

Central Switching Service

CSS Interface Design Document

Prepared for Connecting Participants

19/02/2025

Landmark INFORMATION

Revision History

Version	Comment	Author	Date
1.0	Draft Version	Landmark	21 June 2019
2.0	Draft Version	Landmark	21 June 2019
3.0	Updates to draft version in response to DCC feedback	Landmark	24 June 2019
4.0	Further updates in response to feedback	Landmark	28 June 2019
4.1	Updates in response to feedback from the review on 1 july 2019, interim release	Landmark	24 July 2019
4.2	Interim release	Landmark	26 July 2019
5.0	Release for sign off	Landmark	1 August 2019
5.1	Updates in response to DCC feedback / interim release	Landmark	9 August 2019
5.2	Updates in response to feed back / interim release	Landmark	14 August 2019
5.3	Updates in response to DCC feedback / interim release	Landmark	15 August 2019
6.0	Release for sign off	Landmark	16 August 2019
7.0	Release with updates in response to v6.0	Landmark	04 September 2019
8.0	Release for sign off	Landmark	12 September 2019
8.1	1.2 Updated to only record what versions of abacus and user requirements URL's changed to URLs Updated 4.1 meteringPointEnergyFlowFromDate description	Landmark	23 September 2019
8.2	Updated to remove mpxn from URLs on API routes	Landmark	10 October 2019
8.3	Updated per CR-E46 CR-D001, CR-D008 and CR-D012	Landmark	3 April 2020
8.4	Correction to status code in 5.6.3	Landmark	7 May 2020
8.5	To demonstrate the changes introduced by CR-D013, CR-D017, CR-D035 and CR-D038	Landmark	23 October 2020

Landmark INFORMATION

~~8.6~~	~~Document updated to clarify API route structure, demonstrate changes from CR-D036 and amendment to the REL Address Element 'confidenceScore'~~	~~Landmark~~	~~28 May 2021~~
8.7	~~Updated to clarify API route structure, demonstrate changes from CR-D036 and amendment to the REL Address Element 'confidenceScore'.~~ ~~Further updates per CR-D059, to resolve DEF-797 (DI-1642), DI-1090, DI-1657, DI-2194 and to provide examples of error payloads in OFAF switches.~~ Updated per CR-D036, CR-D059 and CR-D134, to resolve DEF-797 (DI-1642), DI-1090, DI-1657, DI-2194 to clarify API route structure and amendment to the REL Address Element 'confidenceScore' and to provide examples of error payloads in OFAF switches.	Landmark	~~04 March~~ 01 April 2022
8.7.2	Updates to resolve DEF-1288 and DEF1251 3.4.1, 4.1.1, 6.3 Paragraph added to explain that data is a object in JSON and may not be received in order sent 4.1.2 JSON description added 5.6.2 Explanation re correcting future shipperfromdate as unable to cancel Updates to OFAF groups as being non-mandatory under section 5.7.2	Landmark	28/02/2023
8.7.2 R0044 MHHS	Updates to include R0044 (MHHS) changes incorporated and noted as not being applicable until June 2024 Sections 1.0 4.1 4.3.9 5.4 5.6.3 6.1.1 6.2.7	Landmark	04/06/2023
8.8.0 R0067	Add new registration APIs defined under R0067. Sections: 5.6.7; 5.6.8 Added MHHS message ids. Sections: 4.3.9, 5.4, 6.2.7. Add MHHS effective date statement. Sections: 5.6.3	Landmark	13/06/2023

	Minor typo corrections. Sections: 4.1, 4.3.9, 5.6.7, 5.6.8, 6.2.7		
8.8.1 R0067	Registration Status Refresh API definition updated to reflect usage of a new event id in resent messages	Landmark	6 February 2024
8.8.2	Created addendum to highlight that in the supplier arranged appointments webhook notification the mpid contains an upper case I eg; mpId Sections 6.2.1, 6.2.7, 6.2.16 updated CLEAN version from 8.8.2 TRACKED	Landmark	01/07/2024
8.9.0 R0080	Add new message resend API defined under R0080. Sections: 5.8	Landmark	19/02/2025

Landmark INFORMATION

Table of Contents

1 Introduction 9
1.1 Document Purpose 9
1.2 Document Considerations 10
1.3 Out of scope 11
1.4 CSS Functionality 11
1.5 Implementation Overview 12
1.6 AddressBase® Premium 13
2 Connectivity and security 14
3 Transport layer and general principles 15
3.1 OpenAPI Specification 15
3.1.1 Versioning 15
3.1.2 Developer Portal 16
3.2 Further Reading 17
3.2.1 What is REST? 17
3.2.2 OpenAPI Specification 17
3.2.3 OpenAPI Initiative 17
3.3 Generic Asynchronous Message Flow 18
3.4 Webhooks 19
3.4.1 What they are 19
3.4.2 Why do we need them? 19
3.4.3 Subscribing 19
3.4.4 Messages 19
3.4.5 Securing 19
3.4.6 Delivery and Throttling 20
3.5 Error Handling 21
3.5.1 Synchronous Errors 21
3.5.2 Asynchronous Errors 21
3.6 Validation 21
4 CSS Data Definitions 22
4.1 CSS Elements 22
4.1.1 MPL Address Mapping 27
4.1.2 REL Address Elements 30
4.2 CSS Field Data Types 32
4.3 Enumerations 33
4.3.1 fuelType 33

4.3.2 interventionType 33
4.3.3 associationType 33
4.3.4 energyFlow 33
4.3.5 rmpStatus 33
4.3.6 registrationStatus 33
4.3.7 registrationRequestStatus 33
4.3.8 switchEventType 34
4.3.9 eventType 34
4.3.10 eventStatus 36
4.3.11 contextType 36
4.3.12 registrationInitiator 37
4.3.13 addressSource 37
4.3.14 marketParticipantRoleEvent 37
4.3.15 addressType 37
4.3.16 logicalStatus 38
4.3.17 timeBoundaryAlignmentStatus 38
4.4 Element Mappings 39
4.4.1 RMP Gas/Electricity API 39
4.4.2 RMP Associations API 39
4.4.3 RMP Asset Ownership MAP API 39
4.4.4 Registration Gas/Electricity API 39
4.4.5 Registration Supplier Arranged Appointments API 40
4.4.6 Registration Switch Request API 40
4.4.7 Registration Switch Intervention API 40
4.4.8 Registration Deactivation API 41
4.4.9 Domain Data API 41
4.4.10 Market Participant API 42
4.5 Errors Payload 44
5 CSS API specification 46
5.1 General Principles 46
5.1.1 POST 46
5.1.2 PATCH 46
5.1.3 PUT 46
5.1.4 CorrelationId and EventId 47
5.1.5 RegistrationId 48
5.2 Standard Response Body 49
5.3 Error API Request 50

5.4 Webhook API 51
5.5 RMP API Request 54
5.5.1 Gas 54
5.5.2 Electricity 56
5.5.3 Associations 58
5.5.4 Asset Ownership MAP 59
5.6 Registration API Request 60
5.6.1 Initial registration 60
5.6.2 Update Current Active Registration 60
5.6.3 Supplier Arranged Appointments 62
5.6.4 Switch API Request 63
5.6.5 Switch Intervention API Request 64
5.6.6 Registration Deactivation API Request 65
5.6.7 Registration Status Refresh API Request 66
5.6.8 Active Registration Refresh API Request 67
5.7 Market Participant Data API 68
5.7.1 Market Role 68
5.7.2 Market Participant Role 69
5.7.3 Alliance 70
5.7.4 Company 71
5.8 Resend Undelivered Messages API 72
5.8.1 Resend Undelivered Notifications 72
6 CSS Webhook Payload Specification 73
6.1 Payload Structure 73
6.1.1 Static Properties 73
6.1.2 Error Object 74
6.2 Data Payload by Context Type 75
6.2.1 DCCSM 75
6.2.2 MAP 77
6.2.3 MEM 78
6.2.4 DES 79
6.2.5 ECOES 80
6.2.6 GRDA 82
6.2.7 ERDA 84
6.2.8 HHDC, NHHDC 86
6.2.9 HHDA, NHHDA 86
6.2.10 Gaining Supplier 87

6.2.11 Losing Supplier 89
6.2.12 Gaining Shipper 90
6.2.13 Losing Shipper 91
6.2.14 Registered Shipper 92
6.2.15 Webhook Subscription 93
6.2.16 ECOS 93
6.2.17 Registered Supplier 95
6.3 REL Address Payload 97
7 Appendix 98
7.1 Abacus Inbound API Map to Sections 98
7.2 Abacus Outbound Message Map to Webhook Event Type 100

1 Introduction

1.1 Document Purpose

The purpose of this document is to define the Central Switching Service (CSS) Interfaces at a technical level to enable all the participants, Central Data Services/Market Participants, to update their current systems to meet the new requirements.

It is not a standalone document and does not describe the full functionality of the CSS. It should be read in conjunction with the requirement specifications published on the Ofgem website:

https://www.ofgem.gov.uk/publications-and-updates/css-design-and-delivery-products

CSS provides application interfaces for the following Market Roles:

- Suppliers
- Data Enquiry Service (DES)
- Shippers
- Electricity Retail Data Agent (ERDA)
- Gas Retail Data Agent (GRDA)
- Electricity Central Online Enquiry Service (ECOES)
- DCC Smart Metering (DCC SM)
- Data Collectors (DC)
- Data Aggregators (DA)
- Meter Asset Providers (MAPs)
- Metering Equipment Manager (MEMs)
- Electricity Domain Data Governance (EDDG)
- Gas Domain Data Governance (GDDG)
- Retail Energy Code Domain Data Governance (RECDDG)
- Smart Metering Enduring Change of Supplier (ECOS)
- *Smart Meter Data Retrieval (MDR) *Not effective until June 2024*

1.2 Document Considerations

This document is based on the Logical Abacus model version 4, 5 and 6 and D-4.2.1 CSS User Requirements Specification version V2.2.

1.3 Out of scope

This document does not cover the following information, which will be in other policy documents to be generated by the SI and DCC, although some of its content may contribute to those documents. LIG is not the owner of these documents.

- Code of Connection (COCO)
- Data dictionary
- Business rules
- Procedures for the use of CSS – This includes items like the retry on error policy, raising errors to the service desk, etc

1.4 CSS Functionality

The CSS is responsible for the following:

- Mastering the Registrations
- Mastering the Retail Energy Location (REL)
- Distribution of Registerable Measurement Point (RMP)

1.5 Implementation Overview

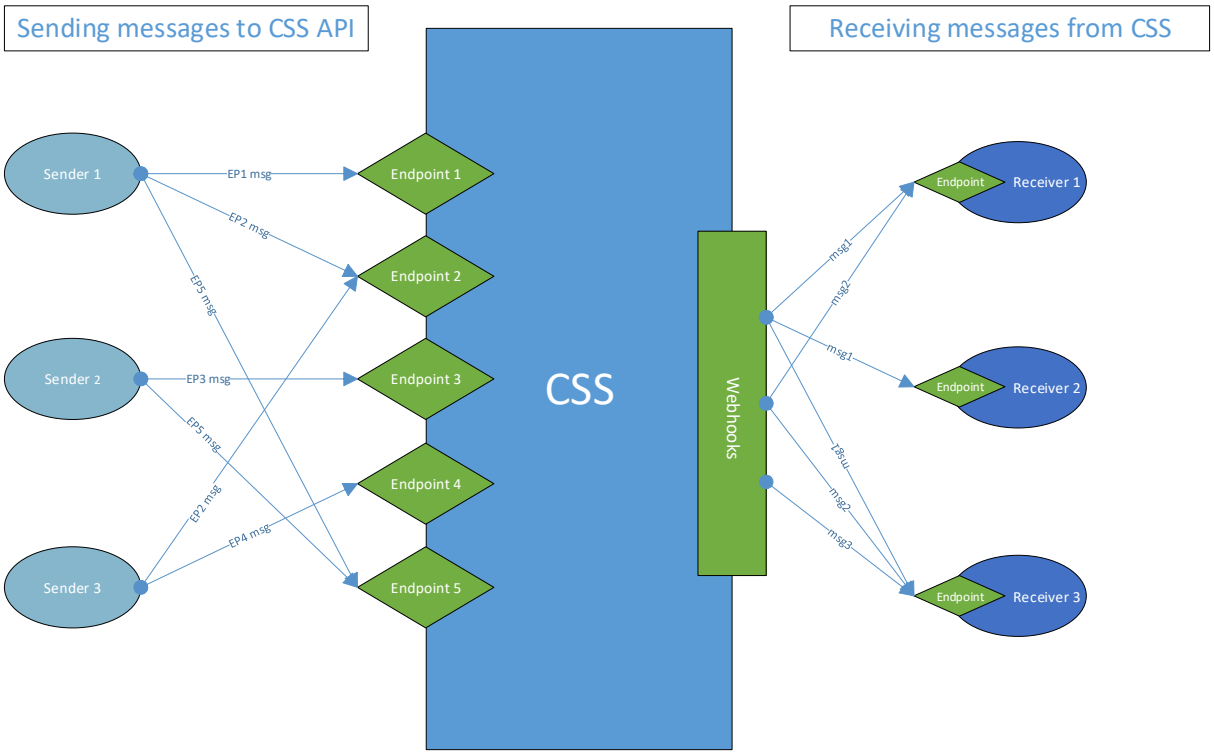
The CSS will be a cloud-based service hosted on the Microsoft Azure Platform. All interaction with the CSS will be real-time messaging, based on a predefined JavaScript Object Notation (JSON[1]) format (See sections 5 and 6).

Inbound messages will be sent by calling pre-determined URLs whilst the outbound messages are delivered to URLs of the recipient's choice. These services are explained in technical detail in the Transport layer and general principles section of this document.

The diagram below depicts the interaction with the CSS from the perspective of sending messages into CSS and receiving messages from CSS.

CSS advertises multiple endpoints for receiving messages each of which fulfils a specific business function, for example, a specific endpoint is used to receive switch requests. Multiple external systems with the same business needs will deliver messages to the same endpoint, for example, all switch requests are sent to the same endpoint regardless of which participant instigated the request.

To receive messages from CSS, a recipient registers target endpoints for each market participant role they hold and all outgoing messages applicable to that role are sent to these endpoints by CSS.



[1] Pronounced Jason, as in "Jason and the Argonauts"

1.6 AddressBase® Premium

The Registration and Addressing elements of the CSS will make use of Ordnance Survey AddressBase® Premium (ABP) data (https://www.ordnancesurvey.co.uk/business-and-government/products/addressbase-premium.html).

The data passed across the interfaces may only be used for the permitted purpose under the licence with Ordnance Survey. The permitted purpose as defined in the agreement is for the "purposes of enabling switching including (without limitation) for the purposes of design, development, testing, integration and live operational use."

The field mappings contained within this document are taken from v2.5 of the ABP technical specification, which can be found here: https://www.ordnancesurvey.co.uk/documents/product-support/tech-spec/addressbase-premium-technical-specification.pdf

2 Connectivity and security

The CSS solution will be built and operated from the Azure Public Cloud.

The network connectivity design is detailed in the code of connection document (COCO), which accompanies the Interface specification. This should be used to determine how to connect with CSS.

3 Transport layer and general principles

This section describes the transport layer and general principles of the CSS Service.

3.1 OpenAPI Specification

Access to the underlying CSS services will be controlled via REST API's fronted by Azure API Management.

The CSS will be using Open API specification version 3.
https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md.

3.1.1 Versioning

When the system has gone live, new functionality may be required or bugs may be found. These changes will be submitted to the Retail Energy Code Company (RECCo) and may result in changes being made to the CSS. In order to rollout those changes, it may be necessary to issue a new version of the API.

