DSD002 Annex Two – Detailed DIP Operational Requirements

Contents

1 Scope and purpose

1.1.1 The purpose of this Annex is to identify any configurable parameters that will be reviewed periodically to cater for changing demand and capacity forecasts, where they have not been identified elsewhere in this DSD. This Annex also defines the operational procedures required by DIP Users and DIP Connection Providers to securely exchange metering and consumption information with the DIP using Public Key Infrastructure (PKI) Digital Certificates.

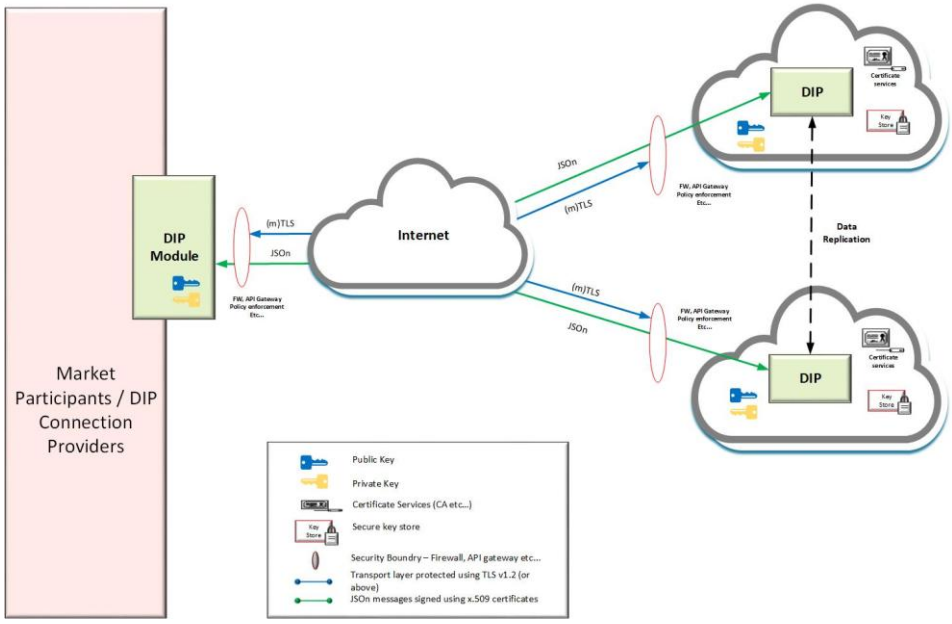
2 DIP Interfaces

2.1 DIP landscape

2.1.1 The DIP is a cloud hosted Event Driven Architecture (EDA) middleware component designed and built to be the central data sharing capability for the business processes created as part of the MHHS Programme.

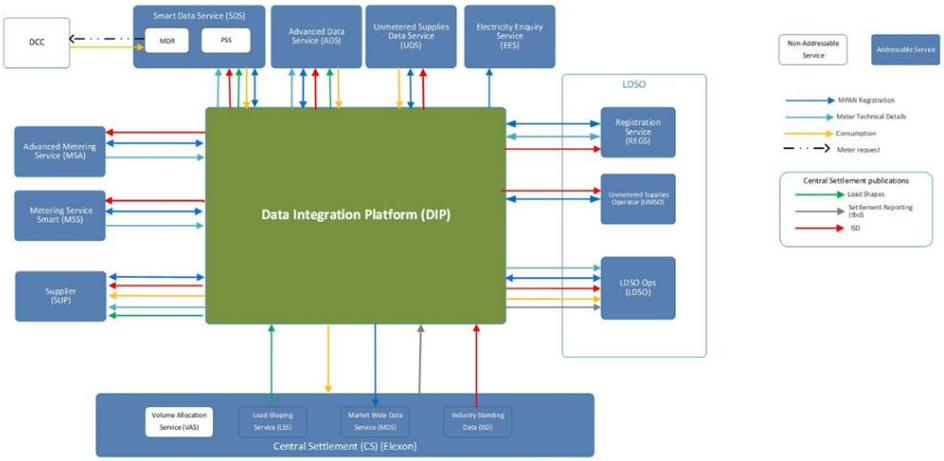
2.1.2 The DIP landscape is a distributed network of services and roles that requires constant communication of data for operational purposes. The EDA is an architectural pattern used for the production, management and consumption of data messages/events that enables the creation of a responsive/reactive, asynchronous, non-blocking/concurrent and de-coupled systems topology.

2.1.3 At the core of this architecture is the DIP operating model which is responsible for brokering the communication between all industry participants operating under that model.

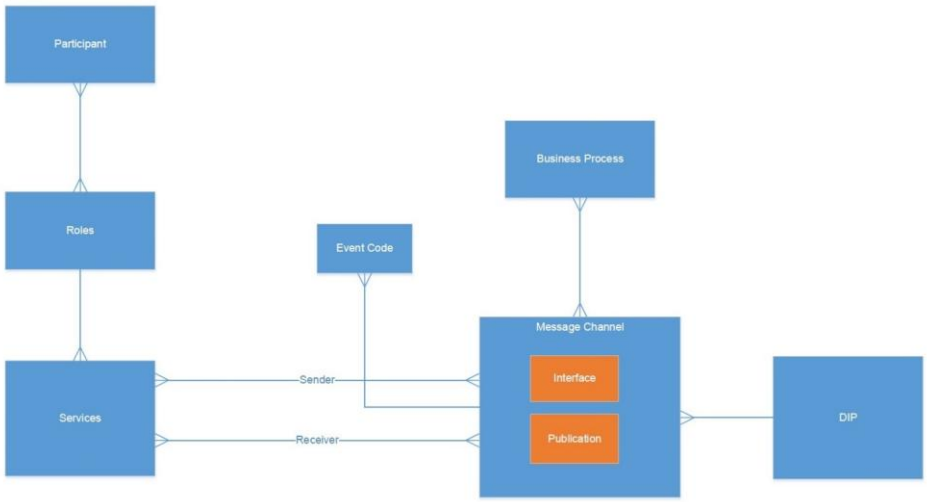


Direct (Internet Access) showing the DIP interfaces in the context of the components with which it interfaces

2.1.4 The following shows how the DIP will interact with systems across the industry and is a high-level view of the flow of data between DIP Users:



2.1.5 The following shows the DIP Logical design:



2.1.6 Inbound Messages are sent to the DIP by calling pre-determined URLs (dependant on environment), whilst the outbound Messages from the DIP are delivered to a Webhook of the recipient's choice. Webhooks are URLs which are event-based, meaning that when a specific event occurs in the DIP, a Message can be delivered to a pre-defined URL.

2.1.7 The DIP is hosted on Microsoft's Azure public cloud independent of the DIP Manager's owning Code Body's other systems i.e. it is a stand-alone system.

2.1.8 To maintain a geographically redundant service, the 'Cloud Provider' execution venue takes advantage of Microsoft's UK South and UK West regions. Within the UK South region the DIP will be deployed across three availability zones and for UK West the DIP will deployed into one availability zone. This configuration provides the capability to 'failover' seamlessly between physically separate availability zones and also between regions.

2.1.9 The DIP is protected against catastrophic failures pertaining to the hosted components that make up the entirety of the service. The granular monitoring of each of the applications and underlying infrastructure is paramount to providing a continuous, truly highly available platform. In the event of a catastrophic failure there would be no impact to the DIP User.

2.1.10 The Microsoft Azure platform delivers a set of efficient, scalable, and dependable internet gateway (IGW) servers which are backed by a scalable bank of firewalls. The firewalls secure the DIP and its corresponding infrastructure. The firewalls are configured to handle the mTLS connections for all incoming and outgoing JSON Messages, thus offering policy-based security for the DIP. The IGWs are configured to manage the mTLS links for all incoming and outgoing Messages.

2.1.11 Behind the IGW, the DIP uses Azure Application Programming Interface (API) management to create and maintain interfaces into the DIP via the internet. Azure API management is used to standardise inbound interfaces within the DIP.

2.1.12 The DIP shall have both a Production environment (for sending actual data) and a non-production environment (for DIP On-Boarding, testing and such) at all times. Additional non-production environments may be created from time-to-time as directed by the DIP Manager. The non-production and production environments are maintained separately at all times.

2.1.13 At the other end of the internet connection is the DIP User's environment. DIP Users are recommended to include both a Policy Enforcement Point (PEP) and one or more API Gateway (depending on their capacity and resilience requirements) or, alternative security controls which meet the minimum security requirements required by the DIP Rules.

2.1.14 The business logic of the DIP is:

a) validate message/event headers;

b) validate the message/event structure; and

c) address and route messages based on content.

2.1.15 The DIP does not validate message body content.

2.1.16 Digital Certificates are used in the communication between DIP Users and the DIP to:

a) to secure the mTLS channel between their PEP and the DIP; and

b) to digitally sign Messages that are sent to the DIP.

2.1.17 Different Digital Certificates are used in the Pre-Production (On-boarding) and Production environments.

2.2 DIP Core Services

2.2.1 The core service areas are:

a) Message Brokering – the core component of the DIP responsible for relaying messages between Market Participants includes the following functions:

i) Message Validation, Security and Repudiation;

ii) Message Archiving;

iii) Message Replay;

b) Message Brokerage Management – this includes the following functions:

i) Manage Message Channels;

ii) Add/Remove Message Channels;

iii) MPAN based addressing;

c) Participant Management:

i) DIP On-boarding and DIP Off-boarding;

ii) Participant administration including DIP Role maintenance;

d) Service Management:

i) Application management;

ii) Incident Management;

iii) Change Management;

iv) Availability Management;

v) Problem Management;

vi) Data Management;

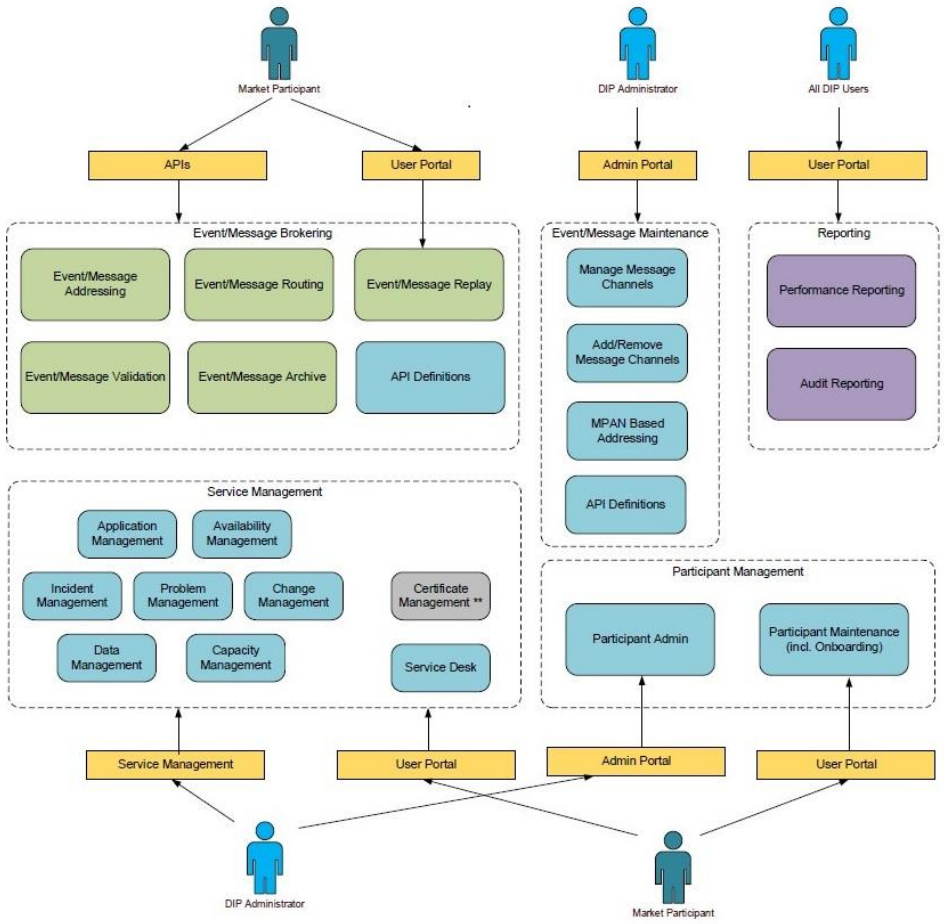
vii) Capacity Management;

viii) Certificate Management – the possibility exists that this function could be carried by the DCC SWIKI services – yet to be agreed;

e) Reporting:

i) Tracking workflows across MHHS business process.

2.2.2 A service orientated view of the DIP is presented below:



The Admin Portal is the DIP Manager's means of administering the DIP, the user portal is for DIP Users

2.3 DIP Connections

2.3.1 Connections between the DIP and DIP Users/DCPs across the internet are subject to cryptographic protection. The cryptographic protection ensures the authenticity, integrity, and confidentiality of the connection. Hence, all communications with the DIP shall be via a secure communications channel. Individual Messages will flow over this secure communications channel.

2.3.2 The communications channel will consist of an encrypted and authenticated session between the DIP and the DIP User's PEP. PEPs will typically be boundary firewalls that can negotiate the security parameters required to set-up a secure TLS session. The TLS session must be configured for mutual authentication (mTLS), such that each entity authenticates the other during the handshake protocol.

2.3.3 The DIP performs Message validation in two phases, a synchronous check (Level 1 validation) at the API that is communicated directly back to the sender, and then further asynchronous checks (Level 2 validation) and addressing that are carried out once the Message has been initially received.

2.3.4 The initial API is intended to be as lightweight as possible thereby increasing Message submission speed. Where no error condition is encountered no event/Message acknowledgement other than the initial API response is sent back to the sender.

2.3.5 The Messages have specific security properties that must be satisfied:

a) Message integrity – to ensure that the Message received is the same as it was when sent and has not been accidentally or maliciously changed in transit;

b) data origin (Message) authentication – assurance that a given entity is the original source of the received data – the Message is authenticated using a cryptographic key. Where required, each Message is individually authenticated to allow the recipient to verify the claimed identifier of the Message;

c) Mutual Authentication – both parties authenticate each other such that both are assured of the identity and authenticity of the other; and

d) Confidentiality – ensures compliance with Data Protection Legislation.

2.4 DIP Users

2.4.1 There are primarily three different types of DIP Users:

a) Active Market Participant;

b) Non-Active Market Participant; and

c) DIP Connection Provider (DCP).

2.4.2 All DIP Users will need to complete DIP On-boarding and register with the DIP Verification Service Provider, who is also responsible for managing DIP Digital Certificates.

2.4.3 As described in DSD002 'Connection and Operations' each DIP User will have a nominated Certificate Admin responsible for all aspects of certificate management for the organisation.

2.4.4 Certificate requests are made via the DIP Portal and owned by the DIP User.

2.4.5 DIP Users should implement mTLS connection pooling both to support their own non-functional requirements (NFRs) and to reduce the centralised and accumulated impact of potentially high traffic volumes on the DIP.

2.5 Active Market Participant

2.5.1 An active Market Participant (MP) is a DIP User directly interfacing with the DIP without the services of a DCP who only requires one dual purpose Digital Certificate (notwithstanding the need for different Digital Certificates between non-Production and Production).

2.5.2 One Digital Certificate authorised for both mTLS and digital signing can be requested. Alternately, separate Digital Certificates for mTLS and digital signing can be requested.

2.6 Non-active Market Participant

2.6.1 A non-active MP is a DIP User utilising the services of a DCP to provide their connections to the DIP.

2.6.2 Where a non-active MP chooses to use multiple DCPs, a unique Digital Certificate will be required for each DCP.

2.6.3 A Non-Active MP – DCP connection to the DIP will adopt the following set-up:

a) the DCP will connect to the DIP using their own Digital Certificate to secure the mTLS connection;

b) the DCP will sign the messages sent on behalf of the Non-Active MP using the DIP User's Digital Certificate (see paragraph 6.3.2).

2.6.4 Following on from the vetting process at the start of the DIP On-boarding process, and registration with the DIP Verification Service Provider, a DIP User can nominate an individual within their contracted DCP to perform the role of a Certificate Admin. The Certificate Admin will be able to login to the DIP and manage Digital Certificates on behalf of the DIP User.

