) |
| Heartbeat | | Heart Beat | REAL32-Q-T | 1.0 | 1.0000 (four decimal points) |
| | **Secondary type Dispatch (*i*COM ICCP)** | | | | |
| Frequency Start Time[22] | FST | DATETIME | INT32-Q-T | **BLOCK 2 format**: 1.0 Seconds since 00:00:00 Coordinated Universal Time (UTC), Thursday, 1 January 1970 – aka epochtime | **BLOCK 2 format:** e.g. 1541541246.0<br><br>Means<br><br>**Human time (GMT):** Tuesday, November 6, 2018 9:54:06 PM<br><br>**Human time (NZ time zone):** Wednesday, November 7, 2018 10:54:06 AM GMT+13:00 |
| Frequency Start Time (current) | FST_C | DATETIME | INT32-Q-T | aka epochtime (same as above) | e.g. 1541541246.0 |
| Ramp Up | RUP | MW | REAL32-Q-T | 1.0 MW | 0.01 MW (i.e. two decimal places) |
| Ramp Down | RDN | MW | REAL32-Q-T | 1.0 MW | 0.01 MW (i.e. two decimal places) |
| Ramp Up Broken | RUPB | MW | REAL32-Q-T | 1.0 MW | 0.01 MW (i.e. two decimal places) |

[22] Note this only applies to non MFK date/time values for Block 2 only; the format of the frequency start time in MFK instructions is not changing.

| Measurement | Type | Units | Data Type | Value of 1.0 equals | Normal Resolution |
|---|---|---|---|---|---|
| Ramp Down Broken | RDNB | MW | REAL32-Q-T | 1.0 MW | 0.01 MW (i.e. two decimal places) |
| Fast Risk | FRK | MW | REAL32-Q-T | 1.0 MW | 0.01 MW (i.e. two decimal places) |
| Sustained Risk | SRK | MW | REAL32-Q-T | 1.0 MW | 0.01 MW (i.e. two decimal places) |
| IG Constraint | IG | Flag | STATUS-Q-T | 1 or 0 | (i.e. no decimal places) |
| Grid Constraint | GC | Flag | STATUS-Q-T | 1 or 0 | (i.e. no decimal places) |
| Non-Dispatchable | ND | Flag | STATUS-Q-T | 1 or 0 | (i.e. no decimal places) |

Note that, where appropriate, other units may be used, but only by mutual agreement between Transpower and the Participant.

Data type ICCP data items will be defined as type "RealQTimeTag" which is defined as a 4-byte floating point number along with quality flag ("Q") and timestamp. Data type StatusQTimeTag will be used for status data.

The "Q" on start-up will be defaulted to bad and when an updated is received from the MS then it will be set good. If no update is received for a dispatch node then the associated data values will all be set to bad.

### 4.6.4 Analog Value Direction (Sign)

ICCP has no native convention for direction of power flow. For those values where the sign indicates the direction of power flow (e.g. MVAR), the sign on the value sent over ICCP shall comply with the following diagram.

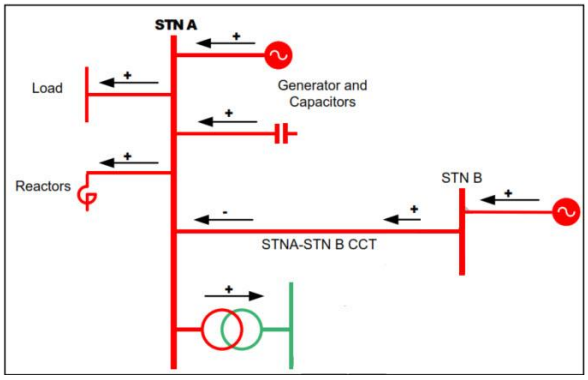



*Figure 10 – Analog Value Direction (Sign) Conventions*

### 4.6.5 Data Sets

Data sets for Dispatch Service are groups of data items (status points, analogs), used to split the list of data items into manageable pieces. It is not necessary to co-ordinate the data sets between the two ends of an ICCP link, as this is carried out automatically by the ICCP protocol. When configuring data sets, the number of data items in each set is limited by the maximum MMS PDU size, and it is recommended that data sets be limited to a maximum of 800 data items, and to:

- Keep the data item names as short as possible;
- Maximise the number of data items using VCC scope (as these only require the exchange of the data item name, and not the domain name).

Transpower will normally define a single data set for each Dispatch Endpoint, and this data set will contain all data items from that Participant. Note that a Participant connection is not limited to a single data set, and additional data sets will be defined if required.

### 4.6.6 Data Attributes

**Block 2:**

ICCP data items can include several data attributes to provide additional information:

- Quality (refer to Section 4.6.7);
- Timestamp;
- Seconds resolution on timestamp (ICCP 2000-08 only);
- COV (multiple change indication).

The Transpower ICCP implementation will only use the State and Real data items, and the attributes enabled on each type shall be as follows:

*Table 8 – Data Attributes (Block 2)*

| Data Item | Quality | Timestamp | Milliseconds | Network Type (DataType) |
|---|---|---|---|---|
| State | Y | Y | By Agreement | STATUS-Q-T |
| Real | Y | Y | N | REAL32-Q-T (RealQTimeTag) |
| Sequence number | Y | Y | N | INT32-Q-T |
| Heartbeat | Y | Y | N | REAL32-Q-T |
| Timestamp | Y | Y | N | INT32-Q-T |
| ACK | Y | Y | N | INT32-Q-T |
| ACKA | Y | Y | N | INT32-Q-T |
| ACKQ | Y | Y | N | INT32-Q-T |
| ACKR | Y | Y | N | INT32-Q-T |

### 4.6.7 Data Item Quality Information

ICCP provides quality codes to indicate the data quality, and these are transferred with each data item. The quality codes are derived from the SCADA system's reliability information for the status, analog and accumulator points stored in its database. Note that the mapping of a SCADA system's data reliability information into the ICCP quality codes is a local implementation issue, as each SCADA system has its own symbols for displaying data quality.

The ICCP data item quality information provided is as follows:

- Data item status or validity. This attribute is the individual data item's quality, and it can have one of the following states:
  - Valid (Good);
  - Invalid (Bad);
  - Held (Point has been out of scan or otherwise removed from service. The value may be the last good value obtained or a value that has been substituted by the Operator);

  - Suspect (Old value due to telemetry failure or failure to acquire a new value within the scan time, or otherwise considered suspect by the data owner);
- Data item source. This attribute indicates where a data item's value is sourced from (which can change at any time), and can have any one of the following states:
  - EnergyReserve (MW, RESF, RESS, DD);
  - Frequency (FREQ);
  - Interruptible Load (INTF, INTS);
  - Voltage (VOLT, MVAR);
  - DNx (MWN, DDN);
  - Status data such as IG (IG Constraint) & BSC (Block Constraint)

### 4.6.8 Data Item Naming

All data items (status points, analogs, controls and setpoints) transferred over ICCP require a unique name. The ICCP data item name is sent over the ICCP link, therefore the ICCP data item name must be identical at each end. ICCP data item names must be in the format specified by the ICCP protocol.

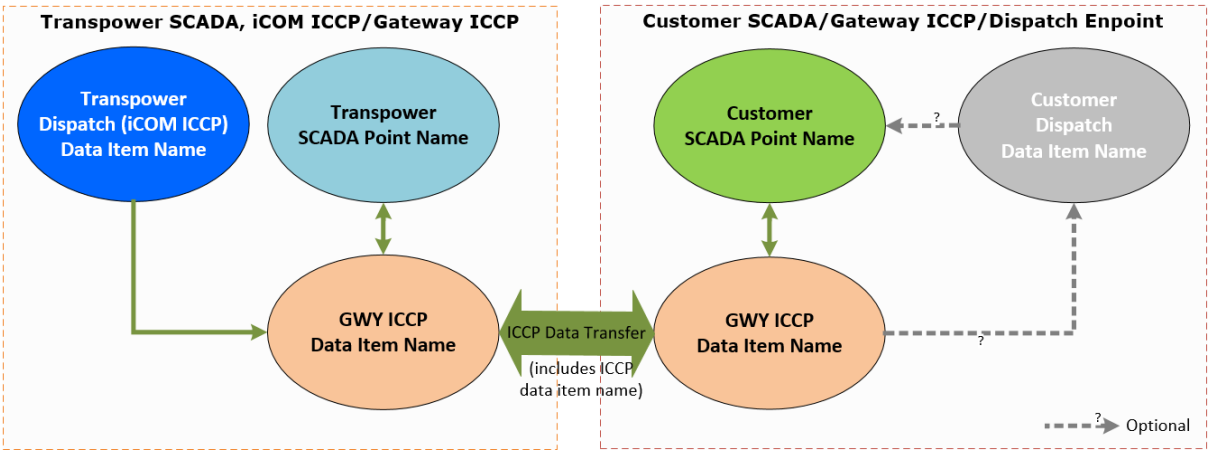



*Figure 11 – ICCP Data Item Name Alignment*

The rules for ICCP data item names are as follows:

- Upper case letters (A…Z), the ten digits (0…9), and underscore;
- No embedded blanks;
- Must begin with a letter;
- Maximum length of 32 characters (for performance, the shorter an ICCP point name the better).

