TOTSCo Hub API specifications v0.5 REVISED

A guide for developers

19/06/2023

TOTSCO | TECH mahindra

Contents

Figures

1 Introduction

This TOTSCo Hub API Specifications document complements and supports the One Touch Switch (OTS) Message Specification and the OTS Industry Process document, and the equivalent documents for the GPLB (gaining provider led business switching) process. For definitions of these processes:

- TOTSCo are the custodians of OTS documentation: www.totsco.org.uk
- FCS are the custodians of GPLB documentation: https://www.fcs.org.uk/gaining-provider-led-business-switching/.

This document contains the definition of the message envelope structure as well as the letterbox API specification

The intended audience of this document is:

- Representatives of commuincations providers (CPs) who are responsible for the technical implementation of the communication between that CP and the TOTSCo Hub.
- Representatives of Tech Mahindra, the vendor chosen by TOTSCo to implement the Hub.

1.1 Communications providers and third party integrators

The TOTSCo Hub will initially provide services for retail CPs to exchange messages in support of the OTS and GPLB processes. In the future the Hub may provide services in support of other processes requiring message exchange between CPs who may not be the retailers. So this document used the generic term of CP.

Not that retail CPs may choose to engage an agent, such as a third party integrator (TPI). Any reference in this document to a requirement against a CP would apply to a TPI providing services to an RCP.

1.2 Change log

Version **Date** **Changed By**	**Reason for change**
V0.1 First draft 27/03/2023 Dave Stubbs	API Specification for the TOTSCo Hub. First release for Hub vendor.
V0.2 Updated draft 6/04/2023 Dave Stubbs	Added specification of the directory API. Added an appendix to describe version control processes for messaging over the TOTSCo Hub.
V0.3 Updated draft 27/04/2023 Dave Stubbs	Corrected the JSON structure of the envelope for the audit data elements. Added an error response JSON structure for HTTP 4xx errors to allow effective communication of synchronous error scenarios Changed the swagger location of the API specification.
V0.4 Updated draft 09/06/2023 Niall Gillespie	This version will be circulated for review to representatives of retail CPs who have registered with TOTSCo for updates.
V0.5 Updated draft 12/06/2023	Draft document containing Letterbox API specification. Directory and Security Information to be added once confirmed
V0.5 Revised draft 19/06/2023	Draft document containing Letterbox API specification and oAuth2 Security Implementation.

1.3 Contributing authors

Author	Organisation
Dave Stubbs	Virgin Media
Niall Gillespie	BT
Hub design team	Tech Mahindra Ltd.

1.4 Stakeholders and document approvals

Stakeholder Name and Title	Role	Reviewer/Approver/ Author/Contributor	Signature/Electronic Approval	Date
Richard Steele	CTO, TOTSCo	Approver		
Tom Merritt	Business Analyst, TOTSCo	Approver		
David Norbury	Head of Delivery, TOTSCo	Approver		
Jason Bird	CSO , TOTSCo	Approver		
Alex Simmons	Technical lead, TOTSCo	Approver		
Niraj Suvarna	Enterprise Architect	Reviewer		
Rahul Mehta	Solution Architect	Reviewer		
Nimesh Soni	Solution Architect	Reviewer		
Premanand Rao	Infrastructure design	Reviewer		
Shailesh Pingle	Delivery Manager	Reviewer		
Vishal Dumbre	Security Design	Reviewer		
Rahul Kumar	OTS Hub Design	Author/Contributor		
Vikash Prasad	OTS Hub Design	Author/Contributor		
Nitesh Barnwal	OTS Hub Design	Author/Contributor		

1.5 Abbreviations and definitions

Abbreviation / term	Meaning / definition
TOTSCo	The One Touch Switching Company www.totsco.org.uk
TOTSCo Hub	This is the formal name used by TOTSCo to refer to the hub which will provide services to CPs in support of OTS and GPLB processes, and possibly for other industry processes in the future. TOTSCo have chosen Tech Mahindra to implement and operate the TOTSCo Hub.
CP	Communications provider

