



თიბონი ბანკი  
T B C B A N K



BSC  
systems and ideas

my | GEMINI

## TBC Direct Banking Interface Specification

## Integration Service for Client Accounting Systems

systems and ideas

Authors: Tomáš Motal  
Ondřej Motlík  
Zaza Chaganava

Version: 1.11  
Date: 23.09.2016

## Document History

Version	Author(s)	Date	Status
1.0	Tomáš Motal Ondřej Motlík Zaza Chaganava	13.08.2014	Initial version prepared for ERP/Accounting software system vendors, solution providers, VARs and other DBI customers.
1.1	Zaza Chaganava	09.10.2014	'AccountMovementDetailIo' object type updated: 'parentMovementId' attribute replaced by 'parentExternalPaymentId' attribute.
1.2	Zaza Chaganava	21.10.2014	<ul style="list-style-type: none"> <li>- PaymentEIService.wsdl (see attached DBI_WSDL&amp;XSD.zip file) updated to be compatible with .NET development environment;</li> <li>- New examples for importing single and batch payment orders added (chapters 6.2.2, 6.3.2).</li> </ul>
1.3	Václav Matouch	29.10.2014	Added .NET client example (chapter 12).
1.4	Václav Matouch	03.11.2014	Added VB client example (chapter 12.2).
1.5	Tomáš Motal Zaza Chaganava	24.11.2014 04.12.2014	<ul style="list-style-type: none"> <li>- GetPaymentOrderStatusResponseIo object type (chapter 6.1): singlePaymentData attribute added;</li> <li>- PaymentStatusDataIo object type (chapter 6.4): attribute obligations changed from mandatory to optional; errorDetailEN and errorDetailGE attributes added;</li> <li>- DBI_WSDL&amp;XSD.zip file (chapter 11.2): PaymentIOMessages.xsd and PaymentIOTypes.xsd files updated according to above changes;</li> <li>- Now OTP sending requirement and signing process of imported payment orders depend on user attributes set by Bank.</li> </ul>
1.6	Radek Šebestík	09.12.2014	Added VBA client example (chapter 12.3).
1.7	Zaza Chaganava	17.12.2014	VBA example (xlsm file) slightly updated.
1.8	Zaza Chaganava	10.03.2015	Comments of additionalDescription attribute (ch.6.4, PaymentOrderIo type) and lastMovementTimestamp filter (ch.7.2, AccountMovementFilterIo type) updated.
1.9	Zaza Chaganava	15.06.2015	<ul style="list-style-type: none"> <li>- Payment order attribute formats added (ch.6, Import payment orders);</li> <li>- Comments of beneficiaryTaxCode, taxpayerCode and taxpayerName updated (ch.6);</li> <li>- Now movementId is mapped to CBS ID instead of GEMINI ID; this avoids duplication of the same movement in client system in case some attributes of movement are updated in the bank system (ch.7, Get account movements);</li> <li>- Comments of paymentId, externalPaymentId and</li> </ul>

			<ul style="list-style-type: none"> <li>statusCode updated (ch.7);</li> <li>- New movement message subtype CREATEDFOREARLIERDATE added (ch.8, Postbox messages);</li> <li>- Comments of movement message subtypes updated, PostboxAcknowledgement request example added (ch.8);</li> <li>- Listing of error codes updated (ch.10, Error handling).</li> </ul>
1.10	Zaza Chaganava	16.06.2015	<ul style="list-style-type: none"> <li>- Postbox messages updated (chapter 8);</li> <li>- Examples of HTTP requests added (chapter 11).</li> </ul>
1.11	Zaza Chaganava	23.09.2016	<ul style="list-style-type: none"> <li>- singlePaymentRequestId optional attribute added to PaymentOrderIo object type (chapter 6.4);</li> <li>- batchPaymentRequestId optional attribute added to ImportBatchPaymentOrderRequestIo object type (chapter 6.3.1);</li> <li>- GetSinglePaymentId WS added (chapter 6.2.1);</li> <li>- GetBatchPaymentId WS added (chapter 6.3.1);</li> <li>- new error codes added (chapter 10);</li> <li>- example 2, faultstring updated (chapter 5.5);</li> <li>- example 4, faultcode updated (chapter 5.5).</li> </ul>

# Contents

DOCUMENT HISTORY .....	2
CONTENTS .....	4
1. INTRODUCTION .....	5
2. REFERENCES .....	5
3. ADMINISTRATION .....	5
4. JOURNALING .....	6
5. SECURITY .....	6
5.1. AUTHENTICATION .....	6
5.2. AUTHORIZATION .....	6
5.3. INTERFACE SPECIFICATION.....	7
5.4. PASSWORD CHANGE WEB SERVICE.....	8
5.4.1. INTERFACE SPECIFICATION.....	8
5.5. EXAMPLES .....	8
6. IMPORT PAYMENT ORDERS .....	10
6.1. GET PAYMENT ORDER STATUS .....	10
6.1.1. INTERFACE SPECIFICATION.....	11
6.2. IMPORT SINGLE PAYMENT ORDERS .....	12
6.2.1. INTERFACE SPECIFICATION.....	12
6.2.2. EXAMPLES .....	17
6.3. IMPORT BATCH PAYMENT ORDER .....	19
6.3.1. INTERFACE SPECIFICATION.....	19
6.3.2. EXAMPLES .....	21
6.4. COMMON OBJECTS USED IN PAYMENT WS.....	23
6.5. PAYMENT ORDER ATTRIBUTE FORMATS .....	25
7. GET ACCOUNT MOVEMENTS .....	26
7.1. FILTER COMBINATIONS .....	26
7.1.1. GET MOVEMENT BY ID .....	26
7.1.2. GET MOVEMENTS BY LAST MOVEMENT TIMESTAMP .....	26
7.1.3. GET MOVEMENTS BY OTHER FILTERING PARAMETERS .....	26
7.2. INTERFACE SPECIFICATION.....	27
7.3. EXAMPLES .....	30
7.3.1. GET ACCOUNT MOVEMENT BY ID .....	30
7.3.2. GET ACCOUNT MOVEMENTS BY DATE RANGE WITH PAGING.....	31
8. POSTBOX MESSAGES.....	34
8.1. INTERFACE SPECIFICATION.....	34
8.2. EXAMPLES .....	37
9. VALIDATIONS .....	39
10. ERROR HANDLING .....	39
11. COMMON OBJECTS USED IN INTERFACES .....	41
11.1. DATE FORMATS .....	42
11.2. WSDL & XSD.....	42
11.3. ABBREVIATIONS .....	42
11.4. EXAMPLES OF HTTP REQUESTS .....	42
12. DEVELOPMENT GUIDE FOR .NET.....	43
12.1. ADDING SOAP SECURITY HEADER IN C# .....	43
12.2. ADDING SOAP SECURITY HEADER IN VISUAL BASIC .....	45
12.3. CALLING DBI WS FROM EXCEL.....	47
12.3.1. CREATING WS CLIENT DLL IN VISUAL BASIC.....	47
12.3.2. REGISTER DLL ON CLIENT MACHINE.....	50
12.3.3. USE REGISTERED DLL IN VBA.....	50
12.3.4. USE EXCEL TO CALL DBI VIA VISUAL BASIC DLL.....	50

## 1. Introduction

TBC Direct Banking Interface (DBI) provides corporate bank clients easier integration of external accounting systems (or Client Systems) with TBC Internet Banking system (my|GEMINI IBS) for data exchange purposes (import payments orders into bank system and get account movements data from bank system).

## 2. References

Referenced resource name	Comment
<a href="#">TBC_Operation_List.xlsx</a>	Defines concrete operation types (rights), roles, role trees for administration console, journaling
<a href="#">Corporates direct channels management and signing rules</a>	Confluence page with mockups, it shows how administration should look like for DBI channel/users
<a href="#">myGEMINI_accountsystems_integration.docx</a>	Pre-analysis of DBI module
<a href="#">ERP Systems Integration</a>	Root confluence page of DBI documentation
<a href="#">catalogs TBC.zip</a> <a href="#">catalogs TBC_part2.zip</a> <a href="#">TBC_System_Catalogs.zip</a>	See DBI catalogs in the file attached herein:  DBI_catalogs.zip

## 3. Administration

A new DBI channel was created for external accounting systems. All administration of DBI channel and its users is done in the administration console. Each corporate client can have one DBI channel with any number of users in this channel. Concrete details of direct channel management and direct channel user management can be found on Confluence (see chapter References).

Following system parameters are introduced:

- **DBI\_CREDENTIALS\_EXPIRATION\_TIME** – specifies in how many days does the password expire and needs to be changed
- **DBI\_PASSWORD\_REGEX** – specifies how the password must look like (how many characters, capital letters, numbers, special characters, etc.) to be accepted
- **DBI\_MAX\_NO\_OF\_INCORRECT\_LOGIN\_ATTEMPTS** – specifies the amount of allowed login attempts before user is temporarily blocked
- **DBI\_RECOVER\_TEMPORARY\_BLOCKED\_DCU\_TIME\_MIN** – specifies after how many minutes should the temporarily blocked users be unblocked
- **DBI\_PAGING\_SIZE** – specifies how many entries can be returned in 1 response. Used in getMovements WS
- **DBI\_MAX\_NO\_OF\_SINGLE\_PAYMENTS\_PER\_REQUEST** – specifies max. number of single payment orders that can be imported per one request. Used in ImportSinglePaymentOrders WS
- **DBI\_MAX\_NO\_OF\_PAYMENTS\_PER\_BATCH** – specifies max. number of payment orders that can be imported per one batch. Used in ImportBatchPaymentOrder WS

Distribution of usernames (login names) and passwords for DBI channel is supposed to be done via security envelopes.

## 4. Journaling

Journaling of activity in DBI channel is done the same way as it is done for other channels. Concrete specification what is journalled can be found in Operation List document.

## 5. Security

Authorization and authentication is done each time any web service is called. TLS 1.2 protocol is used for communication security.

### 5.1. Authentication

Authentication credentials will be different from Internet Banking System (IBS) credentials (different instance in database) and will have a different security policy (expiration of password, length of password etc.).

Authentication is done at each request sent to IBS. The client user identifies himself by sending an username, password and OTP (one time password generated from a security token device that is assigned to the user in admin console) in the header of each request (see section with interface specification for more details). Some clients may use digital certificate based authentication instead of sending of OTP – depends on user attributes set by Bank.