Note: For performance, it is better to keep the length of the data item names shorter. ICCP names are exchanged over the network when data is sent by exception, which is the most common way of transferring data.

The ICCP data item names to be used will be assigned by Transpower, and will be as follows:

- For data points that already exist in National SCADA, the ICCP data item names to the Participant will be the existing ICCP point names;
- For data points received from a Participant that already exist in National SCADA (i.e. where and RTU-RTU link is being replaced with ICCP), the ICCP data item names will be the existing SCADA point names;
- For data points received from Dispatch Data via iCOM ICCP, the ICCP data item name to the Participant shall be in the form of SITEID_DISPATCHGROUP_NODE_DISPATCHTYPE_VALUE TYPE.
- The underscore character (_) shall only be used for delineating between the fields. It shall not be used to separate parts of the name in the device identifier (e.g. BUS A shall be BUSA and not BUS_A);
- While the dollar sign ($) is a permitted delimiter in ICCP, it shall not be used in data item names;

- While lower case letters are permitted in the data item name by ICCP, in Transpower's ICCP system these are automatically switched to upper case letters, therefore, to avoid subsequent confusion, only upper-case letters shall be used;

The data item naming conventions are illustrated in the following examples.

## Transpower Market System Dispatch (*i*COM) ICCP to third party

### Example 1: Energy dispatch (Block 2)

*Table 9 – Energy Dispatch (Block 2)*

| Endpoint Name | Node | Dispatch Group | Dispatch Type | Value Type | Direction | Tag Name |
|---|---|---|---|---|---|---|
| CTY | | EnergyReserve | | RESEND | OUT | CTY_ENERGYRESERVE_RESEND |
| CTY | | EnergyReserve | | RETRYC | OUT | CTY_ENERGYRESERVE_RETRYC |
| CTY | | EnergyReserve | | SEQNUM | OUT | CTY_ENERGYRESERVE_SEQNUM |
| CTY | | EnergyReserve | | ACK | IN | CTY_ENERGYRESERVE_SEQNUM_ACK |
| CTY | | EnergyReserve | | ACKA | IN | CTY_ENERGYRESERVE_SEQNUM_ACKA |
| CTY | | EnergyReserve | | ACKQ | IN | CTY_ENERGYRESERVE_SEQNUM_ACKQ |
| CTY | CTY2201 CTY1 | EnergyReserve | MW | DTIME | OUT | CTY_CTY2201_CTY1_MW_DTIME |
| CTY | CTY2201 CTY1 | EnergyReserve | MW | FRK | OUT | CTY_CTY2201_CTY1_MW_FRK |
| CTY | CTY2201 CTY1 | EnergyReserve | MW | GC | OUT | CTY_CTY2201_CTY1_MW_GC |
| CTY | CTY2201 CTY1 | EnergyReserve | MW | IG | OUT | CTY_CTY2201_CTY1_MW_IG |
| CTY | CTY2201 CTY1 | EnergyReserve | MW | RDN | OUT | CTY_CTY2201_CTY1_MW_RDN |
| CTY | CTY2201 CTY1 | EnergyReserve | MW | RDNB | OUT | CTY_CTY2201_CTY1_MW_RDNB |
| CTY | CTY2201 CTY1 | EnergyReserve | MW | RUP | OUT | CTY_CTY2201_CTY1_MW_RUP |
| CTY | CTY2201 CTY1 | EnergyReserve | MW | RUPB | OUT | CTY_CTY2201_CTY1_MW_RUPB |
| CTY | CTY2201 CTY1 | EnergyReserve | MW | SRK | OUT | CTY_CTY2201_CTY1_MW_SRK |
| CTY | CTY2201 CTY1 | EnergyReserve | MW | VALUE | OUT | CTY_CTY2201_CTY1_MW_VALUE |
| CTY | CTY2201 CTY1 | EnergyReserve | RESF | DTIME | OUT | CTY_CTY2201_CTY1_RESF_DTIME |
| CTY | CTY2201 CTY1 | EnergyReserve | RESF | VALUE | OUT | CTY_CTY2201_CTY1_RESF_VALUE |
| CTY | CTY2201 CTY1 | EnergyReserve | RESS | DTIME | OUT | CTY_CTY2201_CTY1_RESS_DTIME |

| Endpoint Name | Node | Dispatch Group | Dispatch Type | Value Type | Direction | Tag Name |
|---|---|---|---|---|---|---|
| CTY | CTY2201 CTY1 | EnergyReserve | RESS | VALUE | OUT | CTY_CTY2201_CTY1_RESS_VALUE |

**Example 2: Frequency (Block 2)**

*Table 10 – Frequency (Block 2)*

| Endpoint Name | Node | Dispatch Group | Dispatch Type | Value Type | Direction | Tag Name |
|---|---|---|---|---|---|---|
| CTY | | Frequency | | RESEND | OUT | CTY_FREQUENCY_RESEND |
| CTY | | Frequency | | RETRYC | OUT | CTY_FREQUENCY_RETRYC |
| CTY | | Frequency | | SEQNUM | OUT | CTY_FREQUENCY_SEQNUM |
| CTY | | Frequency | | ACK | IN | CTY_FREQUENCY_SEQNUM_ACK |
| CTY | | Frequency | | ACKA | IN | CTY_FREQUENCY_SEQNUM_ACKA |
| CTY | | Frequency | | ACKQ | IN | CTY_FREQUENCY_SEQNUM_ACKQ |
| CTY | CTY2201 CTY1 | Frequency | FREQ | FST | OUT | CTY_CTY2201_CTY1_FREQ_FST |
| CTY | CTY2201 CTY1 | Frequency | FREQ | FST | OUT | CTY_CTY2201_CTY1_FREQ_FST_C |
| CTY | CTY2201 CTY1 | Frequency | FREQ | VALUE | OUT | CTY_CTY2201_CTY1_FREQ_VALUE |
| CTY | CTY2201 CTY1 | Frequency | FREQ | VALUE | OUT | CTY_CTY2201_CTY1_FREQ_VALUE_C |

**Example 3: Voltage (Block 2)**

*Table 11 – Voltage (Block 2)*

| Endpoint Name | Node | Dispatch Group | Dispatch Type | Value Type | Direction | Tag Name |
|---|---|---|---|---|---|---|
| CTY | CTY | Voltage | MVAR | DTIME | OUT | CTY_CTY_MVAR_DTIME |
| CTY | CTY | Voltage | MVAR | VALUE | OUT | CTY_CTY_MVAR_VALUE |
| CTY | CTY | Voltage | VOLT | DTIME | OUT | CTY_CTY_VOLT_DTIME |
| CTY | CTY | Voltage | VOLT | VALUE | OUT | CTY_CTY_VOLT_VALUE |
| CTY | | Voltage | | RESEND | OUT | CTY_VOLTAGE_RESEND |
| CTY | | Voltage | | RETRYC | OUT | CTY_VOLTAGE_RETRYC |
| CTY | | Voltage | | SEQNUM | OUT | CTY_VOLTAGE_SEQNUM |
| CTY | | Voltage | | ACK | IN | CTY_VOLTAGE_SEQNUM_ACK |
| CTY | | Voltage | | ACKA | IN | CTY_VOLTAGE_SEQNUM_ACKA |
| CTY | | Voltage | | ACKQ | IN | CTY_VOLTAGE_SEQNUM_ACKQ |

See Appendix 2 for some detailed examples of what to expect for ICCP tag names.

### 4.6.9 Data Items to be Exchanged

The data items to be exchanged between Transpower and the Participant over ICCP will include:

- The heartbeat analog values (used for diagnostic purposes);
- Dispatch data such as Energy, Frequency, Interruptible Load, Voltage;
- Dispatch Acknowledgements;
- Dispatch status data (flag) for IG Constraint, BSC Constraint and Non-Dispatchable flag.