New versions can be as a result of changes to the API itself, or more likely to underlying business logic directly referred to by the API. This can be problematic when determining breaking changes as the business logic will be distributed across multiple consumers.

New versions will only be issued if the result of a change is deemed to be a breaking change, non-breaking changes will be issued as revisions to the existing version.

A Market Participant can opt into new version changes when the Market Participant is ready. When every Market Participant is consuming the new version, the old version will be deprecated. There will be a time constraint on this, and each Market Participant will be notified when this happens.

3.1.1.1 Breaking Changes

As the very nature of HTTP API's cross application boundaries, trying to determine what defines a breaking change can be complicated as client code can dictate what is deemed breaking.
In general, we can state that the following will cause a breaking change:

- Addition and removal of mission critical properties to the schema model, for example, new properties critical to the business process to ensure data integrity.
- Removal of API endpoints (still being consumed).
- Changing schema model properties (optional, max length, datatype etc) for existing properties.
- New domain values. For example, adding a new Status type. In general, this would be deemed a breaking change as consumers will not be expecting this.

3.1.1.2 Non-Breaking Changes

To aid Market Participants to get a first look at changes to an API, revisions will be used. Revisions enable us to independently release developer previews of upcoming changes. This will help determine if those changes could be deemed as breaking changes, if so, a new version would be created with those changes.

In general, we can state that the following will not cause a breaking change:

- Addition of new noncritical properties to the schema model. **Market Participants should ignore properties on the model which they have not implemented, this will ensure that new properties will not break existing logic**.
- New endpoints added to API's

Versioning of the API's will be handled on the API route, i.e.<CSS URL>/< CSS API Route>/< CSS API Version>/< CSS Operation Route>.

Please see below for examples:

https://centralswitchingservice.com/dccsm/v1.0/rmps/electricity
https://centralswitchingservice.com/dccsm/v1.0/rmps/gas
https://centralswitchingservice.com/supplier/v1.0/registrations
https://centralswitchingservice.com/supplier/v1.0/registrations/switch
https://centralswitchingservice.com/supplier/v1.0/errors

Participants can find information on product specific API routes on the CSS Developer Portal.

Different versions of the API will be available independently in the developer portal and a change log will document any changes made to the new version.

3.1.2 Developer Portal

All versions and revisions of the API's will be documented in the CSS developer portal as OpenAPI Specification Version 3.

The body of the messages described in sections 5 and 6 are provided to give an indication of what will be available when development begins. The open API specification in the portal is the documentation that should be used by developers.

This will be available during the DBT phase, and for the life of the CSS and can be found here:

https://devportal.centralswitchingservice.co.uk/

3.2 Further Reading

3.2.1 What is REST?

Representational state transfer.
"*Representational State Transfer (REST) is a software architectural style that defines a set of constraints to be used for creating Web services. Web services that conform to the REST architectural style, termed RESTful Web services (RWS), provide interoperability between computer systems on the Internet*."

Source: https://en.wikipedia.org/wiki/Representational_state_transfer

3.2.2 OpenAPI Specification

V2 was known as Swagger, V3 is now Open API Specification.
"*The OpenAPI Specification, originally known as the Swagger Specification, is a specification for machine-readable interface files for describing, producing, consuming, and visualizing RESTful web services. Originally part of the Swagger framework, it became a separate project in 2016, overseen by the OpenAPI Initiative, an open source collaborative project of the Linux Foundation. Swagger and some other tools can generate code, documentation and test cases given an interface file.*"

Source: https://en.wikipedia.org/wiki/OpenAPI_Specification

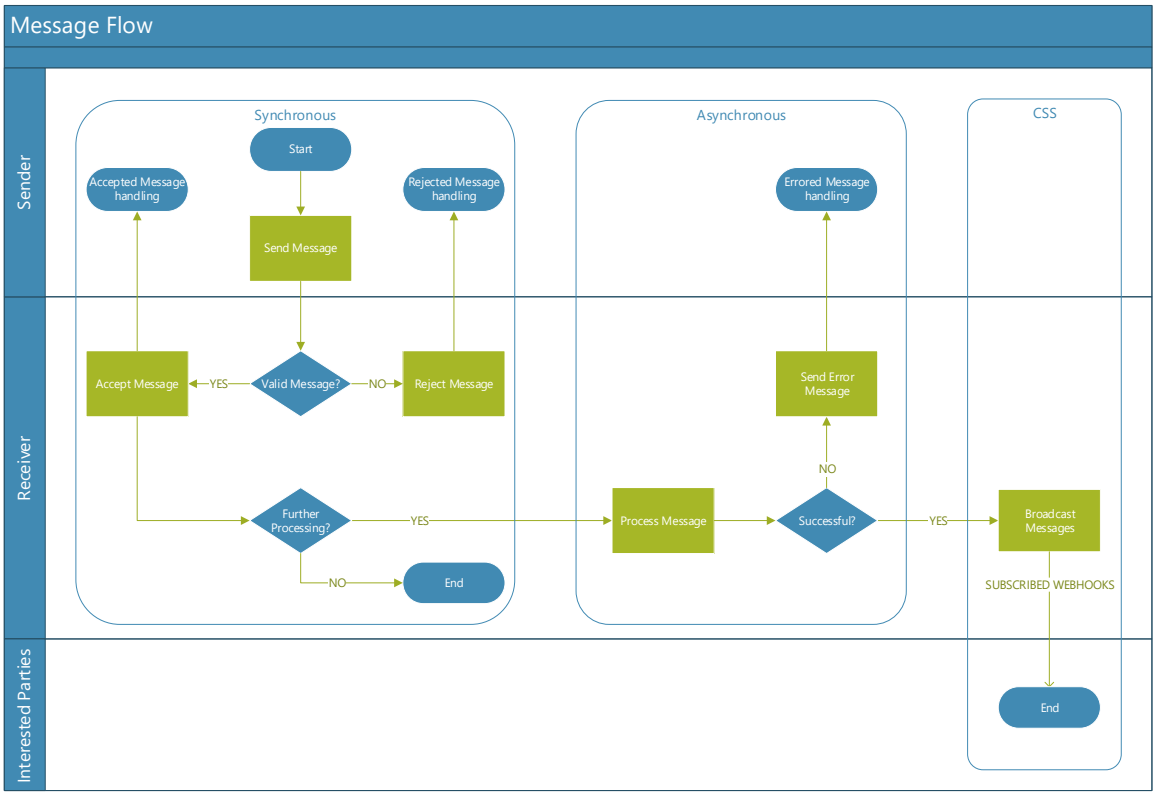
3.2.3 OpenAPI Initiative

"*The OpenAPI Initiative (OAI) was created by a consortium of forward-looking industry experts who recognize the immense value of standardizing on how REST APIs are described. As an open governance structure under the Linux Foundation, the OAI is focused on creating, evolving and promoting a vendor neutral description format.*
APIs form the connecting glue between modern applications. Nearly every application uses APIs to connect with corporate data sources, third party data services or other applications. Creating an open description format for API services that is vendor neutral, portable and open is critical to accelerating the vision of a truly connected world."

Source: https://www.openapis.org/about

3.3 Generic Asynchronous Message Flow

The diagram below depicts a generic message flow sequence applicable to both the CSS and interfacing systems. The exception being the CSS Broadcast message shown represents a CSS specific post-processing step, to illustrate how the CSS will inform market participants.



The key process steps are:

1. Send Message
   This involves posting a message to a predefined endpoint (URL).
2. Accept Message
   On Receipt of the message a synchronous response is sent to the sender accepting the message (202).
3. Reject Message
   A message is rejected based on schema validation (structure/type/content). The sender remains responsible for resolving the rejected request.
4. Process Message
   The receiver has accepted the message and performs their internal processing of the message content. If this is successful there is no requirement to inform the sender.
5. Send Error Message
   If the internal processing of the message content fails for reasons directly related to the message itself the sender is informed by sending an error message. This message is sent asynchronously.
6. Message Generation (CSS specific)
   This is an illustrative CSS specific post processing step to show how CSS will broadcast information messages to subscribers in response to an incoming message. Other systems may have alternative post processing steps at this point.

3.4 Webhooks

3.4.1 What they are

As more and more of what we do with connected apps can be described by events, Webhooks are becoming very popular as a mechanism to react to these events in a resource light decoupled way.

A Webhook (also called a web callback or HTTP push API) is a way for an application or system to provide other applications or systems with real-time information. It delivers data to other applications as it happens, meaning you get data immediately, as opposed to typical APIs where you would need to poll for data very frequently (hammer polling) in order to get it real-time. This makes Webhooks much more efficient for both providers and consumers.

Market participants will need to create an application or system capable of receiving and responding to HTTP requests. CSS will make a HTTP request to your application or system, these will be a POST requests to your Webhook URLs with a JSON body, it is then up to you to process the request and its content.

The data sent to and from CSS is an object in JSON, which is an unordered set of name/value pairs so may not come through in a logical order, examples of data in this document are for information only and do not infer the actual order of receipt.

3.4.2 Why do we need them?

The "messaging" functionality within the CSS is based on **real-time events**, that is, events get triggered by some action within CSS, and these events are then **pushed** out to **subscribers** of those events.

Webhooks give a way for the CSS to push messages out to the various Market Participants who are subscribed to those events in a reactive real-time manner, which makes the system highly performant and scalable.

3.4.3 Subscribing

There will be a way for a market participant to request a CSS registration. They will need to provide at least their Market Participant Id, their role and company name. Their request will be validated, making sure that they are eligible to be given access to the system. Then a subscription key will be issued to the Market Participant. The key will also be stored within CSS.

Once the Market Participant has the subscription key and their Webhooks available, they then need to post to the CSS API as detailed in section 5.4.

Please refer to section 5.4 for the webhook API.

3.4.4 Messages

Once subscribed to the CSS, the market participant will only receive the messages they have subscribed to. They will not receive every possible message sent by CSS. The messages sent out per context type are defined in section 6.2.

3.4.5 Securing

Each Market Participant implementing Webhooks has a need to ensure that requests it receives are from trusted parties i.e. the CSS.

When registering, Market Participants can optionally send an x-api-key http header (normally a UUID generated by themselves), this x-api-key http header will be sent back by CSS on all requests to the Market Participants registered Webhook as the same x-api-key http header. This key can be updated by the Market Participant at any time, by re-registering the Webhook with a new x-api-key.
This is over and above the security which will be in place as documented in the code of connection document (CoCo).

The x-api-key is optional, it is an additional means for Market Participants to ensure that messages came from CSS and for them to be in control of the keys and when they need to rotate them, it can be useful in early development to connect to an API before network security is in place. Best practice would be to have two keys (Primary and Secondary) you would check both against the x-api-key sent in the header. So, let's say the Primary key is registered against your webhook, when you need to rotate the Primary key, register the Secondary key against your webhook, then change the Primary key to a new value, rinse and repeat when rotating the Secondary key.

3.4.6 Delivery and Throttling

CSS is hosted on Azure as an Integration Platform As A Service (IPAAS) in a serverless environment, which allows, in theory, infinite scale out where resources "spin" up and down according to demand.

Event delivery will be via Azure Event Grid. If Event Grid cannot confirm that an event has been received successfully by the subscriber's Webhook, it tries to redeliver the event for a certain amount of time (24 hours by default) using a back off/retry mechanism. After that time is breached, Event Grid will move that event to a "Dead Letter" queue, which will be processed independently by CSS depending on the failure. For more information, see Event Grid message delivery and retry.

To ensure that the CSS does not flood the individual Market Participants with a high volume of events in the cases where the Market Participant Webhook may have been offline (planned maintenance, in a faulted state etc). Market Participants can return a 429 or 503 status code with a **Retry-After** header, CSS will queue event delivery until that point.

The **Retry-After** response HTTP header indicates how long the user agent should wait before making a follow-up request. The two main use cases supported are:

- When sent with a 503 (Service Unavailable) response, this indicates how long the service is expected to be unavailable.
- When sent with a 429 (Too Many Requests) response, this indicates how long to wait before making a new request.

Examples
Retry-After: 2019-08-24T22:57:49.000 (date)
Retry-After: 120 (seconds)

3.5 Error Handling

3.5.1 Synchronous Errors

The goal of error responses is to create a source of information to not only inform the user of a problem, but the possible solution to that problem as well. RFC 7807 provides a standard format for returning problem details from HTTPS APIs, see RFC7807. Whilst we will not follow this to the letter, it does provide some good pointers.

Wrapping an error in an error object gives consumers an easy way to check for an error by simply checking for the existence of the error object. Market Participants should adhere to the format as defined in section 4.5 when sending synchronous error messages to CSS.

3.5.2 Asynchronous Errors

A market participant asynchronous error is where an error occurs in your system as part of your asynchronous processing, after accepting a message from the CSS.

The end point to post the error information to is defined in section 5.3. The CSS will respond to the asynchronous error message using the response defined in section 5.2. These errors will go into the CSS error handling process.

A CSS asynchronous error is where an error occurs in the CSS as part of CSS asynchronous processing. In this scenario the CSS will inform the required market participants of the error, via the market participants Webhook, adhering to the format as defined in 4.54.4.10.

3.6 Validation

Validation of the Webhook payloads will be controlled via an Open API Specification supplied by CSS.

4 CSS Data Definitions

4.1 CSS Elements

The table below details the incoming and outgoing data elements that the CSS uses.

NM denotes that this field is not mastered by CSS; this document cannot provide any further definition.