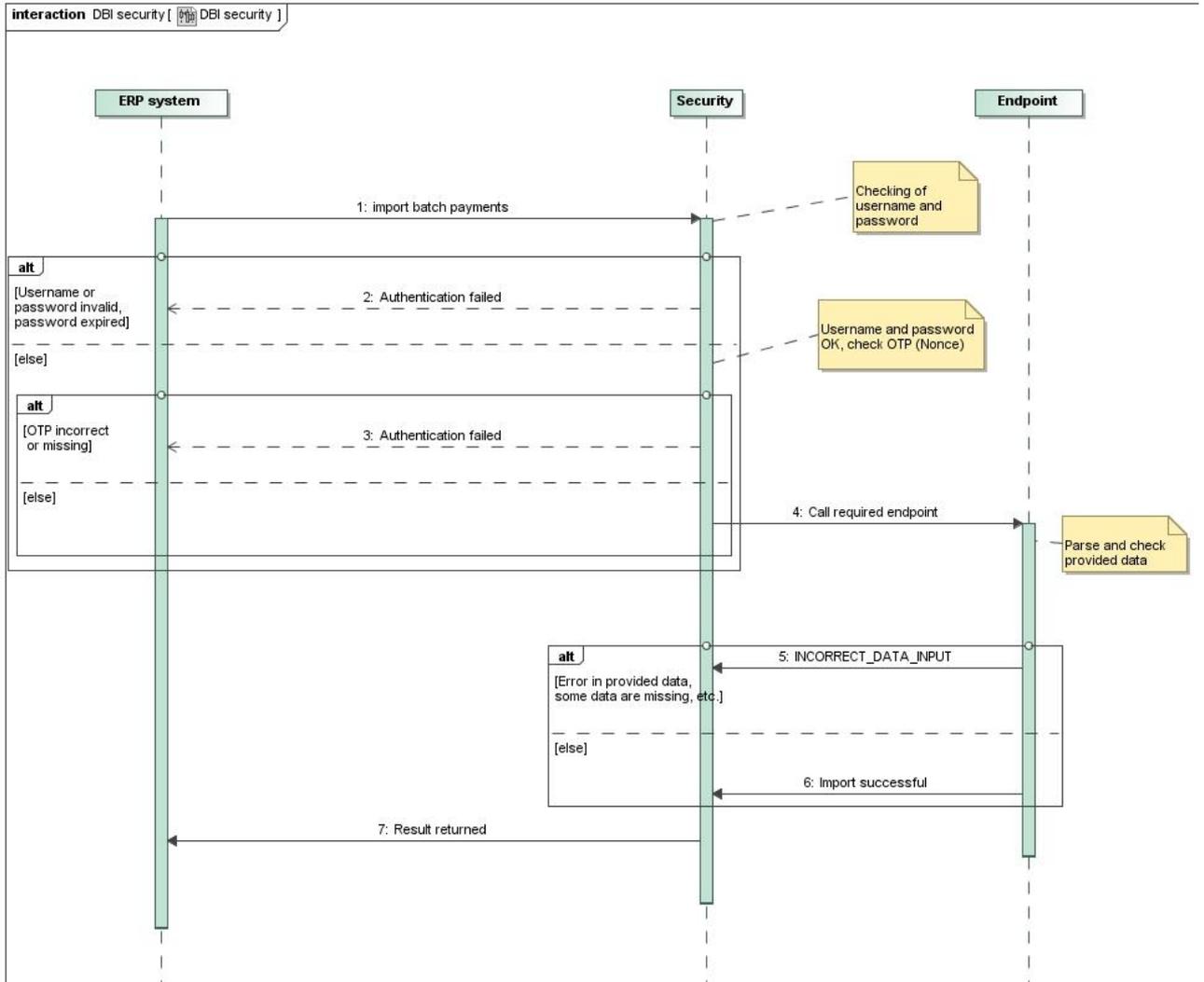
### 5.2. Authorization

Access to any action (downloading data, import of payment orders, etc.) will be controlled via user rights. For access to my|GEMINI via DBI, user needs to have proper access to this channel and needs to be granted proper rights. There are 2 preset profiles used for DBI channel users:

1. DBI\_CORPORATE\_ACTIVE – user has all rights in DBI channel. He can request movements from my|GEMINI IBS and import payment orders to my|GEMINI IBS.
2. DBI\_CORPORATE\_PASSIVE – user has read-only rights. He can request movements from my|GEMINI IBS but is unable to perform active operations like importing payments.

Detailed information about roles and associated rights can be found in Operation List document (see chapter References).

Below is a sequence diagram that shows how security works in DBI channel:



### 5.3. Interface specification

The following attributes are part of every other interface specification part in this document and they must be located in soap header of every web service call in DBI channel. For the security header <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd> is used. Below you can find how does our security header looks like and section 5.5 contains an example of the soap header.

List of request soap header attributes in all web services called:

Object type					
Security					
Attribute	Type	Oblig.	Values	Comment	ASD Code
usernameToken	UsernameToken	1..1	---	Username token object	---
Object type					
UsernameToken					
Attribute	Type	Oblig.	Values	Comment	ASD Code
Username	String	1..1	---	Name of the user for login functionality	---

Password	String	1..1	---	Password of the user for login functionality	---
Nonce	String	0..1	---	OTP generated from a security token device. OTP is NOT REQUIRED in calls of postbox interface and in getPaymentOrderStatus interface	---

**Object definition 1: Security object**

## 5.4. Password change Web service

A web service for changing password for DBI users was implemented. User can change password only to himself (the username and password in soap header will be used for identifying the user whose password is going to be changed). For an example of a request for changing password see chapter 5.5.

There are several situations when the change of password is not allowed:

- 1) User is in status temporarily blocked – user can end in this state if he tries to send a request with incorrect credentials several times (the allowed amount of tries is specified by parameter DBI\_MAX\_NO\_OF\_INCORRECT\_LOGIN\_ATTEMPTS). If user manages to block himself then he can be unblocked automatically after a specified time (this time is specified by parameter DBI\_RECOVER\_TEMPORARY\_BLOCKED\_DCU\_TIME\_MIN) or can be unblocked by bank upon request.
- 2) User is in status blocked
- 3) User is in status deleted

### 5.4.1. Interface specification

Object type					
<u>ChangePasswordRequestIo</u>		Main change password request object			
Attribute	Type	Oblig.	Values	Comment	ASD Code
newPassword	String	1..1		New password	---

**Object definition 2: ChangePasswordRequestIo object**

Object type					
<u>ChangePasswordResponseIo</u>		Main get change password response object			
Attribute	Type	Oblig.	Values	Comment	ASD Code
Message	String	1..1		Informative message	---

**Object definition 3: ChangePasswordResponseIo object**

## 5.5. Examples

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:myg="http://www.mygemini.com/schemas/mygemini"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <soapenv:Header>
    <wsse:Security>

```

```

<wsse:UsernameToken>
  <wsse:Username>USERNAME</wsse:Username>
  <wsse:Password>PASSWORD</wsse:Password>
  <wsse:Nonce>1111</wsse:Nonce>
</wsse:UsernameToken>
</wsse:Security>
</soapenv:Header>
<soapenv:Body>
  .....
</soapenv:Body>
</soapenv:Envelope>

```

**Example 1: Valid SOAP header with username token**

If client user has inserted incorrect username or password (authentication fails) then the following message is returned:

```

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode
xmlns:ns0="http://www.mygemini.com/schemas/mygemini">ns0:INCORRECT_CREDENTIALS</faultcode>
      <faultstring xml:lang="en">Username or Password is incorrect.</faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

