[](https://www.dtmercury.com/)A blue and white background

Description automatically generated

Confidentiality Notice: This document contains private information about Mercury (Pty) Ltd. You can't copy, reproduce, or share its product details, intellectual property, or design methods with anyone else unless you have our written consent.

**Mercury API Specification**

**Secure Token Generation**

**Inbound From Client**

Secure token

* 1. Purpose

The secure token ensures that the payload sent by the client is the same as the payload received by Payments Hub. Should the message be altered while in-flight (after being sent but before it is received by Payments Hub), the secure token validation will fail, and the payload will be rejected.

* 1. Secure Token Generation

Some guidelines:

* Secure token is built from string payload with no null fields included.
* The “secureToken” field is not included when generating or validating secure token.
* The payload used to generate the token must match the payload that is sent (barring the “secureToken” field mentioned above).
* Secure token used a security key/subscriber key to be generated(ex. @tcjAwn$5B9Aet91)
* The secure token will then be validated using the same key that was used to generate the secure token, using the exact payload received by the client.
* If the secure token generated by Payments Hub does not match the secure token sent by the client, validation fails.

The Secure Token should be generated using the string payload that is to be sent with this pseudo-code example:

Function GetSecureTokenforPayload(requestBody)

1. If requestBody is null or empty, throw exception
2. requestBody = RemoveSecureTokenFromPayload(requestBody)
3. requestBody = "{" + requestBody
4. Replace "\" and "\"" and " " in requestBody with ""
5. sourceString = SubscriberKey + trimmed requestBody
6. hash = GetHash(sourceString, SubscriberKey)
7. Return hash

The “GetSecureTokenforPayload” method is responsible for preparing the message and building the secure token with the subscriber key provided.

To prepare the payload for to generate a token with this pseudo-code example:

Function RemoveSecureTokenFromPayload(requestBody)

1. If requestBody contains "SECURETOKEN"
2. Split requestBody by ","
3. Set second element to null
4. Return joined string
5. End If
6. Return requestBody

If the “secureToken” field is present within the message, it will be removed here. The “secureToken” field removal is to make it more convenient when testing payloads after small changes were made.

To generate the token:

Function GetHash(data, secret)

1. hashString = HMACSHA256(secret)
2. hashbytes = ComputeHash(data)
3. Initialize StringBuilder digest
4. For each byte b in hashbytes
5. Append b as hexadecimal to digest
6. Return digest as string

Here the prepared payload is hashed with the security token provided. The returned security token is then provided which is to be placed in the “secureToken” field for the message.

* 1. Example Payloads

Credit Transfer Request:

Secure Token: 1afb5f939144b3dae699f062f6fb6c38148d446f259fb8c01f0d44df94ce7346

{"secureToken": "1afb5f939144b3dae699f062f6fb6c38148d446f259fb8c01f0d44df94ce7346","subscriberID": "902","primaryHostReference": "2023083100001","secondaryHostReference": "78602467923","creationDateTime": "2023-10-03T11:27:53.015708","transactionIdentifier": "1A66C164-0EDA-4170-8D13-FBF","transactionUETR": "41dbef46-4f6e-471e-8e94-120a9586febd","transactionType": "PBAC","transactionAmount": 100.00,"transactionReference": "Micheal Block 230930","payerDetails": {"partyName": "King Price Gauteng"},"payerAccountDetail": {"partyAccountIdentification": "78602467923","partyAccountType": "1","partyAccountName": "King Price Gauteng"},"payeeDetails": {"partyName": "Micheal Block","partyIdentification": {"partyPrivateIdentification": "7001012293083"},"partyContactDetail": {"partyMobileNumber": "+27-717654321","partyEmailAddress": "michealb@kingprice.co.za"}},"payeeAccountDetail": {"partyAccountIdentification": "1268602123","partyAccountType": "2","partyAccountBank": "SASFZAJ0","partyAccountName": "Micheal Block"}}

Secure Token: 0cf516addc0d86cadf8a216dfba8841dd571066c10ae57b6f02267b26dfc43ae

{"secureToken": "0cf516addc0d86cadf8a216dfba8841dd571066c10ae57b6f02267b26dfc43ae","subscriberID": "902","primaryHostReference": "2023083100001","secondaryHostReference": "78602467923","creationDateTime": "2023-10-03T11:27:53.015708","transactionIdentifier": "1A66C164-0EDA-4170-8D13-FBF","transactionUETR": "41dbef46-4f6e-471e-8e94-120a9586febd","transactionType": "PBAC","transactionAmount": 100.00,"transactionReference": "Blocked Michael","payerDetails": {"partyName": "King Price Gauteng"},"payerAccountDetail": {"partyAccountIdentification": "78602467923","partyAccountType": "1","partyAccountName": "King Price Gauteng"},"payeeDetails": {"partyName": "Micheal Block","partyIdentification": {"partyPrivateIdentification": "7001012293083"},"partyContactDetail": {"partyMobileNumber": "+27-717654321","partyEmailAddress": "michealb@kingprice.co.za"}},"payeeAccountDetail": {"partyAccountIdentification": "1268602123","partyAccountType": "2","partyAccountBank": "SASFZAJ0","partyAccountName": "Micheal Block"}}