Abbreviation / term	Meaning / definition
	This is a term defined by Ofcom in their General Conditions of Entitlement as a "means a person who provides an Electronic Communications Network or an Electronic Communications Service".
RCP	Retail CP. This term was first defined in the OTS Industry Process (and re-used in the GPLB documentation) to define those CPs who provide services at the retail level to end-users, both consumer and business.
TPI	Third party integrator This is a commonly used term within the UK telecoms industry to refer to parties who provide integrations services to CPs, but are not themselves CPs.

2 TOTSCo Hub integration specification

The TOTSCo Hub provides the mechanisms to deliver messages from one party to another in an environment where it is impractical for all parties to talk to each other directly. The parties will be CPs or their agents such as TPIs.

The analogy of a post office is appropriate as TOTSCo are the agent who will accept a sender's message and will be responsible for delivering it on their behalf to the intended recipient. Senders do not need to find a way to deliver the message directly. In architecture parlance, this is a hub and spoke mechanism as opposed to point to point.

Any messaging system requires standards to ensure interoperability, and to that end all messages sent via the TOTSCo Hub will be represented in JSON format and delivered using REST APIs.

Messages are made up of:

- an envelope containing the delivery data needed for the TOTSCo Hub to route the message to the correct destination, including a return address for replies and failures,
- and a message body.

The Hub does not need to know anything about the message body – that information is only for the sender and recipient to know and understand.

2.1 Letterbox API specification

The TOTSCo Hub letterbox API specification defines how messages will be sent to and received from the TOTSCo Hub. The API specification is separate from the documentation of the message formats for the industry processes (e.g. OTS and GPLB), as the Hub does not need to know anything about the message format itself. The Hub acts on a routingID which may or may not be related to the message format. Examples of current routing IDs are described in §4 of this document.

The requirement is that both the TOTSCo Hub and the TOTSCo Hub users (the CPs) all implement the same API specification. This makes it simpler to implement the messaging protocols in a uniform way, as well as supporting the ability to perform peer to peer testing.

A sample definition of the letterbox API specification can be found at the following URL:

https://app.swaggerhub.com/apis/TOTSCO/letterbox/0.4.0

The letterbox provides a mechanism to deliver a message via the TOTSCo Hub to an indentified recipient. The TOTSCo Hub will only process the message envelope, to understand what/who it is for and to be able to process it correctly, and will not process the message body (other than to pass it on to the recipient).

For example, certain types of message sent to the Hub will be subject to a delivery timer with backoff and fallout policies. Others will require that the Hub delivers the message no matter what. The policies for delivery messages will be industry-agreed, and will be defined in the Hub and not specified by the sender.

Please note, the letterbox API specification is not specific to any one messaging or industry process. It is designed as a standard reusable interface; to facilitate a hub and spoke message distribution framework; to support the adoption of near real time message processing and guaranteed message delivery; and where peer to peer interfaces and mechanisms (e.g. email, SFTP, etc) would be impractical or lack the security or functionality that the Hub provides.

2.1.1 The letterbox post API Interface

The letterbox post API takes a JSON message from an authorised source and delivers it to an identified destination. The information needed to route that message is defined within the message envelope contained within the JSON message.

In summary:

- Messages are delivered using a “push push” model – i.e. the source CP will push a message to the TOTSCo Hub, and the Hub will onwards push the message to the destination CP.
- Message exchange between CPs via the TOTSCo Hub is asynchronous, so the “push push” model applies to both requests in one direction and responses in the other direction.
- Messages wll be pushed using https post with TLS v1.3.
- OAuth2 token will used be used to authorise the sender of each https request to the recipient.