**Example 2: SOAP fault response in case of authentication failure**

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:myg="http://www.mygemini.com/schemas/mygemini"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <soapenv:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>USERNAME</wsse:Username>
        <wsse:Password>PASSWORD</wsse:Password>
        <wsse:Nonce>1111</wsse:Nonce>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <myg:ChangePasswordRequestIo>
      <myg:newPassword>NEWPASSWORD</myg:newPassword>
    </myg:ChangePasswordRequestIo>
  </soapenv:Body>
</soapenv:Envelope>

```

**Example 3: Valid Change password request**

If the user does not have required rights to perform one of the DBI provided operations then the following message is returned:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode
xmlns:ns0="http://www.mygemini.com/schemas/mygemini">ns0:NOT_AUTHORIZED</faultcode>
      <faultstring xml:lang="en">User is not authorized to import some of the single payment
orders.</faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Example 4: Response in case of unauthorized attempt to import single payment order when user does not have sufficient rights**

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode
xmlns:ns0="http://www.mygemini.com/schemas/mygemini">ns0:CREDENTIALS_MUST_BE_CHANGED</fa
ultcode>
      <faultstring xml:lang="en">Credentials have to be changed.</faultstring>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

**Example 5: SOAP Fault response in case that client user needs to change his password**

## 6. Import payment orders

This feature enables external accounting systems to import payment orders directly into <sup>my</sup>|GEMINI IBS without the need to manually export the required data into a file and then manually import them into <sup>my</sup>|GEMINI IBS (this approach will be still possible though).

There are 2 possible scenarios (depends on user attributes set by Bank):

- 1) The imported payments will be unsigned. In this case <sup>my</sup>|GEMINI IBS will apply its standard flow and handle the signature of the imported payments.
- 2) The imported payments will be signed. In this case <sup>my</sup>|GEMINI IBS will skip part of its standard flow for signing of the imported payments.

### 6.1. Get payment order status

Get payment order status functionality provides user with the possibility to check the status of any imported single or batch payment order. In case a batch payment order status is requested - this interface in addition provides (if available at given moment) information about each payment order in the imported batch: position (this element matches the position element in ImportBatchPaymentOrderRequestIo), paymentId assigned to the payment order by <sup>my</sup>|GEMINI system and status of the payment order.

In this web service the **Nonce** element in security header **is not required!**

### 6.1.1. Interface specification

Object type					
<u>GetPaymentOrderStatusRequestIo</u>		Main get payment order status request object			
Attribute	Type	Oblig.	Values	Comment	ASD Code
singlePaymentId	Long	0..1	---	Single payment ID	---
batchPaymentId	Long	0..1	---	Batch payment ID	---

#### Object definition 4: GetPaymentOrderStatusRequestIo object

Only one of the two attributes (singlePaymentId or batchPaymentId) must be supplied. In case none or both attributes are sent the WS will return a SOAP fault.

Object type					
<u>GetPaymentOrderStatusResponseIo</u>		Main get payment order status response object			
Attribute	Type	Oblig.	Values	Comment	ASD Code
status	String	1..1	For values see catalogs: BatchStatus – for batch payments; Transaction Status – for single payments	Current status of payment	---
singlePaymentData	PaymentStatus DataIo	0..1	---	In case singlePaymentId is supplied in the GetPaymentOrderStatusRequestIo and the single payment is in some error status, this element will be included in the response and will contain errorDetailEN/GE elements.	---
batchPaymentData	PaymentStatus DataIo	0..*	---	In case batchPaymentId is supplied in the GetPaymentOrderStatusRequestIo and the batch has been processed, one such element is provided for each payment in the imported batch.	---

#### Object definition 5: GetPaymentOrderStatusResponseIo object

There are 4 possible scenarios when getting status of batch payment:

- 1) Batch is in REGISTERED, WAITING\_FOR\_CERTIFICATION, CERTIFIED, VERIFIED or FINISHED status – in such case all payment orders in the batch will be returned and batchPaymentData elements will contain: position, paymentId, paymentStatus;

- 2) Batch is in FINISHED\_WITH\_ERRORS status (some payment orders in the batch are in finished status and others are in failed status) or in FAILED status (all payment orders in the batch are in failed status) - all payment orders in the batch will be returned and batchPaymentData elements will contain:
  - a. for finished payments: position, paymentId, paymentStatus;
  - b. for failed payments: position, paymentId, paymentStatus, errorDetailEN, errorDetailGE;
- 3) Batch is in ERROR status - only payment orders with errors will be returned and batchPaymentData elements will contain: position, errorDetailEN, errorDetailGE;
- 4) Batch is in CANCELLED status - only batch status will be returned, no batchPaymentData elements will be returned in the response.

## 6.2. Import single payment orders

Import single payment orders functionality allows external systems to import single payment orders into bank system. In one call 1 to N single payment orders of different types can be imported. For a list of available payment order types see the table below:

Payment order type	xsd element name
Transfer to own account - debit and credit accounts have same currency or have different currencies (currency exchange)	TransferToOwnAccountPaymentOrderIo
Transfer within TBC Bank	TransferWithinBankPaymentOrderIo
Transfer to Other Bank in National Currency	TransferToOtherBankNationalCurrencyPaymentOrderIo
Transfer to Other Bank in Foreign Currency	TransferToOtherBankForeignCurrencyPaymentOrderIo
Treasury Transfer	TreasuryTransferPaymentOrderIo
Treasury Transfer for other person or company	TreasuryTransferPaymentOrderIo

**Table 1: Available payment order types and their xsd element equivalents**

If external system sends multiple single payment orders of different types in one call and user has no right to import at least one of those types then the whole call will be canceled.

### 6.2.1. Interface specification

List of SOAP body request attributes in ImportSinglePaymentOrders WS:

Object type					
ImportSinglePaymentOrdersRequestIo					
Extension	AbstractIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
singlePaymentOrder	PaymentOrderIo	1..*	---	List of single payment orders (can be various types)	---

**Object definition 6: ImportSinglePaymentOrdersRequestIo object**

**PaymentOrderIo** is an abstract element (see chapter 6.4). It has 5 distinct children object types that add payment type specific elements; they are listed below. Columns 'Type' in object definition tables show type and accepted format of payment order attributes; see chapter 6.5 for attribute formats description.

Object type					
TransferToOwnAccountPaymentOrderIo					
Extension	PaymentOrderIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code

**Object definition 7: TransferToOwnAccountPaymentOrderIo object**

Transfer to own account payment order has 3 possible scenarios:

- 1) Transfer between 2 accounts when the both accounts have the same currency;
- 2) Transfer between 2 accounts when the accounts have different currencies:
  - a. Sell order – the amount is specified in debit account currency;
  - b. Buy order – the amount is specified in credit account currency.

The bank system will decide which of the 3 scenarios should be processed based on **amount**, **debitAccount** and **creditAccount** elements' currencies:

e.g. in case debit account currency is USD and credit account currency is GEL, and:

- amount is 100USD – then 100USD will be transferred from debit account, and credit account will receive GEL amount calculated according to currency exchange rate;
- amount is 100GEL – then credit account will receive 100GEL, and from debit account will be transferred USD amount calculated according to currency exchange rate.

Transfer to own account payment order specifics:

- **creditAccount** element is mandatory;
- **creditAccount** element's **accountCurrencyCode** is mandatory;
- **amount** element's currency must match **debitAccount** or **creditAccount** element's currency.

Object type					
TransferWithinBankPaymentOrderIo					
Extension	PaymentOrderIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
beneficiaryName	String 70R	1..1	---	Credit account owner name	INTBC3
beneficiaryTaxCode	String 11n	0..1	---	Tax code (when beneficiary is company) or personal number (when beneficiary is individual person) of credit account owner. This attribute is optional, but in case it is specified, bank system will check and accept order only if the specified credit account really belongs to the company/person that has the specified tax code / personal number.	INTBC12

**Object definition 8: TransferWithinBankPaymentOrderIo object**

Transfer within bank payment order specifics:

- **creditAccount** element is mandatory, but it's **accountCurrencyCode** element is not used.

Object type					
TreasuryTransferPaymentOrderIo					
Extension	PaymentOrderIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
taxpayerCode	String 11n	0..1	---	Tax payer identification code of person or company, for which client makes treasury transfer (mandatory when taxpayerName is filled; if this attribute is filled this means that payment is for other person or company)	TREAS1
taxpayerName	String 100R	0..1	---	Name of person or company, for which client makes treasury transfer (mandatory when taxpayerCode is filled; if this attribute is filled this means that payment is for other person or company)	TREAS2
treasuryCode	String 9!n	1..1	---	Tax payment treasury code	TREAS6

**Object definition 9: TreasuryTransferPaymentOrderIo object**

Treasury transfer payment order specifics:

- **creditAccount** element must not be used. Instead of this element bank system for treasury transfers uses code specified in the **treasuryCode** element;
- **description** element must not be used. It is set automatically by bank system based on the **treasuryCode** element. If description is filled this will cause an error;
- If either the **taxpayerCode** or the **taxpayerName** element is filled then the other is mandatory as well. These two elements **should not be used** when payment is **not** for other person or company.

Object type					
TransferToOtherBankNationalCurrencyPaymentOrderIo					
Extension	PaymentOrderIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
beneficiaryName	String 70R	1..1	---	Credit account owner name	OUTTBC3

beneficiaryTaxCode	String 11n	0..1	---	Tax code or personal number of credit account owner.	OUTTBC13
--------------------	---------------	------	-----	--	----------

**Object definition 10: TransferToOtherBankNationalCurrencyPaymentOrderIo object**

Transfer to other bank in national currency payment order specifics:

- **creditAccount** element is mandatory, but it's **accountCurrencyCode** element is not used;
- **amount** element's **currency** must be GEL.

Object type					
TransferToOtherBankForeignCurrencyPaymentOrderIo					
Extension	PaymentOrderIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
beneficiaryName	String 140S *	1..1	---	Credit account owner name	FPO16
beneficiaryAddress	String 140S *	1..1	---	Credit account owner country	FPO13
beneficiaryBankCode	String 34S	1..1	---	Beneficiary bank identification code	FPO18
beneficiaryBankName	String 140S	0..1	---	Beneficiary bank name (optional if bank SWIFT code is specified)	FPO17
intermediaryBankCode	String 34S	0..1	---	Intermediary bank identification code	FPO24
intermediaryBankName	String 140S	0..1	---	Intermediary bank name (optional if intermediary bank SWIFT code is specified or code is not specified at all)	FPO25
chargeDetails	String 3!A	1..1	For values see catalog FeePayment Method	Detail of charge	FPO5

**Object definition 11: TransferToOtherBankForeignCurrencyPaymentOrderIo object**

Transfer to other bank in foreign currency payment order specifics:

- **creditAccount** element is mandatory, but it's **accountCurrencyCode** element is not used;
- **amount** element's **currency** must be the same as **debitAccount** element's currency;
- Summary length of (**beneficiaryBankName** + ' ' + **beneficiaryAddress**) should not exceed 140 characters.

List of SOAP body response attributes in ImportSinglePaymentOrders WS:

Object type					
<u>ImportSinglePaymentOrdersResponseIo</u>					
Extension					
Attribute	Type	Oblig.	Values	Comment	ASD Code
singlePaymentOrdersResults	PaymentOrderResultIo	0..*	---	---	---
Object type					
<u>PaymentOrderResultIo</u>					
Extension		AbstractIo			
Attribute	Type	Oblig.	Values	Comment	ASD Code
position	Integer	1..1	---	Position copied from request message of given single payment order. Serves for matching which PaymentId belongs to wich payment order in request.	---
PaymentId	String	1..1	---	Payment order ID assigned by my GEMINI	ACCT10

**Object definition 12: ImportPaymentOrdersResponseIo and PaymentOrderResultIo object**

When ImportSinglePaymentOrdersRequestIo is successful, my|GEMINI assigns unique ID to each imported single payment order and returns its value in ImportSinglePaymentOrdersResponseIo, in the PaymentId attribute; in case connection between client system and my|GEMINI IBS is broken before client system gets response, GetSinglePaymentId WS can be used to get ID that was assigned by my|GEMINI to the imported payment order:

Object type					
<u>GetSinglePaymentIdRequestIo</u>		Main get single payment ID request object			
Attribute	Type	Oblig.	Values	Comment	ASD Code
singlePaymentRequestId	Long 19n	1..1	---	Single payment request ID that was specified in the import request	---

**Object definition 13: GetSinglePaymentIdRequestIo object**

Object type					
<u>GetSinglePaymentIdResponseIo</u>		Main get single payment ID response object			
Attribute	Type	Oblig.	Values	Comment	ASD Code
PaymentId	String	0..1	---	Payment order ID assigned by my GEMINI	ACCT10

**Object definition 14: GetSinglePaymentIdResponseIo object**

In case specified single payment request ID is not found for a given client, WS will return a SOAP fault (see chapter "Error handling").

In the GetSinglePaymentId web service the **Nonce** element in security header **is not required!**

## 6.2.2. Examples

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:myg="http://www.mygemini.com/schemas/mygemini" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>USERNAME</wsse:Username>
        <wsse:Password>PASSWORD</wsse:Password>
        <wsse:Nonce>1111</wsse:Nonce>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <myg:ImportSinglePaymentOrdersRequestIo>
      <myg:singlePaymentOrder xsi:type="myg:TransferToOwnAccountPaymentOrderIo">
        <myg:creditAccount>
          <myg:accountNumber>GE61TB1144845062600001</myg:accountNumber>
          <myg:accountCurrencyCode>GEL</myg:accountCurrencyCode>
        </myg:creditAccount>
        <myg:debitAccount>
          <myg:accountNumber>GE69TB1144836020100001</myg:accountNumber>
          <myg:accountCurrencyCode>GEL</myg:accountCurrencyCode>
        </myg:debitAccount>
        <myg:documentNumber>200000</myg:documentNumber>
        <myg:amount>
          <myg:amount>2</myg:amount>
          <myg:currency>GEL</myg:currency>
        </myg:amount>
        <myg:position>1</myg:position>
        <myg:description>Transfer to Own Account</myg:description>
      </myg:singlePaymentOrder>
      <myg:singlePaymentOrder xsi:type="myg:TransferToOwnAccountPaymentOrderIo">
        <myg:creditAccount>
          <myg:accountNumber>GE86TB1144836120100002</myg:accountNumber>
          <myg:accountCurrencyCode>USD</myg:accountCurrencyCode>
        </myg:creditAccount>
        <myg:debitAccount>
          <myg:accountNumber>GE69TB1144836020100001</myg:accountNumber>
          <myg:accountCurrencyCode>GEL</myg:accountCurrencyCode>
        </myg:debitAccount>
        <myg:documentNumber>200001</myg:documentNumber>
        <myg:amount>
          <myg:amount>2</myg:amount>
          <myg:currency>USD</myg:currency>
        </myg:amount>
        <myg:position>2</myg:position>
        <myg:description>Currency Exchange</myg:description>
      </myg:singlePaymentOrder>
      <myg:singlePaymentOrder xsi:type="myg:TransferWithinBankPaymentOrderIo">
        <myg:creditAccount>
          <myg:accountNumber>GE88TB0600000003718633</myg:accountNumber>
```

```

</myg:creditAccount>
<myg:debitAccount>
  <myg:accountNumber>GE69TB1144836020100001</myg:accountNumber>
  <myg:accountCurrencyCode>GEL</myg:accountCurrencyCode>
</myg:debitAccount>
<myg:documentNumber>200002</myg:documentNumber>
<myg:amount>
  <myg:amount>2</myg:amount>
  <myg:currency>GEL</myg:currency>
</myg:amount>
<myg:position>3</myg:position>
<myg:additionalDescription>Transfer within TBC Bank Example</myg:additionalDescription>
<myg:description>Transfer within TBC Bank</myg:description>
<myg:beneficiaryName>Some Company Ltd</myg:beneficiaryName>
</myg:singlePaymentOrder>
<myg:singlePaymentOrder xsi:type="myg:TransferToOtherBankNationalCurrencyPaymentOrderIo">
  <myg:creditAccount>
    <myg:accountNumber>GE33BG0000000263255500</myg:accountNumber>
  </myg:creditAccount>
  <myg:debitAccount>
    <myg:accountNumber>GE69TB1144836020100001</myg:accountNumber>
    <myg:accountCurrencyCode>GEL</myg:accountCurrencyCode>
  </myg:debitAccount>
  <myg:documentNumber>200003</myg:documentNumber>
  <myg:amount>
    <myg:amount>2</myg:amount>
    <myg:currency>GEL</myg:currency>
  </myg:amount>
  <myg:position>4</myg:position>
  <myg:additionalDescription>Transfer to Other Bank in NC Example</myg:additionalDescription>
  <myg:description>Transfer to Other Bank in National Currency</myg:description>
  <myg:beneficiaryName>Some Company Ltd</myg:beneficiaryName>
  <myg:beneficiaryTaxCode>100200300</myg:beneficiaryTaxCode>
</myg:singlePaymentOrder>
<myg:singlePaymentOrder xsi:type="myg:TransferToOtherBankForeignCurrencyPaymentOrderIo">
  <myg:creditAccount>
    <myg:accountNumber>CZ6508000000192000145399</myg:accountNumber>
  </myg:creditAccount>
  <myg:debitAccount>
    <myg:accountNumber>GE86TB1144836120100002</myg:accountNumber>
    <myg:accountCurrencyCode>USD</myg:accountCurrencyCode>
  </myg:debitAccount>
  <myg:documentNumber>200004</myg:documentNumber>
  <myg:amount>
    <myg:amount>2</myg:amount>
    <myg:currency>USD</myg:currency>
  </myg:amount>
  <myg:position>5</myg:position>
  <myg:additionalDescription>EXAMPLE</myg:additionalDescription>
  <myg:description>TRANSFER TO OTHER BANK IN FOREIGN CURRENCY</myg:description>
  <myg:beneficiaryName>SOME COMPANY LTD</myg:beneficiaryName>
  <myg:beneficiaryAddress>COUNTRY</myg:beneficiaryAddress>
  <myg:beneficiaryBankCode>MYBANKCODE</myg:beneficiaryBankCode>
  <myg:beneficiaryBankName>MY SPECIAL BANK NAME</myg:beneficiaryBankName>
  <myg:intermediaryBankCode>BDLEIT21300</myg:intermediaryBankCode>
  <myg:intermediaryBankName></myg:intermediaryBankName>
  <myg:chargeDetails>SHA</myg:chargeDetails>
</myg:singlePaymentOrder>