### 4.6.10 ICCP Association Information Exchange

As a guideline the following data is available for exchanges between Transpower (as the System Operator) and participants for the purpose of dispatch:

- **Data contained in the dispatch instructions issued by Transpower to participants**
- Data contained in the dispatch instructions acknowledgement sent from participants to Transpower

There is some flexibility with some of the items in this table, and the table entries are to be mutually agreed between Transpower and the Participant. Each Transpower – Participant ICCP connection may have to be configured slightly differently, as there may be constraints with some ICCP implementations that necessitate this. This will likely affect the following table entries in particular:

- ICCP Version. The Transpower gateway supports ICCP version 2000-08 (preferred). Note that both ends of an ICCP link must be configured to use the same version. There is no interoperability between versions;
- The various time settings, retries and PDU size may need to be negotiated in order to get a stable connection;
- Some systems impose constraints on the AP Title, AE Qualifier, Presentation Selector (PSEL), Session Selector (SSEL), and Transport Selector (TSEL) entries.

The values in the table below are indicative for information only, and the actual values to be used will be agreed with each Participant at the time of implementation.

*Table 12 – ICCP Association Information Exchange*

| Description | Required | Transpower Detail | Participant Detail | Notes |
|---|---|---|---|---|
| **Company-wide/Server independent information (Gateway ICCP)** | | | | |
| ICCP vendor and platform | M | GE (Alstom Grid) e-terraComm Version 5.9 | | The name of the ICCP vendor, vendor software version. The operating system and hardware platform used for the ICCP servers. |
| Number of possible ICCP servers | R | 8 in total<br>4 Staging<br>4 Realtime | | This is the total number of ICCP servers that may be available to associate with. Include backup servers if they have unique addresses. |
| Domain Name | M | TPNZ | | This is the domain name which is used to access data on the other party's ICCP node. It is to be the 3-character Participant abbreviation as described in Section 4.6.8. |
| Bilateral Table ID | M | BLT1 | | The bilateral table name used when accessing the other party's data. |
| **Gateway ICCP Server specific information** | | | | |
| Server Name | O | RLTICPG1-4 for production servers full names are; NDCSCS-RLTICPG1, NDCSCS-RLTICPG2, SDCSCS-RLTICPG3 & SDCSCS-RLTICPG4 | | The name by which the servers is referred to. This field is not electronically transmitted during any ICCP transactions but is here to facilitate verbal communication between the parties. |
| Server number | R | Four production servers, two NDC and two in SDC. Any one of them may be Active Enabled, Second server in the same data centre will be Active Standby and other tow servers will take Remote Preferred and Remote Standby role. | | If this is the Active Enabled server<br>if this is the Active Standby Server<br>if this is the Remote Preferred Server<br>if this is the Remote Standby Server |
| IP network address | R | Will be advised at the time on implementation | | The IP address for this ICCP server. |

| Description | Required | Transpower Detail | Participant Detail | Notes |
|---|---|---|---|---|
| AP Title | R | Realtime<br>OAGICCP 1 1 1 1<br>OAGICCP2 2 1 1 1<br>OAGICCP3 1 1 1 1<br>Staging<br>OAGICCP 1 1 1 4<br>OAGICCP2 2 1 1 4<br>OAGICCP3 1 1 1 4 | | Optional object identifier representing the Application Process Title given to this application. |
| AE Qualifier | O | 1 | | A long integer (32 bit signed) used to qualify the application entity. |
| Presentation Selector (PSEL) | R | 00 00 00 01 | | 2 or 4-byte number used to select the correct instance of the presentation layer. Typically, 00 09 or 00 00 00 09. |
| Session Selector (SSEL) | R | 00 01 | | 2 or 4-byte number used to select the correct instance of the session layer. Typically, 00 09 or 00 00 00 09. |
| Transport Selector (TSEL) | R | 00 01 | | 2 or 4-byte number used to select the correct instance of the session layer. Typically, 00 09 or 00 00 00 09. |
| Association Type | R | Dual direction Client-Server (where supported by Participant's ICCP implementation) | Dual direction Client-Server (where supported by Participant's ICCP implementation) | Typically, "Dual direction Client-Server", where the ICCP nodes act as Client and Server over one association and information can be sent in either direction per association. The alternative is "Single direction Client-Server", if the ICCP nodes act as either Client or Server over one association, and information can be sent in only one direction per association. |
| Association Initiation | R | Transpower | Transpower | Used if the association type is "Dual direction Client-Server" and indicates which ICCP node shall initiate the association. |

# Appendix 1: Dispatch Web Service Specification (OpenAPI[23] 2.0)

A draft version of the OpenAPI 2.0 specification of the web service API follows. This is for casual inspection only and may not represent the most current version of the web service specification; the formal version can be found as a standalone file named swagger.yml in the integration pack distribution.

Note that this specification is most conveniently viewed by pasting the specification below into http://editor.swagger.io or your favourite swagger editor environment.

---------- swagger.yml ----------

```
swagger: '2.0'
info:
  description: >-
    This API provides a Web Services channel for Market participants to receive Market System
dispatch instructions and acknowledge them.  These APIs can only be used following an enrolment
with the System Operator.  Part of enrolment includes modelling the dispatch endpoints the
participant wishes to utilize, the Dispatch Groups/Products the participant wishes to
'subscribe' to, to dispatch to those endpoints and the channels over which those products are
dispatched.  This API covers dispatching to enrolled endpoints/products over the Web Services
channel. Refer to the Integration Pack for further information.
  version: 1.1.0
  title: Market Dispatch Web Service
  license:
    name: © Transpower 2018
    url: 'https://www.transpower.co.nz/'
host: 'localhost:1024'
basePath: /
tags:
  - name: Acknowledgement
    description: Operations for acknowledgement of dispatch instructions
  - name: Dispatch Instructions
    description: Operations to get dispatch instructions
paths:
  '/api/v1/dispatch/endpoints/{endpoint}/acknowledgements':
    put:
      tags:
        - Acknowledgement
      summary: Acknowledge a dispatch instruction
      description: This operation allows a participant to acknowledge one or more dispatch
instructions.  An acknowledgement indicates the corresponding dispatch instruction through
the combination of dispatch group (product) and sequence number. An Acknowledgement can be
an ACK, ACKA or ACKQ corresponding to a Receipt Acknowledgment an Intention to Comply with
Dispatch or Querying the Dispatch, respectively. Note that as an additional traceability
mechanism, the Correlation-Id header parameter must be set to the correlationId of the
corresponding dispatch instruction.
      consumes:
        - application/json
      parameters:
        - name: Correlation-Id
          in: header
          description: This must be set to the value of the correlationId of the corresponding
dispatch instruction
          type: string
          required: true
        - name: endpoint
          in: path
          description: The dispatch endpoint for which the dispatch instruction was sent
          required: true
          type: string
```