When the TOSCo Hub receives a message from a CP (or a TPI), the OAuth2 credentials of the sender of the message will be matched against the source information in the envelope to ensure the message originates from an authorised sender and is not being spoofed.

Similary when a CP (or a TPI) receives a message from the Hub, they will also check that the OAuth2 credentials to ensure the message originated from the Hub and is not being spoofed.

2.1.2 URI format

The API URI format provided by the TOTSCo Hub and each CP (or TPI) will conform to the following convention.

https://{fqdn}/letterbox/{version}/post

The elements of the URI are as follows.

URI Element	Description	Format
FQDN	The Fully Qualified Domain Name of the provider of the letterbox API. The TOTSCo Hub FQDN values are listed in the next table. Each CP will need to provide their FQDN (possibly for their chosen TPI) as part of their Hub endpoint configuration.	Compliant with standard RFC 1035
Version	This is the version number of the letterbox API. This version will only ever change if there is a substantial update in the way messages are processed by the TOTSCo Hub. If a new version is introduced, the previous versions will remain in service for compatibility with existing processes.	n.n Initially “1.0”

The FQDN values used by the TOTSCo Hub will be:

Environment	FQDN
SIT	sit.api.totsco.co.uk
Pre production	preprod.api.totsco.co.uk
Production	api.totsco.co.uk

Note the use of the totsco.co.uk – this is intentially different to the totsco.org.uk domain used for TOTSCo's public facing website and email addresses.

Each CP/TPI will need to provide their FQDN values, and are encouraged to use similar naming conventions for SIT, pre production and production environments.

2.1.3 API details

Field	Value
API Name	TOTSCo-LetterBoxAPI
Context	letterbox
Version	1.0
Resource	post
Transport Level Security	https, TLS 1.3
Port	The TOSCo Hub will expose their API using the standard port 443 for https. It is recommended that CPs / TPIs also expose their API on port 443, but an alternative port number may be specified in the endpoint configuration if required.
Request Format	application/json
Request Headers	Authorization, Accept, Content-Type: text/plain; charset=UTF-8
Tags	totsco

2.1.4 Version Control

OTS Hub will support API versioning, up to a maximum of five API versions. The current version along with four previous versions. For any major or minor changes, the API version will be updated and notified through TOTSCo communication.

e.g. If there are some minor changes, API version will be upgraded from version 1.0 to 1.1 and for any major changes the API version will be upgraded from version 1.0 to 2.0.

2.1.5 Envelope elements

Every message sent through the Hub letterbox API must have an envelope and a message body

- This document defines the envelope.
- The message format documents for the relevant industry process (e.g. OTS or GPLB) define the body – this document does not repeat those specifications.

The envelope contains the addressable information needed to identify and authenticate the message's originator – the "source", and the intended recipient – the "destination".

The envelope also contains a "routingID" which the Hub will use to determine how and where to send the message. Every message specification will define how the routingID should be populated. The Hub will also use the routingID to determine the delivery policy for the message.

Finally there is an array element called "auditData" that must be used to provide information to TOTSCo for reporting to Ofcom and industry, where defined in each industry process.

The example below shows a completed envelope, with sample values including audit information. The "_messageBody" would be where the specific message content will be defined – the body is not processed by the TOTSCo Hub, and it could contain encrypted data.

```
{
   "envelope": {
      "source": {
         "type": "RCPID",
         "identity": "RBCD",
         "correlationID": "XYZ987"
      },
      "destination": {
         "type": "RCPID",
         "identity": "RCBA",
         "correlationID": "ABC123"
      },
      "routingID": " residentialSwitchMatchRequest",
      "auditData": [
         {
            "name": "auditFieldName",
            "value": "auditFieldValue"
         },
         {
            "name": "auditFieldName",
            "value": "auditFieldValue"
         }
      ]
   },
   "_messageBody": {
      "_comment": "The real message body would appear here, either in plain text,
                  or as secureMessage if encyrption is in use"
   }
}
```