```

```

<myg:singlePaymentOrder xsi:type="myg:TreasuryTransferPaymentOrderIo">
  <myg:debitAccount>
    <myg:accountNumber>GE69TB1144836020100001</myg:accountNumber>
    <myg:accountCurrencyCode>GEL</myg:accountCurrencyCode>
  </myg:debitAccount>
  <myg:documentNumber>200005</myg:documentNumber>
  <myg:amount>
    <myg:amount>2</myg:amount>
    <myg:currency>GEL</myg:currency>
  </myg:amount>
  <myg:position>6</myg:position>
  <myg:additionalDescription>Treasury Transfer Example</myg:additionalDescription>
  <myg:treasuryCode>100071003</myg:treasuryCode>
</myg:singlePaymentOrder>
<myg:singlePaymentOrder xsi:type="myg:TreasuryTransferPaymentOrderIo">
  <myg:debitAccount>
    <myg:accountNumber>GE69TB1144836020100001</myg:accountNumber>
    <myg:accountCurrencyCode>GEL</myg:accountCurrencyCode>
  </myg:debitAccount>
  <myg:documentNumber>200006</myg:documentNumber>
  <myg:amount>
    <myg:amount>2</myg:amount>
    <myg:currency>GEL</myg:currency>
  </myg:amount>
  <myg:position>7</myg:position>
  <myg:additionalDescription>Treasury Transfer for other Example</myg:additionalDescription>
  <myg:taxpayerCode>100200300</myg:taxpayerCode>
  <myg:taxpayerName>Other Company Ltd</myg:taxpayerName>
  <myg:treasuryCode>100071003</myg:treasuryCode>
</myg:singlePaymentOrder>
</myg:ImportSinglePaymentOrdersRequestIo>
</soapenv:Body>
</soapenv:Envelope>

```

**Example 6: Valid SOAP message for different payment order types in single payments import**

### 6.3. Import batch payment order

Import batch payment order functionality allows external systems to import only 1 batch of single payment orders per one request. If external system needs to import more than 1 batch it has to call the WS multiple times. The payment orders in the batch can be of different types. Supported types are displayed in Table 1 in section 1.6.2.

To be able to import batch payments user must be granted appropriate rights.

#### 6.3.1. Interface specification

List of SOAP body request attributes in ImportBatchPaymentOrder WS:

Object type					
ImportBatchPaymentOrderRequestIo					
Extension	AbstractIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
batchPaymentRequestId	Long 19n	0..1	---	Batch payment request ID. This attribute is optional; when specified, myGEMINI will	---

				check whether there is already imported batch payment order with the same request ID for the same client and if yes – then import will fail.	
debitAccountIdentification	AccountIdentificationIo	1..1	---	Debit account identification (account from which all payments in this batch will be made)	---
batchName	String	1..1	---	Name of batch	---
paymentOrder	PaymentOrderIo	1..*	---	List of payment orders to be processed (can be various types)	---

**Object definition 15: ImportBatchPaymentOrderRequestIo object**

List of SOAP body response attributes in ImportBatchPaymentOrder WS:

Object type					
ImportBatchPaymentOrderResponseIo					
Extension	AbstractIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
mygeminiBatchId	Long	0..1	---	Internal ID of batch assigned by my GEMINI	---

**Object definition 16: ImportBatchPaymentOrderResponseIo object**

When ImportBatchPaymentOrderRequestIo is successful, my|GEMINI IBS assigns unique ID to the imported batch and returns its value in ImportBatchPaymentOrderResponseIo, in the mygeminiBatchId attribute; in case connection between client system and my|GEMINI IBS is broken before client system gets response, GetBatchPaymentId WS can be used to get ID that was assigned by my|GEMINI to the imported batch.

Object type					
<u>GetBatchPaymentIdRequestIo</u>					
Main get batch payment ID request object					
Attribute	Type	Oblig.	Values	Comment	ASD Code
batchPaymentRequestId	Long 19n	1..1	---	Batch payment request ID that was specified in the import request	---

**Object definition 17: GetBatchPaymentIdRequestIo object**

Object type					
GetBatchPaymentIdResponseIo		Main get batch payment ID response object			
Attribute	Type	Oblig.	Values	Comment	ASD Code
BatchId	String	0..1	---	Internal ID of batch assigned by myGEMINI	ACCT10

**Object definition 18: GetBatchPaymentIdResponseIo object**

In case specified batch payment request ID is not found for a given client, WS will return a SOAP fault (see chapter "Error handling").