3 DIP Access and participant engagement

3.1 The DIP Portal

3.3.1 The DIP Portal allows DIP Users access to:

a) reporting on audit and monitoring data;

b) manage users and Digital Certificates; and

c) perform DIP functions such as replaying Messages.

3.3.2 The DIP Portal also integrates with the service management system where users may raise and track service incidents

3.3.3 The DIP makes use of standard TLS to ensure the connection between the client (the DIP User's browser) and the server (the DIP) is secured.

3.3.4 DIP Users will authenticate with the DIP using Azure Active Directory (Azure AD). The following describes the process for a user logging in first time:

a) DIP User representatives are invited by the DIP Manager or their DIP User Administrator;

b) the user receives an email welcoming them to the DIP. Within this email is a redemption link;

c) the user clicks upon the link, and they are taken to the DIP Portal where they are asked to sign-in with their company credentials;

d) upon successful sign-in the user is asked to configure Multi-Factor Authentication (MFA). This involves configuring one or more second factor authentication mechanisms (the first factor being the password) such as:

i) Microsoft authenticator app;

ii) mobile phone (text or phone call);

iii) Google Authenticator (Chrome extension);

e) the user will then review and accept the terms of use for the DIP Portal; and

f) once accepted the user will be able to access the DIP Portal.

3.3.5 During subsequent logins, the user will have to complete an MFA challenge if this is the first time, they have logged in within 24 hours. They will not have to accept the terms of use again unless the terms are updated.

3.2 Registration

3.2.1 The process for registration is described in DIP On-Boarding in DSD002 'DIP Connection and Operation'.

3.2.2 Before any DIP User can request Digital Certificates in the DIP, the Certificate Admin must complete the DIP Digital Certificate authority registration and the DIP User must be successfully vetted by the DCA to confirm authenticity.

3.3 PKI Role Governance

3.3.1 The governance of additional Certificate Admin roles is at the discretion of the DIP User. The DIP Manager will not request any employee validation letters for these roles (for clarity, once the organisation has been verified during DIP On-boarding, no further verification will be undertaken by the DIP Manager unless the circumstances of the case require).

Roles privilege table

Privilege	DIP User Administrator	Certificate Administrator
Invite Certificate Admin to undertake organisational vetting and registration	X	
Manage (add/remove) Certificate Admin roles within the DIP User Portal	X	
Organisation Vetting and Registration		First Certificate Admin
Submit a Certificate Signing Request (CSR)		X
Access Digital Certificate from DIP		X
Access IA Trust Chain (public) Certificates from DIP		X
Submit a CRR		X
Submit a Digital Certificate rekey request		X
Get notified from the DIP when Digital Certificate is nearing expiry (90, 60, 30, 1 days)	X	X
Get notified whenever a change of Certificate Admin	X	X
Get notified on a lifecycle status change of CSR		X
Get notified on a lifecycle status change of CRR		X
Get notified on a lifecycle status change of a Digital Certificate rekey		X
Get notified when Digital Certificate available for download		X
Able to download Digital Certificate from DIP		X

3.4 Interfacing with the DIP

3.4.1 The technical interface specification is defined by the Energy Market Data Specification (EMDS). It defines the interface at a logical level to enable DIP Users and DCPs to ensure their systems meet its requirements. It contains instructions on how Messages are to be composed, the structure of the Messages and any specific data validation to be performed.

4 Public Key Infrastructure

4.1 Overview

4.1.1 The DIP and each DIP User are required to hold two cryptographically connected keys:

a) a Public Key that is made widely available and acts as authentication anchor;

b) a Private Key that is used to produce digital signatures on Messages.

4.1.2 Recipients of digitally signed Messages can verify the origin and integrity of a received Message by verifying that the attached signature is valid using the Public Key of the assumed sender. It is the Public Key that is embedded in a Digital Certificate, and the security of Public Key cryptography relies on ensuring that Private Keys are kept secure.

4.2 DIP Certificate Authority

4.2.1 The most common trust model used to validate the validity and authenticity of a Public key is a Certificate Authority and this is the trust model used in the DIP.

4.2.2 The DIP Certificate Authority (DCA - synonymous with an Issuing Authority (IA)) is managed by the DIP Verification Service Provider.

4.2.3 The DCA is integrated into the DIP to issue Digital Certificates to DIP Users. These Digital Certificates are digitally signed by the DCA and bind DIP Users with their Public Keys.

4.3 Chain of trust

4.3.1 The standard trust model used in the DIP holds that if the Digital Certificate is digitally signed by the DCA, then it can be trusted by the Relying Party (see paragraph 6.5).

4.3.2 To do this the Relying Party needs to verify the DCAs Digital Signature, and that certain contents of the Digital Certificate are also correct (e.g. the Digital Certificate has not expired and that the key usage is for Digital Signatures).

4.3.3 To verify the DCAs Digital Signature, the Relying Party must have access to the DCAs Public Key, which itself will be contained in another Digital Certificate signed by a higher-level (intermediate) or top-level (root) CA.

4.3.4 All Digital Certificates that are part of the Digital Certificate Chain will be required to complete the validation process (‘walking the chain’). All Digital Certificates in the Digital Certificate Chain will be in a repository accessible via the DIP.

4.4 Digital Certificate usage

4.4.1 Digital Certificate usage is defined by a Digital Certificate Profile. Digital Certificate Profiles are approved and issued by the DCA. The DCA ensures that Digital Certificates are issued only to DIP Users and are used only for the purposes of:

a) the creation, sending, receiving, and processing of communication within the DIP environments;

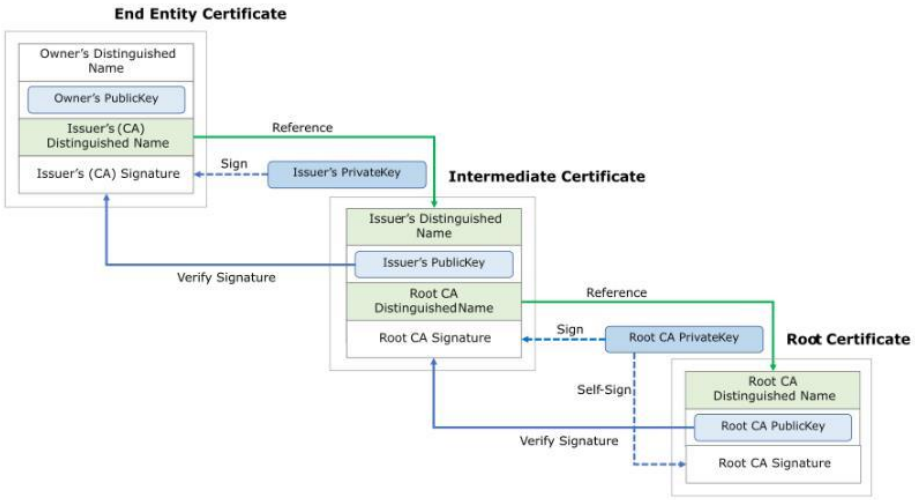
b) Symmetric key generation (Digital Signature, Key Agreement);

c) TLS communication (Digital Signature, Key Agreement, TLS web client authentication, TLS web server authentication);

d) Authentication and Non-Repudiation (Digital Signature, Non-Repudiation, key encipherment, data encipherment, Key Agreement, TLS web client authentication. TLS web server authentication).

4.4.2 Digital Certificates are publicly available, as they do not include either the DIP User or the DCA's Private Keys. As such, they can be used as anchors of trust for authenticating Messages originating from different parties.

4.4.3 CAs also have a Digital Certificate, which they make widely available. This allows the consumers of identities issued by a given CA to verify them by checking that the Digital Certificate could only have been generated by the holder of the corresponding Private Key (the CA).



Digital Certificate Path

4.5 Key management

4.5.1 Since Public Key Cryptography depends on the secrecy of the Private Key, the expectation is that each DIP User should ensure that key material is protected appropriately.

4.5.2 For most organisations, this will involve putting in place a boundary protection device, such as a firewall, that can enforce the controls specified. The device, or PEP, must have logical access to a key store for TLS authentication and be able to create or import keys in a secure fashion such as via a Cryptographic Module.

4.5.3 What provides the PEP capability, whether there is a cryptographic module in operation, and the nature of the local Public Key store is at the discretion of the DIP User and subject to their own risk assessment.

4.5.4 DIP Participants shall notify the DIP Manager as soon as practicable if they become aware of compromise or suspected compromise of any Private Key material associated with a Digital Certificate.

4.6 Digital Certificate Revocation

4.6.1 A Certificate Revocation List (CRL) is a list of references to Digital Certificates that a CA knows to be revoked for one reason or another.

4.6.2 A Digital Certificate must be revoked:

a) when any of the information in the Digital Certificate is known or suspected to be inaccurate;

b) upon suspected or known compromise of the Private Key associated with the Digital Certificate;

c) upon suspected or known compromise of the media holding the Private Key associated with the Digital Certificate.

4.6.3 DIP Users shall submit a Certificate Revocation Request in relation to a Digital Certificate:

a) immediately upon becoming aware that the Digital Certificate has been compromised, or is suspected of having been compromised; or

b) immediately upon ceasing to be the DIP User in respect of that Digital Certificate.

4.6.4 Additional information about CRLs can be found in the DIP-PKI Policy (DSD002 Annex 3 ‘The DIP-PKI (Public Key Infrastructure) Policy’).

5 DIP Security Requirements

5.1 Security requirements

5.1.1 The security requirements in the DIP Rules are based on National Cyber Security Centre Cyber Assessment Framework (NCSC CAF) recommendations that organisations adopt a recognised baseline of security controls such as those prescribed in ISO 27001 (or equivalent, e.g., NIST 800- 53), supplemented with a set of enhanced level controls on which DIP Users are reliant to protect the DIP Ecosystem.

5.1.2 The Messaging security requirements and how and where they are satisfied are:

Objective	Mechanism	Provided by
Confidentiality	Encryption in transit	Transport Layer Security (TLS)
Data Integrity	Message authentication code (MAC)	TLS
	Hash Function	Message signing
	Digital Signature	Message signing
Data Origin Authentication	MAC	TLS
	Digital Signature	Message signing
Non-Repudiation*	Digital signature	Message signing

*Non-Repudiation is not a specific security requirement but can be realised using a Digital Signature.

5.1.3 Messages sent to and received from the DIP will be secured by:

a) Mutual TLS (mTLS) for transport layer security and authentication; this should be TLS 1.3 but shall be TLS 1.2 as a minimum;

b) Message signing for non-repudiation;

c) mTLS for strong security authentication/authorisation at the API; and

d) API key (ingest only) for rate limiting and usage tracking.

Formatted: Danish

5.2 Information Security Management System certification

5.2.1 The DIP Manager is responsible for ensuring the DIP is certified against ISO 27001. Certification shall be provided by a UKAS-certified auditing body.

5.2.2 DIP Users shall carry out regular risk assessments of their systems and interfaces. Details of how to conduct risk assessments can be found in well-established assessment models, such as ISO 27005 or HMG's Information Assurance Standards 1 & 2.

5.3 TLS requirements and configuration

5.3.1 The secure mTLS session will be based on the Transport Layer Security (TLS) v1.2 protocol standard (TLS v1.2) as a minimum, and will make use of authentication using PKCS#3 Ephemeral-Diffie-Hellman key exchange to generate a shared secret (TLS-RSA) with AES-128-GCM-SHA256 for communications encryption.

5.3.2 If this authentication step fails, an 'HTTP 403 Unauthorised' error will be returned.

5.4 TLS key generation and Certificate Signature Requests

5.4.1 The DIP Service Provider issues RSA Digital Certificates for TLS in line with the NCSC guidance with the following parameters: 4096-bit RSA with SHA256.

5.4.2 Public/Private Key pairs must be created following successful completion of a DIP User's registration process. Each DIP User will be responsible for the generation of TLS keys and CSRs used by their Message service interface.

5.4.3 Each DIP User will need to issue an associated PKCS#10 CSR to the DCA to obtain a Digital Certificate. No DIP User may make a CSR which contains:

a) any information that constitutes a trademark unless it is the holder of the Intellectual Property Rights in relation to that trademark; or

b) any confidential information which would be contained in a Digital Certificate issued in response to that CSR.

5.4.4 Where any CSR fails to satisfy the requirements set out in this Annex, the DCA:

a) shall reject it and refuse to issue the Digital Certificate which was the subject of the CSR;

b) shall inform the DIP User which made the CSR why it was rejected.

6 Managing Digital Certificates

6.1 DIP Users responsibilities

6.1.1 DIP Users will be responsible for managing and securing the Digital Certificates they use to communicate with the DIP.

6.1.2 There are four actions in the management of certificate:

a) Issuing Digital Certificates;

b) Renewal of a Digital Certificate – prior to expiry;

c) Reissue of a Digital Certificate; and

d) Revocation of Digital Certificates.

6.1.3 A prerequisite to these actions is that the DIP User has passed organisational vetting and registration as part of the DIP On-boarding process.

6.1.4 Digital Certificates will be issued from the DCA and will be valid for 398 days, which equates to 13 months to allow for a one month overlap between Digital Certificates.

6.2 DNS Domain creation/verification

6.2.1 A prerequisite to requesting a Digital Certificate is to create and validate the DIP User's DNS domain with the DCA. A Certificate Admin can register the DIP User organisation DNS domain in the DIP.

6.3 DIP Digital Certificate requirements

6.3.1 DIP Users will need two sets of Digital Certificates:

a) One for non-production environments; and

b) One for production environments.

6.3.2 Where DIP Users utilise the services of DCPs the responsibility for the management of the Digital Certificates is split:

a) The DIP User owns the Digital Certificate required for Digital Signing and will need to authorise the DCP to use this Digital Certificate on their behalf; and

b) The DCP will need their own Digital Certificate for securing the mTLS connection.

Digital Certificate requirements by DIP User type

Organisation	TLS Certificate	Digital Certificate	Notes
DIP User (active)	Yes	Yes	The Digital Certificate is made available to allow DIP Users to verify signed Messages from the DIP. A Digital Certificate authorised for both mTLS and Digital Signing can be used or, separate Digital Certificates for mTLS and digital signing can be requested.
DIP User (non-active)	No	Yes	These organisations do not themselves connect to the DIP, instead a DCP acts on their behalf. They give authority to DCPs to sign Messages using their organisational Digital Certificates.
DCP	Yes	No	DCPs supporting multiple clients need only a single connection between their service and the DIP, secured by their own Digital Certificate. Messages must be signed using a Digital Certificate provided by the MP they are representing

6.3.3 There are three main scenarios in which different Digital Certificates can be used:

a) Scenario One – DIP User A direct to DIP:

i) DIP User A manages their own connection to the DIP using the same multi-use Digital Certificate for TLS and digital signing;

ii) They have one Digital Certificate for non-production environments, and one Digital Certificate for production environment;

b) Scenario 2 – DIP User B uses DCP X for connection to DIP;

i) DIP User B authorises DCP X to sign Messages on their behalf using DIP User B's Digital Certificate (this certificate does not allow mTLS);

ii) DCP X uses their own Digital Certificate for mTLS;

c) Scenario 3 – DIP User C uses multiple DCPs;

i) DIP User C requests multiple Digital Certificates for DCP X, DCP Y, etc.;

ii) DIP User C authorises both DCP X and DCP Y to sign Messages on their behalf;

iii) DCP X uses their own Digital Certificate for mTLS;

iv) DCP Y uses their own Digital Certificate for mTLS.

6.3.4 DIP Users directly connecting to the DIP (active Market Participant) will have a single Digital Certificate issued for both mTLS and Digital Signing.

6.3.5 DCPs may not use Digital Certificates for different purposes. For example, a Digital Signature may not be used for TLS, and vice versa.

6.3.6 Digital Certificates bind to an environment as well as an identity. This means that DIP Users and DCPs may not re-use Digital Certificates across different environments (non-production and production). Specifically, when a DIP User completes DIP On-boarding and the DIP Manager authorises them to move to the production environment, they will require new Digital Certificates.