The following table defines each element of the envelope:

JSON element	Description	Format	Notes
envelope	A container defining the delivery information for any associated message.	Object	Required
source	A container defines the originator of the message and represents the return address for any message requiring a response.	Object	Required
destination	A container representing the destination of the message and used by the TOTSCo Hub to identify the correct recipient letter box to deliver it to.	Object	Required
source/type destination/type	The name of the directory list where the identity can be found and validated. E.g. "RCPID" for OTS and GPLB messages.	String	Required
source/identity destination/identity	The identity of the sending or receiving entity for the message as defined in the directory list selected. E.g. the RCPID value.	String	Required

JSON element	Description	Format	Notes
source/correlationID destination/correlationID	A string of characters that the message originator will recognise and allow matching of a reply to a request message. In a source element, the correlationID must always be provided, the format can be anything the originator chooses to support their messaging process, but should be sufficiently unique to allow correlation of response with request over a reasonable period of time. In a destination element, the correlationID would only be populated when the message is being sent in response to a message previously sent to you. In that case the correlationID will be the value that was sent by the original sender of the message – i.e. it is being reflected back to them.	String	Required /Optional
routingID	The routingID that the Hub will use to route the message to the recipients desired destination. Each messaging specification will have its own requirements for how this value is populated, but the value must be supported by the Hub and published in the directory.	String	Required
auditData	A list of name value pairs that TOTSCo will use for auditing and reporting. Each messaging specification will have its own requirements for audit data. An example is the error code for a failure response.	Array	Optional
auditData/name	The text name of the property being provided for auditing	String	Required
auditData/value	The value associated to the named entity above.	String	Required
_messageBody	This is the message to be sent to the recipient. The actual element name should be based on the message being sent. Please refer to §4 for the messages supported for the process described in this document.	String	Required

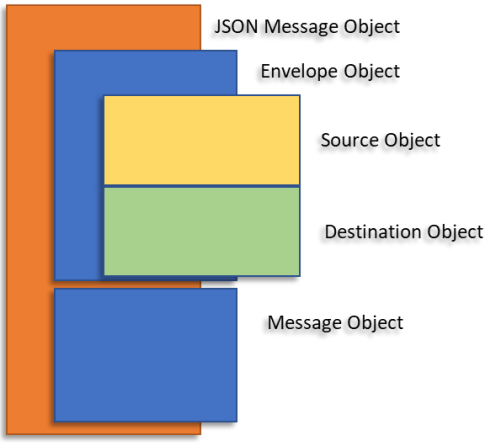

Figure 1 – Post Office JSON Message Envelope Structure

The container structure of a Post Office message is displayed above, the message object is separated from the envelope to allow changes in the content of either structure without affecting each other.

The reason every message must have a source correlation ID is that, as well as the recipient of the message being able to reply to you, in the event of a failure to deliver a message theTOTSCo Hubcan return a delivery failure notification, even if the message was a response message. So you can get delivery failures for response messages as well as request messages.

2.1.6 Auditing requirements

To allow TOTSCo to support the Ofcom reporting requirements for OTS, the audit Data in the envelope must be populated according to the following rules when sending OTS messages.

- messageFormatName – This must be the same value as the JSON element defining the message body sent in the transaction. This applies to all request, responses and failure messages.
- faultCode – If a failure message is being sent through the Hub, the fault code in that failure message must also be replicated in the audit data.