Element	Datatype	Format	Length	Notes
activeRegistrationId	string	UUID	36	The current Active registration Id (UUID)
addressSource	string	string	10	How the address has been derived, see section 4.3.13
allianceId	string	UUID	36	This id is provided by the Market Participant on creation of an alliance, this should be unique, preferably a UUID
allianceNominatorMpid	string	string	NM	The MPID of the organisation that is proposing the alliance
allianceNomineeMpid	string	string	NM	The MPID of the organisation that will be in the alliance
allianceNominatorRole	string	string	NM	The role of the organisation that is proposing the alliance
allianceNomineeRole	string	string	NM	The role of the organisation that will be in the alliance
allianceFromDate	string	datetime	See 4.2	The date at which the alliance starts
allianceToDate	string	datetime	See 4.2	The date at which the alliance ended
associationType	string	string	NM	Internal domain Type, see section 4.3.3
changeOfOccupancyInd	boolean			true or false
commsHubLinkDeviceId	string	EUI-64	NM	The Extended Unique Identifier of the Comms Hub Device
confirmationRequired*	boolean			true or false
contextType	string	string	50	The context in which the Market Participant is receiving the message, e.g. LosingSupplier, see section 4.3.11
correlationId	string	UUID	36	The internal CSS reference used to group and track all related

				events for a given action (for a Switch Request for example) UUID
dccServiceInd	boolean			true or false
domesticPremisesInd	boolean			true or false
erroneousSwitchResolutionInd	boolean			true or false
errorCode	string	string	10	An internal error code mastered in CSS
errorDescription	string	string	5000	An optional description of the error
errorTitle	string	string	255	A short title of the error
errorType	string	string	2048	An optional URL document giving additional information about this error and possible fixes.
eventDate	string	datetime	See 4.2	The date and time when the CSS recorded the submission of the inbound event OR the date and time when CSS generated the outbound event.
eventDescription	string	string	5000	A description of the event the message is related to.
eventId	string	UUID	36	A unique reference for the event. All outbound messages from CSS, i.e. synchronous responses to your API calls and all CSS messages sent via a webhook, will have a unique eventId generated.
eventSubscriptionId	string	UUID	36	The unique id relating to the event types available for a Market Participant to subscribe to
eventType	string	string	NM	The type of event the message is related to, e.g. RegistrationEvent, see section 4.3.9
fuelType	string	string	NM	See section 4.3.1
greenDealFromDate	string	datetime	See 4.2	The date at which the green deal starts for the RMP or market participant
greenDealToDate	string	datetime	See 4.2	The date at which the green deal ends for the

				RMP or market participant
interventionWindowStartDate	string	datetime	See 4.2	The date at which the intervention window starts
interventionType	string	string	NM	The type of intervention being requested see section 4.3.2
marketParticipantRoleEvent	string	string	NM	Type of event that can be recorded against a market participant role. See 4.3.14
meteringPointEnergyFlow	string	string	NM	The direction of energy flow, into or out of the property. Electricity only, see 4.3.4
meteringPointEnergyFlowFromDate	string	datetime	See 4.2	The date at which the change to the Energy Flow is considered to have commenced
meteringPointMeteredInd	boolean			true or false
mpid	string	string	NM	Market Participant Identifier
mpxn	string	string	NM	MPAN or MPRN
networkProvisionMpid	string	string	NM	The MPID of the organisation acting at the network provider
networkProvisionRole	string	string	NM	The role of the organisation acting at the network provider
networkProvisionFromDate	string	datetime	See 4.2	When the organisation became the network provider
pendingRegistrationId	string	UUID	36	The new Pending registration Id initiated from a switch request (UUID)
registrationInitiator	string	string	30	The context of the market participant who initiated the switch. See 4.3.12
registrationId	string	UUID	36	UUID
registrationActiveDate	string	datetime	See 4.2	The date the registration changes to Active
registrationInactiveDate	string	datetime	See 4.2	The date the registration changes to Inactive
registrationPermissionFromDate	string	datetime	See 4.2	The date from which the market participant is allowed to create registrations

registrationPermissionToDate	string	datetime	See 4.2	The date at which the market participant is prevented from creating registrations
registrationStatus	string	string	NM	See section 4.3.6
registrationStatusFromDate	string	datetime	See 4.2	The date the registration status changes
registrationRequestId	string	UUID	36	UUID
registrationRequestStatus	string	string	NM	See section 4.3.7
role	string	string	NM	Market Participant Role
rmpStatus	string	string	NM	See section 4.3.5
rmpStatusFromDate	string	datetime	See 4.2	The date and time that the RMP changed status
shipperMpid	string	string	NM	Shipper Market Participant Identifier
shipperRole	string	string	NM	Shipper Market Participant Role
shipperFromDate	string	datetime	See 4.2	The date the registered shipper changed
statusCode	number		3	The http status code
supplierGeneratedReference	string	string	NM	A supplier generated reference for a switch or initial registration. This is NOT used internally by CSS but will be echoed back on responses.
supplierGeneratedOfafGroupReference	string	string	NM	A supplier generated reference for an OFAF switch. This is NOT used internally by CSS but will be echoed back on responses.
supplierMpid	string	string	NM	Supplier Market Participant Identifier
supplierRole	string	string	NM	Supplier Market Participant Role
supplyStartDate	string	datetime	See 4.2	The date on which the Supplier is requesting to become the supplier of the RMP
timeBoundaryAlignmentStatus	string	string	30	See section 4.3.17
updatedProperties	string array		[100]	An array of properties that have changed in a payload. Denotes all properties which have changed on the payload object. For newly created

				objects this array will be empty.
uprn	number		12	Unique Property Reference Number
version	string	string	30	The version of the API being used

**effective June 2024 under R0044 MHHS*

4.1.1 MPL Address Mapping

(Mappings taken from MAP09 v1.7 - Standard Address Format and Guidance Notes for Address Maintenance)

Where the MPL address is referenced, the following lists the components of the payload and are sent out from CSS. The data is sent by CSS as an object in JSON, which is an unordered set of name/value pairs so may not come through in the order listed.

Field	Type	MAP09 Defined Element	Gas file format	Description
deliveryPointAlias	string	Metering Point Address Line 1	DELIVERY_POINT_ALIAS	Free format, for example plot details or Organisation Name
subBuildingNameOrNumber	string	Metering Point Address Line 2	SUB_BUILDING_NAME	Sub building name as defined in PAF
buildingName	string	Metering Point Address Line 3	BUILDING_NAME	Building name as defined in PAF
buildingNumber	string		BUILDING_NUMBER	Building number as defined in PAF
dependantThoroughfare	string	Metering Point Address Line 4	DEPENDANT_STREET	Dependant thoroughfare descriptor and or dependant thoroughfare name as defined in PAF
thoroughfare	string	Metering Point Address Line 5	PRINCIPAL_STREET	thoroughfare descriptor and or thoroughfare name as defined in PAF
doubleDependantLocality	string	Metering Point Address Line 6	DOUBLE_DEPENDANT_LOCALITY	Double dependant locality as defined in PAF
dependantLocality	string	Metering Point Address Line 7	DEPENDANT_LOCALITY	Dependant locality as defined in PAF
postTown	string	Metering Point Address Line 8	POST_TOWN	post town as defined in PAF
county	string	Metering Point Address Line 9	COUNTY	The county the postTown is located within

postcode	string	Metering Point Address Postcode	POSTCODE	postcode as defined in PAF which is a combination of in code and out code
uprn	number	n/a	n/a	Unique Property Reference Number

4.1.1.1 MPL Address Payload


```
{
        "deliveryPointAlias": "string",
        "subBuildingNameOrNumber": "string",
        "buildingName": "string",
        "buildingNumber": "string",
        "dependantThoroughfare": "string",
        "thoroughfare": "string",
        "doubleDependantLocality": "string",
        "dependantLocality": "string",
        "postTown": "string",
        "county": "string",
        "postcode": "string",
        "uprn": number
}
```

Landmark INFORMATION

4.1.2 REL Address Elements

Field	Type	Length	Description
primaryName	string	90	This is the Primary Addressable Object description. This is normally the name and or number of the property
secondaryName	string	90	This is the Secondary Addressable Object description, e.g. the "Flat 2" in the address "Flat 2, London House, Exeter". This is only relevant for a child property. "London House" in this case will be the Primary Name of the parent property
street1	string	100	LPI[2] - derived from Street DPA[3] - the Thoroughfare
street2	string	80	LPI – Blank DPA - dependant thoroughfare
locality1	string	35	LPI – derived from Street – Using locality code lookup DPA – dependant locality
locality2	string	35	LPI - Blank DPA – double dependant locality
town	string	30	LPI – Derived from Street – Using Town code lookup DPA – Post Town
postcode	string	8	Postcode associated with the address
logicalStatus #	number	1	This is the status of the address. See 4.3.16
language #	string	3	The language of the address (ISO 639-2 Code). For example, in Wales you will usually have an English and Welsh address. It will be cym for welsh.
organisation #	string	60	Current organisation name of the property if one exists
addressType #	string	3	The type of address of this entry in the array. See 4.3.15
confidenceScore #	number	3	A relative confidence score on the match from MPL to REL
classification #	string	6	Classification code of the property as per the AddressBase Premium classification scheme
latitude #	number	10,6	Latitude of the associated property, usually either the centroid of the building polygon or a general internal point within the building polygon
Longitude #	number	10,6	Longitude of the associated property, usually either the centroid of the building polygon or a general internal point within the building polygon

[2] See British Standard 7666

[3] The Delivery Point Address is derived from PAF (Postcode Address File) and identifies a property that receives deliveries from the Royal Mail

Where addressSource is “MPL” or “Manual”, these Address Elements will be NULL and only a single address will be included in the payload.

Where the MPL address is referenced, the following lists the components of the payload and are sent out from CSS. The data is sent by CSS as an object in JSON, which is an unordered set of name/value pairs so may not come through in the order listed.

Note: The mappings contained in the table above are representative of the data mappings as they are contained within OSABP and not intended as presentation mappings.

4.2 CSS Field Data Types

Field	Format	Description
string		Any printable characters.
string	datetime	Will be strings formatted as defined by ISO 8601 examples: (zulu or 0 offset) 2019-08-07T22:57:49.000Z 2019-08-07T22:57:49.000 Offset 1 hour ahead of Coordinated Universal Time (UTC) 2019-08-07T22:57:49.000+01:00 CSS will observe local time for Gate Closure and Supply Start Date. The format used is "date-time" please refer to RFC3339
string	UUID	These will conform to http://www.ietf.org/rfc/rfc4122.txt
boolean		true or false
number		Numbers must be an integer or a floating point

4.3 Enumerations

4.3.1 fuelType

Value	Description
G	Gas
E	Electricity

4.3.2 interventionType

Value	Description
Objection	Intervention being made is an Objection
NoObjection	There is no objection to the switch
Annulment	Intervention being made is an Annulment
Withdrawal	Intervention being made is a Withdrawal

4.3.3 associationType

Value	Description
relatedSecondaryMpxn	The association to the Primary MPxN is a related Secondary MPxN

4.3.4 energyFlow

Value	Description
I	Energy Flow is Import
E	Energy Flow is Export

4.3.5 rmpStatus

Value	Description
C	RMP Status is Created
O	RMP Status is Operational
T	RMP Status is Terminated
D	RMP Status is Dormant

4.3.6 registrationStatus

Value	Description
Pending	Registration Status is Pending
Active	Registration Status is Active
Confirmed	Registration Status is Confirmed
Inactive	Registration Status is Inactive
Cancelled	Registration Status is Cancelled
SecuredActive	Registration Status is Secured Active
SecuredInactive	Registration Status is Secured Inactive

4.3.7 registrationRequestStatus

Value	Description
Submitted *	Registration Request Status is Submitted
Validated	Registration Request Status is Validated
Rejected	Registration Request Status is Rejected

* Whilst a logical state of 'submitted' exists for this item it will never be communicated to market participants.

4.3.8 switchEventType

Value	Description
switchTriggered	Anticipated loss due to a switch being triggered
switchCancelled	Anticipated loss of a switch cancelled
registrationChangeOfShipper	Anticipated loss due to a shipper change

4.3.9 eventType

CSS Event Type	Description	Covers ABACUS Message
RegistrationValidationNotification	This notification message is sent as a result of an initial registration, change to a registration or a new switch	RECM_SN_CSS02380
RegistrationPendingNotification	This notification message is sent as a result of an initial registration or a new switch	RECM_SN_CSS02300
RegistrationPendingSynchronisation	This synchronisation message is sent as a result of a change in status to "Pending" for an initial registration or a switch	RECM_SN_CSS02400 RECM_SN_CSS02500 RECM_SN_CSS02800
RegistrationSecuredActiveNotification	This notification message is sent as a result of a change in status to "SecuredActive" for an initial registration or a switch	RECM_SN_CSS02340 RECM_SN_CSS02370
RegistrationSecuredActiveSynchronisation	This synchronisation message is sent as a result of a change in status to " SecuredActive" for an initial registration or a switch	RECM_SN_CSS02460 RECM_SN_CSS02560 RECM_SN_CSS02860 RECM_SN_CSS03060
RegistrationConfirmedNotification	This notification message is sent as a result of a change in status to "Confirmed" for a switch	RECM_SN_CSS02370

RegistrationSecuredInactiveNotification	This notification message is sent as a result of a change in status to "SecuredInactive" for a switch or deactivation	RECM_SN_CSS02370
RegistrationSecuredInactiveSynchronisation	This synchronisation message is sent as a result of a change in status to " SecuredInactive" for a switch or deactivation	RECM_SN_CSS02460 RECM_SN_CSS02560 RECM_SN_CSS02860 RECM_SN_CSS03060
GainingRegistrationCancelledNotification	This notification message is sent as a result of a change in status to "Cancelled" for a switch or intervention	RECM_SN_CSS02375
RegistrationCancelledNotification	This notification message is sent as a result of a change in status to "Cancelled" for a switch or intervention	RECM_SN_CSS02370
RegistrationCancelledSynchronisation	This synchronisation message is sent as a result of a change in status to " Cancelled" for a switch or intervention	RECM_SN_CSS02460 RECM_SN_CSS02560 RECM_SN_CSS02860 RECM_SN_CSS03060
InvitationToIntervene	This message is sent to the Losing Supplier after a new switch is validated	RECM_SN_CSS02700
RegistrationChangeAnticipatedNotification	This message is sent to losing participants as a result of an anticipated registration change (after invitation to intervene)	RECM_SN_CSS02390
RegistrationEventNotification	This notification message is being sent as a result of a change made to an existing registration	RECM_SN_CSS02350

RegistrationEventSynchronisation	This synchronisation message is being sent as a result of a change made to an existing registration	RECM_SN_CSS02450 RECM_SN_CSS02550 RECM_SN_CSS02850 RECM_SN_CSS03000 RECM_SN_CSS03050 RECM_SN_CSS03200
RMPEventSynchronisation	The synchronisation message is being sent as a result of a change made to an existing RMP or a new RMP being added in CSS	RECM_SN_CSS02900 RECM_SN_CSS03100 RECM_SN_CSS03300
RMPCreatedSynchronisation RMPOperationalSynchronisation RMPTerminatedSynchronisation RMPDormantSynchronisation	These synchronisation messages are sent as a result of a change made to the status of an existing RMP or a new RMP being added in CSS	RECM_SN_CSS02950
RetailEnergyLocationSynchronisation	This synchronisation message is sent as a result of a change to the Retail Energy Location (REL)	RECM_SN_CSS02600
WebhookVerificationEvent	The message is being sent as a result of a Market Participant registering a Webhook with the CSS	
SupplierArrangedAppointmentUpdateNotification*	Provides confirmations of Supplier Arranged Appointment (SAA) updates	RECM_SN_CSS07000

**effective from June 2024 under R0044 MHHS*

4.3.10 eventStatus

Value	Description
Ok	The message is being sent as a result of normal successful processing of the associated request event
Error	The message is being sent as a result of some failure during processing of the associated request event

4.3.11 contextType

Value	Description
LosingSupplier	Receiving message in the context of the Losing Supplier

GainingSupplier	Receiving message in the context of the Gaining Supplier
RegisteredSupplier	Receiving message in the context of the Registered Supplier
LosingShipper	Receiving message in the context of the Losing Shipper
GainingShipper	Receiving message in the context of the Gaining Shipper
RegisteredShipper	Receiving message in the context of the currently Registered Shipper
HHDA	Receiving message in the context of the Half Hourly Data Aggregator
HHDC	Receiving message in the context of the Half Hourly Data Collector
NHHDA	Receiving message in the context of the Non Half Hourly Data Aggregator
NHHDC	Receiving message in the context of the Non Half Hourly Data Aggregator
GRDA	Receiving message in the context of the Gas Retail Data Agent
ECOES	Receiving message in the context of the Electricity Central Online Enquiry Service
ERDA	Receiving message in the context of the Electricity Retail Data Agent
DCCSM	Receiving message in the context of the DCC Smart Metering Data Service
ECOS	Receiving message in the context of the Smart Metering Enduring Change of Supplier
MAP	Receiving message in the context of the Meter Asset Provider
MEM	Receiving message in the context of the Meter Equipment Manager
DES	Receiving message in the context of the Data Enquiry Service (Gas)
WebhookSubscription	Receiving message in the context of a Webhook event subscription

4.3.12 registrationInitiator

Value	Description
GRDA	Registration Initiated by GRDA
GainingSupplier	Registration Initiated by the Gaining Supplier

4.3.13 addressSource

Value	Description
MPL	The MPL address was not matched in the addressing service as a standard address, so one was created using the information of the MPL
Manual	Subsequent to the REL address being created, it has been discovered that the address was wrong and has been manually assigned by a human to an address within the addressing service
Match	The address is derived from a direct match with MPL provided to an address in the addressing service

4.3.14 marketParticipantRoleEvent

Value	Description
greenDeal	Green Deal qualified
registrationPermission	The market participant is able to register an RMP in the CSS

4.3.15 addressType

Value	Description
LPI	Land Property Identifier – The LPI is basically the address of the BLPU in a standard format that uniquely identifies the BLPU in relation to a street as defined and held in the National Street Gazetteer (NSG).