6.4 Certificate Subscriber obligations

6.4.1 Once issued, and before the issued Digital Certificate is used, the accuracy of the information contained within must be reviewed.

6.4.2 By applying for a Digital Certificate and submitting information requested as part of an application, DIP Users (or a DCP acting on their behalf) are requesting the DCA to issue a Digital Certificate to them; and that they are impliedly agreeing to comply with the DIP Rules.

6.4.3 Information provided by DIP Users at the time of DIP On-boarding will be embedded in a Digital Certificate, and may be published in a directory of Digital Certificates and revocation information where required for the purpose of operating the digital certification services.

6.4.4 DIP Users consent to the disclosure of this information for these purposes and understand that they have the right to correct any information as required.

6.4.5 For a device or application, the individual responsible for the device or application must accept these responsibilities. For those entities who hold Digital Certificates, and act on behalf of Certificate Subscribers, the Certificate Subscriber must ensure all responsibilities are met.

6.5 Relying Party Obligations

6.5.1 Relying Parties are those entities that are using a Digital Certificate to authenticate another Digital Certificate subscriber named in the Digital Certificate.

6.5.2 Relying Parties should postpone any transaction which places reliance on/by any Digital Certificate issued by the DCA, until they are satisfied as to the trustworthiness of the Digital Certificate and the measures taken to support their reliance on the Digital Certificate. This applies when they:

a) Submit a query in search of a Digital Certificate;

b) Verify a Digital Signature created with a Private Key corresponding to a Public Key contained in a Digital Certificate;

c) Rely on a Digital Certificate, or on the information contained within a Digital Certificate in support of a transaction;

d) Otherwise rely on, or use any information or services provided by the DCA.

6.5.3 DIP Users accept that their use of Digital Certificate status information and their reliance on (or use of) any Digital Certificate (or the information embedded in it) will be governed by the DIP Rules.

6.5.4 If DIP Users rely on or use a Digital Certificate (including the information embedded in the Digital Certificate), without first meeting these obligations then their reliance on, or use of, that Digital Certificate (or the information embedded in it) will be considered a breach of the DIP Rules.

6.5.5 The DIP Rules define terms of use and reliance on Digital Certificates, together with Digital Certificate status information provided by the DCA. Relying Parties therefore accept that sufficient access to information is provided to ensure that the DIP Manager can make an informed decision as to the extent to which they choose to rely upon, or use a Digital Certificate, or the information embedded in it. Relying Parties further accept that, notwithstanding any determination by the DIP Manager, they are responsible for deciding whether to rely on a Digital Certificate or information embedded in it.

6.5.6 Subscribers to Digital Certificates must inform the DIP Manager should any passwords, passphrases, PINs, Private Keys, or other personal secrets used in obtaining authenticated access to PKI services be suspected of being compromised. On receipt of such a notice the DIP Manager will proceed in accordance with the DIP Rules.

6.5.7 Relying Parties acknowledge the possibility of theft or other form of compromise of a Private Key corresponding to a Public Key contained in a Digital Certificate which may not be detected by the Certificate Subscriber, and of the possibility of the use of a stolen or compromised key to authenticate an organisation or to forge a Digital Signature.

6.6 Certificate Signing Requests

6.6.1 The Certificate Admin should use the DIP to submit a CSR. Once signed by the DIP Verification Service Provider, the Digital Certificate is fulfilled and therefore considered as a Digital Certificate towards the DIP User's quota. The CSR completion only works on the server/service where the CSR was generated; should it be completed elsewhere then it will not complete.

6.6.2 Certificate request details:

Name	Description
Common Name	This value will contain a prefix for the environment and the domain which they are requesting a Digital Certificate for. The prefixes will be as follows: • energydip-nonprod – All Non Production environments • energydip-prod – Production environment For example, the following value could be specified:

	• energydip-nonprod.dipuser.co.uk • energydip-prod.dipuser.co.uk
Organisation Name	The name of the organisation as specified during registration as described in the main part of DSD002 'DIP Connection and Operation'
City	The city of the organisation as specified during registration as described in the main part of DSD002
State	The state of the organisation as specified during registration as described in the main part of DSD002
Country	The country of the organisation as specified during registration as described in the main part of DSD002

6.7 Certificate Revocation Request

6.7.1 Certificate Admins can submit CRRs using the DIP Portal. To revoke a Digital Certificate a reason for revocation must be entered.

6.7.2 On submission of the CRR, the DIP will request the DIP Manager approves revocation of the Digital Certificate. Following DIP Manager approval, the DIP will inform the DIP User that the Digital Certificate is successfully revoked.

6.7.3 Once revoked the Digital Certificate will no longer be valid when calling the DIP as either the mTLS or Message signing certificate. During the process of mTLS or Message signing, the OCSP is called. The OCSP is a property of the Digital Certificate and is an endpoint that specifies the Digital Certificate status (valid/revoked).

6.7.4 Digital Certificate revocation reasons:

Reason	Description
Key compromise	If the DCP's key has been lost, permanently deleted or if an unauthorised entity has been able to take possession of the key, the Digital Certificate must first be revoked before being recreated from scratch with a new key
Cessation of operation	If the DIP User ceases to operate, the Digital Certificate must be revoked. This reason can only be used by the DIP Manager.
Affiliation changes	This is when a key employee (an employee that has access to the Digital Certificate and associated keys) leaves the DIP User organisation
Digital Certificate superseded	If a new Digital Certificate has been produced for any reason, the old Digital Certificate will be superseded and will require revocation
Withdrawal of privilege	The DIP Manager has withdrawn/suspended the DIP User's DIP access in accordance with the DIP Rules, so their Digital Certificate should be revoked
Removal from CRL	If a Digital Certificate is accidentally revoked for any reason and should not be on the CRL, that Digital Certificate will need to be removed from the CRL.

6.8 Digital Certificate renewal

6.8.1 The DIP will notify the DIP User that a Digital Certificate is about to expire and therefore that they should generate a new CSR and get it signed via the DIP; prior to expiry a Certificate Admin should generate a new CSR via the DIP.

6.8.2 Notifications of Digital Certificate expiry will be sent to the DIP User Certificate Admin at the following intervals:

a) 90 days prior to the Digital Certificate expiring;

b) 60 days prior to the Digital Certificate expiring;

c) 30 days prior to the Digital Certificate expiring;

d) 1 day prior to the Digital Certificate expiring.

6.8.3 Renewing a Digital Certificate does not invalidate the existing Digital Certificate. The existing Digital Certificate will remain active for the remainder of the validity period allowing a grace period for seamless transfer.

6.8.4 The new Digital Certificate will start from the date the CSR has been completed, and not the date the existing Digital Certificate expires.

6.8.5 DIP Users requiring a copy of a Digital Certificate (if they are installing on multiple servers or devices for example) can reissue their Digital Certificate.

6.9 Private Keys

6.9.1 The size of IA and any supporting CA-Keys shall be not less than 4096-bit modulus for Rivest-Shamir-Adleman (RSA) encryption. The size of Subscribers' Private Keys shall be not less than 2048-bit modulus for RSA.

6.9.2 DIP Users, who are natural persons, must be authenticated to their cryptographic module before the activation of the Private Key. This authentication may be in the form of a PIN, pass-phrase password, or other activation data.

6.9.3 When deactivated, Private Keys must not be exposed in plaintext form.

6.9.4 Unless unavoidable, Private Keys should never be transmitted. That said, there is a requirement for DCPs to hold the Private Keys of subscribed DIP Users.

6.9.5 Where Private Keys must be moved, the advice is to export and transport Private Keys using a Public Key Cryptographic Standards (PKCS) #12 File (also known as a Personal Information Exchange (PFX) file).

6.9.6 Microsoft Windows servers have a utility through the Microsoft Management Console (MMC) that allows the export of an installed TLS server Digital Certificate along with its corresponding Private Key to a PFX file.

6.9.7 DIP Users with Linux-based servers can use OpenSSL to manage Digital Certificates and keys including creating various file bundles including PKCS #12 archives.

6.9.8 For other types of devices, the instructions will vary depending on the type of device being used. Please consult relevant documentation.

6.9.9 The PKCS #12 file encryption key should never be used for another transfer, even for a different Private Key.

6.10 Encryption Secrets/Password Guidance

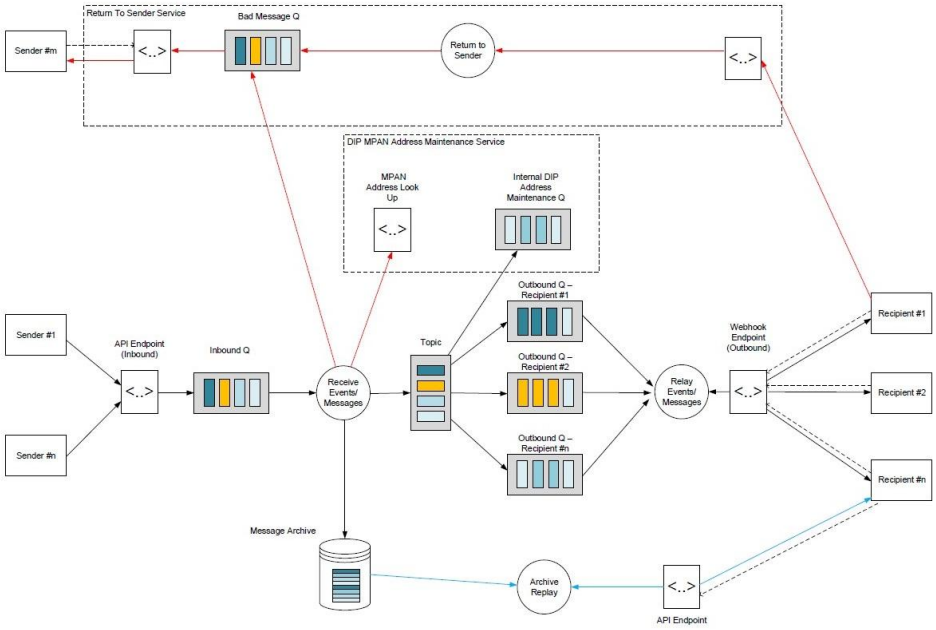
6.10.1 When creating encryption secrets or passwords to protect key material, DIP Users are referred to the NCSC's guidance on password strategies, and specifically the use of password managers and random words and phrases.

7 DIP Messaging

7.1 DIP Message pattern

7.1.1 Messages use a generalised publish and subscribe pattern that allows both many-to-many and one-to-many message exchange and has targeted message recipients.

7.1.2 Messages require a technical acknowledgement for the receipt and the subsequent sending of the message/event by the DIP, i.e. the API HTTP return codes (see Chapter 9) and response bodies to/from DIP Users. The requirement to log all API activity is deemed sufficient to meet any traceability/non-repudiation requirement and provide the acknowledgement back to the message sender.

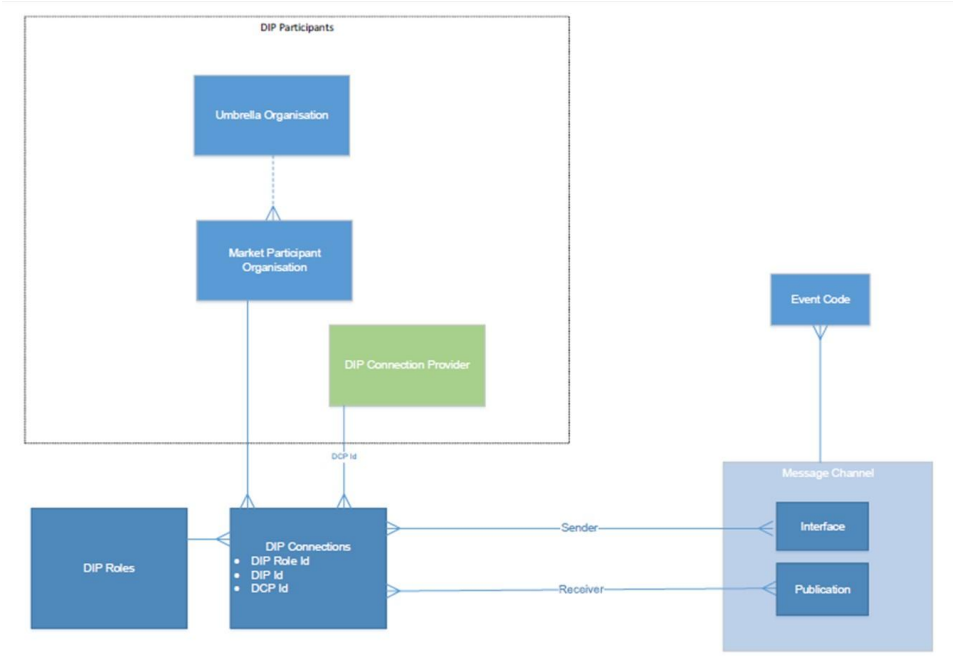


Example of a Messaging Pattern 'A' Template using a message queue-based architecture

7.1.3 Audit reporting facilities are made available to allow participants to check on the progression of individual message exchanges.

7.2 Message Architecture

7.2.1 The following is an overview of message architecture:



7.2.2 Message/event channels orchestrate how messages are exchanged between DIP Users. A message/event channel represents a message exchange between two DIP Users, an inbound flow called an Interface and a corresponding outbound flow called a Publication.

7.2.3 During DIP On-boarding DIP Users are assigned one or many DIP IDs and will have one or many market roles or services, depending on the roles they have registered for and the roles they are permitted to have. Access to individual message/event channels is determined by the DIP Users' DIP Role Code.

7.2.4 DIP Users may appoint a DCP to proxy a specific DIP Role (or all their roles) on their behalf by allocating the chosen role(s) to their DCP.

7.2.5 This ability to provide different proxies for managing different channels does not alter the addressing of messages; messages will still be addressed to the primary DIP ID rather than any of the proxy recipients, i.e. the addressee recipient or sender will always be the logical recipient/sender.

7.2.6 The proxy senders/recipients require their own set of Digital Certificates to interact with the DIP. The DIP is aware of the proxy recipients (appointed DCP) and will digitally sign messages appropriately, i.e. the physical sender or receiver of the message.

7.3 DIP IDs

7.3.1 To support the sharing of data as required by Industry Code processes, DIP Users have a common mechanism for identification across all industry systems – a Market Participant ID (MPID), and MPIDs map directly to DIP IDs.

7.3.2 DIP IDs are used in the following way:

a) DIP ID and DIP Role will be used to identify DIP Users within the DIP

b) DIP Addressing/Routing utilises DIP ID & DIP Role code

c) ISD (entity #M16) enables a 1-2-1 mapping between each DIP ID/ Role combination and MPID

7.4 Message/event channels

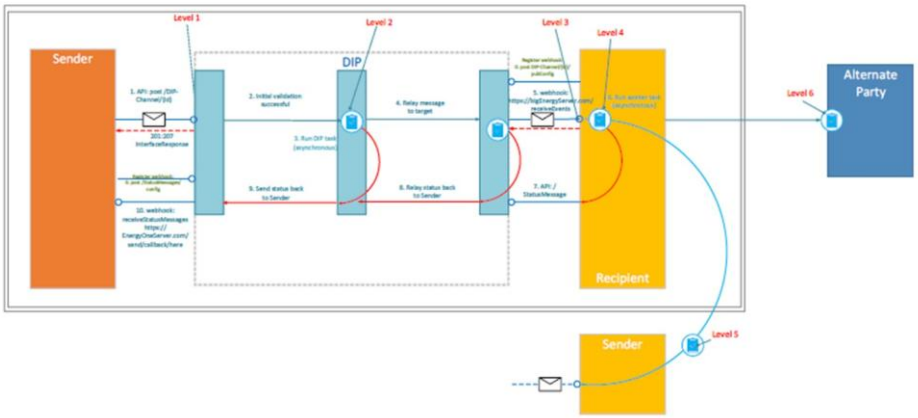
7.4.1 Each message channel uses a standard RESTful architecture for both the inbound Interface and the outgoing publication:

a) a Send Message API for incoming messages (https:/api.{environmment}. energydataintegrationplatform.co.uk/ {version}/ dip-channel/ {IF-xxx}); and

b) a Receive Messages Webhook for outgoing messages.