Here is an example showing an unencrypted fault message sent through the Hub.


```
{
   "envelope": {
      "source": {
         "type": "RCPID",
         "identity": "RBCD",
         "correlationID": "XYZ987"
      },
      "destination": {
         "type": "RCPID",
         "identity": "RCBA",
         "correlationID": "ABC123"
      },
      "routingID": "residentialSwitchMatchFailure",
      "auditData": [
         {
            "name": "messageFormatName",
            "value": "residentialSwitchMatchFailure"
         },
         {
            "name": "faultCode",
            "value": "101"
         }
      ]
   },
```

```
      "residentialSwitchMatchFailure": {
         "faultCode": "101",
         "faultText": "failure to match with the supplied information"
      }
   }
```


2.1.7 Message formats

The TOTSCo Hub API specification defines the envelope of the JSON message only, and the API for sending the messages. These are the only parts of the JSON message the Hub will be responsible for understanding. Each message will also contain a message body, and this is the element that the recipient of the message must understand. Both parts together form the entire JSON message.

There may be hundreds of supported message bodies the Hub will deliver, those defined in this document in section §4 relate only to the process described by this document. Each industry process utilising the TOTSCo Hub (e.g. OTS and GPLB) will have its own message format document that defines the message body.

Should any change be made to the API or envelope specification, that change will apply to every messaging process, so the API and envelope specification must be agnostic of those processes.

2.1.8 Letterbox responses

Synchronous responses

The letterbox API REST interface is synchronous, meaning that when a message is sent to the letterbox it will reply within the same communication session. That reply does not contain a JSON message on a successful post as its purpose is only to acknowledge receipt of the message being delivered to the letter box. However, on a failure, a small JSON error structure will be returned describing the nature of the fault.

The letterbox APIs will acknowledge message receipt with a HTTP 202 response code, the definition of which is as follows: "*The request has been accepted for processing, but the processing has not been completed. The request might or might not be eventually acted upon, and may be disallowed when processing occurs.*"

Where the TOSTCo Hub is the recipient of a message sent by a CP (or TPI), the 202 response will be sent once the Hub has validated the message:

- Validated the OAuth credentials
- Validated the format of the message (e.g. valid JSON message)
- Validated the contents of the envelope
- Verified that the sender is authorised to send on behalf of the defined source
- Verified that the source CP and destination CP is valid

If the message fails to be accepted by the API, various 400 errors may result subject to the OAuth processing of the API, or validation of the received message. For example, 401 – Unauthorised, or 403 – Forbidden for authentication errors, or in the event of a JSON format failure, error 400 – Bad request will be returned.

The following table defines the list of http codes for the different errors during validation of the message.

HTTP Status Code	Exception Name / Code	Error Description
400	BAD REQUEST	The API cannot convert the payload data to the underlying data type. The data is not in the expected format. A required field has not been supplied. A validation error occurred.

HTTP Status Code	Exception Name / Code	Error Description
401	UNAUTHORIZED ERROR	The request requires authentication and valid credentials were not provided
403	FORBIDDEN ERROR	Valid credentials have been provided, but the authorisation level is not sufficient for the request
404	OPERATION NOT FOUND	The requested resource was not found
405	Method Not Allowed	The service does not support the HTTP method (eg. POST, GET)
429	Too Many Requests	You have exceeded your quota. You can access API after YYYY-MMM-DD xx: xx:xx+xxxx UTC.
500	INTERNAL SERVER ERROR	An unexpected error has occurred while processing the request
503	Service Unavailable	The server cannot handle the request for a service due to temporary maintenance.

The synchronous acceptance of the message must extract the source element as a minimum to determine if the sender is valid and authorised to be sending messages into the Hub. An invalid source will result in a 401 or 402 error.

For the fault responses, the following JSON will be returned, there will be no envelope.

```
{
    "errorCode": "9002",
    "errorText": "TOTSCO"
}
```

The content of this message is as follows.

JSON element	Description	Format
errorCode	A numeric description of the specific fault	Integer
errorText	A description that represents the nature of the fault and can be used by the message originator to determine remedial action.	Integer

The following table defines the list of http codes and Fault codes the HUB will generate in the event of a validation error before accepting the message for delivery.