In the GetBatchPaymentId web service the **Nonce** element in security header **is not required!**

### 6.3.2. Examples

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:myg="http://www.mygemini.com/schemas/mygemini" xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
<soapenv:Header>
```

```
<wss:Security>
```

```
<wss:UsernameToken>
```

```
<wss:Username>USERNAME</wss:Username>
```

```
<wss:Password>PASSWORD</wss:Password>
```

```
<wss:Nonce>1111</wss:Nonce>
```

```
</wss:UsernameToken>
```

```
</wss:Security>
```

```
</soapenv:Header>
```

```
<soapenv:Body>
```

```
<myg:ImportBatchPaymentOrderRequestIo>
```

```
<myg:debitAccountIdentification>
```

```
<myg:accountNumber>GE69TB1144836020100001</myg:accountNumber>
```

```
<myg:accountCurrencyCode>GEL</myg:accountCurrencyCode>
```

```
</myg:debitAccountIdentification>
```

```
<myg:batchName>Batch import example</myg:batchName>
```

```
<!--1 or more payment orders:-->
```

```
<myg:paymentOrder xsi:type="myg:TransferToOwnAccountPaymentOrderIo">
```

```
<myg:creditAccount>
```

```
<myg:accountNumber>GE61TB1144845062600001</myg:accountNumber>
```

```
<myg:accountCurrencyCode>GEL</myg:accountCurrencyCode>
```

```
</myg:creditAccount>
```

```
<myg:documentNumber>300000</myg:documentNumber>
```

```
<myg:amount>
```

```
<myg:amount>2</myg:amount>
```

```
<myg:currency>GEL</myg:currency>
```

```
</myg:amount>
```

```
<myg:position>1</myg:position>
```

```
<myg:description>Transfer to Own Account</myg:description>
```

```
</myg:paymentOrder>
```

```

<myg:paymentOrder xsi:type="myg:TransferToOwnAccountPaymentOrderIo">
  <myg:creditAccount>
    <myg:accountNumber>GE86TB1144836120100002</myg:accountNumber>
    <myg:accountCurrencyCode>USD</myg:accountCurrencyCode>
  </myg:creditAccount>
  <myg:documentNumber>300001</myg:documentNumber>
  <myg:amount>
    <myg:amount>2</myg:amount>
    <myg:currency>USD</myg:currency>
  </myg:amount>
  <myg:position>2</myg:position>
  <myg:description>Currency Exchange</myg:description>
</myg:paymentOrder>
<myg:paymentOrder xsi:type="myg:TransferWithinBankPaymentOrderIo">
  <myg:creditAccount>
    <myg:accountNumber>GE88TB0600000003718633</myg:accountNumber>
  </myg:creditAccount>
  <myg:documentNumber>300002</myg:documentNumber>
  <myg:amount>
    <myg:amount>2</myg:amount>
    <myg:currency>GEL</myg:currency>
  </myg:amount>
  <myg:position>3</myg:position>
  <myg:additionalDescription>Transfer within TBC Bank Example</myg:additionalDescription>
  <myg:description>Transfer within TBC Bank</myg:description>
  <myg:beneficiaryName>Some Company Ltd</myg:beneficiaryName>
</myg:paymentOrder>
<myg:paymentOrder xsi:type="myg:TransferToOtherBankNationalCurrencyPaymentOrderIo">
  <myg:creditAccount>
    <myg:accountNumber>GE33BG0000000263255500</myg:accountNumber>
  </myg:creditAccount>
  <myg:documentNumber>300003</myg:documentNumber>
  <myg:amount>
    <myg:amount>2</myg:amount>
    <myg:currency>GEL</myg:currency>
  </myg:amount>
  <myg:position>4</myg:position>
  <myg:additionalDescription>Transfer to Other Bank in NC Example</myg:additionalDescription>
  <myg:description>Transfer to Other Bank in National Currency</myg:description>
  <myg:beneficiaryName>Some Company Ltd</myg:beneficiaryName>
  <myg:beneficiaryTaxCode>100200300</myg:beneficiaryTaxCode>
</myg:paymentOrder>
<myg:paymentOrder xsi:type="myg:TreasuryTransferPaymentOrderIo">
  <myg:documentNumber>300005</myg:documentNumber>
  <myg:amount>
    <myg:amount>2</myg:amount>

```

```

<myg:currency>GEL</myg:currency>
</myg:amount>
<myg:position>5</myg:position>
<myg:additionalDescription>Treasury Transfer Example</myg:additionalDescription>
<myg:treasuryCode>100071003</myg:treasuryCode>
</myg:paymentOrder>
<myg:paymentOrder xsi:type="myg:TreasuryTransferPaymentOrderIo">
  <myg:documentNumber>300006</myg:documentNumber>
  <myg:amount>
    <myg:amount>2</myg:amount>
    <myg:currency>GEL</myg:currency>
  </myg:amount>
  <myg:position>6</myg:position>
  <myg:additionalDescription>Treasury Transfer for other Example</myg:additionalDescription>
  <myg:taxpayerCode>100200300</myg:taxpayerCode>
  <myg:taxpayerName>Other Company Ltd</myg:taxpayerName>
  <myg:treasuryCode>100071003</myg:treasuryCode>
</myg:paymentOrder>
</myg:ImportBatchPaymentOrderRequestIo>
</soapenv:Body>
</soapenv:Envelope>

```

**Example 7: Valid SOAP message for different payment order types in batch payment import**

## 6.4. Common objects used in payment WS

Object type					
PaymentOrderIo					
Extension	AbstractIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
singlePaymentRequestId	Long 19n	0..1	---	Single payment request ID. This attribute is optional; when specified, my GEMINI will check whether there is already imported single payment order with the same request ID for the same client via ImportSinglePaymentOrdersRequestIo or via ImportBatchPaymentOrderRequestIo and if yes – then import will fail.	---
creditAccount	AccountIdentificationIo	0..1	---	Credit account. Is not used for treasury transfers;	---

				for all other transfer types this element is mandatory.	
debitAccount	AccountIdentificationIo	0..1	---	Debit account. When importing single payment order this element is mandatory. When importing batch payment this element should not be used.	---
documentNumber	Long 10n	0..1	---	Document number. Integer numbers up to 2147483647 are accepted. In case external system does not send any value, bank system will generate it automatically based on timestamp.	---
amount	MoneyIo	1..1	---	Amount to transfer from debit account.	---
position	Integer	1..1	---	Position of Payment Order in request. Is mandatory for both single and batch payment orders. For batch payments this value is then returned in GetPaymentOrderStatus so that the calling system is able to match the provided paymentId-s and paymentStatus-es to each payment order sent in a batch. For single payments it is used again to match the paymentOrder data in response to those in request.	---
additionalDescription	String 165S – for transfer to other bank in foreign currency; 255R – for other transfer types.	0..1	---	Additional description. Is not used in transfer to own account orders.	---
description	String 140S – for transfer to other bank in foreign currency;	0..1	---	Description of the payment. Is not used in treasury transfers;	---

	150R – for other transfer types.			for all other transfer types this element is mandatory.	
--	----------------------------------	--	--	---	--

**Object definition 19: PaymentOrderIo object**

Object type					
<u>AccountIdentificationIo</u>					
Extension	AbstractIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
accountNumber	String 34S – for creditAccount in transfer to other bank in foreign currency; 22!S – for all other cases.	1..1	---	Account number in IBAN format (creditAccount in transfer to other bank in foreign currency can have different format)	---
accountCurrencyCode	String 3!A	0..1	---	Account currency code. For debitAccount element this is MANDATORY	---

**Object definition 20: AccountIdentificationIo object**

Object type					
<u>PaymentStatusDataIo</u>					
Extension	AbstractIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
position	Integer	0..1	---	Position of given payment order in Batch import request	
paymentId	String	0..1	---	Payment order ID assigned by my GEMINI	ACCT10
paymentStatus	String	0..1	catalog Transaction Status	Current status of payment order	
errorDetailEN	String	0..1		Detail of error in English language	
errorDetailGE	String	0..1		Detail of error in Georgian language	

**Object definition 21: PaymentStatusDataIo object**

### **6.5. Payment order attribute formats**

Attribute format defines allowed character set and maximal number of characters in the attribute value. Exclamation mark (!) means that value should contain exactly the specified number of characters; e.g.:

- '22!S' – value should contain exactly 22 characters that correspond to 'S' charset;
- '22S' – value can contain up to 22 characters that correspond to 'S' charset;

- '13n.2n' – digits and decimal point: up to 13 digits before point and up to 2 digits after the point.

Format Code	Allowed Charset
n	Digits: 0123456789
A	Capital letters of the Latin alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ
S	Capital letters of the Latin alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ Digits: 0123456789 Characters: / - ? : ( ) . , ' + _ * " ` < > & space  <i>Characters &lt; &gt; &amp; must be represented as: &amp;lt; &amp;gt; &amp;amp;</i>
R	Georgian (Unicode, UTF-8) letters: აბგდეეზთიკლმნოპჟრსტუფკლემზცძწჭხჯჰ Lower case letters of the Latin alphabet: abcdefghijklmnopqrstuvwxyz Capital letters of the Latin alphabet: ABCDEFGHIJKLMNOPQRSTUVWXYZ Digits: 0123456789 Characters: / - ? : ( ) . , ' + space

## 7. Get account movements

This web service interface is used for obtaining account movements data from bank system based on provided filter parameter(s).

Since movements are very frequent business entities in banking environment, it is necessary to prevent long lasting remote requests started by client systems. A simple mechanism is implemented, where bank fills system parameter, which defines maximum number of returned movements in one response (DBI\_PAGING\_SIZE – see Chapter 3 for more information). Such message will also contain information whether all movements were returned or only part of them. This is done by paging objects. Client system is supposed to read paging metadata and send the same request but with different index number of required page. Downloading process of movements will end when client system downloads all pages.

### 7.1. Filter combinations

There are several combinations of values which are considered as **correct**; the rest should return an error code INCORRECT\_INPUT\_DATA.

#### 7.1.1. Get movement by ID

If movement ID parameter is filled, then other filtering parameters must be empty; if not then request will fail and an error code will be returned.

#### 7.1.2. Get movements by last movement timestamp

If last movement timestamp is filled, then other filter parameters must be empty; if not then request will fail and an error code will be returned.

#### 7.1.3. Get movements by other filtering parameters

All filter parameters can be filled except movement ID and last movement timestamp, otherwise the request will fail and an error code will be returned.

## 7.2. Interface specification

The interface for downloading account movements from myGEMINI has the following attributes:

List of SOAP body request attributes in GetAccountMovements WS:

Object type					
<u>GetAccountMovementsRequestIo</u>		Main get account movement request object			
Attribute	Type	Oblig.	Values	Comment	ASD Code
accountMovementFilterIo	AccountMovementFilterIo	1..1	---	---	---
Object type					
<u>AccountMovementFilterIo</u>		Account movement filter object			
Extension		BaseFilterIo			
Attribute	Type	Oblig.	Values	Comment	ASD Code
accountNumber	String	0..1	---	Account number	ACCT32
accountCurrencyCode	String	0..1	---	Account currency; is mandatory when accountNumber is specified	ACC4
periodFrom	DateTime	0..1	---	Period interval "from" (inclusive)	ACCT3
periodTo	DateTime	0..1	---	Period interval "to" (inclusive)	ACCT3
movementId	String	0..1	---	Movement ID	ACCT30
lastMovementTimestamp	DateTime	0..1	---	Last movement timestamp – system will return movements that have higher timestamp (e.g. if 2013-10-25T15:23:12.000 is specified, then system will return movements that were created or updated after that time)	---

**Object definition 22: GetAccountMovementRequestIo object**

List of SOAP body response attributes in GetAccountMovements WS:

Object type					
<u>GetAccountMovementsResponseIo</u>		Main get account movement response object			
Attribute	Type	Oblig.	Values	Comment	ASD Code
accountMovement	AccountMovementDetailIo	1..*	---	---	---
Result	BaseQueryResultIo	0..1	---	---	---

Object type					
AccountMovementDetailIo	Account movement detail object				
Extension	IndexIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
movementId	String	1..1	---	Movement ID assigned by CBS	ACCT30
paymentId	String	0..1	---	ID of payment order based on which this movement was generated (assigned by my GEMINI)	ACCT10
externalPaymentId	String	1..1	---	ID of payment order based on which this movement was generated (assigned by CBS)	ACCT19
debitCredit	Integer	1..1	0 = Debit 1 = Credit	Operation type: is debit or credit operation	ACCT5
valueDate	DateTime	1..1	---	Operation date	ACCT3
Description	String	0..1	---	Description	ACCT24
Amount	MoneyIo	1..1	---	Amount and currency of movement	ACCT6, ACC4
accountNumber	String	1..1	---	Account number	ACCT32
accountName	String	0..1	---	Account name	ACCT33
additionalInformation	String	0..1	---	Additional information (partner info)	ACCT49
documentDate	DateTime	0..1	---	Document date	ACCT4
documentNumber	String	0..1	---	Document number	ACCT43
partnerAccountNumber	String	0..1	---	Partner's account number	ACCT13
partnerName	String	0..1	---	Partner's name	ACCT14
partnerTaxCode	String	0..1	---	Partner's tax code	ACCT38
partnerBankCode	String	0..1	---	Partner's bank code	ACCT15
partnerBank	String	0..1	---	Partner's bank	ACCT37
intermediaryBankCode	String	0..1	---	Intermediary bank code	ACCT45
intermediaryBank	String	0..1	---	Intermediary bank	ACCT46
chargeDetail	String	0..1	catalog FeePayment Method	Charge details	ACCT42
taxpayerCode	String	0..1	---	Taxpayer code	ACCT41
taxpayerName	String	0..1	---	Taxpayer name	ACCT40
treasuryCode	String	0..1	---	Treasury code	ACCT39
operationCode	String	0..1	---	Operation code	ACCT44

additionalDescription	String	0..1	---	Additional description	ACCT31
exchangeRate	String	0..1	---	Exchange rate (for currency exchange operations)	ACCT59
partnerPersonalNumber	String	0..1	---	Partner's personal number (for cash operations)	ACCT52
partnerDocumentType	String	0..1	catalog Identification Document Type	Partner's ID document type (for cash operations)	ACCT53
partnerDocumentNumber	String	0..1	---	Partner's ID document number (for cash operations)	ACCT54
parentExternalPaymentId	String	0..1	---	Payment order ID of parent movement assigned by CBS; e.g. this attribute can be filled for charge movements to link them with original payment movements	ACCT48
statusCode	String	0..1	catalog CBS Transaction Status	Authorization status code in CBS ( <b>Client system can guess that payment order is fully accepted and processed by Bank only after movement gets Authorized status</b> )	ACCT28
transactionType	String	0..1	catalog CBS Transaction Subtype	Transaction type	ACCT8

**Object definition 23: GetAccountMovementsResponseIo object**

## 7.3. Examples

### 7.3.1. Get account movement by ID

Below is a typical request message asking for account movement data by its unique ID. Request consists of security header and body with filled filter object.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:myg="http://www.mygemini.com/schemas/mygemini" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <soapenv:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>USERNAME</wsse:Username>
        <wsse:Password>PASSWORD</wsse:Password>
        <wsse:Nonce>1111</wsse:Nonce>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <myg:GetAccountMovementsRequestIo>
      <myg:accountMovementFilterIo>
        <myg:movementId>001157711468.1</myg:movementId>
      </myg:accountMovementFilterIo>
    </myg:GetAccountMovementsRequestIo>
  </soapenv:Body>
</soapenv:Envelope>
```