7.4.2 Each message channel has both synchronous and asynchronous methods for reporting status/error messages back to the Sender.

7.4.3 A standard event/message channel has the following pattern:



The points at which the different level of validation occurs are also shown.

7.4.4 In a message exchange there will be 4 levels of message acceptance/validation within the message channel itself:

a) Level 1 - initial synchronous validation by the DIP;

b) Level 2 - secondary asynchronous validation by the DIP;

c) Level 3 - initial synchronous validation by the DIP User;

d) Level 4 - secondary asynchronous validation by the DIP User.

7.4.5 Then, outside of the message channel other validation may occur, levels 5 &6, which relates to the workflow being undertaken:

a) Level 5 – Response through a new message channel where the original recipient becomes the new sender e.g. IF-006 provides a response to IF-005;

b) Level 6 – where the response is via a third-party, e.g. Service Appointment by the Supplier (IF-031) via the Registration Service - the outcome and if appropriate error reason code will be contained within the body of a subsequent message (IF-035).

7.4.6 The processing within a message channel follows the following pattern:

a) Recipient registers Webhook for receipt of publication; sender registers Webhook for the receipt of status messages;

b) Sender sends message/events to the DIP via the post /DIP-Channel/{IF-xxx} API;

c) Initial receipt of events/messages by the DIP is accompanied with an auditable acknowledgement. This will include the Level 1 validation HTTP response;

d) The message traverses through the DIP where it undertakes internal processing where Level 2 validation also occurs;

e) Message is relayed to the intended recipients;

f) Onward delivery of message/events from the DIP to the recipient occurs via registered Webhook with an auditable acknowledgement and Level 3 validation;

g) The recipient will undertake further validation checks (Level 4);

h) If an error condition is found these are reported back to DIP via the Status Message API;

i) Both the Level 3 synchronous and Level 4 asynchronous response reported back to Sender;

j) DIP relays all Level 2/3/4 responses to the Sender;

k) Sender receives all Level 2/3/4 responses via the Status Messages Webhook.

7.4.7 The DIP handles the message in two phases:

a) a synchronous check at the API that is communicated directly back to the Sender, and

b) further asynchronous checks and addressing that are carried out once the message has been initially received.

7.4.8 The initial API is intended to be as lightweight as possible thereby increasing message submission speed.

7.4.9 Where no error condition is encountered no event/message acknowledgement other than the initial API response is sent back to the Sender.

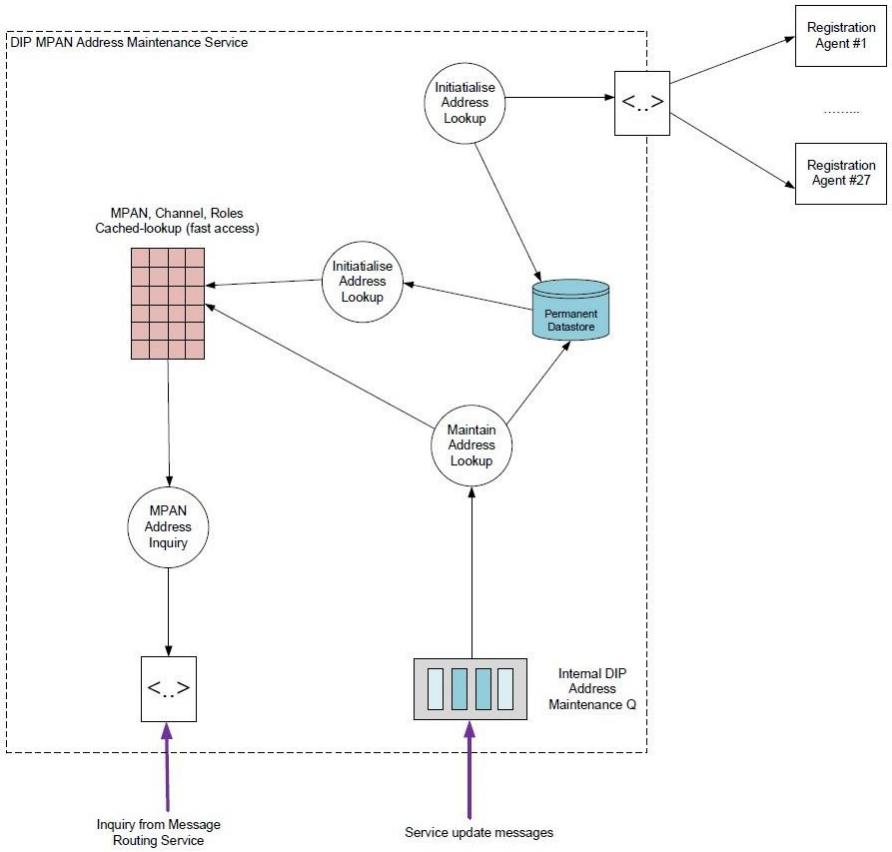
7.4.10 Some process acknowledgements, either positive or negative, are recognised as distinct message/event flows.

7.4.11 Security between DIP Users and the DIP is ensured by securing a mTLS connection via TLS handshake.

7.4.12 On receipt of message a DIP User should endeavour to validate the message contents as much as feasibly possible and relay all validation errors back to the Sender i.e. not stop at the first error.

7.5 MPAN Address Maintenance Service

7.5.1 To facilitate the addressing and routing of messages, the DIP has the DIP MPAN Address Maintenance Service (DIP MAMS). The DIP MAMS is responsible for maintaining a routing table that provides the messaging services with an instant address lookup for incoming messages based on MPAN. The lookup table will be based on MPAN, Message Channel, intended recipient roles and date/time.



DIP MAMs

7.5.2 The DIP MAMS works in the following way:

a) Each Registration Service must present an API that the DIP can query to initialise its permanent data store;

b) Data will need to be kept in two places: a permanent and a data cache;

c) The cache is populated from the permanent store and provides a fast lookup and the permanent store provides the permanent store;

d) The MAMS receives an input MPAN and message channel, and returns a list of downstream recipients;

e) After initialisation, the DIP is responsible for maintaining MAMs data in real time;

f) All message channels pertinent to the registration data flow will send a message to an internal queue within the DIP – This queue provides the MAMS function information;

g) Both permanent and cache are updated with the new details; and

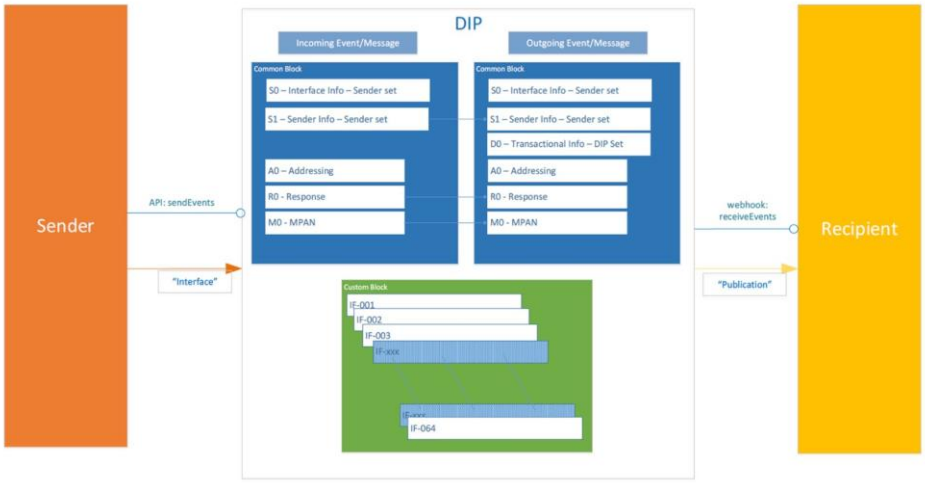
h) MAMS can recognise a change of details and retain data for the historical lookup as 'old' messages may arrive that need to be sent to a previously responsible party;

i) For example, late consumption data (INTF-011) would need to be sent to the set of secondary parties that are pertinent for the specific day for the meter read – in this scenario, the header would define an applicable date/time field used for addressing.

7.6 Message structure

7.6.1 Each message channel is defined by a specific interface definition, which defines the incoming messages, and also a publication, which defines the corresponding outgoing message. There is a direct 1:1 mapping between interfaces and publications.

7.6.2 Message structures are specified through the definition of a JSON schema. The definition of the schema to be used for the DIP can be found in the EMDS.



7.6.3 Each message will have two distinct components a common block and a custom block. The contents of the common block in the interface incoming message will differ from that of

the outgoing message in the publication whilst the custom block passes through the DIP unchanged.

7.6.4 In a standard message exchange the following blocks are sent and received:

		Sender	DIP	Archive	Receiver
Interface		API/ DIP-Channel/ {IF-xxx}			MP Webhook
Header/S0	Sender	X		#	#
Header/S1	Sender	X		#	#
Header/A0	Address	O	O	#	#
Header/R0	Response	O	O	#	#
Header/D0	DIP		X	#	#
Header/M0	MPAN	X		#	#
Message Body		+		#	#

Formatted: Italian (Italy)

7.6.5 When a message is sent, the Sender sends a message with the blocks indicated above. The blocks with a 'X' are the sole responsibility of the Sender or the DIP. For those with 'O' some of the fields are populated by both the Sender and DIP. The '+' denotes the potential for some data fields to be obfuscated by the DIP. The message blocks written to archive and sent to the recipient are marked '#'.

7.6.6 Messages adopt the following convention for representing optional and required data items in messages:

a) all data items properties are marked as required, where data items are optional then they will be marked as nullable.

b) The underlying datatype for each data item will reflect the nullable property, i.e., nullable – true/false.

7.6.7 Worked examples for different data types can be seen in the table below (This is also valid for any enumerations that allow nullables.):

Data Type	Nullable value to use
String	Null
Boolean	Null
Number	Null

7.6.8 Messages exchanged over the DIP have a discernible audit path as they progress from initial receipt to final delivery via the API transaction logs and the message archive.

7.6.9 Each individual message will be uniquely identified with a Sender Unique Reference by the Sender, and the DIP will provide a corresponding Transaction ID for each message that will be returned to the Sender in the API HTTP response body of the Send Messages API call.

7.7 Message routing

7.7.1 The DIP will address messages between DIP Users in one of three ways:

a) Always – This option does not require specific addressing action by the Sender. Configuration on the DIP will ensure that certain Messages are always delivered to certain recipients. For a specific message channel a message is either always sent to a named DIP User or all DIP Users assigned to a designated role (role is assigned to the message channel)

b) Secondary – The DIP holds Routing Table of 'Recognised Parties' e.g. Registration Service, LDSO, Current Supplier, Current Service Providers, etc. The DIP uses this routing table to address certain messages. The MPAN core in the M0 block and optionally the Event Code and/or other fields (to determine the targeted roles) provides the lookup to the MPAN lookup address function.

c) Primary – Set by the Sender of the message. The purpose of this is allow a message to be addressed to a specific DIP User or more commonly where only the sender is aware of the correct recipient. DIP uses the ‘Primary Recipient List’ in the A0 block in the message header.

7.7.2 The DIP can apply a single or all types of addressing to a single message channel. Messages are sent to DIP ID and/or DIP Role Codes, hence a DIP User can receive a message more than once if they undertake multiple roles that are the recipients for the message. However, each message will only be sent once to a single DIP ID/DIP Role Code pairing.

7.7.3 The Interface ID and event Code/other fields determine which type of addressing is appropriate if message is inappropriately addressed by the Sender i.e. primary recipient is undefined when it is required, then an error message is returned.

7.7.4 The DIP will add the correct addresses to the Addressing block (A0) to the message before the message is archived.

7.7.5 Each message format within the EMDS details who the ‘Always’, ‘Secondary’, and ‘Primary’ recipients are for that Message.

7.8 Message/event obfuscation

7.8.1 The DIP will obfuscate the contents of a message based on the recipient's role within a particular message channel. This requirement is implemented so that the message sender does not have to send multiple messages to different recipients containing a subset of data;

instead, a single message is sent to multiple recipients. As all data items within a data block are mandatory, the DIP will replace the data with the flowing values:

a) String values - multiple 'x' for the maximum length of the field

b) Date values – the date '1900-01-01 00:00' is used

c) Numeric values – no requirement at present

d) Boolean values – true

e) DIP IDs – '9999999999' (to ensure it remains a numeric string for regex validation)

7.8.2 It is the responsibility of the recipient to know that they are receiving obfuscated data on a specific message channel and must not solely rely on message contents.

7.9 Message Compression Handling

7.9.1 Message compression is used in the BSC systems reports & LDSO E-Bill Interfaces to ensure that the report size is within a notional upper limit for the size of messages accepted by the DIP, as well as BSC systems reports received by upstream & downstream DIP Users. This message pattern for compressed payload involves using GZIP to compress the data, then BASE 64 to encode the data. The reverse is true for decompression.

7.9.2 This pattern for compressed payloads is used in the following reports:

a) REP-002 Supplier report for DUoS – aggregated data;

b) REP-002A Embedded Network report for DUoS – aggregated data;

c) REP-002B LDSO report for DUoS – aggregated data;

d) REP-003 BM Unit Allocated Demand Volumes to Market Participants;

e) REP-003A Aggregated BM Unit Allocated Demand Volumes to Supplier;

f) REP-004 Supplier Deemed Take Report;

g) REP-006 Aggregated Uncorrected volumes by CCC to Market Participants;

h) REP-007 VAS Exception Report to Suppliers and BSCCo;

i) REP-008 MDS Exception Report to LDSOs;

j) REP-009 EMRS Report for Suppliers;

k) REP-900 LDSO – DUoS E-Bill;

l) REP-901 LDSO – Aggregated DUoS Charges;

7.9.3 Compression and Decompression are concerned with the data contained within the Custom Block.

7.9.4 The following steps happen for compression as follows:

a) The sender compresses the contents of the Custom Block of the report using GZIP;

b) The sender takes this compressed data and then BASE 64 encodes the data;

c) The sender takes this BASE64 encoded data and inserts this into the (empty) CustomBlock (P03 property as below).

7.9.5 Any data/ schema validation should be completed by the sender prior to data compression steps above. Message processing continues as with other interfaces and the Common Block/ HTTP Header is not changed. Message signing will take place on message payload post data compression.

7.9.6 In the DIP Open API Definitions (DIP Swagger), the Custom Block for the BSC systems reports contains the following:

a) P03 property – to be used for the compressed payload following the instructions above; and

b) Schema property – this is the uncompressed message to be used for validation and will not be populated/ contain any data.

7.10 Message De-compression Handling

7.10.1 The recipient will know that a specific message flow is compressed, and hence will need to decompress the payload. The following steps happen for decompression as follows:

a) The receiver BASE64 decodes the contents of the CustomBlock (P03 property as per previous section);

b) The receiver takes this BASE 64 data and decompresses using GZIP;

c) The receiver will now have the decompressed payload for validation and onward processing.

7.10.2 Any data/ schema validation should be completed by the receiver after the data compression steps above. Message processing continues as with other interfaces and the Common Block/ HTTP Header is not changed. Message signing will take place on message payload pre data de-compression.

7.11 Message configuration requirements

7.11.1 For most Messages sent to or from the DIP, there is a requirement to apply Message signatures – the exception being some HTTP responses/acknowledgements which do not need to be signed.

7.11.2 Receiving applications must verify the Message signatures, effectively performing authentication and authorisation of the Message sender, and confirming that the Message has not been tampered with in transit. Verification of each Message should be performed using the digital signature relevant to the environment being used (non-Production/Production).

7.11.3 On receipt of a Message, and where JSON schema validation is successful, the recipient will authenticate the Message by verifying the Message signature which is required on all Messages. This Digital Signature uses an RSA 256 key.

7.11.4 Messages that fail signature verification will not be processed by the DIP. In this instance, the DIP User will be advised of authentication failure as part of the HTTP response or acknowledgment generated and sent to the requesting DIP User, along with an appropriate response code.