No.	Validation	http code	Fault code
1	oAUTH2 token validation	401	NA
2	Validate JSON Message Object structure	400	NA
3	Validate Message Envelope attributes	400	NA
4	Validate if Destination Id is valid	400	9001
5	Validate if Source Type is valid	400	9002
6	Validate if Source RCPD Id is valid	400	9003
7	Validate if Source RCPD ID account status is valid	403	9003
8	Validate Sender Network Location	401	9004
9	Validate if Routing Id is mapped to the Source RCP	400	9010
10	Validate RoutingID	400	9012

Post office faults and messages

In the event of the post office being unable to deliver a message to its intended recipient, or if status update messages are sent as a result of a delivery policy, the post office will create a message to the originator of the message in the following format.

```
{
   "envelope": {
      "source": {
         "type": "RCPID",
         "identity": "TOTSCO"
      },
      "destination": {
         "type": "RCPID",
         "identity": "DCBA",
         "correlationID": "ABC123"
      },
      "routingID": "messageDeliveryFailure",
      "auditData": [{
            name: "originalDestinationType",
            value: "RCPID"
         },
         {
            name: "originalDestination",
            value: "XYZ3211"
         },
         {
            name: "originalRoutingID",
            value: "residentialSwitchMatchRequest"
         },
         {
            name: "faultCode",
            value: "9020"
         }
      ]
   },
   "postOfficeMessage": {
      "code": "9020",
      "text": " Unauthorised use of source code",
      "severity": "failure"
   }
}
```

The post Office Message body describes a notification to the sender of the original message of a failure to deliver the message. The source information will represent the TOTSCO Hub, and the audit data will contain the original intended message recipient and destination. The originators correlation ID will be returned in the destination information. Note that the source does not contain a correlationID, the postOfficeMessage cannot be replied to as it is a notification, and therefore no correlationID is required.

The content of this message then describes the notification information.

JSON element	Description	Format
postOfficeMessage	Container for messages from the post office	Object
Code	A number that represents the nature of the fault and can be used by the message originator to determine remedial action.	Integer
Text	A description of the associated response code	String
Severity	An indicator of the nature of the message about the processing of the originators' message. Values will include, “information”, “warning”, “failure”.	String

The following table defines the list of response codes the post office will generate in the event of an message delivery failure.

Code	Text	Severity
9005	Unable to deliver the message to the destination, no valid route.	Failure
9006	Unable to deliver the message to the destination, rejected, invalid message format.	Failure
9007	Recipient rejected message.	Failure
9008	Unable to deliver the message to the destination, timed out.	Failure

3 Security Implementation

3.1 Transport Layer Security (TLS)

Messages will be pushed from the sender to the recipient (e.g. CP/TPI to Hub, or Hub to CP/TPI) using https. The TOTSCo Hub will expose their API using the standard https port of 443. It is recommended that CPs / TPIs do the same, but they may choose an alternative port as part of their endpoint configuration in the TOTSCo CP portal.

The version of TLS supported will be 1.3. As all parties will be building new endpoints for implementation of OTS/GPLB, there will be **no** backward compatibility to v1.2 or earlier.The TLS will be one-way TLS. For inbound communication (message coming into the hub), the CP would be given the hub's public key which CP would need to install into their truststore. This public key will be shared with the CPs and the time of onboarding or whenever the public key will be renewed.

For outbound communication (message going out to CP), the hub will install each CP's public key into its truststore. CPs must provide their public key to the HUB at the time of onboarding or whenever the keys are renewed.

The server end will require a digital certificate issued by a certificate authority (CA). The issuing CA must be in a certificate hierarchy with a root CA that is commonly trusted.