#### Example 8: Get movement by ID request

A typical response to previous request follows. It contains account movement detail object.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:GetAccountMovementsResponseIo xmlns:ns2="http://www.mygemini.com/schemas/mygemini">
      <ns2:accountMovement>
        <ns2:movementId>001157711468.1</ns2:movementId>
        <ns2:externalPaymentId>821912922</ns2:externalPaymentId>
        <ns2:debitCredit>0</ns2:debitCredit>
        <ns2:valueDate>2014-02-22T00:00:00+04:00</ns2:valueDate>
        <ns2:description>HEALTH INSURANCE</ns2:description>
        <ns2:amount>
          <ns2:amount>34</ns2:amount>
          <ns2:currency>USD</ns2:currency>
        </ns2:amount>
        <ns2:accountNumber>GE05TB0000000077720096</ns2:accountNumber>
        <ns2:accountName>my account</ns2:accountName>
        <ns2:additionalInformation>HICO, BIDCCUHH, ABAX20000000415</ns2:additionalInformation>
      </ns2:accountMovement>
    </ns2:GetAccountMovementsResponseIo>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```

<ns2:documentDate>2014-02-22T00:00:00+04:00</ns2:documentDate>
<ns2:partnerAccountNumber>ABAX20000000415</ns2:partnerAccountNumber>
<ns2:partnerName>HICO</ns2:partnerName>
<ns2:partnerBankCode>BIDCCUHH</ns2:partnerBankCode>
<ns2:partnerBank>TBBA</ns2:partnerBank>
<ns2:intermediaryBankCode>CITIUS33</ns2:intermediaryBankCode>
<ns2:intermediaryBank>CITIBANK N.A.</ns2:intermediaryBank>
<ns2:operationCode>*IBT*</ns2:operationCode>
<ns2:additionalDescription>MOVEMENT EXAMPLE</ns2:additionalDescription>
<ns2:statusCode>1</ns2:statusCode>
<ns2:transactionType>30</ns2:transactionType>
</ns2:accountMovement>
</ns2:GetAccountMovementsResponseIo>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

#### Example 9: Get movement by ID response

### 7.3.2. Get account movements by date range with paging

First request does not define any paging data, because it is not known how many results will be returned. There is used implicit parameter defined by bank to determine maximum page size.

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:myg="http://www.mygemini.com/schemas/mygemini" xmlns:wss="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <soapenv:Header>
    <wss:Security>
      <wss:UsernameToken>
        <wss:Username>USERNAME</wss:Username>
        <wss:Password>PASSWORD</wss:Password>
        <wss:Nonce>1111</wss:Nonce>
      </wss:UsernameToken>
    </wss:Security>
  </soapenv:Header>
  <soapenv:Body>
    <myg:GetAccountMovementsRequestIo>
      <myg:accountMovementFilterIo>
        <myg:periodFrom>2012-10-25T15:23:12.000</myg:periodFrom>
        <myg:periodTo>2013-10-25T15:23:12.000</myg:periodTo>
      </myg:accountMovementFilterIo>
    </myg:GetAccountMovementsRequestIo>
  </soapenv:Body>
</soapenv:Envelope>

```

#### Example 10: Get movements by date range request

Response returns data block with paging information. System automatically returns first page, which has index equal to zero. Since paging information in response contains total count and actual page size, calling system can easily count how many pages are needed to download all items matching the filter criteria.

```

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:GetAccountMovementsResponseIo xmlns:ns2="http://www.mygemini.com/schemas/mygemini">
      <ns2:result>
        <ns2:pager>
          <ns2:pageIndex>0</ns2:pageIndex>
          <ns2:pageSize>50</ns2:pageSize>
        </ns2:pager>
        <ns2:totalCount>64</ns2:totalCount>
      </ns2:result>
      <ns2:accountMovement>
        ... ..
      </ns2:accountMovement>
      <ns2:accountMovement>
        ... ..
      </ns2:accountMovement>
      <ns2:accountMovement>
        ... ..
      </ns2:accountMovement>
    </ns2:GetAccountMovementsResponseIo>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

**Example 11: Get movements by date range response - first page**

Calling system should send another request with page index equal to 1 to obtain the rest of the data (2nd page in this example).

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:myg="http://www.mygemini.com/schemas/mygemini" xmlns:wsse="http://docs.oasis-
  open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <soapenv:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>USERNAME</wsse:Username>
        <wsse:Password>PASSWORD</wsse:Password>
        <wsse:Nonce>1111</wsse:Nonce>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <myg:GetAccountMovementsRequestIo>
      <myg:accountMovementFilterIo>
        <myg:pager>
          <myg:pageIndex>1</myg:pageIndex>

```

```

<myg:pageSize>50</myg:pageSize>
</myg:pager>
<myg:periodFrom>2012-10-25T15:23:12.000</myg:periodFrom>
<myg:periodTo>2013-10-25T15:23:12.000</myg:periodTo>
</myg:accountMovementFilterIo>
</myg:GetAccountMovementsRequestIo>
</soapenv:Body>
</soapenv:Envelope>

```

**Example 12: Get second page of movements by date range request**

System produces similar response to the first one with the specified page index and the rest of the requested account movements.

```

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:GetAccountMovementsResponseIo xmlns:ns2="http://www.mygemini.com/schemas/mygemini">
      <ns2:result>
        <ns2:pager>
          <ns2:pageIndex>1</ns2:pageIndex>
          <ns2:pageSize>50</ns2:pageSize>
        </ns2:pager>
        <ns2:totalCount>64</ns2:totalCount>
      </ns2:result>
      <ns2:accountMovement>
        ... ..
      </ns2:accountMovement>
      <ns2:accountMovement>
        ... ..
      </ns2:accountMovement>
      <ns2:accountMovement>
        ... ..
      </ns2:accountMovement>
    </ns2:GetAccountMovementsResponseIo>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

**Example 13: Get second page of movements by date range response**

## 8. Postbox messages

When a change in client account movements' data occurs that the client should be informed about - the Postbox functionality comes in play, where bank system will place messages containing information about the occurred changes. Client system can periodically request messages by using the `getPostboxMessage` functionality and after receiving the information should acknowledge bank system that the messages were downloaded successfully. This allows the DBI channel to filter which messages the client has already received and which should be provided to him. Bank system will remove acknowledged messages from the Postbox and archive them. Bank system can remove unread and unacknowledged messages from Postbox after a predefined time period is expired.

In all requests of the Postbox interface the **Nonce** element in security header **is not required!**

### 8.1. Interface specification

List of SOAP body request attributes in `GetMessagesFromPostbox WS`:

Object type					
<u>GetPostboxMessagesRequestIo</u>		Main get postbox messages request object			
Attribute	Type	Oblig.	Values	Comment	ASD Code
messageType	String	0..1	catalog Postbox Message Type	Message type	CSIMSG03

#### Object definition 24: GetPostboxMessagesRequestIo object

There are 2 message types available:

SIMPLE_MESSAGE	Messages of this type contain only text in the <code>messageText</code> element. Usually it will be only short message from bank.
MOVEMENT_MESSAGE	Messages of this type contain data about changes in movements data (e.g.: attributes of some movement were updated in the bank system, some movement was deleted in the bank system, etc.)