DPA	A Delivery Point Address (DPA) is defined as a property that receives deliveries from Royal Mail.

4.3.16 logicalStatus

Value	Description
0	N/A (for Delivery Point Address which does not have a logical status)
1	Approved
3	Alternative
6	Provisional
8	Historic

4.3.17 timeBoundaryAlignmentStatus

Value	Description
Settlement Day	Settlement is daily
Settlement Period	Settlement is intraday
Unaligned	The alignment is unknown

4.4 Element Mappings

4.4.1 RMP Gas/Electricity API

ABACUS Field	CSS Field
Supply Meter Point Reference Number	mpxn
MPAN Core	mpxn
RMP Network Provision Market Participant Identifier	networkProvisionMpid
RMP Network Provision Effective From Date	networkProvisionFromDate
RMP Lifecycle Status Type Identifier Reference	rmpStatus
RMP Lifecycle Status From Date	rmpStatusFromDate
RMP Event Effective From Date	greenDealFromDate
RMP Event Effective Through Date	greenDealToDate
RMP Time Boundary Alignment Status	timeBoundaryAlignmentStatus
Metering Point Metered Indicator	meteringPointMeteredInd
Metering Point Energy Flow	meteringPointEnergyFlow
Metering Point Energy Flow Effective From Date	meteringPointEnergyFlowFromDate
RMP CommsHub Link Device GUID	commsHubLinkDeviceId

4.4.2 RMP Associations API

ABACUS Field	CSS Field
Supply Meter Point Reference Number	mpxn
Primary MPAN Core	mpxn
RMP Association Type Identifier Reference	associationType
Secondary MPAN Core	associatedMpxn

4.4.3 RMP Asset Ownership MAP API

ABACUS Field	CSS Field
Supply Meter Point Reference Number	mpxn
MPAN Core	mpxn
RMP Asset Ownership MAP Identifier	mpid

4.4.4 Registration Gas/Electricity API

ABACUS Field	CSS Field
Registration Management Request Supplier Market Participant Role Identifier	supplierMpid
Registration Effective From Date	supplyStartDate
Registration Event Shipper Market Participant Role Identifier	shipperMpid

Registration Event Domestic Premises Indicator	domesticPremisesInd
Registration Event Registration Identifier	registrationId

4.4.5 Registration Supplier Arranged Appointments API

ABACUS Field	CSS Field
Supplier Arranged Appointment Market Participant Role Identifier	mpid
Supplier Arranged Appointment Registration Identifier	registrationId

4.4.6 Registration Switch Request API

ABACUS Field	CSS Field
Registration Effective From Date	supplyStartDate
Supply Meter Point Reference Number	mpxn
MPAN Core	mpxn
RMP Fuel Type	fuelType
Registration Management Request Supplier Market Participant Role Identifier	supplierMpid
Switch Request Change Of Occupancy Indicator	changeOfOccupancyInd
Registration Erroneous Switch Resolution Indicator	erroneousSwitchResolutionInd
Registration Event Shipper Market Participant Role Identifier	shipperMpid
Registration Event Domestic Premises Indicator	domesticPremisesInd
Switch Request Group One Fail All Fail Supplier Generated Group Reference	supplierGeneratedOfafGroupReference
Registration Management Request Supplier Generated Reference	supplierGeneratedReference
Registration GT Initiated Indicator	registrationInitiator

4.4.7 Registration Switch Intervention API

ABACUS Field	CSS Field
Registration Management Request Supplier Market Participant Role Identifier	mpid
Supply Meter Point Reference Number	mpxn
MPAN Core	mpxn
Registration Intervention Reason (2200)/ Withdrawal Request	interventionType

Target Registration Request Identifier (2100)	
Registration Identifier (2100)/ Registration Intervention Pending Registration Identifier (2200)	pendingRegistrationId

4.4.8 Registration Deactivation API

ABACUS Field	CSS Field
Registration Management Request Supplier Market Participant Role Identifier	mpid
Supply Meter Point Reference Number	mpxn
MPAN Core	mpxn
Registration Identifier	registrationId

4.4.9 Domain Data API

ABACUS Field	CSS Field
Domestic Objection Window Duration	domesticObjectionWindowDuration
Domestic Objection Window Duration Unit of Measure	domesticObjectionWindowDurationUnitofMeasure
Non Domestic Objection Window Duration	nonDomesticObjectionWindowDuration
Non Domestic Objection Window Duration Unit Of Measure	nonDomesticObjectionWindowDurationUnitOfMeasure
Registration Effective Time Of Day	registrationEffectiveTimeOfDay
Registration Effective Time Of Day TimeZone	registrationEffectiveTimeOfDayTimeZone
Execution Window Duration	executionWindowDuration
Execution Window Duration Unit Of Measure	executionWindowDurationUnitOfMeasure
Maximum Switch Period Note: An additional day will be added internally to the value of this SDD item to ensure validation aligns with DB4/REC i.e. for a maximum switch period value of 28 days, CSS will hold a value of 29 to ensure the full period is available to Suppliers.	maximumSwitchPeriod
Maximum Switch Period Unit Of Measure	maximumSwitchPeriodUnitOfMeasure
Domestic Minimum Switch Period	domesticMinimumSwitchPeriod
Domestic Minimum Switch Period Unit Of Measure	domesticMinimumSwitchPeriodUnitOfMeasure
Non Domestic Minimum Switch Period	nonDomesticMinimumSwitchPeriod:number
Non Domestic Minimum Switch Period Unit Of Measure	nonDomesticMinimumSwitchPeriodUnitOfMeasure
DCC Serviced Meter Standstill Duration	DCCServicedMeterStandstillDuration

DCC Serviced Meter Standstill Duration Unit Of Measure	DCCServicedMeterStandstillDurationUnitOfMeasure
Non DCC Serviced Meter Standstill Duration	nonDCCServicedMeterStandstillDuration
Non DCC Serviced Meter Standstill Duration Unit Of Measure	nonDCCServicedMeterStandstillDurationUnitOfMeasure
Registration Request Mode	registrationRequestMode
Annulment Processing Indicator	annulmentProcessingInd
COO Objection Processing Indicator	COOObjectionProcessingInd
Calendar Date	calendarDate
RMP Event Type Name	name
RMP Event Type Description	description
RMP Association Type Name	name
RMP Association Type Description	description
RMP Lifecycle Status Type Name	name
RMP Lifecycle Status Type Description	description
Registration Lifecycle Status Type Name	name
Registration Lifecycle Status Type Description	description
Market Participant Role Event Type	name
Market Participant Role Event Type	description
Market Alliance Constituent Nominator Market Role	marketAllianceConstituentNominatorMarketRole
Market Alliance Constituent Nominee Market Role	marketAllianceConstituentNomineeMarketRole

4.4.10 Market Participant API

ABACUS Field	CSS Field
Retail Energy Company Group Name	companyGroupName
Retail Energy Company Name	companyName
Retail Energy Company Registration Number	companyRgistrationNumber
Market Participant Role Event Effective From Date	greenDealFromDate
Market Participant Role Event Effective Through Date	greenDealToDate
Market Participant Role Event Effective From Date	registrationPermissionFromDate
Market Participant Role Event Effective Through Date	registrationPermissionToDate
Retail Energy Company Registration Authority	companyRegistrationAuthority
Market Role Name	role

Market Role Description	description
Market Participant Role MPID	mpid
Market Participant Role Market Role Name	roleName
Market Participant Role Alliance Nominator Participant	allianceNominatorMpid
Market Participant Role Alliance Nominee Participant	allianceNomineeMpid
Market Participant Role Alliance Nominator Participant	allianceNominatorRole
Market Participant Role Alliance Nominee Participant	allianceNomineeRole
Market Participant Role Alliance Effective From Date	allianceFromDate
Market Participant Role Alliance Effective Through Date	allianceToDate

4.5 Errors Payload

The errors data payload will be sent out to Market Participants as a result of errors within the CSS and also for receiving errors into CSS via the errors API. The http status code for a synchronous response will be the same as the statusCode property within the error object as defined below. Note: There could be multiple error objects in the "errors" element depending on the event in error.

```
{
        "errors": [ {
                "statusCode": 400,
                "errorCode": "V1200",
                "errorTitle": "The switch request is invalid",
                "errorDescription": "Property x is invalid, max length is 30 chars", (optional)
                "errorType": "https://cssProblems.net/errorCode/V1200" (optional)
            }
        ]
}
```

To accommodate OFAF switch requests, where more than a single mpxn is involved, CSS will return errors in an array against each mpxn and indicate this using a single error payload at the end of the array.

```
"data": [
        {
          "registrationRequestId": "xxx",
          "registrationRequestStatus": "Rejected",
          "mpxn": "aaaaaaaaaaaa",
          "supplierGeneratedReference": "xxx",
          "supplierGeneratedOfafGroupReference": "xxx",
          "errors": [
            {
              "statusCode": 404,
              "errorCode": "1080",
              "errorTitle": "Pending Registration ID not known",
              "errorDescription": "An RMP with Mpxn 'aaaaaaaaaaaa' could not be found",
              "errorType": "https://devportal.centralswitchingservice.co.uk/errorCodes#1080"
            },
            {
              "statusCode": 400,
              "errorCode": "1156",
              "errorTitle": "Supply Start Date falls within the objection window",
              "errorDescription": "The supply start date 2020-11-08T00:00:00.000+00:00 must be after the
objection window closure date of ",
              "errorType": "https://devportal.centralswitchingservice.co.uk/errorCodes#1156"
            }
          ]
        },
        {
          "registrationRequestId": "xxx",
          "registrationRequestStatus": "Rejected",
          "mpxn": "xxxxxxxxxx",
          "supplierGeneratedReference": "xxx",
          "supplierGeneratedOfafGroupReference": "xxx",
          "errors": [
            {
```

```
            "statusCode": 404,
            "errorCode": "1080",
            "errorTitle": "Pending Registration ID not known",
            "errorDescription": "An RMP with Mpxn 'xxxxxxxxxx' could not be found",
            "errorType": "https://devportal.centralswitchingservice.co.uk/errorCodes#1080"
          },
          {
            "statusCode": 400,
            "errorCode": "1156",
            "errorTitle": "Supply Start Date falls within the objection window",
            "errorDescription": "The supply start date 2020-11-08T00:00:00.000+00:00 must be after the
objection window closure date of ",
            "errorType": "https://devportal.centralswitchingservice.co.uk/errorCodes#1156"
          }
        ]
      },
      {
        "registrationRequestId": "xxx",
        "registrationRequestStatus": "Rejected",
        "mpxn": "yyyyyyyyyyyyy",
        "supplierGeneratedReference": "xxx",
        "supplierGeneratedOfafGroupReference": "xxx",
        "errors": [
          {
            "statusCode": 404,
            "errorCode": "1080",
            "errorTitle": "Pending Registration ID not known",
            "errorDescription": "An RMP with Mpxn 'yyyyyyyyyyyyy' could not be found",
            "errorType": "https://devportal.centralswitchingservice.co.uk/errorCodes#1080"
          },
          {
            "statusCode": 400,
            "errorCode": "1156",
            "errorTitle": "Supply Start Date falls within the objection window",
            "errorDescription": "The supply start date 2020-11-08T00:00:00.000+00:00 must be after the
objection window closure date of ",
            "errorType": "https://devportal.centralswitchingservice.co.uk/errorCodes#1156"
          }
        ]
      }
    ],
    "errors": [
      {
        "statusCode": 400,
        "errorCode": "1154",
        "errorTitle": "OFAF switch failed",
        "errorDescription": "3 of 3 registrations in the OFAF group failed validation",
        "errorType": "https://devportal.centralswitchingservice.co.uk/errorCodes#1154"
      }
    ]
```


For non-OFAF switch requests (which contain a single request) any errors will be returned in the outer error block.

5 CSS API specification

5.1 General Principles

In the sections below, where the field is mandatory in the body, the item is bold and commented with *.

All data passed to CSS should already have had any padding removed.

5.1.1 POST

Use POST when you want to create a new resource or initiate a process.

5.1.2 PATCH

Use PATCH when you want to update a subset of a resource.
When sending a PATCH, the fields you wish to “Update” should be sent in the “data” element. You only need to add the properties in the data element that you wish to update, you do not need to send the whole entity. Some properties in the data element may have a dependency on your Role within the CSS.

If a data item is not supplied, it will not be updated.

data:

```
{
        "item1": "item1value"
}
```

If an object in the system looks like the example above:

- Item 1 is an item that just has a value
    - To set item1 to not have a value, send "item1": null
    - To update item1 to have a new value send "item1": "new value"

Use PATCH when you want to delete a resource on a /delete route.

5.1.3 PUT

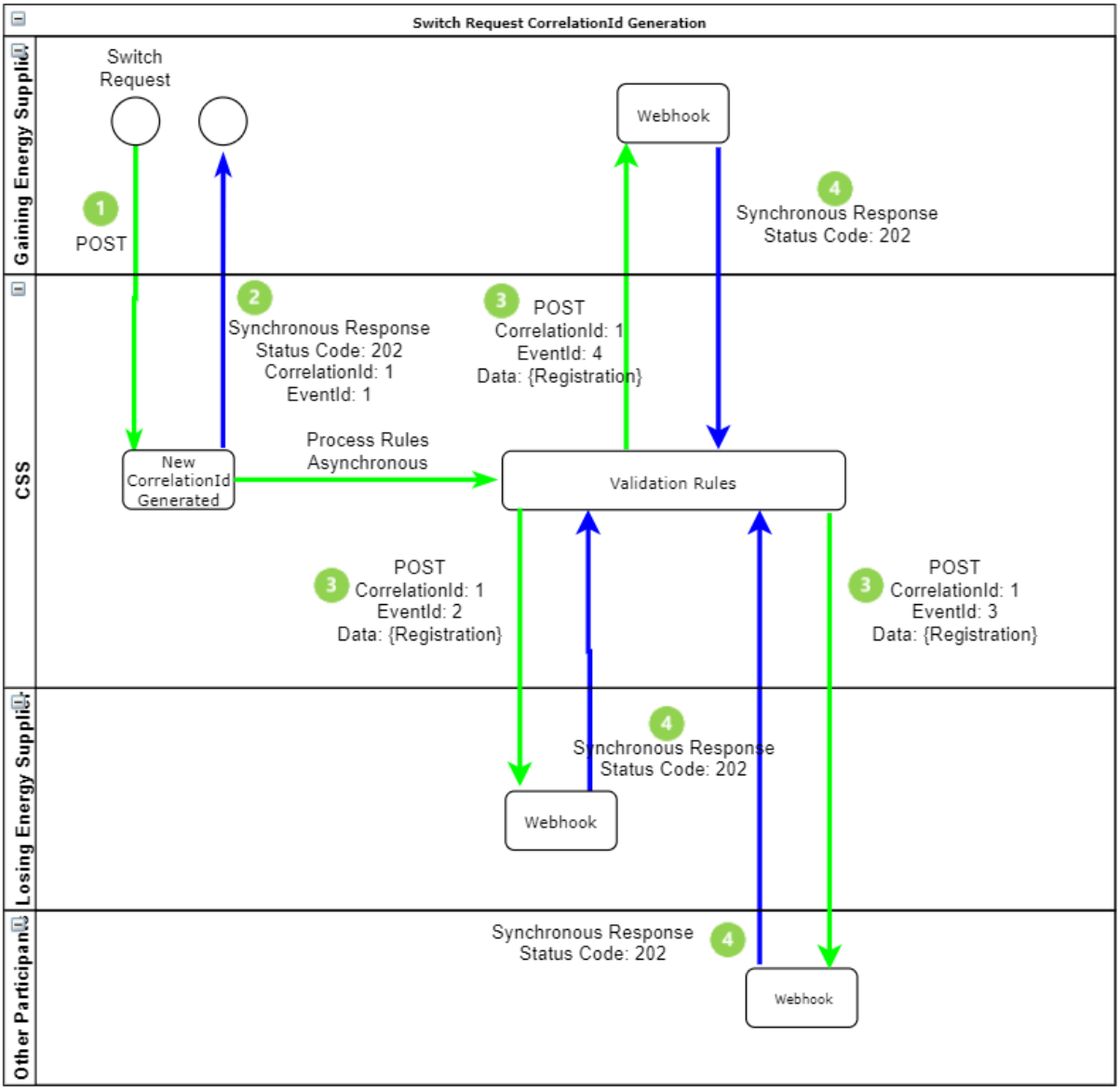
Use PUT when you need to replace a resource in its entirety.
Use of the PUT method essentially updates the given resource at the request URI. If the request URI refers to an already existing resource – an update operation will happen, otherwise a create operation should happen if the request URI is a valid resource URI.