[23] Formerly, and often still informally, known of as, "Swagger".

```
        - in: body
          name: acknowledgement
          description: >-
            The representation of an Acknowledgement corresponding to the dispatch instruction
being acknowledged.
          required: true
          schema:
            $ref: '#/definitions/Acknowledgement'
      responses:
        200:
          description: OK
        400:
          description: Bad Request
        401:
          description: Unauthorized
        403:
          description: Forbidden
        405:
          description: Method Not Allowed
        500:
          description: Internal Server Error
      deprecated: false
  '/api/v1/dispatch/endpoints/{endpoint}/instructions':
    get:
      tags:
        - Dispatch Instructions
      summary: >-
        Get dispatch instructions for a given endpoint as an event stream.
      description: >-
        This operation establishes a 'long polling' connection with the server and receives
dispatch instructions as an event stream.  On first connection, the server will send an array
of the most recent dispatch instructions. For Dispatch Groups, other than Voltage, at most
one instruction will be sent, whether or not it has been acknowledged. For Voltage the
situation is different and all unacknowledged instructions are sent. After that the server
will send new instructions as they arrive. This means that, in practice, most of the time the
client will receive an array with one dispatch instruction although the client should not
constrain itself to expect this behaviour. Each array of Dispatch Instructions is sent as the
value of the 'data' property of the event stream.  The client needs only to subscribe to
event 'dispatch' to receive the dispatch instruction stream.
        <p>Refer         to         https://developer.mozilla.org/en-US/docs/Web/API/Server-
sent_events/Using_server-sent_events#Event_stream_format for further information on the event
stream format.
      produces:
        - text/event-stream
      parameters:
        - name: endpoint
          in: path
          description: The dispatch endpoint the Participant wants to receive dispatch
instructions for
          required: true
          type: string
      responses:
        200:
          description: Array of dispatch instructions, at most one per dispatch group (except
possibly for Voltage)
          schema:
            type: array
            items:
              $ref: '#/definitions/DispatchInstruction'
        400:
          description: Bad Request
        401:
          description: Unauthorized
        403:
          description: Forbidden
```

```
        405:
          description: Method Not Allowed
        500:
          description: Internal Server Error
      deprecated: false
  '/api/v1/dispatch/endpoints/{endpoint}/instructions/latest':
    get:
      tags:
        - Dispatch Instructions
      summary: >-
        Get the latest dispatch instructions for a given endpoint using request/response
interaction style.
      description: >-
        This operation gets, from the server, an array of the most recent dispatch
instructions.  For Dispatch Groups, other than Voltage, at most one instruction will be
included, whether or not it has been acknowledged. For Voltage the situation is different and
all unacknowledged instructions are included.
      produces:
        - application/json
      parameters:
        - name: endpoint
          in: path
          description: The dispatch endpoint the Participant wants to receive dispatch
instructions for
          required: true
          type: string
      responses:
        200:
          description: Array of dispatch instructions, at most one per dispatch group (except
possibly for Voltage)
          schema:
            type: array
            items:
              $ref: '#/definitions/DispatchInstruction'
        400:
          description: Bad Request
        401:
          description: Unauthorized
        403:
          description: Forbidden
        405:
          description: Method Not Allowed
        500:
          description: Internal Server Error
      deprecated: false
  '/api/v1/dispatch/endpoints/{endpoint}/instructions/products/{product}/latest':
    get:
      tags:
        - Dispatch Instructions
      summary: Get latest dispatch instruction for given endpoint and product/dispatch group.
      description: >-
        This operation gets, from the server, the most recent dispatch instruction(s), for
the given product (dispatch group). For Dispatch Groups, other than Voltage, at most one
instruction will be included, whether or not it has been acknowledged. For Voltage the
situation is different and all unacknowledged instructions are included.  If the product is
not subscribed a 404 Not Found error is returned; if there are no current instructions for
the given product a 200 OK is returned with an empty body.
      produces:
        - application/json
      parameters:
        - name: endpoint
          in: path
          description: The dispatch endpoint the Participant wants to receive dispatch
instructions for
          required: true
```

```
          type: string
        - name: product
          in: path
          description: The product/dispatch group the Participant wants to receive dispatch
instructions for
          required: true
          type: string
          enum:
            - energy
            - frequency
            - interruptibleload
            - voltage
      responses:
        200:
          description: Array of dispatch instructions, at most one per dispatch group (except
possibly for Voltage)
          schema:
            type: array
            items:
              $ref: '#/definitions/DispatchInstruction'
        400:
          description: Bad Request
        401:
          description: Unauthorized
        403:
          description: Forbidden
        404:
          description: Not Found; No current subscription found for given product for this
endpoint
        405:
          description: Method Not Allowed
        500:
          description: Internal Server Error
      deprecated: false

definitions:
  DispatchGroupName:
    description: >-
      Dispatch group name, aka product group name.
    type: string
    example: energy
    enum:
      - energy
      - frequency
      - interruptibleload
      - voltage
      - dnx

  Acknowledgement:
    description: >-
      Representation of a Participant acknowledgement for a given dispatch instruction. Note
that the dispatchGroupName and sequenceNumber are required to uniquely identify the
corresponding dispatch instruction.
    type: object
    properties:
      ackType:
        type: string
        description: >-
          The type of Acknowledgement. ACK, ACKA, ACKQ, ACKR corresponds to Receipt
Acknowledgement, Intention to Comply with Dispatch and Acknowledged but Queried, respectively.
        example: ACKA
        enum:
          - ACK
          - ACKA
          - ACKQ
```

```
          - ACKR
      dispatchGroupName:
        $ref: '#/definitions/DispatchGroupName'
      sequenceNumber:
        type: integer
        format: int32
        description: Sequence number of the dispatch instruction being acknowledged
    required:
      - ackType
      - dispatchGroupName
      - sequenceNumber

  DispatchInstruction:
    description: >-
      A dispatch instruction holds the current dispatched values for a given dispatch endpoint
and product group. Note that ONLY one of the properties: energyDispatch, frequencyDispatch,
interruptibleLoadDispatch or voltageDispatch will be present corresponding to the value
specified for dispatchGroupName
    type: object
    required:
      - correlationId
      - dispatchGroupName
      - sequenceNumber
      - dispatchEndpointOwner
      - isUserResend
      - dispatchEndpointName
    properties:
      correlationId:
        type: string
        description: The unique correlation id of this dispatch instruction.
        example: '889ffe5f-ab14-421a-a83f-38a9598ff61d'
      dispatchGroupName:
        $ref: '#/definitions/DispatchGroupName'
      sequenceNumber:
        type: integer
        format: int32
        description: >-
          The sequence number is a monotonically increasing value (incrementing by 1) that
is unique for each dispatch endpoint and dispatchGroupName combination. i.e. each subscribed
dispatchGroupName for a given endpoint has its own sequence number. The sequence number and
dispatchGroupName must be included within the dispatch Participant's acknowledgement message
to uniquely resolve the dispatch instruction.
        example: 299792458
      dispatchEndpointOwner:
        type: string
        example: ACME
        description: Owner of the dispatch endpoint. This is not necessarily the same as the
traderCode defined at the node/block level
      dispatchEndpointName:
        type: string
        example: acme-endpoint-2
        description: The unique dispatch endpoint name for the participant
      isUserResend:
        type: boolean
        description: >-
          Indicator whether this instruction has been resent by an Energy Coordinator (or
other user) or not. Note that a 'resend' is an explicit action performed by an Energy
Coordinator if, for example, the participant hasn't responded with a business acknowledgement
or some other such situation.  Note that a resend is treated as a new dispatch instruction
and therefore acquires a new sequence number even though the product content is otherwise the
same.  This is different from a system level redelivery attempt (prior to the Market System
having received an ACK) where it may retry delivery of the same instruction, with the same
sequence number.
        example: false
        default: false
```

```
      messageSentTime:
        type: string
        format: date-time
        description: >-
          The (ISO 8601 with zone indicator format) date/time dispatch instruction left the
Transpower WS
        example: '2019-06-20T15:49:00.123Z'
      # ONLY one of the following will be present, corresponding to the dispatchGroupName for
this dispatch instruction
      energyDispatch:
        $ref: '#/definitions/EnergyDispatch'
      frequencyDispatch:
        $ref: '#/definitions/FrequencyDispatch'
      interruptibleLoadDispatch:
        $ref: '#/definitions/InterruptibleLoadDispatch'
      voltageDispatch:
        $ref: '#/definitions/VoltageDispatch'
      energyNotificationDispatch:
        $ref: '#/definitions/EnergyNotificationDispatch'

    example:
      {
        "correlationId": "889ffe5f-ab14-421a-a83f-38a9598ff61d",
        "dispatchGroupName": "energy",
        "sequenceNumber": 299792458,
        "dispatchEndpointOwner": "ACME",
        "dispatchEndpointName": "acme-endpoint-2",
        "isUserResend": false,
        "energyDispatch": {
          "blocks": [
          {
            "name": "ACME_BLK",
            "traderCode": "ACME",
            "primaryValues": [
            {
              "dispatchType": "MW",
              "dispatchValue": 150,
              "dispatchTime": "2018-10-16T15:49:00.123Z",
              "dispatchIssueTime": "2018-10-16T15:49:00.123Z",
              "secondaryRealValues": [
              {
                "dispatchType": "RUP",
                "dispatchValue": 30
              }
              ],
              "secondaryConstraintValues": [
              {
                "dispatchType": "IG",
                "dispatchValue": true
              }
              ]
            },
            {
              "dispatchType": "RESF",
              "dispatchValue": 230.2,
              "dispatchTime": "2018-10-16T15:39:00.123Z",
              "dispatchIssueTime": "2018-10-16T15:39:00.123Z"
            },
            {
              "dispatchType": "RESS",
              "dispatchValue": 130.2,
              "dispatchTime": "2018-10-16T15:00:00.123Z",
              "dispatchIssueTime": "2018-10-16T15:00:00.123Z"
            }
            ],
```

```
            "nodes": [
            {
              "name": "ACME_1",
              "traderCode": "ACME",
              "primaryValues": [
              {
                "dispatchType": "MW",
                "dispatchValue": 50,
                "dispatchTime": "2018-10-16T15:49:00.123Z",
                "dispatchIssueTime": "2018-10-16T15:49:00.123Z"
              }
              ]
            },
            {
              "name": "ACME_2",
              "traderCode": "ACME",
              "primaryValues": [
              {
                "dispatchType": "MW",
                "dispatchValue": 100,
                "dispatchTime": "2018-10-16T15:49:00.123Z",
                "dispatchIssueTime": "2018-10-16T15:49:00.123Z"
              }
              ]
            }
            ]
          }
          ],
          "nodes": [
          {
            "name": "ACME_3",
            "traderCode": "ACME",
            "primaryValues": [
            {
              "dispatchType": "MW",
              "dispatchValue": 65.2,
              "dispatchTime": "2018-10-16T15:49:00.123Z",
              "dispatchIssueTime": "2018-10-16T15:49:00.123Z"
            }
            ]
          }
          ]
        }
      }

  EnergyNotificationDispatch:
    type: object
    description: Representation of the dispatch of the Energy Notification dispatch/product
group
    properties:
      blocks:
        type: array
        description: Array of (0 or more) Blocks that Energy Notification is being dispatched
to
        items:
          $ref: '#/definitions/EnergyNotificationBlock'
      nodes:
        type: array
        description: Array of (0 or more) Blocks that Energy Notification is being dispatched
to
        items:
          $ref: '#/definitions/EnergyNotificationNode'

  EnergyNotificationNode:
    type: object
```

```
    description: Representation of the Energy Notification dispatch for a Node
    properties:
      name:
        type: string
        description: The unique name of the node
        example: OHAU_1234
      traderCode:
        type: string
        example: ACME
        description: Four letter code of the trader for this node
      primaryValues:
        type: array
        description: Primary dispatch type values
        items:
          $ref: '#/definitions/PrimaryEnergyNotificationValue'
    required:
      - name
      - traderCode
      - primaryValues

  EnergyNotificationBlock:
    type: object
    description: Representation of Energy Notification dispatch for a Block
    allOf:
      - type: object
        properties:
          nodes:
            type: array
            description: Array of 0 or more Nodes that comprise the Block
            items:
              $ref: '#/definitions/EnergyNotificationNode'
      - $ref: '#/definitions/EnergyNotificationNode'