7.12 Message Egress

7.12.1 The DIP has operational service limits that can be enforced when constructing messages for egress.

7.12.2 For Message egress, a maximum number of events and maximum payload size shall be used to construct the outgoing payload size in the call back request. The DIP will combine Messages for a DIP User to send based on the maximum payload size and maximum number of messages per payload. A worked example of this is below.

				Transaction (Tx) Sizes (kb)						
Message size (kb)	**No of Messages**	**Max payload size (kb)**	**Outcome example**	**Tx1**	**Tx2**	**Tx3**	**Tx4**	**Tx5**	**Tx6**	**Tx7**
100	3	1000	Messages pushed after [XX]ms wait	300	-	-	-	-	-	-
100	13	1000	Ten messages pushed out instantly Three messages pushed out after the initial transaction completes	1000	300	-	-	-	-	-
400	3	1000	two messages pushed out instantly One messages pushed out after the initial transaction completes	800	400	-	-	-	-	-
1300	1	1000	One message pushed after [XX]ms wait	1300	-	-	-	-	-	-

1300	7	1000	Seven messages pushed after [XX]ms wait	1300	1300	1300	1300	1300	1300	1300
5000	1	1000	One message pushed after [XX]ms wait	5000	-	-	-	-	-	-

7.12.3 In the above table the Maximum payload size is defined by the MP and the outcome example is from the DIP to the MP.

7.12.4 DIP Users are expected to adopt a retry with exponential back off (up to a configurable maximum wait time) if there is a failure in connecting to the DIP.

7.13 Message Ingress

7.13.1 When sending messages, DIP Users are expected to adopt a 'retry with exponential backoff' (up to a configurable maximum wait time) process if there is a failure in connecting to the DIP, i.e. a corresponding HTTP response of 408, 429, 500-504 (inc).

7.14 Capacity Management

7.14.1 In the event of DIP Users or DCPs exhibiting extreme patterns of data submission, the DIP Manager may (following investigation) implement controls that will limit API access.

7.15 Message Security

7.15.1 Some of the data flowing through the DIP includes personal data which is protected under the Data Protection Legislation. As such, there is a requirement to ensure the security of the data, which is achieved by the use of an mTLS connection between DIP and DIP User.

7.15.2 There is also a requirement for message repudiation, i.e., ensuring a message is sent from only the expected party, hence all messages sent to the DIP are digitally signed by the Sender. To help simplify the message privacy classification the DIP has four different security categories:

1. Public Data – digitally signed;
2. MPAN – digitally signed;
3. MPAN + personal data – digitally signed;
4. MPAN + Consumption data – digitally signed.

7.15.3 Each of the MHHS interfaces is assigned one of these categories.

7.15.4 All JSON messages should be encoded using UTF-8 as a standard, to ensure correct encoding and decoding across DIP users.

7.16 Message Alarms

7.16.1 Where a message/event channel's latency has extended beyond a predefined threshold and messages have been moved to 'dead letter' queues, an alarm will alert the DIP Manager. This is to indicate that messages are not being processed promptly, and as such the DIP Manager shall investigate the reason.

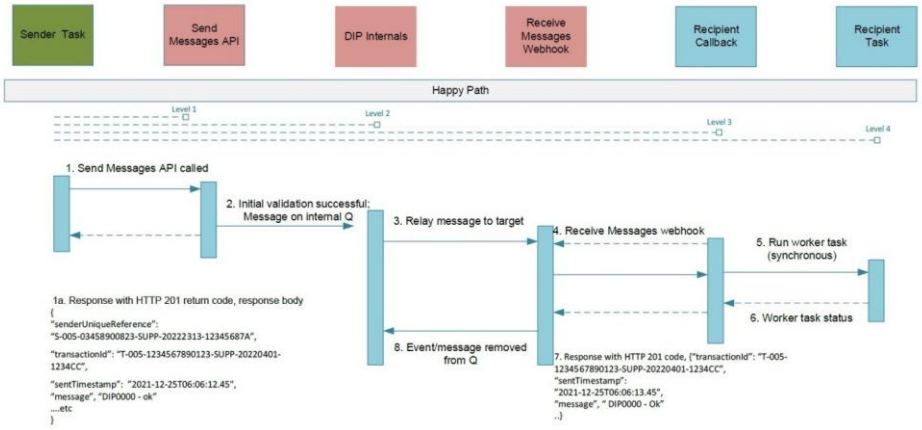
8 Message choreography

8.1 Message choreography

8.1.1 The following diagrams describe the 6 different scenarios that can occur on a simple message exchange between a single DIP User service and the DIP and includes the different levels of validation.

8.1.2 For diagram clarity, the step showing message being written to the DIP archive are not shown on any of the diagrams. The diagrams only show a simple scenario where a single message is exchanged between Sender and the DIP, and then relayed by the DIP to recipient.

8.2 Simple Message Exchange



Standard Exchange

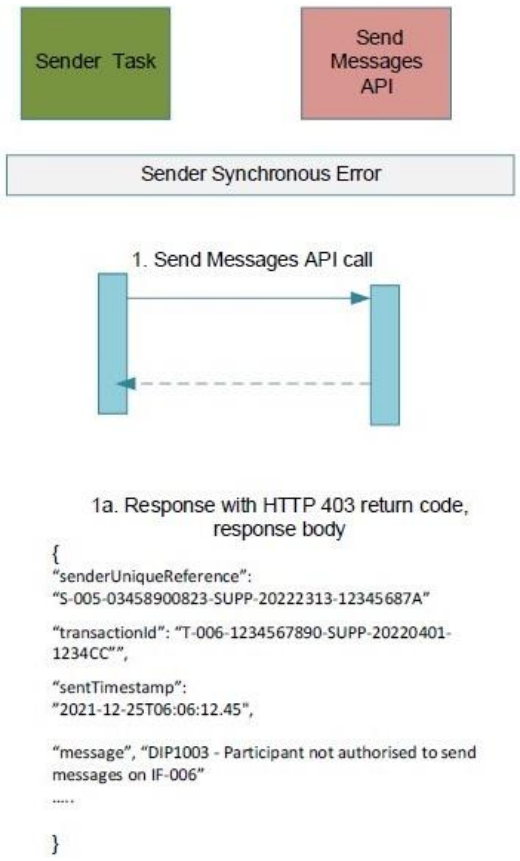
8.2.3 In this exchange the message recipient will only receive the initial synchronous acknowledgement from the DIP (Level 1 validation), there is no acknowledgement from the recipient of the message back to the sender. However, there is an acknowledgement from the recipient to the DIP (Level 3 validation) for the message but this is not relayed back to the sender.

8.2.4 This follows the principle that only processing and data errors are returned to the sender, and successful processing is not routinely reported. Both synchronous responses, initially by the DIP on the initial API call (steps 1&1a above) and the Webhook API call back (steps 4&7) are logged in the DIP to meet the audit requirements.

8.2.5 Message auditing logging is not shown. The audit trail is available to sender via a DIP report.

8.3 Level 1 validation - DIP Synchronous rejection

8.3.1 In the scenario a message is synchronously rejected by the DIP. In the example below the message has been rejected as the sender does not have the privilege to send messages on IF-006, so the response body clearly states the sender is not allowed to send messages on IF-006.



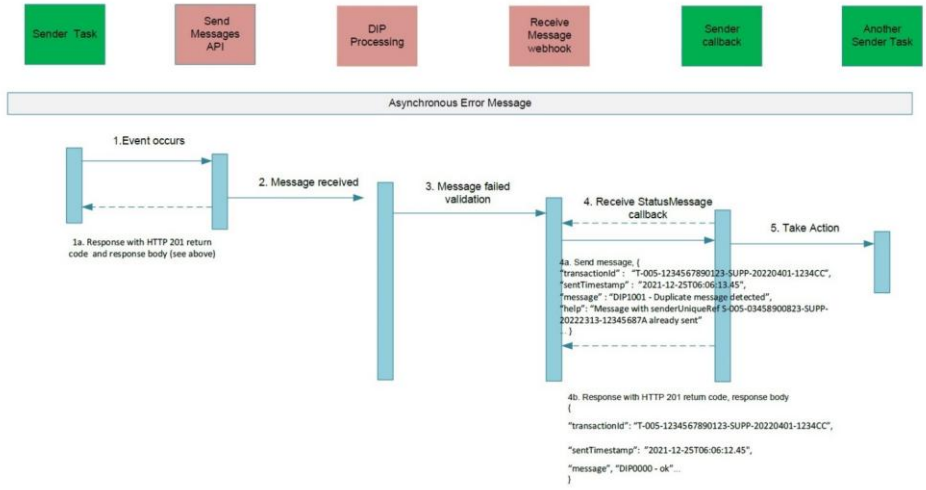
Level 1 DIP synchronous error

8.4 Level 2 validation – DIP asynchronous rejection

8.4.1 After the initial successful receipt of the message the DIP encounters an error with a sender's message; in this example the DIP finds the signature invalid.

8.4.2 The DIP informs the sender of the problem, a message is sent back via the 'Status Message' Webhook, with return message containing the Transaction ID, an error code and a description of the problem.

8.4.3 The Webhook call back responds that the status message has been received and the sender will need to take some action accordingly.

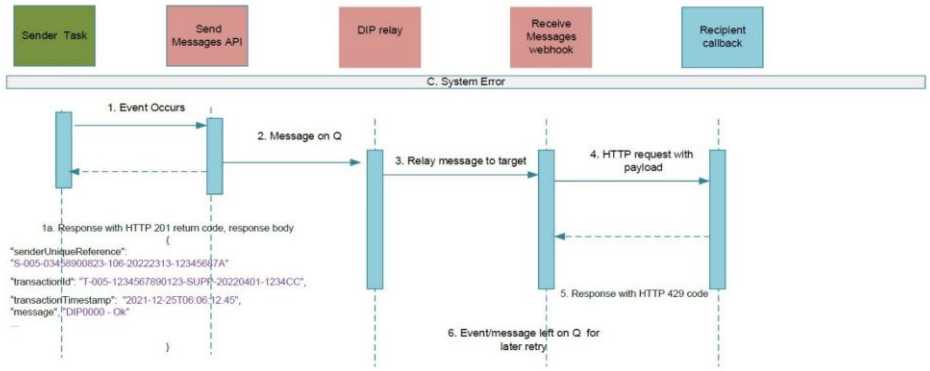


Level 2 – DIP asynchronous error

8.5 Level 3 response - Recipient Synchronous Error (with retry)

8.5.1 Recipient Synchronous Error with retry where the response from the recipient Webhook indicates a system error, e.g. with an HTTP 429 'Too Many Requests' response.

8.5.2 The message has not been consumed and is retained in the queue for later processing. The DIP has the logic to attempt to resend the message after a timeout period. If the message is still not sent after a maximum retry period has elapsed then the message is moved to the 'Dead Letter Queue'.

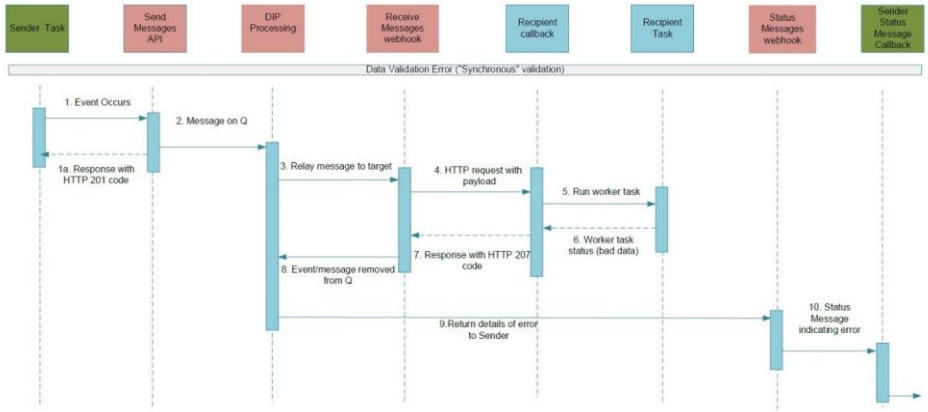


Recipient synchronous error (retry)

8.6 Level 3 validation - Recipient Synchronous Response (no-retry)

8.6.1 'Recipient Synchronous Error' with no-retry is the case where the recipient's system rejects the message via the Webhook call, i.e. the message has been consumed and rejected.

8.6.2 In this scenario, the Webhook reports a mixed/error return, and the DIP relays the error received from the 'Receive Messages' Webhook back via the 'Status Messages' Webhook to the originating DIP User and the message is removed from the outbound queue.



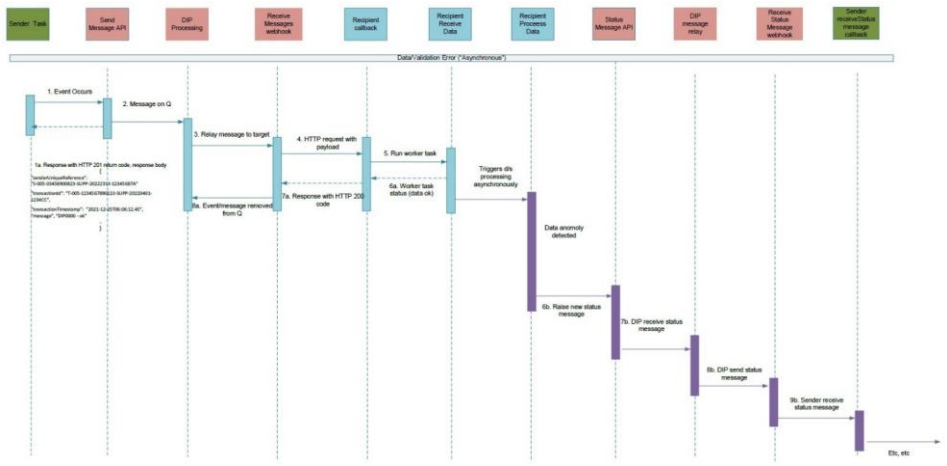
Recipient synchronous error (no-retry)

Formatted: Spanish (Spain)

8.7 Level 4 validation - Recipient validation response (asynchronous)

8.7.1 In this scenario the recipient of the message discovers an error with the message and needs to report the response back to the sender. i.e. the recipient's system can consume the data, however, there is an inconsistency with the data which is not reported on the initial call back.

8.7.2 The initial message exchange is identical to the standard use case described above however, there is an additional message sent back via the Send Status Message API.



Recipient data error – asynchronous reporting

8.8 Consumption Replay

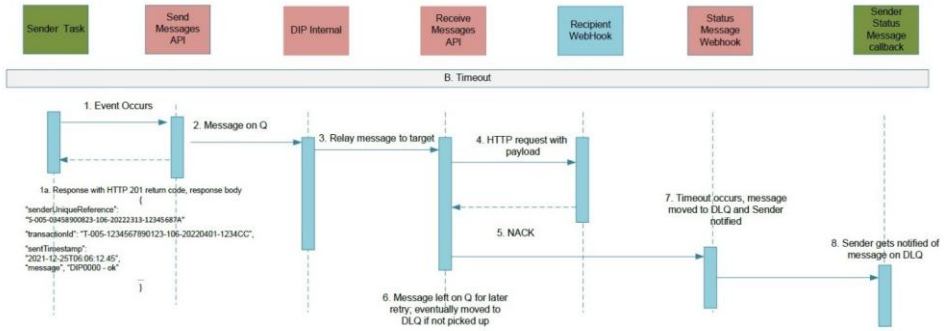
8.8.1 Message Channels 15 & 16 are used for the replay of historic consumption data and do not follow the standard design pattern of the other message channels. Channels 15 &16 perform a request/response type of exchange where the initial message via interface 15 is a request for historic consumption data which the DIP received and processes. The DIP will search the message archive for the requested data and send the corresponding messages back via publication 16.

8.8.2 This transaction is only available to Advanced Data Services. All consumption records (IF-021 / IF-013) submitted/received by previous Advanced Data Services for a period of four months prior to SSD will be replayed. The rationale for splitting into two channels is due to the different formats of the incoming request and the outgoing messages containing consumption data.