5.1.4 CorrelationId and EventId

The correlationId will appear in HTTP headers, standard responses, and will also be sent out as part of the webhook payload for related messages.

In HTTP headers it will appear in the response field X-Correlation-Id
e.g. `X-Correlation-Id: f058ebd6-02f7-4d3f-942e-904344e8cde5`

The correlationId can be used by Market Participants to relate messages that are part of the same workflow.
The below is a depiction of how the correlationId and eventId would flow in an asynchronous scenario.

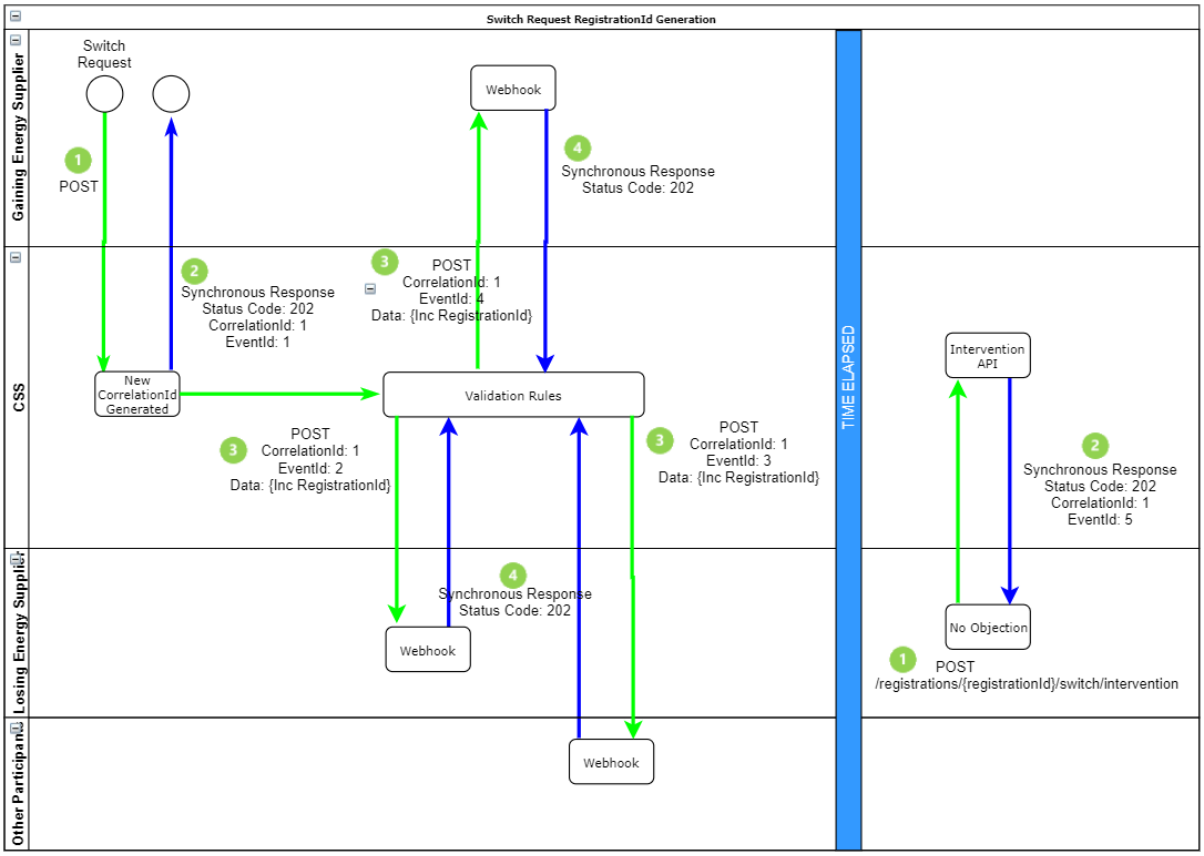


5.1.5 RegistrationId

The CSS will use registration identifiers (registrationId) to uniquely identify each registration. CSS will generate a new registrationId upon receipt of an initial registration or a switch request. All market participants who receive registration lifecycle events will receive the registrationId as part of the Registration Data Payload.

If the losing supplier wishes to object to the proposed switch, they will have to use the newly generated registrationId when intervening, see section 5.6.5. Other market participants will need to store the registrationId for future updates from the CSS and to update the CSS regarding a registration.

The below diagram depicts the basic flow of when and where the registrationId is generated and propagated. The adoption or creation of registrationIds when migrating data prior to go-live will be covered in the data migration document.



5.2 Standard Response Body

This is the standard response body sent out synchronously by CSS whenever a message is received by CSS. The errors property will only be populated if the request is in error.

Body:

{
"**version**": "string", --*
"**correlationId**":"UUID", --*
"**eventId**":"UUID", --*
"**eventDate**":"string", --*
"errors": [{Object see section 4.5}]
}

5.3 Error API Request

This is the request body expected by the CSS if there is an error to report after processing.

Available to: Any market participant known to CSS

Route: /errors
Method: POST
Body:
{
"**eventId**": "string", --*
"**correlationId**": "string", --*
"errors": [{Object see section 4.5}]
}
Note: The eventId should be the eventId sent as response to the request which initiated the error.

Response:
Status Codes: 201, 400, 401, 403, 429, 500, 503
Body:
{
Standard Response Body
}

5.4 Webhook API

This API is used to subscribe, unsubscribe and view the Webhooks associated with a Market Participant and each role they perform for each eventType available to them, see section 7.2. As the available event types are mastered by CSS, Market Participants will need to call the GET endpoint first to get the list of events they can subscribe to.

The general subscription flow will be

1. Call the POST endpoint to get the event types available for you to subscribe to
2. Call the POST endpoint to subscribe to **all** the mandatory events and any optional events
3. Use the PATCH and PUT endpoints to manage your subscriptions

You can use the same Webhook URL for all event subscriptions if you wish or have separate Webhook URLs for each event subscription.

On registering Webhook URLs , CSS will send WebhookVerificationEvent messages to those Webhook URLs to verify they are valid HTTPS endpoints, if a 202 is not returned for all those requests, or a request times out, the whole subscription will fail. You must then correct any errors and try to register again, until the verification succeeds. The verification body will follow the standard Webhook structure defined in section 6.

Outgoing messages from CSS will be POSTed to the Webhook URLs defined against each eventType you have subscribed to.

There is no need to send in your MPID when using this API as CSS will derive this information from your initial registration to CSS see section 3.4.3. CSS also holds information regarding the Roles associated with your MPID.

Available to: All Market Participants.

This endpoint provides all the event types available for a Market Participant to subscribe to and whether those events have been subscribed to or not.
Route: /system/webhook/mpid
Method: POST
Responses:
Status Codes: 200
Body:

```
{
        Standard Response Body,
        "data":[ {
                "eventSubscriptionId": "string", --*
                "mpid": "string", --*
                "role": "string", --*
                "eventType": "string", --*
                "mandatory": boolean, --*
                "webhookUrl": "string"
        }]
}
```

Status Codes: 400, 401, 403, 429, 500, 503
Body:

```
{
```

Standard Response Body
}

The POST endpoint is used to subscribe your webhook URLs against each eventType you wish to register for.
Note: Some event types are mandatory for certain roles, you must subscribe to all of these, failure to do so will result in an error response. You can get the list of event types available to you from the POST endpoint above.

Route: /system/webhook
Method: POST
Body:
[{
"**eventSubscriptionId**": "string", --*
"webhookUrl": "string" --*
}]
Response:
Status Codes: 201, 400, 401, 403, 429, 500, 503
Body:
{
Standard Response Body
}

This endpoint is used to unregister an event subscription.
Route: /system/webhook/{eventSubscriptionId}/delete
Method:-PATCH
Body:
{
}

Response:
Status Codes: 200, 400, 401, 403, 404, 429, 500, 503
Body:
{
Standard Response Body
}
Note: An error will be returned if an attempt is made to Delete a mandatory event subscription. Use the PUT below if you want to change the Webhook URL for an event subscription.

This endpoint is used to update a Webhook URL associated to an event subscription.
Route: /system/webhook/{eventSubscriptionId}
Method: PUT
Body:
{
"**webhookUrl**": "string" --*
}
Response:
Status Codes: 200, 400, 401, 403, 404, 429, 500, 503
Body:
{
Standard Response Body
}

This endpoint is used to confirm to CSS the outcome of processing any successfully received webhook message. Although the API definition supports any webhook message the initial implementation is limited to DCCSM confirmation of Supplier Arranged Appointment updates received via a RegistrationEventSynchronisation message . This endpoint will not become effective until June 2024 under R0044 MHHS.

Messages id:
RECM_SN_CSS06000

Available: *DCCSM*

Route*: /system/webhook/message/confirmation*
Method*: POST*
Body*:*
{
*"**eventId**": "string", --**
*"**correlationId**": "string", --**
"errors": [{error object}]
}

Response:
Status Codes*: 200, 400, 401, 403, 429, 500, 503*

Body:
{
Standard Response Body
}

5.5 RMP API Request

A request would be made to this API when creating a new RMP (POST) or updating information belonging to that RMP (PATCH).

Available to: GRDA, ERDA, DCC SM
Availability Notes:

- Only GRDA/ERDA can create (POST)
- GRDA/ERDA can update (PATCH) all fields bar commsHubLinkDeviceId
- DCC SM can only update (PATCH) the commsHubLinkDeviceId field

5.5.1 Gas

Abacus messages covered:
RECM_SN_CSS00100
RECM_SN_CSS01100
RECM_SN_CSS01200
RECM_SN_CSS01300
RECM_SN_CSS01400
RECM_SN_CSS01600

Route: /rmps/gas
Method: POST
Body:

```
{
        "mpxn": "string", --*
        "networkProvisionMpid ": "string", --*
        "networkProvisionRole":  "string", --*
        "networkProvisionFromDate":"string", --*
        "rmpStatus": "string", --*
        "rmpStatusFromDate": "string", --*
        "mplAddress": {MPLAddress}, -- *
        "timeBoundaryAlignmentStatus": "string",
        "dccServiceInd": boolean,
        "assetOwnershipMaps: [{
                "mpid": "string" -- *,
                "role": "string" --*
        }]
}
```

Response:
Status Codes: 202, 400, 401, 403, 429, 500, 503
Body:

```
{
        Standard Response Body
}
```

Route: /rmps/gas
Method: PATCH

Body:

```
{
```

"**mpxn**": "string", --*
"**data**": { --*
"networkProvisionMpId": "string",
"networkProvisionRole": "string",
"networkProvisionFromDate":"string",
"rmpStatus": "string",
"rmpStatusFromDate": "string",
"mplAddress": {MPLAddress},
"commsHubLinkDeviceId": "string",
"timeBoundaryAlignmentStatus": "string",
"dccServiceInd": boolean
}
}

Response:
Status Codes: 202, 400, 401, 403, 404, 429, 500, 503
Body:
{
Standard Response Body
}

5.5.2 Electricity

Abacus messages covered:
RECM_SN_CSS00100
RECM_SN_CSS00300
RECM_SN_CSS00400
RECM_SN_CSS00600
RECM_SN_CSS00700
RECM_SN_CSS01000

Route: /rmps/electricity
Method: POST
Body:

```
{
        "mpxn": "string", --*
        "networkProvisionMpid":  "string", --*
        "networkProvisionRole":  "string", --*
        "networkProvisionFromDate":"string", --*
        "meteringPointEnergyFlow":  "string", --*
        "meteringPointEnergyFlowFromDate":  "string", --*
        "meteringPointMeteredInd": boolean,
        "rmpStatus": "string", -- *
        "rmpStatusFromDate": "string", -- *
        "mplAddress": {MPLAddress}, -- *
        "timeBoundaryAlignmentStatus": "string",
        "dccServiceInd": boolean,
        "greenDealFromDate": "string",
        "greenDealToDate": "string",
        "assetOwnershipMaps: [{
                "mpid": "string", -- *
                "role": "string" -- *
        }],
        "associations: [{
                "associationType": "string", -- *
                "associatedMpxn": "string" -- *
        }]
}
```

Responses:

Status Codes: 202, 400, 401, 403, 429, 500, 503
Body:

```
{
        Standard Response Body
}
```

Route: /rmps/electricity
Method: PATCH
Body:

```
{
        "mpxn": "string", --*
        "data": {--*
```

```
                "dccServiceInd": boolean,
                "networkProvisionMpid":  "string",
                "networkProvisionRole":  "string",
                "networkProvisionFromDate":  "string",
                "meteringPointEnergyFlow":  "string",
                "meteringPointEnergyFlowFromDate":  "string",
                "meteringPointMeteredInd": boolean,
                "greenDealFromDate": "string",
                "greenDealToDate": "string",
                "rmpStatus": "string",
                "rmpStatusFromDate": "string",
                "mplAddress": {MPLAddress},
                "commsHubLinkDeviceId": "string",
                "timeBoundaryAlignmentStatus": "string"
        }
}
```


Response:
Status Codes: 202, 400, 401, 403, 404, 429, 500, 503
Body:


```
{
        Standard Response Body
}
```

5.5.3 Associations

These endpoints are used to update the associations for the RMP. It enables them to be managed as a separate entity if required. The CSS needs to be informed of all the current active Associations whenever they change. To clear the array of Associations completely, send an empty array on the request body i.e. "data": [].

Abacus messages covered:
RECM_SN_CSS00500

Available to: ERDA

Route: /rmps/associations
Method: PATCH
Body:

```
{
        "mpxn": "string", --*
        "data": [{ --*
                "associationType": "string", -- *
                "associatedMpxn": "string" -- *
        }]
}
```

Responses:
Status Codes: 200, 400, 401, 403, 404, 429, 500, 503
Body:

```
{
        Standard Response Body
}
```

5.5.4 Asset Ownership MAP

These endpoints are used to update the asset ownership MAPs for the RMP. It enables them to be managed as a separate entity if required. The CSS needs to be informed of all the current active Asset Ownership MAPs whenever they change. To clear the array of Asset Ownership MAPs completely, send an empty array on the request body i.e. "data": []

Abacus messages covered:
RECM_SN_CSS00400
RECM_SN_CSS01200

Available to: GRDA, ERDA

Route: /rmps/assetownershipmaps
Method: PATCH
Body:

```
{
        "mpxn": "string", --*
        "data": [{ --*
                "mpid": "string", -- *
                "role": "string" --*
        }]
}
```

Responses:
Status Codes: 200, 400, 401, 403, 404, 429, 500, 503
Body:

```
{
        Standard Response Body
}
```

5.6 Registration API Request

This API will be used to add (POST) initial registrations and (PATCH) updates to existing registrations.