  PrimaryEnergyNotificationValue:
    type: object
    description: Holds one primary Energy Notification value for a given dispatch node
    properties:
      dispatchType:
        type: string
        description: The type of primary Energy Notification value being dispatched.
        example: MWN
        enum:
          - MWN
          - DDN
      dispatchValue:
        type: number
        format: double
        description: The value of the type being dispatched
        example: 123.2
      dispatchTime:
        type: string
        format: date-time
        description: >-
          The (ISO 8601 with zone indicator format) date/time the dispatch was generated by
the Market System
        example: '2018-10-16T15:49:00.123Z'
      dispatchIssueTime:
        type: string
        format: date-time
        description: >-
          The (ISO 8601 with zone indicator format) date/time the dispatch was sent by the
Market Dispatch System to the Dispatch Participant
        example: '2018-10-16T15:49:00.123Z'
      secondaryRealValues:
        type: array
```

```
        description: Real valued Secondary dispatch type values. Applicable only for MWN
primary dispatch type
        items:
          $ref: '#/definitions/SecondaryEnergyNotificationRealValue'
      secondaryConstraintValues:
        type: array
        description: Constraint (boolean) dispatch type values. Applicable only for MWN
primary dispatch type
        items:
          $ref: '#/definitions/SecondaryEnergyConstraintValue'
    required:
      - dispatchType
      - dispatchValue
      - dispatchTime

  SecondaryEnergyNotificationRealValue:
    type: object
    description: Representation of a secondary dispatch double (floating point) value for a
Node
    properties:
      dispatchType:
        type: string
        description: The type of secondary dispatch value being dispatched. The rules defining
which secondaries may accompany a primary dispatch type are documented elsewhere, not
expressed in this model.
        example: RRUP
        enum:
          - RUP
          - RDN
          - RUPB
          - RDNB
      dispatchValue:
        type: number
        format: double
        description: The value of the type being dispatched
        example: 123.2
    required:
      - dispatchType
      - dispatchValue


  EnergyDispatch:
    type: object
    description: Representation of the dispatch of the Energy dispatch/product group
    properties:
      blocks:
        type: array
        description: Array of (0 or more) Blocks that Energy is being dispatched to
        items:
          $ref: '#/definitions/EnergyBlock'
      nodes:
        type: array
        description: Array of (0 or more) Blocks that Energy is being dispatched to
        items:
          $ref: '#/definitions/EnergyNode'

  EnergyBlock:
    type: object
    description: Representation of Energy dispatch for a Block
    allOf:
      - type: object
        properties:
          nodes:
            type: array
            description: Array of 0 or more Nodes that comprise the Block
```

```
            items:
              $ref: '#/definitions/EnergyNode'
      - $ref: '#/definitions/EnergyNode'

  EnergyNode:
    type: object
    description: Representation of the Energy dispatch for a Node
    properties:
      name:
        type: string
        description: The unique name of the node
        example: OHAU_1234
      traderCode:
        type: string
        example: ACME
        description: Four letter code of the trader for this node
      primaryValues:
        type: array
        description: Primary dispatch type values
        items:
          $ref: '#/definitions/PrimaryEnergyValue'
    required:
      - name
      - traderCode
      - primaryValues

  PrimaryEnergyValue:
    type: object
    description: Holds one primary Energy value for a given dispatch node
    properties:
      dispatchType:
        type: string
        description: The type of primary Energy value being dispatched.
        example: MW
        enum:
          - MW
          - RESS
          - RESF
          - DD
      dispatchValue:
        type: number
        format: double
        description: The value of the type being dispatched
        example: 123.2
      dispatchTime:
        type: string
        format: date-time
        description: >-
          The (ISO 8601 with zone indicator format) date/time the dispatch was generated by
the Market System
        example: '2018-10-16T15:49:00.123Z'
      dispatchIssueTime:
        type: string
        format: date-time
        description: >-
          The (ISO 8601 with zone indicator format) date/time the dispatch was sent by the
Market Dispatch System to the Dispatch Participant
        example: '2018-10-16T15:49:00.123Z'
      secondaryRealValues:
        type: array
        description: Real valued Secondary dispatch type values. Applicable only for MW
primary dispatch type
        items:
          $ref: '#/definitions/SecondaryEnergyRealValue'
      secondaryConstraintValues:
```

```
        type: array
        description: Constraint (boolean) dispatch type values. Applicable only for MW primary
dispatch type
        items:
          $ref: '#/definitions/SecondaryEnergyConstraintValue'
    required:
      - dispatchType
      - dispatchValue
      - dispatchTime
      - secondaryRealValues
      - secondaryConstraintValues

  SecondaryEnergyRealValue:
    type: object
    description: Representation of a secondary dispatch double (floating point) value for a
Node
    properties:
      dispatchType:
        type: string
        description: The type of secondary dispatch value being dispatched. The rules defining
which secondaries may accompany a primary dispatch type are documented elsewhere, not
expressed in this model.
        example: RRUP
        enum:
          - RUP
          - RDN
          - RUPB
          - RDNB
          - FRK
          - SRK
      dispatchValue:
        type: number
        format: double
        description: The value of the type being dispatched
        example: 123.2
    required:
      - dispatchType
      - dispatchValue

  SecondaryEnergyConstraintValue:
    type: object
    description: Representation of a secondary dispatch constraint (boolean) value for a Node
    properties:
      dispatchType:
        type: string
        description: The type of secondary dispatch (constraint) value being dispatched. The
rules defining which secondaries may accompany a primary dispatch type are documented
elsewhere, not expressed in this model. Note IG == Intermittent Generation, GC == Grid
Constraint, ND == Non Dispatchable
        example: IG
        enum:
          - IG
          - GC
          - ND
      dispatchValue:
        type: boolean
        description: The value of the type being dispatched
        example: true
    required:
      - dispatchType
      - dispatchValue

  FrequencyDispatch:
    type: object
    description: Representation of the dispatch of the Frequency dispatch/product group
```

```
    properties:
      blocks:
        type: array
        description: Array of (0 or more) Blocks that Frequency is being dispatched to
        items:
          $ref: '#/definitions/FrequencyNode'
      nodes:
        type: array
        description: Array of (0 or more) Nodes that Frequency is being dispatched to
        items:
          $ref: '#/definitions/FrequencyNode'

  FrequencyNode:
    type: object
    description: Representation of the Frequency dispatch for a Node or Block
    properties:
      name:
        type: string
        description: The unique name of the node
        example: OHAU_1234
      traderCode:
        type: string
        example: ACME
        description: Four letter code of the trader for this node
      primaryValues:
        type: array
        maxItems: 1
        description: Primary dispatch type values.  Note there is only ever one value for
Frequency.
        items:
          $ref: '#/definitions/PrimaryFrequencyValue'
    required:
      - name
      - traderCode
      - primaryValues

  PrimaryFrequencyValue:
    type: object
    description: Holds one primary Frequency value for a given dispatch node
    properties:
      dispatchType:
        type: string
        description: The type of primary value being dispatched.
        example: FREQ
        enum:
          - FREQ
      dispatchValue:
        type: number
        format: double
        description: The value of the type being dispatched
        example: 50.4
      dispatchTime:
        type: string
        format: date-time
        description: >-
          The (ISO 8601 with zone indicator format) date/time the dispatch was generated by
the Market System
        example: '2018-10-16T15:49:00.123Z'
      dispatchIssueTime:
        type: string
        format: date-time
        description: >-
          The (ISO 8601 with zone indicator format) date/time the dispatch was sent by the
Market Dispatch System to the Dispatch Participant
        example: '2018-10-16T15:49:00.123Z'
```

```
      secondaryDateTimeValues:
        type: array
        description: Date/Time valued Secondary dispatch type values
        items:
          $ref: '#/definitions/SecondaryFrequencyDateTimeValue'
    required:
      - dispatchType
      - dispatchValue
      - dispatchTime
      - secondaryDateTimeValues

  SecondaryFrequencyDateTimeValue:
    type: object
    description: Holds one secondary Frequency value for a given dispatch node
    properties:
      dispatchType:
        type: string
        description: The type of secondary value being dispatched.
        example: FST
        enum:
          - FST
      dispatchValue:
        type: string
        format: date-time
        description: >-
          The (ISO 8601 with zone indicator format) date/time value of the type being
dispatched
        example: '2018-10-16T15:49:00.123Z'
    required:
      - dispatchType
      - dispatchValue

  InterruptibleLoadDispatch:
    type: object
    description: Representation of the dispatch of the Interruptible Load dispatch/product
group
    properties:
      blocks:
        type: array
        description: Array of (0 or more) Blocks that Interruptible Load is being dispatched
to
        items:
          $ref: '#/definitions/InterruptibleLoadBlock'
      nodes:
        type: array
        description: Array of (0 or more) Nodes that Interruptible Load is being dispatched
to
        items:
          $ref: '#/definitions/InterruptibleLoadNode'

  InterruptibleLoadBlock:
    type: object
    description: Representation of Interruptible Load dispatch for a Block
    allOf:
      - type: object
        properties:
          nodes:
            type: array
            description: Array of 0 or more Nodes that comprise the Block
            items:
              $ref: '#/definitions/InterruptibleLoadNode'
      - $ref: '#/definitions/InterruptibleLoadNode'

  InterruptibleLoadNode:
    type: object
```

```
    description: Representation of the Interruptible Load dispatch for a Node
    properties:
      name:
        type: string
        description: The unique name of the node
        example: OHAU_1234
      traderCode:
        type: string
        example: ACME
        description: Four letter code of the trader for this node
      primaryValues:
        type: array
        description: Primary dispatch type values
        items:
          $ref: '#/definitions/InterruptibleLoadValue'
    required:
      - name
      - traderCode
      - primaryValues

  InterruptibleLoadValue:
    description: Holds one primary Interruptable Load value for a given dispatch node
    type: object
    properties:
      dispatchType:
        type: string
        description: The type of Interruptable Load value being dispatched.
        example: INTF
        enum:
          - INTF
          - INTS
      dispatchValue:
        type: number
        format: double
        description: The value of the type being dispatched
        example: 123.2
      dispatchTime:
        type: string
        format: date-time
        description: >-
          The (ISO 8601 with zone indicator format) date/time the dispatch was generated by
the Market System
        example: '2018-10-16T15:49:00.123Z'
      dispatchIssueTime:
        type: string
        format: date-time
        description: >-
          The (ISO 8601 with zone indicator format) date/time the dispatch was sent by the
Market Dispatch System to the Dispatch Participant
        example: '2018-10-16T15:49:00.123Z'
    required:
      - dispatchType
      - dispatchValue
      - dispatchTime

  VoltageDispatch:
    type: object
    description: Representation of the dispatch of the Voltage dispatch/product group
    properties:
      blocks:
        type: array
        description: Array of (0 or more) Blocks that Voltage is being dispatched to
        items:
          $ref: '#/definitions/VoltageNode'
      nodes:
```

```
        type: array
        description: Array of (0 or more) Nodes that Voltage is being dispatched to
        items:
          $ref: '#/definitions/VoltageNode'

  VoltageNode:
    type: object
    description: Representation of the voltage dispatch for a Node or Block
    properties:
      name:
        type: string
        description: The unique name of the node
        example: OHAU_1234
      traderCode:
        type: string
        example: ACME
        description: Four letter code of the trader for this node
      primaryValues:
        type: array
        maxItems: 1
        description: Primary dispatch type values. There is only ever one primary value for
a given node at a point in time.
        items:
          $ref: '#/definitions/VoltageDispatchValue'
    required:
      - name
      - traderCode
      - primaryValues

  VoltageDispatchValue:
    type: object
    description: Holds one primary Voltage value for a given dispatch node
    properties:
      dispatchType:
        type: string
        description: The type of value being dispatched
        example: VOLT
        enum:
          - VOLT
          - MVAR
      dispatchValue:
        type: number
        format: double
        description: The value of the type being dispatched
        example: 123.2
      dispatchTime:
        type: string
        format: date-time
        description: >-
          The (ISO 8601 with zone indicator format) date/time the dispatch was generated by
the Market System
        example: '2018-10-16T15:49:00.123Z'
      dispatchIssueTime:
        type: string
        format: date-time
        description: >-
          The (ISO 8601 with zone indicator format) date/time the dispatch was sent by the
Market Dispatch System to the Dispatch Participant
        example: '2018-10-16T15:49:00.123Z'
    required:
      - dispatchType
      - dispatchValue
      - dispatchTime
```