8.9 Recipient timeout and 'Dead-Letter Queue' handling

8.9.1 The diagram below shows the case where there has been no response from the recipient Webhook and the call back times out. The message is retained in the queue for later processing. The DIP has the logic to attempt to resend the message after a timeout period.

8.9.2 If a message/event has not be retrieved by a recipient within 14 days, the DIP will write the message to a 'Dead-Letter Queue' (DLQ). Messages on the DLQ will be reported back to the message's sender via the Status Message Webhook.



8.10 Error Handling & Message Distribution Patterns

8.10.1 The majority of the DIP message flows originate from either the Registration system(s) or the BSC Central Systems (LSS, MDD, ISD) and the principle adopted is that these systems are considered to be the 'System of Truth' in the message exchanges, and hence discrepancies are raised by DIP Users that either send or receive messages to/from these services.

8.10.2 Most message exchanges via the DIP do not involve a simple message being sent from a sender to a single recipient. Depending on the addressing requirements of each message channel, messages are often sent to multiple recipients. All the message distribution (MD) patterns assume that the original message has been consumed by one or more recipients, and the recipient has sent a 'Status Message' back to the sender via the DIP detailing the issue with the data.

MD	Pattern	Description	Interfaces	Validation resolution
MD#1	Registration In	A single DIP User sends a message solely to the Registration Service	IF-005, IF-007, IF-025, IF-031, IF-034, IF-038	Registration will send validation errors back to the sender. Sender will aim to correct data and resend; if sender considers the data to be correct then raise support ticket on Registration.

MD	Pattern	Description	Interfaces	Validation resolution
MD#2	Registration Out	The Registration Service sends out a broadcast message, typically a confirmed update from the Registration system, to many DIP Users	IF-001, IF-002, IF-006, IF-008, IF-009, IF-018, IF-026, IF-032, IF-033, IF-035, IF-036, IF-037, IF-039, IF-043, IF-044, IF-045, IF-050	Registration data is considered correct. If participant interface raises an error, they will need to track the error via the DIP. Check if other DIP Users have raised an error on the message. If sender still considers there is an issue with the data then raise a support ticket on Registration (or update existing ticket)
MD#3	BSC systems Out (LSS, MDS, ISD, VAS)	One of the BSC systems (LSS, MDS, ISD, VAS) sends out a broadcast message to multiple DIP Users	IF-013, IF-014, IF-022, IF-023, IF-040, IF-047	LSS, MDS, VAS, ISD data are considered correct. If participant raises an error, they will need to track the error via the DIP. Check if other DIP Users have raised an error on the message. If sender still considers there is an issue with the data then raise a support ticket on LSS/ MDS/ VAS/ ISD (or update existing ticket)
MD#4	IF-021	The Half Hourly consumption data interface is considered separately	IF-021	With IF-021 the default position is that the responsibility will be for the SDS to send correct data. If an error message is received from MDS then the SDS must take initial responsibility to trace the issue. If IF-021 is rejected by another DIP User other than the SDS then the onus is on that DIP User to pursue any action, i.e. raise a support ticket, noting they should establish whether the

MD	Pattern	Description	Interfaces	Validation resolution
				MDS successfully received the same data via the DIP audit trail.
MD#5	Point-to-Point	Messages between two single DIP Users	IF-024, IF-027, IF-028, IF-004	With point-to-point interfaces the onus is on the recipient to raise any issues with errors detected in the message exchange.
MD#6	Request	DIP User sends a data request to the DIP (normal BAU, not event replay)	IF-015	In the event of an error being returned by the DIP, the onus is on the sender to raise a support ticket if they see no issues with the original request.
MD#6	Response	Participant receives a data request to the DIP (normal BAU, not event replay)	IF-016	In the event of an error being returned by the DIP, the onus is on the sender to raise a support ticket if they see no issues with the original request.

8.11 Batch Message Handling

8.11.1 For ease of understanding, the examples in this chapter describe the processing of individual messages rather than a group of messages. In practice DIP Users send batches of messages rather than individual messages via the 'Send Messages' API and the call back from receive events Webhook within a single HTTP transaction.

8.11.2 In this scenario the course of individual messages will follow the sequences above, however, as each connection can only result in a single HTTP return code that will indicate the success or otherwise of the complete transaction and this be reflected as follows:

Response	Meaning
201 – Created	All messages created
207 – Multi status	Some messages created; see response body for details
4xx/5xx	See DIP Open API Definitions (DIP Swagger)

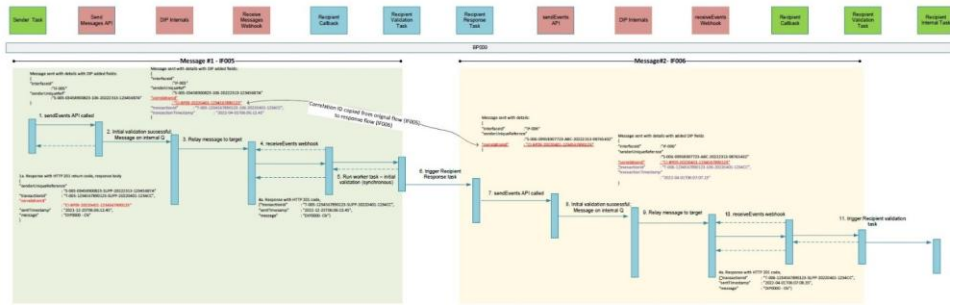
8.11.3 In addition the DIP will undertake schema validation on the whole batch of messages rather than individual messages. If there is a validation issue with a single message in a group then the whole transaction will fail and no messages are accepted. The DIP will respond with a 403 response and a MSG1001 code.

8.12 Workflow message handling

8.12.1 The message flows described above are for a single instance of a message over an interface/publication along with acceptance/rejection paths for that message. For most workflows an instance of each workflow will compromise of a series of messages with the receipt of one message triggering a processing step that will result in subsequent message. In order to link a single thread of a workflow process, the 'Workflow Correlation ID' is used.

8.12.2 The 'Workflow Correlation ID' is generated by the DIP on receipt of the first message in a workflow sequence; the DIP generates 'Workflow IDs' for specific interfaces. Recipients of the initial message, and any further recipients of messages in the workflow sequence, will have the responsibility of transcribing the 'Workflow Correlation ID' from the transaction block from the received message to the transaction block in any subsequent return messages.

8.12.3 Participants will also be able to track specific threads of workflows by tracking 'Correlation ID' and MPAN via the DIP audit reports. This is best explained with the aid of an example for the 'Change of Meter' process. This process consists of two message flows; IF-005 and IF-006. IF-005 is the initial message, and IF-006 is the response.



Correlation Id example with IF-005 & IF-006

8.12.4 A worked example on this is as follows for IF-005 / 6:

1. For IF-005, a Metering Service Smart (MSS) or Metering Service Advanced (MSA) will send a message to the Registration Service, using the DIP;

2. As part of Level 1 validation, the DIP will respond synchronously back to the MSS or MSA and this will contain the Correlation ID (DT-011). The MSS/ MSA will need to store this 'Correlation ID';

3. The DIP will (also) add the Correlation ID to block D0 and send PUB-005 to the Registration Service with the added information (in the common blocks), as well as the information in the original messages (in the common and custom blocks). The Registration Service (in this example) will also need to store this 'CorrelationID'; and

4. At this point in time, both the MSS / MSA and Registration Service will have / store this "Correlation ID".

5. At a future point in time, as per the process, the Registration Service will send a message, using IF-006.

6. For IF-006, the Registration Service will populate the S1 block with the sender 'Correlation ID' (as this is a message that requires correlation). This is populated from Step 3 above (the one the Registration Service stored from IF-005).

7. The DIP will not change the data in the S1 block, it will pass this information through as per the design. The DIP will not populate the D0 block with a new Correlation ID. This way we have one Correlation ID that is consistent across all related messages.

8. The DIP will send this message to the MSS/ MSA (in this example) using the PUB-006 message and this will contain the 'Correlation ID'. The MSS/ MSA can look this up in their systems against the 'Correlation ID' stored previously (as described in Step 2 above) to continue processing as per the design.

8.13 System non-availability

8.13.1 Where a DIP User's system will be unavailable, whether as a result of planned maintenance or otherwise, the DIP Manager shall be informed. The DIP Manager will inform other DIP Users of such outages if an outage means that DIP Users may not be able to deliver or receive messages within the time frames required elsewhere.

8.13.2 DIP Users shall not plan routine outages during the period in which a Registration Service is processing Secured Active messages in accordance with BSCP706 'Supplier Meter Registration Service for MHHS Metering Systems'.

9 Use of APIs and Webhooks

9.1 Send Messages API

9.1.1 The send event API (https:/api.{environmment}.energydataintegrationplatform.co.uk/{version}/dip-channel/{IF-xxx}) is used to submit messages to the DIP. The full Open API definition can be found in the DIP Open API Definitions (DIP Swagger).

9.1.2 The message sender will need to send messages to ~~channel-specific~~channel-specific endpoints, i.e., all messages need to be of the same IF. The messages within a single transaction (API call) can have mixed MPANs.

9.1.3 The return code and response body ~~provides~~ provide the sending DIP User with an auditable trace on the success of the receipt of the messages by the DIP.

9.1.4 On sending a set of events/messages, the Sender shall undertake the following:

a) **Message Construction** – On sending a message, the Sender is responsible for ~~construction~~ constructing the message header and payload as described in ~~the~~ S~~s~~ection ~~below~~9.1.5. The use of Sender Unique Reference is described ~~below~~in the S1 Sender blocks descriptions – Mandatory table in Section 9.1.5.

b) **Message Addressing** – When a message is sent, the DIP User will need to be aware of the type of addressing that needs to be applied and address the message accordingly. The type of addressing for each message channel is defined in the EMDS. The ~~Sender~~ DIP User sending a Message is only responsible for primary addressing~~,~~; the DIP is responsible for secondary and always addressing.

c) **Message Signing** – Once the message has been constructed, it needs a digital signature written to the message header.

d) **API Call** – The sender ~~has the ability to~~can control the number of messages sent within each API call, and ~~to~~ send multiple ~~event~~events/messages in a single API call. Messages can only be sent to a single message channel within the same API call.

e) **API Response** – The message sender will need to process the API response. The HTTP return code will provide the return for the whole transaction and needs to be considered with the response body, as it will provide the details for the acceptance or otherwise for each message sent, and will provide Transaction IDs for each message referenced by the Sender Unique Reference.

f) **Back Off and Retry** – DIP Users are expected to adopt a retry with exponential back off (up to a configurable maximum wait time) if there is a failure in connecting to the DIP.

g) **Multiple Connections** – DIP User may have multiple connections to the API endpoint. DIP Users may wish to logically separate their own interfaces and have a single connection per interface, and the DIP will support this pattern.

9.1.5 The sender will need to populate the following blocks ~~on~~ when sending a message. The tables below detail the validation that the DIP will undertake on each of the message fields:

Field	Description	DIP Validation
Message Interface ID	The Interface ID	Check valid Interface ID
Event Code	Used to route messages to specific parties	Check valid Interface ID/Event Code pair is valid
Schema Version	Version of the schema used in the message	Check version is valid for the interface in question

Sender (S0) block descriptions – mandatory

Field	Description	DIP Validation
Environment	The environment indicator, i.e., PROD, PREPROD, SIT, UIT, DEV	Check the environment is in the allowed list of values. Check that the environment is correct for the current instance.
Sub Text	Optional field used in testing to uniquely identify a set of messages	n/a
Sender Unique Reference	Unique identifier for the event/message provided by the message Sender. ~~See note below that provides guidance on setting this field~~	Check S~~s~~ender U~~u~~nique R~~r~~eference obeys the format specified, which is: ~~below~~ • Interface, i.e., IF-0XX, • Participant Identification (Id), i.e., 0123456789. • Role Id, i.e., SUP. • Date, i.e., Year-Month-Day (2026-03-24). • Sequence/reference that must be alphanumeric, i.e., 12345687a1234567a • Hence, i.e., S-IF-0XX-0123456789-SUP-20222313-12345687a1234567a.
Sent Timestamp	Date/time (UTC) message was sent	Check valid/date, i.e. complies with RFC-3339
Sender ID	The DIP User who sent the message.	Check valid DIP User

Formatted: Font: Bold, Complex Script Font: Not Bold

Formatted: Font: Bold, Complex Script Font: Not Bold

Formatted: Font: (Default) Times New Roman, (Asian) Times New Roman, 12 pt, Complex Script Font: Times New Roman, 12 pt

Formatted: Font: (Default) Times New Roman, (Asian) Times New Roman, 12 pt, Complex Script Font: Times New Roman, 12 pt

Formatted: Left

Formatted: Font: (Default) Times New Roman, (Asian) Times New Roman, 12 pt, Complex Script Font: Times New Roman, 12 pt, French (France)

Formatted: French (France)

Formatted: French (France)

Formatted: Font: (Default) Times New Roman, (Asian) Times New Roman, 12 pt, Complex Script Font: Times New Roman, 12 pt, French (France)

Formatted: Font: (Default) Times New Roman, (Asian) Times New Roman, 12 pt, Complex Script Font: Times New Roman, 12 pt

Formatted: Font: (Default) Times New Roman, (Asian) Times New Roman, 12 pt, Complex Script Font: Times New Roman, 12 pt

Formatted: Font: (Default) Times New Roman, (Asian) Times New Roman, 12 pt, Complex Script Font: Times New Roman, 12 pt

Formatted: Font: (Default) Times New Roman, (Asian) Times New Roman, 12 pt, Complex Script Font: Times New Roman, 12 pt

Formatted: Font: (Default) Times New Roman, (Asian) Times New Roman, 12 pt, Complex Script Font: Times New Roman, 12 pt

Formatted: Elexon Body, Space After: 0 pt, Bulleted + Level: 1 + Aligned at: 0.63 cm + Indent at: 1.27 cm

Sender DIP Role	The capacity in which the message was sent	Check the sender ID/ sender DIP Role is entitled to send a message
DCP ID	If message sent by a nominated third-party	Check DCP ID/ Sender DIP Role is authorised to send messages on behalf of sender ID
Sender Correlation Id	Populated when the IF is used in the response to the primary message in the workflow.	Check obeys format

Sender (S1) block descriptions – mandatory

9.1.6 The Sender blocks are split between S0 & S1. Both are fixed formats, however, the S0 block has different enumerated values for each of the respective fields, whilst the S1 blocks does not have this separation. Consequentially, each interface definitions has a bespoke S0 block.

9.1.7 The response block is only used when the message is a "response" to an initial message, e.g. IF-006 is a response to an IF-005 message and the Response Correlation Id is that sent in the original message.

Field	Description	DIP Validation
Response Code	Response Code	n/a
Response Message	Response Message	n/a

Response (R0) block - optional

9.1.8 The address block is only populated on an interface-by-interface basis, i.e. not all Interfaces require primary addressing.

Field	Description	DIP Validation
Primary recipient DIP ID(s)	Used to route messages to targeted parties (Primary Addressing)	Check DIP User has the corresponding role (permission) to receive message

Address (A0) block - optional

9.1.9 The Message Body MPAN (M0) is an optional block.

Field	Description	DIP Validation
MPAN Core	Used to route messages to specific parties (Secondary Addressing)	n/a

Formatted: Complex Script Font: Not Bold

Formatted: Font: Bold, Complex Script Font: Not Bold

Formatted: Font: Bold, Complex Script Font: Not Bold

Distributor Id	The market-wide unique reference for the distributor who is responsible for the distribution network that a metering point is connected to.	n/a
GSP Group Id	Identifies the distinct grid supply point group (physical region of the country) where the metering point is located.	Check valid GSP Id (enumerated values)

Message Body (M0) block - optional

9.1.10 The Message Body MPAN (M1) is an optional block but is only used in IF-019 and IF-020 as the equivalent of the M0 block.