5.6.1 Initial registration

Abacus messages covered:
RECM_SN_CSS01700
RECM_SN_CSS01750

Available to: Energy Suppliers, GRDA

Route: /registrations
Method: POST
Body:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "supplierMpid": "string", --*
        "supplierRole": "string", --*
        "supplyStartDate": "string", --*
        "domesticPremisesInd": boolean, --*
        "supplierGeneratedReference": "string",
        "shipperMpid": "string", -**
        "shipperRole": "string" --**
}
```

** required for Gas

Response:
Status Codes: 202, 400, 401, 403, 429, 500, 503
Body:

```
{
        Standard Response Body
}
```

5.6.2 Update Current Active Registration

Provide any of the properties in the "data" object below that you wish to be updated, some properties may be dependent on your Role within the CSS.

Abacus messages covered:
RECM_SN_CSS02000

Available to: Energy Suppliers
Availability Notes:
All Energy Suppliers can update (PATCH) the domesticPremisesInd property.
Only Gas Energy Suppliers can update (PATCH) the shipperMpid, shipperRole and shipperFromDate property.

It is not possible to cancel an Update Current Active Registration request where a future dated change of shipper has been requested.

Future-dated shipper requests are actioned in shipperFromDate Datetime order, so shipperFromDate Datetimes must be used to sequence corrections.

For a correction that must be made on the same calendar day as an already-submitted request, add a minimum of 5 minutes to the time element of the shipperFromDate to ensure that the correction is queued, and therefore actioned, later than the original request.

Route: /registrations/{registrationId}
Method: PATCH
Body:

```
{
        "mpxn": "string", --*
        "data": {--*
                "shipperMpid": "string",
                "shipperRole": "string",
                "shipperFromDate": "string",
                "domesticPremisesInd": boolean
        }
}
```

Response:
Status Codes: 202, 400, 401, 403, 404, 429, 500, 503
Body:

```
{
        Standard Response Body
}
```

5.6.3 Supplier Arranged Appointments

These endpoints are used to update the supplier arranged appointments for the registration. It enables each appointment to be managed as a separate entity if required. The CSS needs to be informed of all the current active Supplier Arranged Appointments whenever they change. To clear the array of supplier arranged appointments completely, send an empty array on the request body i.e. []

Abacus messages covered:
RECM_SN_CSS00200

Available to: GRDA, ERDA

Route: /registrations/{registrationId}/supplierarrangedappointments
Method: PUT
Body:

```
[{
        "mpid": "string", --*
        "role": "string", --*
        "fromDate": "string", --*
        "toDate": "string"
}]
```

Response:
Status Codes: 200, 400, 401, 403, 404, 429, 500, 503
Body:

```
{
        Standard Response Body
}
```

Query string parameters listed below are supported for this API.*

Parameter name	Data Type	Usage
confirmationRequired	boolean	**true** – API caller requests confirmation that the update to appointments have been applied successfully or not. **false** – API caller does not expect a confirmation message for the update. **omitted** – API caller does not expect a confirmation message for the update

The confirmationRequired parameter is optional. Usage is limited to:

- ERDA use the parameter to indicate whether a confirmation message is expected from DCCSM for a SAA update affecting MDR role appointments only.
- GRDA cannot use the above parameter.

**effective from June 2024 under R0044 MHHS*

5.6.4 Switch API Request

A request is made to this API by an energy supplier when they wish to initiate a registration switch. To achieve a One Fail All Fail (OFAF) switch, all MPxN related properties involved in the OFAF need to be included in the registrations property array. All entries in the "registrations" property array are presumed to form an OFAF grouping. There is no way to inform the CSS of multiple switches using the registrations property array without creating an OFAF. To perform an individual MPxN switch simply provide one entry in the "registrations" property array.

Abacus messages covered:
RECM_SN_CSS01800

Available to: Energy Suppliers

Route: /registrations/switch
Method: POST
Body:

```
{
        "supplyStartDate": "string", --*
        "supplierGeneratedOfafGroupReference": "string",
        "registrations": [{--*
                "mpxn": "string", --*
                "fuelType":  "string", -- *
                "supplierMpid": "string", -- *
                "supplierRole": "string", --*
                "changeOfOccupancyInd":  boolean, -- *
                "erroneousSwitchResolutionInd ": boolean, -- *
                "domesticPremisesInd": boolean, -- *
                "shipperMpid": "string", -- **
                "shipperRole": "string", -- **
                "supplierGeneratedReference": "string"
        }]
}
```

**mandatory for gas only

"registrations": Is a mandatory array of the registrations to switch, at least 1, if there are more than 1, then these will form an OFAF group with an auto generated group id generated by CSS.

Response:
Status Codes: 202, 400, 401, 403, 429, 500, 503
Body:

```
{
        Standard Response Body
}
```

5.6.5 Switch Intervention API Request

A request is made to this API when an energy supplier needs to object, annul or withdraw a switch.

Abacus messages covered:
RECM_SN_CSS02100
RECM_SN_CSS02200

Available to: Gaining Supplier, Losing Supplier,
Availability Notes:

- Losing Supplier can use interventionType of “Objection”, "NoObjection" or “Annulment”
- Gaining Supplier can use interventionType of “Withdrawal”

Route: /registrations/{pendingRegistrationId}/switch/intervention
Method: POST
Body:
{
"**mpxn**": "string", --*
"**interventionType**": "string" --*

}

Responses:
Status Codes: 202, 400, 401, 403, 404, 429, 500, 503
Body:
{
Standard Response Body
}

5.6.6 Registration Deactivation API Request

This request is made by energy suppliers to indicate that the registration is no longer active and is not being switched to another supplier.

Abacus messages covered:
RECM_SN_CSS01900

Available to: Energy Suppliers

Route: /registrations/{registrationId}/deactivation
Method: POST
Body:
{
"**mpxn**": "string" --*
}
Responses:
Status Codes: 202, 400, 401, 403, 404, 429, 500, 503
Body:
{
Standard Response Body
}

5.6.7 Registration Status Refresh API Request

Request a re-send of a registration status synchronisation message as appropriate to the market participant role provided.

The message delivered is a resend of the original message previously sent to the requesting party with the original message EventId value replaced with a newly created unique EventId value specific to the resend request. If multiple API calls are made to resend the same original message, each resent message will contain different EventId values. All other original message attributes are included and unchanged in a resent message.

The following registration status messages are supported:

- RegistrationSecuredActiveSynchronisation
- RegistrationSecuredInactiveSynchronisation
- RegistrationCancelledSynchronisation

If the optional correlationId is provided, the message returned will reflect the registration status as set in the context of the business process associated with the correlationId otherwise the message returned will reflect the current registration status within CSS.

Message Id:
RECM_SN_CSS06010

Available to: GRDA

Route: /registrations/{registrationId}/statusrefresh
Method: POST
Body:

```
{
        "mpxn": "string" --*,
        "role": "string" --*,
        "correlationId": "string"
}
```

Responses:
Status Codes: 202, 400, 401, 403, 404, 429, 500, 503
Body:

```
{
        Standard Response Body
}
```

5.6.8 Active Registration Refresh API Request

Request a registration synchronisation message (RegistrationEventSynchronisation) for the current active registration for a RMP as appropriate to the market participant role provided.

Message Id:
RECM_SN_CSS06020

Available to: GRDA

Route: /registrations/activerefresh
Method: POST
Body:
{
"**mpxn**": "string" --*,
"**role**": "string" --*
}

Responses:
Status Codes: 202, 400, 401, 403, 429, 500, 503
Body:
{
Standard Response Body
}

5.7 Market Participant Data API

5.7.1 Market Role

Abacus messages covered:
RECM_SN_CSS05100

Available to:-RECDDG

Route: /domain/marketrole
Method: POST
Body:
{
"**role**": "string", --*
"**description**": "string", --*
}

Response:
Status Codes: 201, 400, 401, 403, 409, 429, 500, 503
Body:
{
Standard Response Body,
}

Route: /domain/marketrole/{role}
Method: PATCH
Body:
{
"**description**": "string", --*
}

Response:
Status Codes: 200, 400, 401, 403, 404, 429, 500, 503
Body:
{
Standard Response Body
}

Route: / domain/marketrole/{role}/delete
Method: PATCH
Body:
{
}

Response:
Status Codes: 200, 400, 401, 403, 404, 429, 500, 503
Body:
{
Standard Response Body
}

5.7.2 Market Participant Role

Abacus messages covered:
RECM_SN_CSS04200
RECM_SN_CSS05200

Available to: RECDDG

Route: /domain/marketparticipantrole
Method: POST
Body:

```
{
        "mpid": "string", --*
        "role": "string", --*
        "companyRegistrationNumber": "string" --*
        "ofafCompanyGroupId": "string",
        "greenDealFromDate": "string",
        "greenDealToDate": "string",
        "registrationPermissionFromDate": "string",
        "registrationPermissionToDate": "string"
}
```

Response:
Status Codes: 201, 400, 401, 403, 409, 429, 500, 503
Body:

```
{
        Standard Response Body
}
```

Route: /domain/marketparticipantrole/mpid/{mpid}/role/{role}
Method: PATCH
Body:

```
{
        "companyRegistrationNumber": "string",
        "ofafCompanyGroupId": "string",
        "greenDealFromDate": "string",
        "greenDealToDate": "string",
        "registrationPermissionFromDate": "string",
        "registrationPermissionToDate": "string"
}
```

Response:
Status Codes: 200, 400, 401, 403, 404, 429, 500, 503
Body:

```
{
        Standard Response Body
}
```

Route: / domain/marketparticipantrole/mpid/{mpid}/role/{role}/delete
Method: PATCH
Body:



{
}

Response:
Status Codes: 200, 400, 401, 403, 404, 429, 500, 503
Body:
{
Standard Response Body
}

5.7.3 Alliance

Abacus messages covered:
RECM_SN_ CSS00250
Available to: ERDA, GRDA

Route: /domain/alliance
Method: POST
Body:
{
"**allianceId**":"string", --*
"**allianceNominatorMpid**": "string", -- *
"**allianceNominatorRole**": "string", -- *
"**allianceNomineeMpid**": "string", -- *
"**allianceNomineeRole**": "string", -- *
"**allianceFromDate**":"string", --*
"allianceToDate":"string"
}

Response:
Status Codes: 201, 400, 401, 403, 409, 429, 500, 503
Body:
{
Standard Response Body,
}

Route: /domain/alliance/{allianceId}
Method: PATCH
Body:
{
"allianceFromDate":"string",
"allianceToDate":"string",
}

Response:
Status Codes: 200, 400, 401, 403, 404, 429, 500, 503
Body:
{
Standard Response Body
}

Route: /domain/alliance/{allianceId}/delete

Method: PATCH
Body:
{
}

Response:
Status Codes: 200, 400, 401, 403, 404, 429, 500, 503
Body:
{
Standard Response Body
}

5.7.4 Company

Abacus messages covered:
RECM_SN_CSS05000

Available to: RECDDG

Route: /domain/company
Method: POST
Body:
{
"**companyName**": "string", --*
"**companyRegistrationNumber**": "string", --*
"**companyRegistrationAuthority**": "string" --*
}

Response:
Status Codes: 201, 400, 401, 403, 409, 429, 500, 503
Body:
{
Standard Response Body
}

Route: /domain/company/companyRegistrationNumber/{companyRegistrationNumber}/companyRegistrationAuthority/{companyRegistrationAuthority}
Method: PATCH
Body:
{
"companyName": "string",
}

Response:
Status Codes: 200, 400, 401, 403, 404, 429, 500, 503
Body:
{
Standard Response Body
}

Route: /domain/company/companyRegistrationNumber/{companyRegistrationNumber}/companyRegistrationAuthority/{companyRegistrationAuthority}/delete
Method: PATCH
Body:
{
}

Response:
Status Codes: 200, 400, 401, 403, 404, 429, 500, 503
Body:
{
Standard Response Body
}

5.8 Resend Undelivered Messages API

5.8.1 Resend Undelivered Notifications

Permits Market Participants to request the re-send of undelivered notification messages.

Message Id:
RECM_SN_CSS06030

Available to: Any market participant receiving messages from CSS.

Route: /system/messages/resend
Method: POST
Body:
{
"**role**": "string" --*
}

Responses:
Status Codes: 202, 400, 401, 403, 404, 429, 500, 503
Body:
{
Standard Response Body
}

6 CSS Webhook Payload Specification

All messages sent from CSS will follow the format detailed below.
Market Participants should respond with 202 (Accepted) if they regard the message payload as valid, this validation may take place out of process, in which case a 202 (Accepted) would still be expected and any errors found later after processing, can be posted to the error API see 5.3.
For any other errors the appropriate http status code should be returned along with an "errors" property see section 4.5.

6.1 Payload Structure

Method: POST
Body:

```
{
        "version": "string",
        "eventId": "string",
        "eventType": "string",
        "eventStatus": "string",
        "eventDate": "string",
        "contextType": "string",
        "correlationId": "string",
        "eventDescription": "string",
        “confirmationRequired”: “bool”*
        "updatedProperties": ["string"],
        "data": {
                --see data payload sections below
        },
        "errors": [{object See section 4.5}]
}
```

Responses:
Status Codes: 202
Body: n/a

Status Codes: 400, 401, 403, 404, 429, 500, 503
Body:

```
{
        "errors": [{object See section 4.5}]
}
```

*”confirmationRequired” is an optional property coming into effect in June 2024 under R0044 MHHS

6.1.1 Static Properties

Each event payload sent from CSS to Market Participant's webhooks will have static properties in the body relating to the event. Refer to section 7.2 for the mapping of the logical messages to the webhook eventType.

```
{
        "version": "1.0",
        "eventId": "ba5861ff-eddc-4cb7-a9bd-5a22c3b8912d",
        "eventType": " RegistrationPendingSynchronisation",
        "eventStatus": "Ok",
```

"**eventDate**": "2019-08-07T22:57:49.000Z",
"**contextType**": " ECOES",
"**correlationId**": " ba5861ff-eddc-4cb7-a9bd-5a22c3b8912d",
"**eventDescription**": "Registration status changed to Pending",
"updatedProperties": ["registrationStatus", " registrationStatusFromDate"],
"data": {
"registrationStatus": "Pending",
"registrationStatusFromDate": "2019-08-07T15:00:00.000Z",
… omitted for brevity
},
“errors”: see 4.5
}

6.1.2 Error Object

If the CSS has generated errors during processing, then the payload will contain an errors property. In this case the eventStatus would be "Error". See 4.5

Only the ‘Registration Validation Message’ message type can carry an ‘errors’ property.

6.2 Data Payload by Context Type

In addition to the static properties there is a data element (unless it is an error). This dynamic "data" object relates to the "Action" which triggered this event. It will be the RMP data, or the data for a registration.

If a property in the data object is not present, then it is not set.

The following sections detail the data object that is expected for each type of message that is being sent out by CSS. Where the field is mandatory in the body, the item is bold and commented with *.

The following webhook payloads are defined below by Context Type 4.3.11.

6.2.1 DCCSM

Abacus Message Id	eventType
RECM_SN_CSS03060	RegistrationSecuredActiveSynchronisation RegistrationSecuredInactiveSynchronisation RegistrationCancelledSynchronisation
RECM_SN_CSS03000 RECM_SN_CSS03050 RECM_SN_CSS03200	RegistrationEventSynchronisation
RECM_SN_CSS02900 RECM_SN_CSS02960 RECM_SN_CSS03100 RECM_SN_CSS03300	RMPEventSynchronisation
RECM_SN_CSS02950	RMPCreatedSynchronisation RMPOperationalSynchronisation RMPTerminatedSynchronisation RMPDormantSynchronisation

ContextType: DCCSM

The data elements and their associated event types are listed below.

RegistrationEventSynchronisation
Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string", --*
        "supplierMpid": "string", --*
        "supplierRole": "string", --*
        "supplyStartDate": "string", --*
        "domesticPremisesInd": boolean, --*
        "supplierArrangedAppointments": [{
                "mpId": "string", --*
```

```
                "role": "string", --*
                "fromDate": "string",
                "toDate": "string"
        }]
}
```

NOTES:

CSS will not include updated supplierArrangedAppointments details for registrations with a status of Inactive or Cancelled. This will apply as of June 2024 under R0044 MHHS

The supplier arranged appointment update notification message key Name "mpId" for the Market Participant Identifier is conveyed as "mpId" with a capital 'I' instead of a lower case 'i'.