# Appendix 2: SAMPLE ICCP TAGS

For data going to ICCP endpoint ABC, modelling sites DEF, GHI and JKL:

Site DEF is configured as a block with nodes:

- **DEF2201 DEF0**
- **DEF2201 DEF1**

Site DEF is dispatched for:

- **Generation – and reserves**
- **Frequency at Block level**
- **Voltage at Block Level**
- **Subject to Block Security Constraints**

Site GHI is configured as a block with nodes:

- **GHI2201 GHI0**
- **GHI2201 GHI1**
- **PQR110 PQR0**

Site GHI is dispatched for:

- **Generation – and reserves**
- **Frequency at node GHI2201 GHI1 only**
- **Voltage at Block Level**
- **Not subject to BSCs**

Site JKL is configured as a single node not a block for node JKL0331 JKL0.

Site JKL is a wind farm and can do:

Generation

The tags for the above endpoint configuration is shown in the table below (tags for blocks DEF, GHI and JKL are background coloured yellow, blue and green, respectively.

*Table 13 – Example tag configuration for endpoint ABC*

| Description | Default Tag |
|---|---|
| EndPoint Heartbeat | ABC_HEARTBEAT |
| Energy Value | ABC_DEF_MW_VALUE |
| Energy Dispatch Time | ABC_DEF_MW_DTIME |
| Fast Reserve Value | ABC_DEF_RESF_VALUE |
| Fast Reserve Dispatch Time | ABC_DEF_RESF_DTIME |
| Sustained Reserve Value | ABC_DEF_RESS_VALUE |
| Sustained Reserve Dispatch Time | ABC_DEF_RESS_DTIME |
| Ramp Up Value | ABC_DEF_MW_RUP |