List of SOAP body response attributes in `GetMessagesFromPostbox WS`:

Object type					
<u>GetPostboxMessagesResponseIo</u>		Main get postbox messages response object			
Attribute	Type	Oblig.	Values	Comment	ASD Code
Messages	PostboxMessageIo	1..*	---	List of postbox messages	---
Object type					
<u>PostboxMessageIo</u>		Message from postbox			
<b>Extension</b>		AbstractIo			
Attribute	Type	Oblig.	Values	Comment	ASD Code
messageId	Long	1..1	---	Unique message ID	CSIMSG01
messageText	String	1..1	---	Textual message	CSIMSG02
messageType	String	1..1	Catalog Postbox Message Type	Message type	CSIMSG03
messageStatus	String	1..1	Catalog	Message status	CSIMSG04

			Postbox Message Status	indicating whether it was already sent to client system	
additionalAttributes	AdditionalAttributeIo	0..*	---	Additional attributes for movement messages; simple messages have no add.attributes.	CSIMSG05

**Object definition 25: GetPostboxMessagesResponseIo object**

additionalAttributes of messages of MOVEMENT\_MESSAGE type may contain the following parameters:

<name> element	<value> element
UPDATED	Movement ID
DELETED	Movement ID
CREATEDFOREARLIERDATE	Movement ID
DATE	Date (yyyy-MM-dd)
ACCOUNT_IBAN	Account IBAN number
ACCOUNT_CURRENCY	Account Currency
FROM_DATE	1st date of the date range (yyyy-MM-dd)
TO_DATE	Last date of the date range (yyyy-MM-dd)

Postbox may return 5 subtypes of messages of MOVEMENT\_MESSAGE type:

- 1) In case one or more attributes of some movement are updated in the bank system (e.g. some payment orders need to be authorized by bank employee; movement generated based on such payment order gets 'Waiting for authorization' status at first; and after bank employee authorizes the payment order, the movement gets 'Authorized' status and movement's attribute 'statusCode' is updated) - the message will contain additionalAttributes like this:

```
<ns2:additionalAttributes>
  <ns2:name>UPDATED</ns2:name>
  <ns2:value>001157711468.1</ns2:value>
</ns2:additionalAttributes>
```

Client system can redownload updated movement based on Movement ID - see movementId filter in GetAccountMovements WS.

- 2) In case some movement is deleted in the bank system - the message will contain additionalAttributes like this:

```
<ns2:additionalAttributes>
  <ns2:name>DELETED</ns2:name>
  <ns2:value>001157711468.1</ns2:value>
</ns2:additionalAttributes>
```

Client system can remove deleted movement from the own database based on Movement ID.

- 3) In case there is created a movement that has valueDate attribute earlier than the current date (e.g. valueDate is 2014-02-22, while current date is 2014-02-23) - the message will contain additionalAttributes like this:

```
<ns2:additionalAttributes>
  <ns2:name>CREATEDFOREARLIERDATE</ns2:name>
  <ns2:value>001157711468.1</ns2:value>
</ns2:additionalAttributes>
```

Client system can download the movement data based on Movement ID - see movementId filter in GetAccountMovements WS.

- 4) In case movement data of some account is updated for particular date - then the message will contain additionalAttributes like this:

```
<ns2:additionalAttributes>
  <ns2:name>DATE</ns2:name>
  <ns2:value>2014-02-22</ns2:value>
</ns2:additionalAttributes>
<ns2:additionalAttributes>
  <ns2:name>ACCOUNT_IBAN</ns2:name>
  <ns2:value>GE75TB000000077720097</ns2:value>
</ns2:additionalAttributes>
<ns2:additionalAttributes>
  <ns2:name>ACCOUNT_CURRENCY</ns2:name>
  <ns2:value>GEL</ns2:value>
</ns2:additionalAttributes>
```

This means that all movements that had matching accountNumber, amount.currency and valueDate, were deleted in the bank system and then account's movement data for that date was recreated.

Client system can remove old data from the own database and redownload updated data using accountNumber, accountCurrencyCode, periodFrom, periodTo filters in GetAccountMovements WS.

- 5) In case movement data of some account is updated for some date range - then the message will contain additionalAttributes like this:

```
<ns2:additionalAttributes>
  <ns2:name>FROM_DATE</ns2:name>
  <ns2:value>2014-02-22</ns2:value>
</ns2:additionalAttributes>
<ns2:additionalAttributes>
  <ns2:name>TO_DATE</ns2:name>
  <ns2:value>2014-02-23</ns2:value>
</ns2:additionalAttributes>
<ns2:additionalAttributes>
  <ns2:name>ACCOUNT_IBAN</ns2:name>
```

```

<ns2:value>GE75TB0000000077720097</ns2:value>
</ns2:additionalAttributes>
<ns2:additionalAttributes>
  <ns2:name>ACCOUNT_CURRENCY</ns2:name>
  <ns2:value>GEL</ns2:value>
</ns2:additionalAttributes>

```

This means that all movements that had matching accountNumber, amount.currency and valueDate, were deleted in the bank system and then account's movement data for that date range was recreated.

Client system can remove old data from the own database and redownload updated data using accountNumber, accountCurrencyCode, periodFrom, periodTo filters in GetAccountMovements WS.

List of SOAP body request attributes in AcknowledgePostboxMessages WS:

Object type					
<u>PostboxAcknowledgementRequestIo</u>		Main postbox acknowledgement request object			
Attribute	Type	Oblig.	Values	Comment	ASD Code
messageIds	Long	1..*	---	Postbox message IDs	CSIMSG01

**Object definition 26: PostboxAcknowledgementRequestIo object**

List of SOAP body response attributes in AcknowledgePostboxMessages WS:

Object type					
<u>PostboxAcknowledgementResponseIo</u>		Main postbox acknowledgement response object			
Attribute	Type	Oblig.	Values	Comment	ASD Code
responseText	String	1..1	---	Message with acknowledgement result	---

**Object definition 27: PostboxAcknowledgementResponseIo object**

## 8.2. Examples

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:myg="http://www.mygemini.com/schemas/mygemini" xmlns:wss="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <soapenv:Header>
    <wss:Security>
      <wss:UsernameToken>
        <wss:Username>USERNAME</wss:Username>
        <wss:Password>PASSWORD</wss:Password>
      </wss:UsernameToken>
    </wss:Security>
  </soapenv:Header>
  <soapenv:Body>
    <myg:GetPostboxMessagesRequestIo>

```

```

<myg:messageType>MOVEMENT_MESSAGE</myg:messageType>
</myg:GetPostboxMessagesRequestIo>
</soapenv:Body>
</soapenv:Envelope>

```

**Example 14: Example of getPostboxMessage request**

```

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <ns2:GetPostboxMessagesResponseIo xmlns:ns2="http://www.mygemini.com/schemas/mygemini">
      <ns2:messages>
        <ns2:messageId>1626504</ns2:messageId>
        <ns2:messageText>The following movements have been deleted:</ns2:messageText>
        <ns2:messageType>MOVEMENT_MESSAGE</ns2:messageType>
        <ns2:messageStatus>WAITING_FOR_ACKNOWLEDGE</ns2:messageStatus>
        <ns2:additionalAttributes>
          <ns2:name>DELETED</ns2:name>
          <ns2:value>001157711467.1</ns2:value>
        </ns2:additionalAttributes>
        <ns2:additionalAttributes>
          <ns2:name>DELETED</ns2:name>
          <ns2:value>001157711467.2</ns2:value>
        </ns2:additionalAttributes>
        <ns2:additionalAttributes>
          <ns2:name>DELETED</ns2:name>
          <ns2:value>001157711468.1</ns2:value>
        </ns2:additionalAttributes>
      </ns2:messages>
    </ns2:GetPostboxMessagesResponseIo>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

**Example 15: Example of getPostboxMessage response**

```

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:myg="http://www.mygemini.com/schemas/mygemini" xmlns:wsse="http://docs.oasis-
  open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <soapenv:Header>
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>USERNAME</wsse:Username>
        <wsse:Password>PASSWORD</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <myg:PostboxAcknowledgementRequestIo>

```

```

<myg:messageIds>1626504</myg:messageIds>
<myg:messageIds>1626509</myg:messageIds>
<myg:messageIds>1626510</myg:messageIds>
<myg:messageIds>1626512</myg:messageIds>
</myg:PostboxAcknowledgementRequestIo>
</soapenv:Body>
</soapenv:Envelope>

```

**Example 16: Example of PostboxAcknowledgement request**

## 9. Validations

Postel's Law says: "Be conservative in what you do; be liberal in what you accept from others."

DBI module adopts the law in terms of XSDs, which are used in remote communication. It means DBI validates request and response very strictly based on XSDs, but XSDs are written very liberally.

my|GEMINI validates every request and response automatically, in case of validation error system will return generated SOAP fault response.

```

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <SOAP-ENV:Fault>
      <faultcode>SOAP-ENV:Client</faultcode>
      <faultstring xml:lang="en">Validation error</faultstring>
      <detail>
        <spring-ws:ValidationError xmlns:spring-ws="http://springframework.org/spring-ws">cvc-
complex-type.2.4.b: The content of element 'myg:GetAccountMovementsRequestIo' is not complete. One
of '{"http://www.mygemini.com/schemas/mygemini":accountMovementFilterIo}' is expected.</spring-
ws:ValidationError>
      </detail>
    </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

```

**Example 17: Automatic schema validation - validation error**

## 10. Error handling

This chapter defines error codes and their meaning. All errors are propagated to client systems as SOAP fault messages defined in corresponding standard.

Error code	Description
Client	Error code generated by automatic validation systems (e.g. by schema validator). Try to check documentation and examples of usage.
GENERAL_ERROR	This is a common error code used for unexpected error states. Bank should be contacted for help in most cases.
CBS_GENERAL_ERROR	This error code is returned in case there is a problem in core banking system. Bank should be contacted for help.