RegistrationSecuredActiveSynchronisation

Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string", --*
        "registrationActiveDate": "string", --*
        "supplierMpid": "string", --*
        "supplierRole": "string", --*
        "supplyStartDate": "string", --*
        "domesticPremisesInd": boolean --*
}
```

RegistrationSecuredInactiveSynchronisation

Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string", --*
        "registrationInactiveDate": "string" --*
}
```

RegistrationCancelledSynchronisation

Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string" --*
}
```

RMPEventSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**networkProvisionMpid**": "string", --*
"**networkProvisionRole**": "string", --*
"**networkProvisionFromDate**": "string", --*
"**rmpStatus**": "string", --*
"**rmpStatusFromDate**": "string", --*
"**mplAddress**": [MPL Address Payload – section 4.1.1.1], --*
"meteringPointEnergyFlow": "string", --**
"meteringPointEnergyFlowFromDate": "string" --**
}
** Electricity only

RMPCreatedSynchronisation
RMPOperationalSynchronisation
RMPTerminatedSynchronisation
RMPDormantSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**rmpStatus**": "string", --*
"**rmpStatusFromDate**": "string" --*
}

6.2.2 MAP

Abacus Message Id	eventType
RECM_SN_CSS02340	RegistrationSecuredActiveNotification
RECM_SN_CSS02370	RegistrationSecuredInactiveNotification

ContextType: MAP

The data elements and their associated event types are listed below.

RegistrationSecuredActiveNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*
"**supplierMpid**": "string", --*
"**supplierRole**": "string", --*
"**registrationActiveDate**": "string" --*
}

RegistrationSecuredInactiveNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*
"**registrationInactiveDate**": "string" --*
}

6.2.3 MEM

Abacus Message Id	eventType
RECM_SN_CSS02370	RegistrationSecuredInactiveNotification
RECM_SN_CSS02390	RegistrationChangeAnticipatedNotification

ContextType: MEM

The data elements and their associated event types are listed below.

RegistrationSecuredInactiveNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*
"**registrationInactiveDate**": "string" --*

}

RegistrationChangeAnticipatedNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**anticipatedInactiveFromDate**": "string", --*
"**switchEvent**": "string" --*
}

6.2.4 DES

Following the approval of CR-D036 on 23 October 2020, all webhooks to provide data to DES have been removed. No outbound messages will be sent and webhook registration is not required for the following event types.

DES will be updated via UK Link with CSS Switching data however this section is being retained in case a direct connection to CSS becomes preferred in the future.

Abacus Message Id	eventType
RECM_SN_CSS02500	RegistrationPendingSynchronisation
RECM_SN_CSS02560	RegistrationSecuredActiveSynchronisation RegistrationSecuredInactiveSynchronisation RegistrationCancelledSynchronisation
RECM_SN_CSS02550	RegistrationEventSynchronisation
RECM_SN_CSS02600	RetailEnergyLocationSynchronisation

ContextType: DES
The data elements and their associated event types are listed below.

RegistrationPendingSynchronisation
RegistrationEventSynchronisation
Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string", --*
        "supplierMpid": "string", --*
        "supplierRole": "string", --*
        "shipperMpid": "string", --*
        "shipperRole": "string", --*
        "shipperFromDate": "string", --*
        "supplyStartDate": "string", --*
        "domesticPremisesInd": boolean --*
}
```

RegistrationSecuredActiveSynchronisation
Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string", --*
        "registrationActiveDate": "string", --*
        "supplierMpid": "string", --*
```

"**supplierRole**": "string", --*
"**supplyStartDate**": "string", --*
"**domesticPremisesInd":** boolean, --*
"**shipperMpid**": "string", --**
"**shipperRole**": "string", --**
"**shipperFromDat**e": "string" --**
}
** only populated for gas

RegistrationSecuredInactiveSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*
"**registrationInactiveDate**": "string" --*
}

RegistrationCancelledSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string" --*
}

RetailEnergyLocationSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**addressSource**": "string", --*
"**uprn**": number, --*
"**relAddress**": [{ REL Address Payload - section 6.3 }] --*
}

6.2.5 ECOES

Abacus Message Id	eventType
RECM_SN_CSS02400	RegistrationPendingSynchronisation
RECM_SN_CSS02460	RegistrationSecuredActiveSynchronisation RegistrationSecuredInactiveSynchronisation RegistrationCancelledSynchronisation
RECM_SN_CSS02450	RegistrationEventSynchronisation
RECM_SN_CSS02600	RetailEnergyLocationSynchronisation

ContextType: ECOES

The data elements and their associated event types are listed below.

RegistrationPendingSynchronisation
RegistrationEventSynchronisation
Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string", --*
        "supplierMpid": "string", --*
        "supplierRole": "string", --*
        "supplyStartDate": "string", --*
        "domesticPremisesInd": boolean --*
}
```

RegistrationSecuredActiveSynchronisation
Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string", --*
        "registrationActiveDate": "string", --*
        "supplierMpid": "string", --*
        "supplierRole": "string", --*
        "supplyStartDate": "string", --*
        "domesticPremisesInd": boolean, --*
        "shipperMpid": "string", --**
        "shipperRole": "string", --**
        "shipperFromDate": "string" --**
}
```

** only populated for gas

RegistrationSecuredInactiveSynchronisation
Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string", --*
        "registrationInactiveDate": "string" --*
}
```

RegistrationCancelledSynchronisation

Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string" --*
}
```

RetailEnergyLocationSynchronisation

Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "addressSource": "string", --*
        "uprn": number, --*
        "relAddress": [{ REL Address Payload - section 6.3 }] --*
}
```

6.2.6 GRDA

Abacus Message Id	eventType
RECM_SN_CSS02800	RegistrationPendingSynchronisation
RECM_SN_CSS02860	RegistrationSecuredActiveSynchronisation RegistrationSecuredInactiveSynchronisation RegistrationCancelledSynchronisation
RECM_SN_CSS02380	RegistrationValidationNotification
RECM_SN_CSS02850	RegistrationEventSynchronisation
RECM_SN_CSS02600	RetailEnergyLocationSynchronisation

ContextType: GRDA

The data elements and their associated event types are listed below.

RegistrationPendingSynchronisation
RegistrationEventSynchronisation

Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string", --*
        "supplierMpid": "string", --*
        "supplierRole": "string", --*
        "shipperMpid": "string", --*
        "shipperRole": "string", --*
        "shipperFromDate": "string", --*
        "supplyStartDate": "string", --*
        "domesticPremisesInd": boolean --*
}
```

RegistrationSecuredActiveSynchronisation
Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string", --*
        "registrationActiveDate": "string", --*
        "supplierMpid": "string", --*
        "supplierRole": "string", --*
        "supplyStartDate": "string", --*
        "domesticPremisesInd": boolean, --*
        "shipperMpid": "string", --**
        "shipperRole": "string", --**
        "shipperFromDate": "string" --**
}
```

** only populated for gas

RegistrationSecuredInactiveSynchronisation
Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string", --*
        "registrationInactiveDate": "string" --*
}
```

RegistrationCancelledSynchronisation
Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string" --*
}
```

RegistrationValidationNotification
Data:

```
[ {
        "registrationRequestId": "string", --*
        "registrationRequestStatus": "string", --*
        "mpxn": "string", --*
        "errors": [{Object see section 4.5}] **
} ]
```

** only populated if registrationRequestStatus is "Rejected"

RetailEnergyLocationSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**addressSource**": "string", --*
"**uprn**": number, --*
"**relAddress**": [{ REL Address Payload - section 6.3 }] --*
}

6.2.7 ERDA

Abacus Message Id	eventType
RECM_SN_CSS02800	RegistrationPendingSynchronisation
RECM_SN_CSS02860	RegistrationSecuredActiveSynchronisation RegistrationSecuredInactiveSynchronisation RegistrationCancelledSynchronisation
RECM_SN_CSS02850	RegistrationEventSynchronisation
RECM_SN_CSS02600	RetailEnergyLocationSynchronisation
RECM_SN_CSS07000	SupplierArrangedAppointmentUpdateNotification*

**effective from June 2024 under R0044 MHHS*

ContextType: ERDA

The data elements and their associated event types are listed below.

RegistrationPendingSynchronisation
RegistrationEventSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*
"**supplierMpid**": "string", --*
"**supplierRole**": "string", --*
"**supplyStartDate**": "string", --*
"**domesticPremisesInd**": boolean --*
}

RegistrationSecuredActiveSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*

"**registrationActiveDate**": "string", --*
"**supplierMpid**": "string", --*
"**supplierRole**": "string", --*
"**supplyStartDate**": "string", --*
"**domesticPremisesInd":** boolean --*
}

RegistrationSecuredInactiveSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*
"**registrationInactiveDate**": "string" --*
}

RegistrationCancelledSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string" --*
}

RetailEnergyLocationSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**addressSource**": "string", --*
"**uprn**": number, --*
"**relAddress**": [{ REL Address Payload - section 6.3 }] --*
}

SupplierArrangedAppointmentUpdateNotification
Data:
{
"**confirmingMpid**": "string", --*
"**registrationId**": "string", --*
"**supplierArrangedAppointments**": [{
"**mpId**": "string", --*
"**role**": "string", --*
"**fromDate**": "string", --*
"toDate": "string"
}], --*
"**updateSucceeded**": boolean --*
}

NOTES:
updateSucceeded
Where 'FALSE', the Error Object of the webhook payload static properties will be populated with the associated errors.
Is not a required field until June 2024 under R0044 MHHS.

The supplier arranged appointment update notification message key Name "mpId" for the Market Participant Identifier is conveyed as "mpId" with a capital 'I' instead of a lower case 'i'.

6.2.8 HHDC, NHHDC

Abacus Message Id	eventType
RECM_SN_CSS02390	RegistrationChangeAnticipatedNotification
RECM_SN_CSS02370	RegistrationSecuredInactiveNotification

ContextType: HHDC, NHHDC

The data elements and their associated event types are listed below.

RegistrationChangeAnticipatedNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**anticipatedInactiveFromDate**": "string", --*
"**switchEvent**": "string" --*
}

RegistrationSecuredInactiveNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*
"**registrationInactiveDate**": "string" --*
}

6.2.9 HHDA, NHHDA

Abacus Message Id	eventType
RECM_SN_CSS02390	RegistrationChangeAnticipatedNotification
RECM_SN_CSS02370	RegistrationSecuredInactiveNotification

ContextType: HHDA, NHHDA

The data elements and their associated event types are listed below.

RegistrationChangeAnticipatedNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**anticipatedInactiveFromDate**": "string", --*
"**switchEvent**": "string" --*
}

RegistrationSecuredInactiveNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*
"**registrationInactiveDate**": "string" --*
}

6.2.10 Gaining Supplier

Abacus Message Id	eventType
RECM_SN_CSS02380	RegistrationValidationNotification
RECM_SN_CSS02300	RegistrationPendingNotification
RECM_SN_CSS02370	RegistrationConfirmedNotification RegistrationSecuredActiveNotification
RECM_SN_CSS02375	GainingRegistrationCancelledNotification

ContextType: GainingSupplier

The data elements and their associated event types are listed below.

RegistrationValidationNotification
Data:
[{
"**registrationRequestId**": "string", --*
"**registrationRequestStatus**": "string", --*
"**mpxn**": "string", --*
"supplierGeneratedReference": "string",
"supplierGeneratedOfafGroupReference": "string",
"errors": [{Object see section 4.5}] **
}]
** only populated if registrationRequestStatus is "Rejected"

RegistrationPendingNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*

"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*
"**supplierMpid**": "string", --*
"**supplierRole**": "string", --*
"**supplyStartDate**": "string", --*
"**domesticPremisesInd**": boolean, --*
"**registrationInitiator**": "string", --*
"shipperMpid": "string", -- **
"shipperRole": "string", -- **
"shipperFromDate": "string", --**
"changeOfOccupancyInd": boolean,
"erroneousSwitchResolutionInd": boolean,
"supplierGeneratedReference": "string",
"supplierGeneratedOfafGroupReference": "string"
}
** only populated for gas

RegistrationConfirmedNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*
"supplierGeneratedReference": "string",
"supplierGeneratedOfafGroupReference": "string"
}

RegistrationSecuredActiveNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*
"**registrationActiveDate**": "string", --*
"supplierGeneratedReference": "string",
"supplierGeneratedOfafGroupReference": "string"
}

GainingRegistrationCancelledNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*

"**registrationCancellationReason**": "string", --*
"supplierGeneratedReference": "string",
"supplierGeneratedOfafGroupReference": "string"
}

6.2.11 Losing Supplier

Abacus Message Id	eventType
RECM_SN_CSS02380	RegistrationValidationNotification
RECM_SN_CSS02370	RegistrationSecuredInactiveNotification RegistrationCancelledNotification
RECM_SN_CSS02700	InvitationToIntervene

ContextType: LosingSupplier

The data elements and their associated event types are listed below.