The HUB will restrict the permissible ciphers to the following values, as both server and client:

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384"

The security provided by TLS 1.3 and the associated digital certificate is that the end-point genuinely corresponds with the specified domain name (either the FQDN, or the certificate may have a CN of form *.domain).
There are some responsibilities to ensure this security is meaningful:

- CPs / TPIs must ensure that the FQDN they have configured for the TOTSCo Hub is based on a valid source, such as documentation retrieved directly from the TOTSCo website.
- CPs / TPIs must ensure that the FQDN they configure for their endpoint(s) via the TOTSCo CP portal are valid for their organisation.
- The private keys associated with digital certificates must be held securely by the owning party.
- TOTSCo will ensure their implementation respects the values configured via the TOTSCo CP portal, and that requests for changes are not accepted from unverified parties.

3.2 API Authentication

As described above, the use of TLS allows the sender to have confidence that messages are being sent to a genuine recipient. To enable the recipient to validate that a message has been sent by a genuine sender, OAuth 2.0 will be used.

OAuth uses token based validation and will be used for all communication between CP/TPI and OTS Hub.

Inbound Communication from CP to OTS Hub:

CP will use OAuth2 Access Token to call the Letter Box API in OTS Hub. The CP will be provided with a client Id , secret key, OAuth2 token generation webservice URL for each of the source locations that will be configured by the CP for sending messages to the HUB. The Token generation URL, Client Id and Client Secret must be stored by RCP in their trusted database store.

The CP client systems will have to use the OAuth2 token generation URL to request an access token from OTS Hub, using the client Id and secret key provided. Please refer to section 3.3.1 for the process to generate access token.

The CP client systems can then use the access token to call the OTS Hub webservice for posting the messages. The OAuth2 token have a validity of 1 hour and the RCP client systems will have to ensure that the tokens are not expired before posting the messages to the Hub.

Outbound Communication from OTS Hub to CP:

OTS Hub will use OAuth2 Access Token to send the outbound message to the CP. To support the authentication mechanism, CP will have to provide the clientid and secretkey, Token key generation webservice URL during the setup for each of their endpoints that will be configured for receiving the messages from HUB. Where an CP has taken the service of TPIs (Third party integrators) for their message journey, the RCP can configure their end points to the TPI connection points and provide the above details for the TPI endpoints.

HUB will store the clientid and secretkey, Token key generation webservice URL in the HUB secure database. Before sending a message to the RCP endpoint, HUB will use the token key generation webservice URL, clientId and secretkey to generate an OAuth2 access token. This token will be sent in the http header of the message, to the RCP endpoint.

3.2.1 Process to generate OAuth2 Access Token

3.2.1.1 Access Token Request

Given below is format of oAUTH2 access token request.

Request URL: https://{fqdn}/oauth2/token

fqdn: fully qualified domain name with the host and port e.g. https://host:port

Use the provided request URL to generate OAuth access token using POST method.

Required Parameters:

Header Parameter		**Comments**
Field	Value	
Authorization	Basic <Base64-encoded client_key:client_secret>	This will convert the passed **Client_Id** and **Client_Secret** into Base64-encoded.
Content-Type	application/x-www-form-urlencoded	
Body parameter		
Field	Value	
grant_type	client_credentials	**grant_type** and its value **client_credentials** need to be passed inside **x-www-form-urlencoded.**

Field	Description	Type	Notes
client_id	Client Id (will be provided by OTS Hub during onboarding)	String	Required

Field	Description	Type	Notes
client_secret	Client Secret (will be provided by OTS Hub)	String	Required

3.2.1.2 Access Token Response

In a successful authorization grant type, the response will contain access token and expiry time. Below is an example of a successful response:

```
HTTP/1.1 200 OK
 Content-Type: application/json;charset=UTF-8
 Cache-Control: no-store
 Pragma: no-cache
{
 "access_token":"{OAuth2 Access Token}",
 "token_type":"bearer",
"scope":"default"
 "expires_in":3600
 }
```

Field	Description	Type	Default Value
access_token	Access token will be used to call the API.	String	-
token_type	Token type describes the type of the token.	String	Bearer
scope	Scope of the access token.	String	default
expires_in	Indicates the validity of token in seconds.	String	3600

3.2.1.3 Exception

Below exceptions would be sent in oAUTH2 token generation response:

Error Code	Description	Cause
400	Bad Request	error found when invalid request value passed
401	Unauthorised	Incorrect client credentials provided
404	Not Found	when incorrect token generation URL is passed
405	Method Not Allowed	when incorrect method type is passed
415	Unsupported Media Type	when mandatory parameters are not passed

4 Routing ID Summary

The TOTSCo Hub will be capable of routing messages of any kind from many different industry processes. Below are a sample list of the current known and proposed routing IDs.