| Description | Default Tag |
|---|---|
| Ramp Up Broken Flag | ABC_DEF_MW_RUPB |
| Ramp Down Value | ABC_DEF_MW_RDN |
| Ramp Down Broken Flag | ABC_DEF_MW_RDNB |
| Fast Risk Value | ABC_DEF_MW_FRK |
| Sustained Risk Value | ABC_DEF_MW_SRK |
| Generation Constrained Flag | ABC_DEF_MW_GC |
| Intermittent Generation Flag | ABC_DEF_MW_IG |
| EndPoint EnergyReserve Resend Flag | ABC_ENERGYRESERVE_RESEND |
| EndPoint EnergyReserve Retry Counter | ABC_ENERGYRESERVE_RETRYC |
| EndPoint EnergyReserve Sequence Number | ABC_ENERGYRESERVE_SEQNUM |
| Endpoint EnergyReserve System Acknowledgement Tag | ABC_ENERGYRESERVE_SEQNUM_ACK |
| Endpoint EnergyReserve Business Acknowledgement - Affirmative Tag | ABC_ENERGYRESERVE_SEQNUM_ACKA |
| Endpoint EnergyReserve Business Acknowledgement - Query Tag | ABC_ENERGYRESERVE_SEQNUM_ACKQ |
| Energy Value | ABC_DEF2201_DEF0_MW_VALUE |
| Energy Dispatch Time | ABC_DEF2201_DEF0_MW_DTIME |
| Fast Reserve Value | ABC_DEF2201_DEF0_RESF_VALUE |
| Fast Reserve Dispatch Time | ABC_DEF2201_DEF0_RESF_DTIME |
| Sustained Reserve Value | ABC_DEF2201_DEF0_RESS_VALUE |
| Sustained Reserve Dispatch Time | ABC_DEF2201_DEF0_RESS_DTIME |
| Ramp Up Value | ABC_DEF2201_DEF0_MW_RUP |
| Ramp Up Broken Flag | ABC_DEF2201_DEF0_MW_RUPB |
| Ramp Down Value | ABC_DEF2201_DEF0_MW_RDN |
| Ramp Down Broken Flag | ABC_DEF2201_DEF0_MW_RDNB |
| Fast Risk Value | ABC_DEF2201_DEF0_MW_FRK |
| Sustained Risk Value | ABC_DEF2201_DEF0_MW_SRK |
| Generation Constrained Flag | ABC_DEF2201_DEF0_MW_GC |
| Intermittent Generation Flag | ABC_DEF2201_DEF0_MW_IG |
| Energy Value | ABC_DEF2201_DEF1_MW_VALUE |
| Energy Dispatch Time | ABC_DEF2201_DEF1_MW_DTIME |
| Fast Reserve Value | ABC_DEF2201_DEF1_RESF_VALUE |

| Description | Default Tag |
|---|---|
| Fast Reserve Dispatch Time | ABC_DEF2201_DEF1_RESF_DTIME |
| Sustained Reserve Value | ABC_DEF2201_DEF1_RESS_VALUE |
| Sustained Reserve Dispatch Time | ABC_DEF2201_DEF1_RESS_DTIME |
| Ramp Up Value | ABC_DEF2201_DEF1_MW_RUP |
| Ramp Up Broken Flag | ABC_DEF2201_DEF1_MW_RUPB |
| Ramp Down Value | ABC_DEF2201_DEF1_MW_RDN |
| Ramp Down Broken Flag | ABC_DEF2201_DEF1_MW_RDNB |
| Fast Risk Value | ABC_DEF2201_DEF1_MW_FRK |
| Sustained Risk Value | ABC_DEF2201_DEF1_MW_SRK |
| Generation Constrained Flag | ABC_DEF2201_DEF1_MW_GC |
| Intermittent Generation Flag | ABC_DEF2201_DEF1_MW_IG |
| Voltage Value | ABC_DEF_VOLT_VALUE |
| Voltage Dispatch Time | ABC_DEF_VOLT_DTIME |
| Mvar Value | ABC_DEF_MVAR_VALUE |
| Mvar Dispatch Time | ABC_DEF_MVAR_DTIME |
| EndPoint Voltage Resend Flag | ABC_VOLTAGE_RESEND |
| EndPoint Voltage Retry Counter | ABC_VOLTAGE_RETRYC |
| EndPoint Voltage Sequence Number | ABC_VOLTAGE_SEQNUM |
| Endpoint Voltage System Acknowledgement Tag | ABC_VOLTAGE_SEQNUM_ACK |
| Endpoint Voltage Business Acknowledgement - Affirmative Tag | ABC_VOLTAGE_SEQNUM_ACKA |
| Endpoint Voltage Business Acknowledgement - Query Tag | ABC_VOLTAGE_SEQNUM_ACKQ |
| Frequency Keeping Band Value | ABC_DEF_FREQ_VALUE |
| Frequency Keeping Band Value current | ABC_DEF_FREQ_VALUE_C |
| Frequency Keeping Start Time | ABC_DEF_FREQ_FST |
| Frequency Keeping Start Time (current) | ABC_DEF_FREQ_FST_C |
| EndPoint Frequency Resend Flag | ABC_FREQUENCY_RESEND |
| EndPoint Frequency Retry Counter | ABC_FREQUENCY_RETRYC |
| EndPoint Frequency Sequence Number | ABC_FREQUENCY_SEQNUM |
| Endpoint Frequency System Acknowledgement Tag | ABC_FREQUENCY_SEQNUM_ACK |
| Endpoint Frequency Business Acknowledgement - Affirmative Tag | ABC_FREQUENCY_SEQNUM_ACKA |

| Description | Default Tag |
|---|---|
| Endpoint Frequency Business Acknowledgement - Query Tag | ABC_FREQUENCY_SEQNUM_ACKQ |
| Energy Value | ABC_GHI_MW_VALUE |
| Energy Dispatch Time | ABC_GHI_MW_DTIME |
| Fast Reserve Value | ABC_GHI_RESF_VALUE |
| Fast Reserve Dispatch Time | ABC_GHI_RESF_DTIME |
| Sustained Reserve Value | ABC_GHI_RESS_VALUE |
| Sustained Reserve Dispatch Time | ABC_GHI_RESS_DTIME |
| Ramp Up Value | ABC_GHI_MW_RUP |
| Ramp Up Broken Flag | ABC_GHI_MW_RUPB |
| Ramp Down Value | ABC_GHI_MW_RDN |
| Ramp Down Broken Flag | ABC_GHI_MW_RDNB |
| Fast Risk Value | ABC_GHI_MW_FRK |
| Sustained Risk Value | ABC_GHI_MW_SRK |
| Generation Constrained Flag | ABC_GHI_MW_GC |
| Intermittent Generation Flag | ABC_GHI_MW_IG |
| Energy Value | ABC_GHI2201_GHI0_MW_VALUE |
| Energy Dispatch Time | ABC_GHI2201_GHI0_MW_DTIME |
| Fast Reserve Value | ABC_GHI2201_GHI0_RESF_VALUE |
| Fast Reserve Dispatch Time | ABC_GHI2201_GHI0_RESF_DTIME |
| Sustained Reserve Value | ABC_GHI2201_GHI0_RESS_VALUE |
| Sustained Reserve Dispatch Time | ABC_GHI2201_GHI0_RESS_DTIME |
| Ramp Up Value | ABC_GHI2201_GHI0_MW_RUP |
| Ramp Up Broken Flag | ABC_GHI2201_GHI0_MW_RUPB |
| Ramp Down Value | ABC_GHI2201_GHI0_MW_RDN |
| Ramp Down Broken Flag | ABC_GHI2201_GHI0_MW_RDNB |
| Fast Risk Value | ABC_GHI2201_GHI0_MW_FRK |
| Sustained Risk Value | ABC_GHI2201_GHI0_MW_SRK |
| Generation Constrained Flag | ABC_GHI2201_GHI0_MW_GC |
| Intermittent Generation Flag | ABC_GHI2201_GHI0_MW_IG |
| Energy Value | ABC_GHI2201_GHI1_MW_VALUE |
| Energy Dispatch Time | ABC_GHI2201_GHI1_MW_DTIME |