Payment Status Request:  
  
Secure Token: ddf912d13544779007edcad023505a6475fad1581a30e5e28f107ba35f0290e7

{"secureToken": "ddf912d13544779007edcad023505a6475fad1581a30e5e28f107ba35f0290e7","subscriberID": "902","primaryHostReference": "2023083100001","secondaryHostReference": "78602467923","creationDateTime": "2023-10-03T11:27:53.015708","transactionIdentifier": "1A66C164-0EDA-4170-8D13-FBF9E495B680","originalTransactionIdentifier": "CAE4F68F-BFB1-4FB2-B2B0-4AA94096142E","originalUETR": "41dbef46-4f6e-471e-8e94-120a9586febd"}

List All Banks:

Secure Token: 344ff3499b9e44d0eac45add6ad88f9b055a844c7ed07222c5a79f7212e4b7ad

{"secureToken": "344ff3499b9e44d0eac45add6ad88f9b055a844c7ed07222c5a79f7212e4b7ad","subscriberID": "902","creationDateTime": "2023-10-03T11:27:53.015708","transactionIdentifier": "54000001"}

* 1. Example of a secure token process

The process demonstrated here is to showcase how the string value used to generate the secure token is processed and what is ultimately used to generate the secure token.  
  
**Step 1:**  
Request is sent to Direct Transact Mercury API. List All Banks will be used in this case with the following payload:

{"secureToken": "61eea680c5bf7112a69cc641e3bc89886d02185ed646139f5919c1577c0a8a13","subscriberID": "902","creationDateTime": "2023-10-03T11:27:53.015708","transactionIdentifier": "54000001"}

**Step 2:**

In the JSON payload from Step 1, the “secureToken” field and its value is removed from the payload:

"subscriberID": "902""creationDateTime": "2023-10-03T11:27:53.015708""transactionIdentifier": "54000001"}

Note that the payload above has no starting bracket (“{”) and no comma’s (“,”)

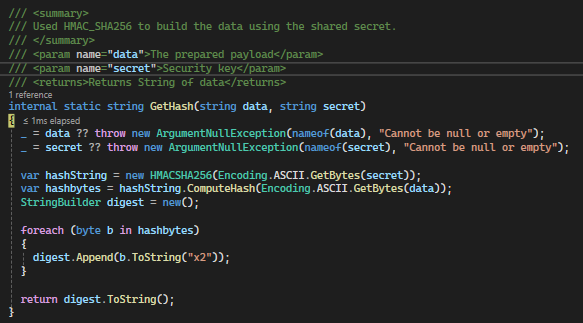
**Step 3:**

The missing starting bracket mentioned in Step 2 is added to the payload:  
  
{"subscriberID": "902""creationDateTime": "2023-10-03T11:27:53.015708""transactionIdentifier": "54000001"}

**Step 4:**

The client-unique security key is then prepended to the message payload. The payload is also trimmed to remove any whitespaces before and after the payload:  
  
aBcD123ExampleKeyHere{subscriberID:902creationDateTime:2023-10-03T11:27:53.015708transactionIdentifier:54000001}

**Step 5:**

The payload from Step 4 is then hashed with the security key. Here’s an example method of how hashing can be done:  
The digest would be the secure token: 344ff3499b9e44d0eac45add6ad88f9b055a844c7ed07222c5a79f7212e4b7ad

**Step 6.1 – Generating token to be sent:**

The secure token field is then added to the payload:

{"secureToken": "344ff3499b9e44d0eac45add6ad88f9b055a844c7ed07222c5a79f7212e4b7ad","subscriberID": "902","creationDateTime": "2023-10-03T11:27:53.015708","transactionIdentifier": "54000001"}

This is the exact payload that should be received by Direct Transact, otherwise secure token validation will fail.

**Step 6.2 – Validating token when message is received:**

A new secure token is generated (Step 1 – Step 5) and this secure token must match the incoming payload’s secure token. If they tokens do not match, validations must fail, processing must stop and an appropriate negative acknowledgement must be returned.