The TOTSCo Hub will allow configuration of adding new routing IDs as and when required.

Industry Process / Function	**routingID**	**routingID**
Post Office	messageDeliveryFailure	
One Touch Switch	residentialSwitchMatchRequest	residentialSwitchOrderUpdateRequest
	residentialSwitchMatchConfirmation	residentialSwitchOrderUpdateConfirmation
	residentialSwitchMatchFailure	residentialSwitchOrderUpdateFailure
	residentialSwitchOrderRequest	residentialSwitchOrderTriggerRequest
	residentialSwitchOrderConfirmation	residentialSwitchOrderTriggerConfirmation
	residentialSwitchOrderFailure	residentialSwitchOrderTriggerFailure
		residentialSwitchOrderCancellationRequest
		residentialSwitchOrderCancellationConfirmation
		residentialSwitchOrderCancellationFailure

5 Appendix A – Messaging version control

All messaging specifications that will be used by industry over the TOTSCo Hub will have the ability to define their own requirements for version control. However, some basic principles for versioning are outlined below for guidance in the event a message format is changed, to help define a standard process for versioning.

Normal practice for APIs is to use a version number in the URI, and indeed the letterbox API's do have such a version number so that if that API was to functionally change then a new version can be created and both supported.

With messaging you can go one of two ways, either include a version number within a message or rename the message itself.

In a distributed environment, it is important to know what the party you are sending messages to supports. So the key decision factor in determining the versioning approach is how to inform the sender what format they can use to communicate with the recipient, as in all cases the recipient is the gating factor.

The Hub directory is where all message formats are listed that a recipient accepts – although not explicitly defined as the formats, they are actually routing ID's, they serve the same function.

To version control a message format without changing its name would require another level of information within the Hub to relate the version of the message and then logic on both the sender and receiver side to interpret the message in specific ways depending on that version and its contents.

Taking the approach of simply introducing a new message format name makes this much simpler, no changes needed to the Hub, and a much clearer communication to the builder and processor of that message what to do with it right up front. This is therefore the recommendation for how new versions of messages should be managed within processes using the Hub.

The next question relates to what drives a message format version change. The consideration here is where a change can be considered minor, adding optional elements to a message for example, or major, renaming an element.

Any system parsing JSON messages should discard elements they are not expecting, most if not all JSON parsers support that capability and was one of the main reasons for choosing JSON as the messaging standard.

Consider a switch match request requires adding a new element called clientContactNumber, that can be considered a minor, non-destructive change and the element made optional. Consumers would update their systems when they are ready to make use of that element but it does not break the process or messaging if you have not yet added support for it and simply ignore the element.

Now consider we rename the services element in the matching request (a bad thing to do, but as an example), and we change it to serviceList. That is clearly a major, destructive change. In this case the message format name should be changed from residentialSwitchMatchRequest to residentialSwitchMatchRequestv2 as the structural format is different and therefore the processing of existing consumers of that message would break.

As the directory holds the supported message formats, it will be incumbent on the message sender to ensure that they use the appropriate format for the intended recipient. If provider A supports v1 and v2 of the match request but provider B only supports v1, then when provider A is creating a match request to BT it would know it had to create the message in the v1 format as v2 isn't yet supported by provider B.

Through industry consultation, process owners will agree on the message format changes, the most appropriate way of updating them, and how to control the versions.

End of Document