Field	Description	DIP Validation
Principal MPAN	Used to route messages to specific parties (Primary Addressing)	n/a
Distributor Id	The market-wide unique reference for the distributor who is responsible for the distribution network that a metering point is connected to.	n/a
GSP Group Id	Identifies the distinct grid supply point group (physical region of the country) where the metering point is located.	Check valid GSP Id (enumerated values)

Message Body (M1) block – optional (IF-019 and IF-020 only)

9.1.11 The custom block within a message body is dependent on which interface is being submitted. These definitions are not repeated in this document, instead they can be found in the DIP Open API Definitions (DIP Swagger) and EMDS.

9.2 Send message DIP Processing (Level 1 Validation)

9.2.1 The purpose of the initial API checks is to provide basic syntax checking of the message header to ascertain whether the message is syntactically correct and can be processed further. A ‘handshake’ return message provides a reference between the message sender’s unique reference and the DIP’s Transaction ID.

9.2.2 On first error the DIP will abort the transaction and return the corresponding status code and error messages. The DIP will perform the following for each separate message received.

a) Validation of API key and establish Sender;

b) Ensure Interface Id is correct;

c) Schema validation checks are undertaken via the definitions described in the DIP Open API Definitions (DIP Swagger). If schema validation fails on a single message within a transaction then all messages within the transaction are rejected with a '400' code and a MSG1001 message in the response. Some of the checks are undertaken here include:

i) Check message structure, i.e. mandatory blocks are present for the corresponding interface and event code;

ii) Check the message validation checks for the common blocks:

1. Check the environment is set correctly;

2. Check Interface Id/Event Code pair is an allowable combination;

3. Check the message Schema version aligns (API enumeration defines allowable values);

d) The following are undertaken outside the schema validation checks:

i) Check the Sender Id ("logical" sender) is entitled to send message for the Event Channel for the given role;

ii) Check the DCP is entitled to send message on behalf of the "logical" sender for the given role/channel;

iii) Check the Sender Unique Reference is formatted correctly and unique;

e) Generate a Transaction ID (see below);

f) Generate a Transaction timestamp, i.e. record the current date/time (UTC);

g) If the message channel is configured to generate a Workflow Correlation Id;

h) Generate service desk/audit links;

i) Write the Transaction ID, Transaction timestamp, Workflow correlation Id, status code, status message and audit/service desk links to the response body;

j) Generate return code based on the results of the above checks for the all the messages received in the single transaction.

9.2.3 The DIP will enforce capitalisation checks on message ingress for path names and message structure. Data items within messages are also case sensitive and the DIP will not convert between cases: data will pass through unaltered. This follows OpenAPI standards.

9.2.4 Enumeration checks undertaken by the DIP are case sensitive, e.g., for the Common Block/ S1/ environment the term 'PROD' is accepted, whilst 'prod' or 'Prod' are rejected. A significant amount of legacy data items where case is important pass through the DIP and hence the reason why these checks are applied.

9.2.5 The list of bespoke response codes are described in the DIP Open API Definitions (DIP Swagger).

9.3 Receive Message Webhooks

9.3.1 A DIP User's Webhook can either be configured within the DIP Portal, or via the API call to register the Webhook. Either can be used to register the Webhook that will be used to relay publication events/messages from the DIP to the corresponding message recipient. The latest version of the full Open API definition can be found in the DIP Swagger.

9.3.2 The Webhook for each message channel (publication) needs to be registered separately. The receiving DIP User has the option of defining multiple call backs (one for each publication) or a single call back for the receipt of all messages (which would need to be registered on each channel). It is the receiving DIP User's responsibility to ensure that all publications they are due to receive are covered by all the call backs registered.

9.3.3 The messages within a single call back can have mixed MPANs.

9.3.4 The return code and response body of the call back API will provide the receiving DIP User with an auditable trace on the success of sending the messages by the DIP. If the Webhook request has originated from a DCP, the DIP will check that the DCP is authorised to receive messages on behalf of the logical recipient.

9.3.5 The API has the following path parameters:

a) The message channel ID that the corresponding Webhook needs to service, eg. IF-001

b) The DIP ID of the message recipient, eg. 1001012345

9.3.6 The common block described above in the send Events API will be augmented with the D0 block which is added to the message by the DIP.

Field	Description
Transaction ID	Unique DIP Transaction ID
Transaction timestamp	DIP Receipt timestamp
Publication ID	Publication ID

Field	Description
DIP Correlation ID	Provides unique identification for a specific workflow instance. Only set for those channels were a correlation ID is required
Replay Indicator	If set, then message has arrived through a replay request rather than a standard message
Service Ticket URL	URL to create/view appropriate service ticket

DIP (D0) block – mandatory

9.4 Receive Message Call back Request structure

9.4.1 The call back request will have the following properties:

a) the URL for the call back should not specify a port number as all communications are through standard HTTP ports, i.e. 443 for HTTPS;

b) the maximum number of messages the recipient can receive in a single transaction; and

c) the maximum payload size - the payload size sent in the call back (from the DIP to the DIP User) will not exceed this limit. This limit is ignored (by the DIP) if a single message cannot be transmitted.

9.5 Recipient responsibilities

9.5.1 On receipt of a batch of messages the recipient will need to validate those messages. Validation is split into two phases:

a) Synchronous (Level 3); and

b) Asynchronous (Level 4).

9.5.2 Where validation occurs, either at L3 or L4, is down to the discretion of the recipient. It is recognised that different recipients will design their interfacing systems differently, hence, the responses can either be synchronous (L3) or asynchronous (L4).

9.5.3 The expectation is that the recipient will undertake some initial validation checks and employ schema validation checks on their own gateway. At this point the transaction has not been de-serialised and individual messages not recognised within the payload; hence if an error is encountered then the whole transaction is rejected with a 400 response code. At this juncture no Transaction ID would be quoted back in the response.

9.5.4 Once the initial checks have been undertaken and the individual messages identified, then the processing and reporting can be undertaken at message level. These subsequent checks are more likely to be L4 checks.

9.5.5 If the recipient is receiving the event/message through a replay request then the subsequent downstream processing and subsequent reply needs to be cognisant that the event/message may have already been processed.

9.6 Replay events

9.6.1 Where DIP Users have lost any messages (following successful delivery previously), or to support DIP Users to recover from a system failure (alongside several other scenarios), the 'Replay Events' functionality enables DIP Users to retrieve messages from the DIP archive.

9.6.2 Two different APIs are available:

a) Replay API; and

b) Re-queue API.

9.6.3 The replay API will return the messages directly from the API call. The re-queue API will place the messages on the queue to be pulled by the corresponding Webhook.

9.6.4 The furthest a query can retrieve data is dependent upon the retention time set with each message channel (the default retention is 2 years).

9.6.5 The DIP User can make a single request to return an array of events (50,000). The message will contain data i.e., timestamps as per the original submission.

9.6.6 The original recipient DIP ID needs to be specified as the DIP will need to ascertain which messages the DIP User is able to replay. This 'Replay Events' functionality will also be available via the DIP Portal to manually request a replay.

9.6.7 The Replay Events query parameters are:

a) Message Channel

b) Message Recipient DIP ID

9.6.8 Message request object contains:

a) Event Code (optional)

b) MPAN

c) Transaction IDs – array of message Transaction IDs to replay, or

d) Date/time from – transaction time from which the first message needs to be replayed.

e) Date/time to (optional) – transaction time message to be replayed

f) Version – message version (optional)

9.6.9 The replayed message will have the same structure as the message received via the Webhook, so in addition to the message body and Sender message blocks (P0, M0) (sender) the message will also have the added the amended D0 block.

9.6.10 There will be two methods by which the Event Replay facility can be initiated:

a) Through the UI interface on the DIP

b) Through an API Call

9.6.11 As well as being delivered via a separate API, the event replay will deliver the message under a new transaction wrapper with a Transaction ID and Correlation ID (with a reply prefix) so that the event/message are identified as a replayed message, and hence downstream processes are not triggered. The initial message transaction wrapper will also be sent. Only messages the DIP User is entitled to view will be sent.

9.6.12 DIP Users need to be aware where messages are replayed from archive that message payloads are replayed in the schema version that they were originally submitted in, i.e. the messages are not 'uplifted' to the current live format when replayed. DIP Users requiring full replay facilities will need message gateways and software that have multi-version support for message ingress during replay.

9.7 Message idempotency

9.7.1 All DIP Users' systems are expected to be idempotent in terms of handling duplicate messages. Duplicate messages will be identified using the Sender Unique Reference and the Transaction ID.

9.7.2 For clarity, if a DIP User's systems make an identical API call, it is expected that the effect in the DIP User's system will be the same as if a single call is made.

9.8 Message Throughput

9.8.1 DIP Users' systems shall have the capacity to process messages according to the following volumes:

a) Average daily volume of 66,000 messages/day;

b) Peak daily volume of 300,000 messages/day;

c) Average hourly volume of 2,750 messages/hour;

d) Peak hourly volume of 35,000 messages/hour; and

e) Annual volume of 24,000,000 messages/year.

9.8.2 In the case of IF-035, IF-036 or IF-037, the following shall apply (where PC is the Portfolio Coefficient):

a) Peak daily volume of 1,143,000*PC messages/day; and

b) Peak hourly volume of 1,143,000*PC messages/hour.

9.8.3 For the purposes of this paragraph, PC shall be the percentage of the total number of MPANs associated with that DIP User.

9.8.4 The above requirements in 9.8.1 and 9.8.2 do not apply to the handling of IF-021. Nor shall they apply to PUB-013, PUB-014, PUB-021, REP002, REP002A, REP900 or REP901 where used in Distribution Use of System (DUoS) billing by a Distributor.

9.9 Message return codes and response body

9.9.1 Each connection will result in a HTTP return code that will indicate the success or otherwise of the complete transaction. The detailed list of expected behaviour and response codes, including any expected retry behaviour is shown in the tables below (be it automatic or manually initiated). The HTTP codes and retry polices attempt to follow, what is considered, standard industry practice. All HTTP response codes are also included in the DIP Open API Definitions (DIP Swagger).

Code	Messages	Automated Retry	Reason	Action	Retry Behaviour
2xx	**Successful**				
201	Messages Created		Messages successfully received by DIP and passed L1 validation (All messages have a MSG0000 in the corresponding response block)		
207	Some Messages Created	No	Some or no messages are created, i.e. some messages have a MSG0000 message in the corresponding response block, others will have a response code in the range MSG1000 to 1012.	Reform failed messages and resend in new transaction. If problem persists, contact DIP 1st line support	

Code	Messages	Automated Retry	Reason	Action	Retry Behaviour
2xx	Other 200 messages		The DIP will only send 201 or 207 in the successful receipt of messages		
4xx	**Client Errors**				
400	Bad Request	No	Malformed messages or HTTP Header content.	Reform message to align with DIP Open API Definitions (DIP Swagger). If submitting messages in-batch submit in smaller batches to establish problem message. If problem persists contact DIP 1st line support	
401	Unauthorised Error	No	Issues related to Message Signing Certificates, Header problems or Account Issue (this includes any errors related to the X-API Key).	Ensure certificate validity; check cert has not expired. If problem persists, contact DIP 1st line support	After rectifying cert issue reattempt sending messages

Code	Messages	Automated Retry	Reason	Action	Retry Behaviour
403	Forbidden	No	Issues related to TLS Certificates (including authentication failures), alongside other general 403 related issues i.e., could be IP blocking	Contact DIP 1st line support	Retry after new security measures (cert/account) in place
404	Not Found	No	Resource not found	If problem persists, contact DIP 1st line support	Resource could be temporarily unavailable, hence assume a periodic retry
405	Method Not Allowed	No	Requested method unsupported	Contact DIP 1st line support	
406	Not Acceptable	No	Requested method unsupported	Contact DIP 1st line support	
408	Request Timeout	Yes	System timeout waiting for resource	If problem persists, contact DIP 1st line support	Adopt an automated back-off and retry algorithm for sending messages.
413	Payload Too Large	No	Request is too large for firewall/gateway	Reduce payload size where possible, if not possible contact support.	Retry after dialogue with 1st line support

Code	Messages	Automated Retry	Reason	Action	Retry Behaviour
429	Too Many Requests	Yes	Rate limiting in force.	Wait, if symptom persists after cool-off period then contact support	Adopt an automated back-off and retry algorithm for sending messages.
4xx	Other 400 messages		The DIP will send any other 400 messages	Contact DIP 1st line support	
5xx	**Server Errors**				
500	Internal Server Error	Yes	The DIP is aware that it has encountered an error or is otherwise incapable of performing the request		
502	Bad Gateway	Yes	Retry, but if problem persists contact DIP 1st line support		
503	Service Unavailable	Yes	Adopt an automated back-off and retry algorithm for sending messages.		
504	Gateway Timeout	Yes			
505	HTTP Version Not Supported	No	Contact support	Contact DIP 1st line support	
5xx	Other 500 messages		The DIP will not send any other 500 messages		

9.9.2 The table above introduces the ingress of messages to the DIP. The automated retry and retry behaviour columns presents the suggested behaviour a DIP User's system is expected to follow, but not mandated as they will have their own mitigation policies already defined.

9.9.3 The response uses the common Standard Response body:

Field	Description
Transaction ID	Unique DIP Transaction ID
Sent timestamp	DIP Receipt timestamp
Sender Unique Reference	The original Sender Unique Reference
Sender ID	DIP identified
Correlation ID	Correlation ID relayed back (optional)
Recipient ID	Recipient of the message
DCP ID	DCP (optional)
Message	Information on the message status
Help	Extra help information
Service Ticket URL	URL to SNOW ticket related to the message

9.9.4 The response body of the HTTP call will deliver the Sender a Transaction ID and optionally a correlation ID against the Sender's Unique Reference for each message.

9.10 DIP API Actions

9.10.1 On sending a set of events/messages the DIP will undertake the following:

a) **Message Construction** – The DIP will augment the original message with the added information detailed below in section 4.6.4.

b) **API Call** – The DIP will call the API declared in the call back; only messages on the same messages channels will be sent in the same API call. The payload size will not exceed the size specified in the call back request; the DIP will collate all outstanding messages on the channel and if there is a message that exceeds the limit requested, then that message will be sent in the next transaction.

c) **API Response** – The HTTP return code will provide the return for the whole transaction and needs to be considered with the response body as it will provide the details for the acceptance or otherwise for each message (see below for response code).

d) **Back Off and Retry** – The DIP has a retry with exponential backoff approach (up to a maximum wait time) and additional “circuit breakers” to ensure efficient handling of broken connections to DIP Users.

9.11 API Version Control

9.11.1 Versioning is a crucial part of API design. It gives developers the ability to improve their API without breaking the client's applications when new updates are rolled out. Due to the complexity of the data exchange between DIP Users and the DIP being mostly in the format of the messages where minor-versioning is required. API versioning is noted through custom headers. However, entity versioning, where there are major changes to the API, versioning is through URI path.

9.11.2 As a principle, all API definitions need to be backwardly compatible. In the situation where this does not hold, then a significant transition period must be provided for DIP Users to move from old to new version.

9.11.3 API version control must adopt semantic versioning as the common standard, i.e. Given a version number MAJOR.MINOR.PATCH, increment:

a) **MAJOR** version – when you make incompatible API changes;

b) **MINOR** version – when you add functionality in a backwards compatible manner; and

c) **PATCH** version – when you make backwards compatible bug fixes.

9.11.4 Additional labels for pre-release and build metadata are available as extensions to the MAJOR. MINOR. PATCH format.

9.12 Message channel versioning

9.12.1 Each separate message channel has its own version number and is defined by the schema version item in the payload/ Common Block/ S0, for example:

9.12.2 Messages will only be submitted in a single version of the message. If a message channel requires an upgrade then this will happen via a controlled outage where all message ingress/egress is quiesced, the channel is then modified and then the API will only accept messages in the new format. Subsequently, if messages are submitted that align to the previous schema version then they will be rejected with a schema validation error.