RegistrationValidationNotification
Data:
[{
"**registrationRequestId**": "string", --*
"**registrationRequestStatus**": "string", --*
"**mpxn**": "string", --*
"errors": [{Object see section 4.5}] **
}]
** only populated if registrationRequestStatus is "Rejected"

RegistrationSecuredInactiveNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*
"**registrationInactiveDate**": "string" --*
}

RegistrationCancelledNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string" --*
}

InvitationToIntervene

Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "activeRegistrationId": "string", --*
        "pendingRegistrationId": "string", --*
        "gainingSupplierMpid": "string", --*
        "gainingSupplierRole": "string", --*
        "supplyStartDate": "string", --*
        "interventionWindowStartDate": "string", --*
        "objectionWindowEndDate": "string", --*
        "annulmentWindowEndDate": "string", --*
        "changeOfOccupancyInd": boolean, --*
        "erroneousSwitchResolutionInd": Boolean --*
}
```

6.2.12 Gaining Shipper

Abacus Message Id	eventType
RECM_SN_CSS02300	RegistrationPendingNotification
RECM_SN_CSS02370	RegistrationSecuredActiveNotification RegistrationConfirmedNotification RegistrationCancelledNotification
RECM_SN_CSS02350	RegistrationEventNotification

ContextType: GainingShipper

The data elements and their associated event types are listed below.

RegistrationPendingNotification

Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string", --*
        "supplierMpid": "string", --*
        "supplierRole": "string", --*
        "shipperMpid": "string", --*
        "shipperRole": "string", --*
        "shipperFromDate": "string", --*
        "supplyStartDate": "string", --*
        "domesticPremisesInd": boolean, --*
        "registrationInitiator": "string", --*
        "changeOfOccupancyInd": boolean,
        "erroneousSwitchResolutionInd": boolean
}
```

RegistrationConfirmedNotification

RegistrationCancelledNotification
Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate ": "string" --*
}
```

RegistrationSecuredActiveNotification
Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate ": "string", --*
        "registrationActiveDate": "string", --*
        "supplierMpid": "string", --*
        "supplierRole": "string", --*
        "shipperMpid": "string", --**
        "shipperRole": "string", --**
        "shipperFromDate": "string", --**
        "supplyStartDate": "string", --*
        "domesticPremisesInd": boolean, --*
        "registrationInitiator": "string", --*
        "changeOfOccupancyInd": boolean,
        "erroneousSwitchResolutionInd": boolean
}
```

** only populated for gas

RegistrationEventNotification
Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "domesticPremisesInd": Boolean, --*
        "losingShipperMpid": "string", --*
        "losingShipperRole": "string", --*
        "gainingShipperMpid": "string", --*
        "gainingShipperRole": "string", --*
        "gainingShipperFromDate": "string", --*
        "supplierMpid": "string", --*
        "supplierRole": "string" --*

}
```

6.2.13 Losing Shipper

Abacus Message Id	eventType

RECM_SN_CSS02370	RegistrationSecuredInactiveNotification
RECM_SN_CSS02390	RegistrationChangeAnticipatedNotification
RECM_SN_CSS02350	RegistrationEventNotification

ContextType: LosingShipper

The data elements and their associated event types are listed below.

RegistrationEventNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**domesticPremisesInd**": Boolean, --*
"**losingShipperMpid**": "string", --*
"**losingShipperRole**": "string", --*
"**gainingShipperMpid**": "string", --*
"**gainingShipperRole**": "string", --*
"**gainingShipperFromDate**": "string", --*
"**supplierMpid**": "string", --*
"**supplierRole**": "string" --*
}

RegistrationSecuredInactiveNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*
"**registrationInactiveDate**": "string" --*
}

RegistrationChangeAnticipatedNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**anticipatedInactiveFromDate**": "string", --*
"**switchEvent**": "string" --*
}

6.2.14 Registered Shipper

Abacus Message Id	eventType
RECM_SN_CSS02350	RegistrationEventNotification

ContextType: RegisteredShipper

The data elements and their associated event types are listed below.

RegistrationEventNotification
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**domesticPremisesInd**": boolean --*
}

6.2.15 Webhook Subscription

ContextType: WebhookSubscription

The data elements and their associated event types are listed below.

WebhookVerificationEvent
Data: n/a

6.2.16 ECOS

Abacus Message Id	eventType
RECM_SN_CSS03060	RegistrationSecuredActiveSynchronisation RegistrationSecuredInactiveSynchronisation RegistrationCancelledSynchronisation
RECM_SN_CSS03000 RECM_SN_CSS03050 RECM_SN_CSS03200	RegistrationEventSynchronisation
RECM_SN_CSS02900 RECM_SN_CSS02960 RECM_SN_CSS03100 RECM_SN_CSS03300	RMPEventSynchronisation
RECM_SN_CSS02950	RMPCreatedSynchronisation RMPOperationalSynchronisation RMPTerminatedSynchronisation RMPDormantSynchronisation

ContextType: ECOS

The data elements and their associated event types are listed below.

RegistrationEventSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*

"**supplierMpid**": "string", --*
"**supplierRole**": "string", --*
"**supplyStartDate**": "string", --*
"**domesticPremisesInd**": boolean, --*
"supplierArrangedAppointments": [{
"**mpId**": "string", --*
"**role**": "string", --*
"**fromDate**": "string",
"toDate": "string"
}]
}

NOTES:

CSS will not include updated supplierArrangedAppointments details for registrations with a status of Inactive or Cancelled.

The supplier arranged appointment update notification message key Name "mpId" for the Market Participant Identifier is conveyed as "mpId" with a capital 'I' instead of a lower case 'i'.

RegistrationSecuredActiveSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*
"**registrationActiveDate**": "string", --*
"supplierMpid": "string", --*
"supplierRole": "string", --*
"supplyStartDate": "string", --*
"domesticPremisesInd": boolean --*
}

RegistrationSecuredInactiveSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string", --*
"**registrationInactiveDate**": "string" --*
}

RegistrationCancelledSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*

"**registrationId**": "string", --*
"**registrationStatus**": "string", --*
"**registrationStatusFromDate**": "string" --*
}

RMPEventSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**networkProvisionMpid**": "string", --*
"**networkProvisionRole**": "string", --*
"**networkProvisionFromDate**": "string", --*
"**rmpStatus**": "string", --*
"**rmpStatusFromDate**": "string", --*
"**mplAddress**": [MPL Address Payload – section 4.1.1.1], --*
"meteringPointEnergyFlow": "string", --**
"meteringPointEnergyFlowFromDate": "string" --**
}
** Electricity only

RMPCreatedSynchronisation
RMPOperationalSynchronisation
RMPTerminatedSynchronisation
RMPDormantSynchronisation
Data:
{
"**mpxn**": "string", --*
"**fuelType**": "string", --*
"**rmpStatus**": "string", --*
"**rmpStatusFromDate**": "string" --*
}

6.2.17 Registered Supplier

Abacus Message Id	eventType
RECM_SN_CSS02380	RegistrationValidationNotification
RECM_SN_CSS02370	RegistrationSecuredInactiveNotification

ContextType: RegisteredSupplier

The data elements and their associated event types are listed below.

RegistrationValidationNotification
Data:
[{
"**registrationRequestId**": "string", --*
"**registrationRequestStatus**": "string", --*
"**mpxn**": "string", --*
"errors": [{Object see section 4.5}] **
}]

****** only populated if registrationRequestStatus is "Rejected"

RegistrationSecuredInactiveNotification

Data:

```
{
        "mpxn": "string", --*
        "fuelType": "string", --*
        "registrationId": "string", --*
        "registrationStatus": "string", --*
        "registrationStatusFromDate": "string", --*
        "registrationInactiveDate": "string" --*
}
```

6.3 REL Address Payload

Where address (REL) information is sent out, the following is what will be in the address payload:


```
{
        "secondaryName": "string",
        "primaryName": "string",
        "street1": "string",
        "street2": "string",
        "locality1": "string",
        "locality2": "string",
        "town": "string",
        "postcode": "string",
        "logicalStatus": number,
        "language": "string",
        "organisation": "string",
        "addressType": "string",
        "confidenceScore": number,
        "classification":"string",
        "latitude": number,
        "longitude": number
}
```


The REL address will be returned as an array. Where available one of these will be a delivery point address, this is denoted by the addressType field. See 4.1.2 for a description of the data elements. Where the MPL address is referenced, the above lists the components of the payload and are sent out from CSS. The data is sent by CSS as an object in JSON, which is an unordered set of name/value pairs so may not come through in the order listed.

7 Appendix

Messages ids with numeric component of 6000 or greater (RECM_SN_CSS06000 and above) are allocated post Abacus usage and are not therefore included in the tables below.

7.1 Abacus Inbound API Map to Sections

Logical Message	Logical Description	Direction	Market Participant Role	API section
RECM_SN_CSS00100	Communications Hub Data Link	Inbound	DCC SM	5.5.1
RECM_SN_CSS00100	Communications Hub Data Link	Inbound	DCC SM	5.5.2
RECM_SN_CSS00200	Supplier Arranged Appointment synchronisation	Inbound	GRDA ERDA	5.6.3
RECM_SN_CSS00250	Market Participant Role Alliance	Inbound	ERDA GRDA	5.7.3
RECM_SN_CSS00300	Metering Point Synchronisation	Inbound	ERDA	5.5.2
RECM_SN_CSS00400	Metering Point RMP Event Synchronisation	Inbound	ERDA	5.5.2
RECM_SN_CSS00400	Metering Point RMP Event Synchronisation	Inbound	ERDA	5.5.4
RECM_SN_CSS00500	Metering Point RMP Association Synchronisation	Inbound	ERDA	5.5.3
RECM_SN_CSS00600	Metering Point RMP Location Synchronisation	Inbound	ERDA	5.5.2
RECM_SN_CSS00700	Metering Point RMP Lifecycle Synchronisation	Inbound	ERDA	5.5.2
RECM_SN_CSS01000	Metering Point RMP Network Operator Synchronisation	Inbound	ERDA	5.5.2
RECM_SN_CSS01100	Supply Meter Point Synchronisation	Inbound	GRDA	5.5.1
RECM_SN_CSS01200	Supply Meter Point RMP Event Synchronisation	Inbound	GRDA	5.5.1
RECM_SN_CSS01200	Supply Meter Point RMP Event Synchronisation	Inbound	GRDA	5.5.4
RECM_SN_CSS01300	Supply Meter Point RMP Location Synchronisation	Inbound	GRDA	5.5.1
RECM_SN_CSS01400	Supply Meter Point RMP Lifecycle Synchronisation	Inbound	GRDA	5.5.1
RECM_SN_CSS01600	Supply Meter Point RMP Network Provision Synchronisation	Inbound	GRDA	5.5.1
RECM_SN_CSS01700	Initial Registration Request	Inbound	Gas Supplier	5.6.1
RECM_SN_CSS01700	Initial Registration Request	Inbound	Electricity Supplier	5.6.1
RECM_SN_CSS01750	GT Initiated Registration Request	Inbound	GT Registration Initiator	5.6.1
RECM_SN_CSS01800	Switch Request	Inbound	Gaining Supplier	5.6.4
RECM_SN_CSS01900	Registration Deactivation Request	Inbound	Supplier	5.6.6

RECM_SN_CSS02000	Registration Action Request	Inbound	Gas Supplier	5.6.2
RECM_SN_CSS02000	Registration Action Request	Inbound	Electricity Supplier	5.6.2
RECM_SN_CSS02100	Withdrawal Request	Inbound	Supplier	5.6.5
RECM_SN_CSS02200	Switch Intervention	Inbound	Supplier	5.6.5
~~RECM_SN_CSS04000~~	~~Retail Market Domain Specification~~	~~Inbound~~	~~RECDDG~~	~~5.7.1~~
~~RECM_SN_CSS04050~~	~~Retail Market Processing Calendar~~	~~Inbound~~	~~RECDDG~~	~~5.7.3~~
~~RECM_SN_CSS04100~~	~~Retail Energy Company Group Membership~~	~~Inbound~~	~~RECDDG~~	~~5.7.4~~
RECM_SN_CSS04200	Market Participant Role Event	Inbound	RECDDG	~~5.8.2~~ 5.7.2
~~RECM_SN_CSS04300~~	~~Market Alliance Constituent Market Role~~	~~Inbound~~	~~RECDDG~~	~~5.7.1~~
~~RECM_SN_CSS04400~~	~~RMP Event Type~~	~~Inbound~~	~~RECDDG~~	~~5.7.4~~
~~RECM_SN_CSS04410~~	~~RMP Association Type~~	~~Inbound~~	~~RECDDG~~	~~5.7.4~~
~~RECM_SN_CSS04420~~	~~RMP Lifecycle Status Type~~	~~Inbound~~	~~RECDDG~~	~~5.7.4~~
~~RECM_SN_CSS04430~~	~~Registration Lifecycle Status Type~~	~~Inbound~~	~~RECDDG~~	~~5.7.4~~
~~RECM_SN_CSS04440~~	~~Market Participant Role Event Type~~	~~Inbound~~	~~RECDDG~~	~~5.7.4~~
RECM_SN_CSS05000	Retail Energy Company	Inbound	~~EDDG~~ ~~GDDG~~ RECDDG	~~5.8.4~~ 5.7.4
RECM_SN_CSS05100	Market Role	Inbound	~~EDDG~~ ~~GDDG~~ RECDDG	~~5.8.4~~ 5.7.1
RECM_SN_CSS05200	Market Participant Role Ownership	Inbound	~~EDDG~~ ~~GDDG~~ RECDDG	~~5.8.4~~ 5.7.2

7.2 Abacus Outbound Message Map to Webhook Event Type

eventType	ABACUS Msg Id	contextType	Section
RegistrationValidationNotification	RECM_SN_CSS02380	GainingSupplier	6.2.10
		LosingSupplier	6.2.11
		Registered Supplier	6.2.17
		GRDA	6.2.6
RegistrationPendingNotification	RECM_SN_CSS02300	GainingSupplier	6.2.10
		GainingShipper	6.2.12
RegistrationPendingSynchronisation	RECM_SN_CSS02400	ECOES	6.2.5
	RECM_SN_CSS02500	DES	6.2.4
	RECM_SN_CSS02800	GRDA	6.2.6
		ERDA	6.2.7
RegistrationSecuredActiveNotification	RECM_SN_CSS02340	MAP	6.2.2
	RECM_SN_CSS02370	GainingSupplier	6.2.10
		GainingShipper	6.2.12
RegistrationSecuredActiveSynchronisation	RECM_SN_CSS02460	ECOES	6.2.5
	RECM_SN_CSS02560	DES	6.2.4
	RECM_SN_CSS02860	GRDA	6.2.6
		ERDA	6.2.7
	RECM_SN_CSS03060	DCCSM	6.2.1
		ECOS	6.2.16
RegistrationConfirmedNotification	RECM_SN_CSS02370	GainingSupplier	6.2.10
		GainingShipper	6.2.12
RegistrationSecuredInactiveNotification	RECM_SN_CSS02370	LosingSupplier	6.2.11
		Registered Supplier	6.2.17
		MEM	6.2.3
		MAP	6.2.2
		LosingShipper	6.2.13
		HHDC	6.2.8
		HHDA	6.2.9
		NHHDC	6.2.8
		NHHDA	6.2.9
RegistrationSecuredInactiveSynchronisation	RECM_SN_CSS02460	ECOES	6.2.5
	RECM_SN_CSS02560	DES	6.2.4
	RECM_SN_CSS02860	GRDA	6.2.6
		ERDA	6.2.7
	RECM_SN_CSS03060	DCCSM	6.2.1
		ECOS	6.2.16
GainingRegistrationCancelledNotification	RECM_SN_CSS02375	GainingSupplier	6.2.10
RegistrationCancelledNotification	RECM_SN_CSS02370	GainingShipper	6.2.12
	RECM_SN_CSS02370	LosingSupplier	6.2.11

Landmark INFORMATION

RegistrationCancelledSynchronisation	RECM_SN_CSS02460	ECOES	6.2.5
	RECM_SN_CSS02560	DES	6.2.4
	RECM_SN_CSS02860	GRDA	6.2.6
		ERDA	6.2.7
	RECM_SN_CSS03060	DCCSM	6.2.1
		ECOS	6.2.16
RegistrationChangeAnticipatedNotification	RECM_SN_CSS02390	MEM	6.2.3
		HHDC	6.2.8
		HHDA	6.2.9
		NHHDC	6.2.8
		NHHDA	6.2.9
		LosingShipper	6.2.11
RegistrationEventNotification	RECM_SN_CSS02350	GainingShipper	6.2.12
		LosingShipper	6.2.13
		RegisteredShipper	6.2.14
RegistrationEventSynchronisation	RECM_SN_CSS02450	ECOES	6.2.5
	RECM_SN_CSS02550	DES	6.2.4
	RECM_SN_CSS02850	GRDA	6.2.6
	RECM_SN_CSS03000	DCCSM ECOS	6.2.1 6.2.16
	RECM_SN_CSS03050		
	RECM_SN_CSS03200		
InvitationToIntervene	RECM_SN_CSS02700	LosingSupplier	6.2.11
RMPEventSynchronisation	RECM_SN_CSS02900	DCCSM ECOS	6.2.1 6.2.16
	RECM_SN_CSS02960		
	RECM_SN_CSS03100		
	RECM_SN_CSS03300		
RMPCreatedSynchronisation	RECM_SN_CSS02950		
RMPOperationalSynchronisation			
RMPTerminatedSynchronisation			
RMPDormantSynchronisation			
RetailEnergyLocationSynchronisation	RECM_SN_CSS02600	DES	6.2.4
		ECOES	6.2.5
		GRDA	6.2.6
		ERDA	6.2.7