| Description | Default Tag |
|---|---|
| Fast Reserve Value | ABC_GHI2201_GHI1_RESF_VALUE |
| Fast Reserve Dispatch Time | ABC_GHI2201_GHI1_RESF_DTIME |
| Sustained Reserve Value | ABC_GHI2201_GHI1_RESS_VALUE |
| Sustained Reserve Dispatch Time | ABC_GHI2201_GHI1_RESS_DTIME |
| Ramp Up Value | ABC_GHI2201_GHI1_MW_RUP |
| Ramp Up Broken Flag | ABC_GHI2201_GHI1_MW_RUPB |
| Ramp Down Value | ABC_GHI2201_GHI1_MW_RDN |
| Ramp Down Broken Flag | ABC_GHI2201_GHI1_MW_RDNB |
| Fast Risk Value | ABC_GHI2201_GHI1_MW_FRK |
| Sustained Risk Value | ABC_GHI2201_GHI1_MW_SRK |
| Generation Constrained Flag | ABC_GHI2201_GHI1_MW_GC |
| Intermittent Generation Flag | ABC_GHI2201_GHI1_MW_IG |
| Energy Value | ABC_PQR1101_PQR0_MW_VALUE |
| Energy Dispatch Time | ABC_PQR1101_PQR0_MW_DTIME |
| Fast Reserve Value | ABC_PQR1101_PQR0_RESF_VALUE |
| Fast Reserve Dispatch Time | ABC_PQR1101_PQR0_RESF_DTIME |
| Sustained Reserve Value | ABC_PQR1101_PQR0_RESS_VALUE |
| Sustained Reserve Dispatch Time | ABC_PQR1101_PQR0_RESS_DTIME |
| Ramp Up Value | ABC_PQR1101_PQR0_MW_RUP |
| Ramp Up Broken Flag | ABC_PQR1101_PQR0_MW_RUPB |
| Ramp Down Value | ABC_PQR1101_PQR0_MW_RDN |
| Ramp Down Broken Flag | ABC_PQR1101_PQR0_MW_RDNB |
| Fast Risk Value | ABC_PQR1101_PQR0_MW_FRK |
| Sustained Risk Value | ABC_PQR1101_PQR0_MW_SRK |
| Generation Constrained Flag | ABC_PQR1101_PQR0_MW_GC |
| Intermittent Generation Flag | ABC_PQR1101_PQR0_MW_IG |
| Frequency Keeping Band Value | ABC_GHI2201_GHI1_FREQ_VALUE |
| Frequency Keeping Band Value Current | ABC_GHI2201_GHI1_FREQ_VALUE_C |
| Frequency Keeping Start Time | ABC_GHI2201_GHI1_FREQ_FST |
| Frequency Keeping Start Time (current) | ABC_GHI2201_GHI1_FREQ_FST_C |
| Voltage Value | ABC_GHI_VOLT_VALUE |

| Description | Default Tag |
|---|---|
| Voltage Dispatch Time | ABC_GHI_VOLT_DTIME |
| Mvar Value | ABC_GHI_MVAR_VALUE |
| Mvar Dispatch Time | ABC_GHI_MVAR_DTIME |
| Energy Value | ABC_JKL0331_JKL0_MW_VALUE |
| Energy Dispatch Time | ABC_JKL0331_JKL0_MW_DTIME |
| Fast Reserve Value | ABC_JKL0331_JKL0_RESF_VALUE |
| Fast Reserve Dispatch Time | ABC_JKL0331_JKL0_RESF_DTIME |
| Sustained Reserve Value | ABC_JKL0331_JKL0_RESS_VALUE |
| Sustained Reserve Dispatch Time | ABC_JKL0331_JKL0_RESS_DTIME |
| Ramp Up Value | ABC_JKL0331_JKL0_MW_RUP |
| Ramp Up Broken Flag | ABC_JKL0331_JKL0_MW_RUPB |
| Ramp Down Value | ABC_JKL0331_JKL0_MW_RDN |
| Ramp Down Broken Flag | ABC_JKL0331_JKL0_MW_RDNB |
| Fast Risk Value | ABC_JKL0331_JKL0_MW_FRK |
| Sustained Risk Value | ABC_JKL0331_JKL0_MW_SRK |
| Generation Constrained Flag | ABC_JKL0331_JKL0_MW_GC |
| Intermittent Generation Flag | ABC_JKL0331_JKL0_MW_IG |

# Appendix 3: FREQUENCY DISPATCH

The logic in the ICCP Adapter has been updated and released on 12-Dec-2019:

Whenever there is a frequency dispatch for the current trading period, we will always populate both the current and next trading period ICCP tags where the next period value is dictated by the MDB (Market Database) sending only a current dispatch value or both the current and next period values.

*i.e. When we detect the FST_C in a trading period we also generate an entry for the next period*

| Generated Time | MW band | Effective Time |
|---|---|---|
| 15:15 | VALUE=15 | FST=15:30 |
| 15:36 | VALUE_C=0 | FST_C=15:36 |
| 15:36 | VALUE=0 | FST=16:00 |

Frequency dispatch consists of **_C** tags. The **_C** represents current (trading period). The tags without _C represents start of the next trading period.

| CTY | CTY2201 CTY1 | Frequency | FREQ | FST | OUT | CTY_CTY2201_CTY1_FREQ_FST |
|---|---|---|---|---|---|---|
| CTY | CTY2201 CTY1 | Frequency | FREQ | FST | OUT | CTY_CTY2201_CTY1_FREQ_FST_C |
| CTY | CTY2201 CTY1 | Frequency | FREQ | VALUE | OUT | CTY_CTY2201_CTY1_FREQ_VALUE |
| CTY | CTY2201 CTY1 | Frequency | FREQ | VALUE | OUT | CTY_CTY2201_CTY1_FREQ_VALUE_C |

where

_FST_C = Current frequency start time in epoch time

_FST = frequency start time for the next trading period in epoch time

_FREQ_VALUE_C = Current Frequency Band

_FREQ_VALUE = Frequency Band for the start of the next trading period

## 3.1 LOGIC SPECIFICALLY PERTAINING TO ICCP FOR THE TAGS VALUE, VALUE_C, FST, AND FST_C

**Rule 1: At the start of every trading period, the VALUE is always current**

In the standard frequency dispatching scenario, the frequency values (VALUE) are dispatched halfway through the current trading period for the next trading period. The frequency start time (FST) that is dispatched with the value will correspond to the start time of the next trading period.

Values are persistent; If there is no change to the frequency between trading periods then no dispatch will be sent and the previous VALUE remains in effect.

**Rule 2: Frequency dispatch overrides are only applicable to the current period of the dispatch**

In the non-standard scenario where a frequency dispatch override is sent, the override value is sent for the current period as VALUE_C with the current period start time (FST_C) corresponding to the time with dispatch was created. Since rule #1 must apply, the dispatch will also include the VALUE and FST for the next period such that the current period override is only applicable for the remainder of the current trading period.

To condense that logic:

All frequency dispatches will contain a VALUE and FST that are for the next trading period.

VALUE_C overrides are only ever applicable to the current period.

At the start of each trading period the VALUE will be "current"

## 3.2 EXAMPLE (ICCP BLOCK 2):

CTY1 unit was manually selected for Frequency keeping for the current trading period (i.e. 1100 trading period) and it is coming off Frequency keeping for the following trading period.

| | |
|---|---|
| CTY_CTY2201_CTY1_FREQ_FST | 1571697000 |
| CTY_CTY2201_CTY1_FREQ_FST_C | 1571696086 |
| CTY_CTY2201_CTY1_FREQ_VALUE | 0 |
| CTY_CTY2201_CTY1_FREQ_VALUE_C | 15 |

Where Epoch time 1571697000 = Tuesday, October 22, 2019 11:30:00 AM
Where Epoch time 1571696086 = Tuesday, October 22, 2019 11:14:46 AM

## 3.3 FAQ:

| | | |
|---|---|---|
| 1 | Are the _C tags only used when the current frequency dispatch needs to be changed outside of the normal 15-minute window? | *It's used any time the current (as opposed to future) frequency keeper is changed regardless of time window* |
| 2 | Will the _C tags be used to change the current frequency dispatch band? | *Yes* |
| 3 | Will the _C tags be sent individually, or will they always be accompanied by normal FST and FREQ_VALUE tags? | *They will be accompanied by normal FST and FREQ_VALUE tags only when the band is different from the _C values for that site within the next 15min* |
| 4 | If there are accompanying normal FST and FREQ_VALUE tags, will these always be used to turn frequency control off at the start of the next period? | *They will be used to either turn the control off, on or change the band for the control* |

# Appendix 4: DATA QUALITY/VALIDATION

## 4.1 DEFECT(S) IDENTIFIED AFTER THE FIRST CUSTOMER TRANSITIONED TO DISPATCH ICCP CHANNEL

Soon after a participant's transition to ICCP, an unplanned server failover resulted in the dispatch system sending bad quality data from the newly enabled server. A hot fix released on 23 April 2020 minimises ICOM/ICCP stale data issues with dispatch ICCP.

## 4.2 VALIDATION RULES AT THE DISPATCH ENDPOINTS

Transpower strongly recommends participants to have validation rules in place to discard data values with bad quality.

## 4.3 EXAMPLE VALIDATION RULES IMPLEMENTED BY DISPATCH PARTICIPANT(S) FOR ICCP CHANNEL.

a. Inhibit all dispatch sequence numbers from propagating through the system unless valid data exists on the energy sequence number for at least 10 seconds;
b. Add validation logic to the SCADA system for the following:
    - When TPNZ rolls the SEQNUM over from 9999 to 1;
    - In the event a SEQNUM is received that's less than SEQNUM_ACK value then ignore this dispatch.
    - data quality check
    - ability to update/rest sequence manually when required.