9.12.3 Like API Versioning, message channel versioning also follows the MAJOR. MINOR. PATCH format:

a) **MAJOR** version – a new major version of the API will be created when a new piece of functionality is required, or a change is required that will completely break the existing API. Under this scenario all interfaces will only have forwards looking data definitions and not support previous versions of data.

b) **MINOR** version – Where a new message channel is created, then a new minor version of the API will be created. This will not require a new API to be created, however, all the corresponding interface and publications data definitions will be need to be updated to accommodate the new channel.

c) **PATCH** version – Where the data definition of a single interface changes, then that will necessitate a patch version of the current version of the interface. Under 'normal' circumstances the incoming interfaces will only support the new version of the data definitions. If there is a requirement to support multiple version of the same interfaces (incoming) then a new minor version is required. In this scenario the publication of the messages will also support just the new current version. All data must have already traversed through the DIP.

9.12.4 Where a message channel has been versioned to accommodate a change to the message structure, the message replay supports all versions of the message. If a DIP User requests a message replay, it is their responsibility to ensure that all pertinent versions of the message will be accepted.

9.13 Multi-version support – API and message channels

9.13.1 All new major, minor and patch versions, i.e. x+1, or Y+1, or Z+1, will be available prior to release for industry testing for major releases in a non-production environment.

9.13.2 Backward API compatibility is maintained for a short period after the roll out of a new major version. For minor releases multi-version support is dependent on the type of underlying change.

9.14 API Keys

9.14.1 API Keys are included within the DIP design to provide the capability to enforce rate limiting, quota management and analytics. This functionality is performed by the DIP. 9.14.2

9.14.3 The DIP allows DIP Users to obtain API Keys via the DIP ID management screen.

9.14.4 Two keys are provided by default (primary/secondary) for each DIP ID to allow API Key(s) to be rotated independently on demand.

9.14.5 If an invalid API Key is used a 401 (unauthorised) error will be returned. If there is a mismatch between the API key and Sender ID, a 400 (bad request) error will be returned, with DIP Error code.

9.15 API Key rotation

9.15.1 There is no automatic key rotation and as such, DIP Users should rotate their API Keys at least once a year.

9.15.2 Keys should also be rotated immediately following a suspected compromise of an API Key, and such event shall be reported to the DIP Manager.

9.15.3 New API keys can be generated in the DIP portal using the 'Regenerate' option. Primary and secondary keys can be regenerated independently to ensure service continuity.

9.16 API Key(s) and DCPs

9.16.1 Where a DIP User has chosen to use a DCP, the DIP User will be responsible for providing the API Key(s) to the DCP in a secure manner. See the NCSC guidance for protecting Private Keys.

10 Signatures

10.1 Digital signatures

10.1.1 Authentication within the DIP is based on two form factors:

a) the first is covered by the establishment of a secure communications channel provided by mTLS, as described above;

b) the second is the authentication of individual Messages which is achieved through the application of a digital signature.

10.1.2 Applying a Digital Signature to a Message also adds a layer of data integrity assurance. Digital Signatures are applied to hashes of JSON Messages (sometimes called Message digests) and are used to detect unauthorised modifications to data, as well as to authenticate the identity of the signatory.

10.1.3 In addition, the recipient of signed data can use the digital signature as evidence in demonstrating to a third-party that the signature was, in fact, generated by the claimed signatory.

10.2 Digital signature formats

10.2.1 A DIP Message signature represents digitally signed content using JSON data structures and Base64 encoding. A Message contains the following values:

RFC7515 Logical Values

Header Value	Format	Details
X-DIP-Signature	BASE64 encoded. UTF-8 Character set. Padding RSA PKCS-1	Ensure Base64 encoding of the Message Signature
X-DIP-Signature-Date	Plain Text	The Message signature date
X-DIP-Signature-Certificate	BASE64 encoded. UTF-8 Character set.	Ensure Base64 encoding of Public Key of the sender (Private Key used to generate signature)
X-DIP-Content-Hash	BASE64 encoded. UTF-8 Character set.	SHA256 Hash of JSON Message Body. Base64 encoding of the SHA256 Hash (using UTF-8 Character set)

10.3 Message signing

10.3.1 To create a Message, the following steps should be followed:

a) generate the content of the Message, this is the Message body;

b) create a SHA256 hash of the Message body. BASE64 encode the output from the SHA256 hash (this is the SHA256 content hash);

c) generate an ISO8601 timestamp in the format: YYYY-MMDDTHH:MM:SS.XXX:Z. This is the Signature Date;

d) concatenate the Signature String in the following format:

i) Verb; Destination; Signature Date; SHA256 Content Hash;

ii) Verb = Uppercase only, Destination = Lowercase only;

e) create a SHA256 hash of the Signature String (this is the SHA256 Signature string);

f) sign the SHA256 signature string using the Private Key of the signing certificate. This is the signed signature. Utilise RSA PKCS1 signature padding;

g) create BASE64 encoded string of the signed signature;

h) add to the request the header: X-DIP-Signature with the value of the BASE64 encoded signed signature;

i) add to the request the header: X-DIP-Signature-Date the value of the Signature date;

j) add to the request the header: X-DIP-Signature-Certificate with the BASE64 encoded signing certificate (this will contain only the Public Key);

k) add to the request the header: X-DIP-Content-Hash with the SHA256 Content Hash.

10.4 Verifying signatures

10.4.1 When validating a Message signature, the following steps MUST be performed. If any of the listed steps fails, then the signed content MUST be rejected:

a) verify the Digital Certificate received (X-DIP-Signature-Certificate) is trusted and valid;

b) take the value of the header: X-DIP-Signature and decode using BASE64. This is the BASE64 Decoded Signature;

c) build a comparison string:

i) Verb - from HTTP Request received;

ii) Destination - from HTTP Request (the url of the endpoint the Message has been received on);

iii) Signature Date - ISO8601 timestamp in the format: YYYY-MM-DDTHH:MM:SS.XXX:Z (retrieved from the X-DIP-Signature-Date);

iv) Content Hash - Create a SHA 256 hash of the JSON Message body received. Base 64 encode the output from the SHA 256 hash. For any request with an empty body (GET, DELETE requests) assume an empty JSON body – {}

d) create a SHA256 hash of the comparison string. This is the SHA256 hashed comparison string;

e) verify the SHA256 hashed comparison string with the BASE64 Decoded Signature, using the Public Key of the Digital Certificate received (X-DIP-Signature-Certificate).

10.5 Signature key generation and Certificate Signing Requests

10.5.1 The DCA issues Digital Certificates for Digital Signatures with the following parameters: RSA-4096

10.5.2 Public/Private key pairs must be created following successful completion of the registration process. Each DIP User will be responsible for the generation of mTLS keys and CSR used by their Message service interface.

10.5.3 A DIP User will need to create an associated PKCS#10 CSR and upload the CSR to the DIP, where the CSR will be used to generate a signed Digital Certificate.

10.5.4 The DIP User should use the distribution mechanisms for CSRs and Digital Certificates outlined elsewhere in this Annex.

10.5.5 Tools for Public/Private key pair and CSR generation will depend on the local environment.

10.5.6 DIP Users must ensure that they adhere to the Certificate Profiles in the tables below when creating CSRs so that the Digital Signature meets the required shape and form. The Digital Certificates must be built or configured as indicated in the profile to ensure they work correctly, and CSRs are not rejected.

DIP Certificate Profiles – non-Production

Name of field	DIP-NON-PRODUCTION_TLS	Critical	DIP-NON-PRODUCTION_SIG	Critical
Version	1		1	
Serial Number	yes		yes	
Signature Algorithm	Sha256RSA		Sha256RSA	
Issue rName				
Common Name (CN)	GlobalSign		GlobalSign	
Org Unit (OU)				
Organisation (O)	MHHS-DIP		MHHS-DIP	
City				
State				
Country (C)	GB		GB	
Subject Name	Supplied in CSR		Supplied in CSR	
Email address (E)				

Name of field	DIP-NON-PRODUCTION_TLS	Critical	DIP-NON-PRODUCTION_SIG	Critical
Common Name	energydip-nonprod <MPID>		energydip-nonprod.<MPID>	
Org Unit				
Org Unit	Non-Production		Non-Production	
Org	<MPO/DCP Organisation name>		<MPO/DCP Organisation name>	
City				
State				
Country	GB		GB	
Validity Period				
Months	12		12	
Public Key				
Algorithm	RSA		RSA	
Key Size	4096		4096	
Basic Constraints				
CA				
Path Len Constraint				
Key Identifiers				
Authority Key Identifier	yes		yes	
Subject KeyIdentifier	yes		yes	
Key Usage		**x**		**x**
Digital Signature	yes		yes	
Non Repudiation			yes	
Key Encipherment	yes			
Data Encipherment				
Key Agreement	yes			
Key Cert Sign				
CRL Sign				
Encipher Only				
Decipher Only				
Digital Certificate Signing				
Off-line CRL Signing				
Extended Key Usage				
Server Authentication	yes			
Client Authentication	yes			
Secure Email				
Encrypting File System (1.3.6.1.4.1.311.10.3.4)				
Code Signing				
Smart Card Logon				
OCSP Signing				
IPSec Usage				
Document Signing (1.3.6.1.4.1.311.10.3.12)				
OCSP NoCheck				

Name of field	DIP-NON-PRODUCTION_TLS	Critical	DIP-NON-PRODUCTION_SIG	Critical
Digital Certificate Policy				
Policy OID				
User Notice				
CPS Pointer				
CRLDistributionPoints				
CDP name (1)				
CDP Full Name				
CDP name (2)				
CDP Full Name				
Authority Information Access				
Access Method				
Alternative Name				
Access Method				
Alternative Name				
Subject Alternative Name	Supplied in CSR		Supplied in CSR	
DNS Name				
DNS Name				
IP Address				
UPN OtherName (1.3.6.1.4.1.311.20.2.3)				
RFC 822 email address				
Enrolment requirements				
Autoenrol				
Enrol permissions				
Subject name				
Digital Certificate manager approval rquired				
Private key exportable				

DIP Certificate Profiles –Production

Name of field	DIP-PRODUCTION-TLS	Critical	DIP-PRODUCTION-SIG	Critical
Version	1		1	
Serial Number	yes		yes	
Signature Algorithm	Sha256RSA		Sha256RSA	
Issuer Name				
Common Name (CN)	GlobalSign		GlobalSign	
Org Unit (OU)				
Organisation (O)	MHHS-DIP		MHHS-DIP	
City				
State				
Country (C)	GB		GB	
Subject Name	Supplied in CSR		Supplied in CSR	
Email address (E)				

Name of field	DIP-PRODUCTION-TLS	Critical	DIP-PRODUCTION-SIG	Critical
Common Name	energydip-prod.<MPID>		energydip-prod.<MPID>	
Org Unit				
Org Unit	Production		Production	
Org	<MPO/DCP Organisation name>		<MPO/DCP Organisation name>	
City				
State				
Country	GB		GB	
Validity Period				
Months	12		12	
Public Key				
Algorithm	RSA		RSA	
Key Size	4096		4096	
Basic Constraints				
CA				
Path Len Constraint				
Key Identifiers				
Authority Key Identifier	yes		yes	
Subject KeyIdentifier	yes		yes	
Key Usage		**x**		**x**
Digital Signature	yes		yes	
Non Repudiation			yes	
Key Encipherment	yes			
Data Encipherment				
Key Agreement	yes			
Key Cert Sign				
CRL Sign				
Encipher Only				
Decipher Only				
Digital Certificate Signing				
Off-line CRL Signing				
Extended Key Usage				
Server Authentication	yes			
Client Authentication	yes			
Secure Email				
Encrypting File System (1.3.6.1.4.1.311.10.3.4)				
Code Signing				
Smart Card Logon				
OCSP Signing				
IPSec Usage				
Document Signing (1.3.6.1.4.1.311.10.3.12)				
OCSP NoCheck				
Digital Certificate Policy				

Name of field	DIP-PRODUCTION-TLS	Critical	DIP-PRODUCTION-SIG	Critical
Policy OID				
User Notice				
CPS Pointer				
CRLDistributionPoints				
CDP name (1)				
CDP Full Name				
CDP name (2)				
CDP Full Name				
Authority Information Access				
Access Method				
Alternative Name				
Access Method				
Alternative Name				
Subject Alternative Name	Supplied in CSR		Supplied in CSR	
DNS Name				
DNS Name				
P Address				
UPN OtherName (1.3.6.1.4.1.311.20.2.3)				
RFC 822 email address				
Enrolment requirements				
Autoenrol				
Enrol permissions				
Subject name				
Digital Certificate manager approval required				
Private key exportable				

11 DIP capabilities

11.1 Number of channels and DIP Users

11.1.1 The DIP has the ability to accommodate (with the potential for annual increase):

a) Fifty message channels (five to ten channel annual increase);

b) Two hundred senders (twenty senders annual increase);

c) Two hundred recipients (twenty recipient annual increase).

11.2 Transactional volumes

Formatted: French (France)

11.2.1 Transaction volume capability:

Data Entity	Interface	Target	Annual Volumes		Annual Data Size	
			Typical	Maximum	Typical	Maximum
Registration	Appointment	Metering Service				
		Data Service				
	De-appointment	Metering Service				
		Data Service				
	Accept/ Reject appointment	Registration Service				
	Updates	Metering Service				
		Data Service				
		BSC systems				
Consumption	Daily SP level data	BSC systems				

Data Entity	Interface	Target	Annual Volumes		Annual Data Size	
			Typical	Maximum	Typical	Maximum
	Load Shape	Data Service				
	All Entities	n/a				

11.3 Message Latency

11.3.1 The DIP provides near real-time message delivery, the requirement is for messages internally processed and available for recipient egress (measured from the time of initial HTTP response on the message ingress) within the following parameters:

a) up to average hourly volume, 90th percentile time of 3s or less;

b) up to average hourly volume, 99th percentile response time of 10s or less;

c) from average hourly to peak hourly volume, 90th percentile response time of 6s or less;

d) from average hourly to peak hourly volume, 99th percentile response time of 30s or less.

11.3.2 In order to future-proof the DIP, it is recognised that the platform will need to be able to support near-real-time, i.e. <1 second, message flows.

11.3.3 All DIP User systems (excluding for the handling of Excludes IF-021) shall provide an initial synchronous response to a message within the following timeframes:

a) up to average hourly volume, mean response time of 2s or less;

b) up to average hourly volume, 90th percentile response time of 4s or less;

c) from average hourly to peak hourly volume, mean response time of 5s or less;

d) from average hourly to peak hourly volume, 90th percentile response time of 8s or less.

11.4 System availability

11.4.1 The DIP is expected to have the following availability requirements:

e) percentage of uptime – 99.95% (unplanned);

f) Mean time to recovery (MTTR) – 60 minutes;

g) Mean time between failures (MTBR) – n/a;

h) Recovery time objective (RTO) – 60 minutes;

i) Recovery point objective (RPO) – 0.

11.4.2 All DIP Users' systems (except those stated in sections 11.4.3 and 11.4.4 below) shall provide an asynchronous response (Level 4 validation) to a message within the following timeframes:

a) up to average hourly volume, mean response time of 6s or less;

b) up to average hourly volume, 90th percentile response time of 12s or less;

c) from average hourly to peak hourly volume, mean response time of 10s or less;

d) from average hourly to peak hourly volume, 90th percentile response time of 16s or less.

11.4.3 The BSC's Data Acquisition Hub (DAH) shall have an asynchronous response time of 10 minutes or less.

11.4.4 All Registration Services (REGS), Unmetered Supplies Operators (UMSO) and Distributors (DNOs/iDNOs) shall have an asynchronous response time of 60 minutes or less within normal operating hours (05:00 to 18:00 and 21:00 23:00 Working Days only).

11.4.5 The above requirements in 11.4.2, 11.4.3 and 11.4.4 do not apply to the handling of IF-021. Nor shall they apply to PUB-013, PUB-014, PUB-021, REP002, REP002A, REP900 or REP901 where used in DUoS billing by a Distributor.

Amendment Record

Version	Date	Description of Change	Approval Reference
1.0	01/10/2024	01 October 2024 Non-Standard Release	P353/08
2.0	27/02/2025	27 February 2025	N/A
3.0	26/06/2025	26 June 2025 Standard Release	N/A
4.0	05/08/2025	05 August 2025 Non-Standard Release	N/A
5.0	22/10/2025	22 October 2025 Non-Standard Release	N/A