INCORRECT_INPUT_DATA	Incorrect input data error code means that incorrect data was sent to the system. Try to check documentation and examples of usage.
USER_IS_NOT_ACTIVE	This error code is returned in case user is not active and therefore cannot use the DBI channel.
USER_IS_BLOCKED	This error code is returned in case user is in a blocked state (e.g. is temporarily blocked due to too many unsuccessful login attempts).
CREDENTIALS_MUST_BE_CHANGED	This error code is returned in case user must change password.
SECURITY_POLICIES_NOT_MET	During password change this error can be returned in case new password does not match the criteria specified by bank. This error can be returned also in case client is required to use digital certificate, but tries to establish connection without it.
INCORRECT_CREDENTIALS	This error code is returned in case incorrect user name or password is provided and authentication fails.
OTP_FAILED	This error code is returned in case provided OTP is invalid, is expired or is not supplied at all.
NOT_AUTHORIZED	This error code is returned in case user has no right to perform requested operation (e.g. to import payment order).
DUPLICATED_SINGLE_PAYMENT_REQUEST	This error code is returned in case web service ImportSinglePaymentOrders or web service ImportBatchPaymentOrder is called and there is already imported single payment order with the same request ID for the same client.
DUPLICATED_BATCH_PAYMENT_REQUEST	This error code is returned in case web service ImportBatchPaymentOrder is called and there is already imported batch payment order with the same request ID for the same client.
SINGLE_PAYMENT_REQUEST_NOT_FOUND	This error code is returned in case web service GetSinglePaymentId is called and specified single payment request ID is not found for a given client.
BATCH_PAYMENT_REQUEST_NOT_FOUND	This error code is returned in case web service GetBatchPaymentId is called and specified batch payment request ID is not found for a given client.

**Table 2: Error codes exposed to client systems**

## 11. Common objects used in interfaces

Object type					
<u>MoneyIo</u>	Wrapper object for money definition - value and currency together.				
<b>Extension</b>	AbstractIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
Amount	Decimal 13n.2n	1..1	---	Numeric representation of amount	---
Currency	String 3!A	1..1	---	Currency ISO code	---

**Object definition 28: MoneyIo object**

Object type					
<u>AdditionalAttributeIo</u>	Additional attribute serves as non-invasive mechanism how to use more attributes in XML schema				
<b>Extension</b>	AbstractIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
Name	String	1..1	---	Attribute name	---
Value	String	1..1	---	Attribute value	---

**Object definition 29: AdditionalAttributeIo object**

Object type					
<u>BaseQueryResultIo</u>	Base object for query results				
<b>Extension</b>	AbstractIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
Pager	BasePagerIo	0..1	---	Pager object	---
totalCount	Integer	1..1	---	Total count of elements on all pages together	---

**Object definition 30: BaseQueryResultIo object**

Object type					
<u>BasePagerIo</u>	Object which holds information about paging.				
<b>Extension</b>	AbstractIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
pageIndex	Integer	1..1	---	Index of page starting from zero	---
pageSize	Integer	1..1	---	Number of elements on 1 page; specifies how many entries can be returned in 1 response	---

**Object definition 31: BasePagerIo object**

Object type					
<u>BaseFilterIo</u>	Predecessor for all filters. Holds paging wrapper object and additional attributes.				
<b>Extension</b>	AbstractIo				
Attribute	Type	Oblig.	Values	Comment	ASD Code
Pager	BasePagerIo	0..1	---	Wrapper object used for paging	---
additionalAttributes	AdditionalAttributeIo	0..*	---	List of additional attributes	---

**Object definition 32: BaseFilterIo object**

Object type	
<u>AbstractIo</u>	Abstract predecessor for all interface objects.

**Object definition 33: AbstractIo object**

### ***11.1. Date formats***

**DateTime** – in **requests** sent to DBI channel DateTime must be in format:

yyyy-MM-dd'T'HH:mm:ss.SSS  
 e.g.: 2014-02-22T15:23:12.000

**DateTime** – in **responses** received from DBI channel DateTime is in format:

yyyy-MM-dd'T'HH:mm:ssZ  
 e.g.: 2014-02-22T00:00:00+04:00

### ***11.2. WSDL & XSD***

See WSDL and XSD files in the zip file attached herein:



DBI\_WSDL&XSD.zip

### ***11.3. Abbreviations***

CBS – Core Banking System  
 IBS – Internet Banking System  
 DBI – Direct Banking Interface  
 WS – Web Service

### ***11.4. Examples of HTTP requests***

See examples of making of HTTP requests in the zip file attached herein:



DBI\_HTTPRequests.zip

## 12. Development guide for .NET

### 12.1. Adding SOAP Security header in C#

According to chapter 5 the SOAP request must contain security header. The current version of the WSDL does not specify the header, thus the code for it is not present in classes generated by the wsdl.exe.

For each service you must declare the Security class as following:

```

/// <remarks/>
[System.CodeDom.Compiler.GeneratedCodeAttribute("wsdl", "4.0.30319.33440")]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute("code")]
[System.Web.Services.WebServiceBindingAttribute(Name="PaymentServiceBinding",
Namespace="http://www.mygemini.com/schemas/mygemini")]
[System.Xml.Serialization.XmlIncludeAttribute(typeof(AbstractIo))]
public partial class PaymentService : System.Web.Services.Protocols.SoapHttpClientProtocol
{
    private System.Threading.SendOrPostCallback ImportSinglePaymentOrdersOperationCompleted;

    private System.Threading.SendOrPostCallback ImportBatchPaymentOrderOperationCompleted;

    private System.Threading.SendOrPostCallback GetPaymentOrderStatusOperationCompleted;

    // BEGIN manual addition for Security header
    [XmlRoot(Namespace = "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd")]
    public partial class Security : SoapHeader
    {
        public UsernameToken UsernameToken { get; set; }
    }

    public partial class UsernameToken
    {
        public string Username { get; set; }
        public string Password { get; set; }
        public string Nonce { get; set; }
    }

    public Security secHeader;
    // END manual addition for Security header

```

Then for each method you have to add the SoapHeader decorator:

```

/// <remarks/>
[System.Web.Services.Protocols.SoapDocumentMethodAttribute("http://www.mygemini.com/schemas
/mygemini/ImportSinglePaymentOrders",
Use=System.Web.Services.Description.SoapBindingUse.Literal,
ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Bare)]
[return:
System.Xml.Serialization.XmlArrayAttribute("ImportSinglePaymentOrdersResponseIo",
Namespace="http://www.mygemini.com/schemas/mygemini")]
[return: System.Xml.Serialization.XmlArrayItemAttribute("PaymentOrdersResults",
IsNullable=false)]
[SoapHeader("secHeader", Direction = SoapHeaderDirection.In)]
public PaymentOrderResultIo[]
ImportSinglePaymentOrders([System.Xml.Serialization.XmlArrayAttribute(Namespace =

```

```
"http://www.mygemini.com/schemas/mygemini")]
[System.Xml.Serialization.XmlArrayItemAttribute("singlePaymentOrder", IsNullable = false)]
PaymentOrderIo[] ImportSinglePaymentOrdersRequestIo)
{
    object[] results = this.Invoke("ImportSinglePaymentOrders", new object[] {
        ImportSinglePaymentOrdersRequestIo});
    return ((PaymentOrderResultIo[])(results[0]));
}
```

Basically, you can declare the class anywhere in your project - just the instance must be visible for the decorator.

Finally, sample usage of the ImportSinglePaymentOrders :

```
class Program
{
    static void Main(string[] args)
    {
        var ps = new PaymentService();
        addSecurityHeader(ps);
        ps.Url = "https://test.tbconline.ge/dbi/dbiService";
        var paymentOrder = createSamplePaymentOrder();
        var paymentOrders = new TransferToOwnAccountPaymentOrderIo[1] { paymentOrder };
        var paymentOrderResult = ps.ImportSinglePaymentOrders(paymentOrders);
        System.Diagnostics.Debug.WriteLine("Resulting paymentId: {0}",
            paymentOrderResult[0].paymentId);
    }

    private static TransferToOwnAccountPaymentOrderIo createSamplePaymentOrder()
    {
        var paymentOrder = new TransferToOwnAccountPaymentOrderIo();
        paymentOrder.creditAccount = new AccountIdentificationIo();
        paymentOrder.creditAccount.accountNumber = "GE86TB1144836120100002";
        paymentOrder.creditAccount.accountCurrencyCode = "USD";
        paymentOrder.debitAccount = new AccountIdentificationIo();
        paymentOrder.debitAccount.accountNumber = "GE69TB1144836020100001";
        paymentOrder.debitAccount.accountCurrencyCode = "GEL";
        paymentOrder.amount = new MoneyIo();
        paymentOrder.amount.amount = 1;
        paymentOrder.amount.currency = "USD";
        paymentOrder.description = "my best description";
        return paymentOrder;
    }

    private static void addSecurityHeader(PaymentService ps)
    {
        ps.secHeader = new PaymentService.Security();
        ps.secHeader.UsernameToken = new PaymentService.UsernameToken();
        ps.secHeader.UsernameToken.Username = "USERNAME";
        ps.secHeader.UsernameToken.Password = "PASSWORD";
        ps.secHeader.UsernameToken.Nonce = "1111";
    }
}
```

## 12.2. Adding SOAP Security header in Visual Basic

According to chapter 5 the SOAP request must contain security header. The current version of the WSDL does not specify the header, thus the code for it is not present in classes generated by the wsdl.exe.

For each service you must declare the Security class as following:

```
'''<remarks/>
<System.CodeDom.Compiler.GeneratedCodeAttribute("wsdl", "4.0.30319.33440"), _
System.Diagnostics.DebuggerStepThroughAttribute(), _
System.ComponentModel.DesignerCategoryAttribute("code"), _
System.Web.Services.WebServiceBindingAttribute(Name="PaymentServiceBinding",
[Namespace]="http://www.mygemini.com/schemas/mygemini"), _
System.Xml.Serialization.XmlIncludeAttribute(GetType(AbstractIo))> _
Partial Public Class PaymentService
    Inherits System.Web.Services.Protocols.SoapHttpClientProtocol

    Private ImportSinglePaymentOrdersOperationCompleted As
System.Threading.SendOrPostCallback

    Private ImportBatchPaymentOrderOperationCompleted As
System.Threading.SendOrPostCallback

    Private GetPaymentOrderStatusOperationCompleted As System.Threading.SendOrPostCallback

    'BEGIN manual addition for Security header
    <XmlRoot(Namespace="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd")> _
    Partial Public Class Security
        Inherits SoapHeader

        Public Property UsernameToken As UsernameToken
    End Class

    Partial Public Class UsernameToken
        Public Property Username As String
        Public Property Password As String
        Public Property Nonce As String
    End Class

    Public Property secHeader As Security
    'END manual addition for Security header
```

Then for each method you have to add the SoapHeader decorator:

```
'''<remarks/>
<System.Web.Services.Protocols.SoapDocumentMethodAttribute("http://www.mygemini.com/schemas
/mygemini/ImportSinglePaymentOrders",
Use:=System.Web.Services.Description.SoapBindingUse.Literal,
ParameterStyle:=System.Web.Services.Protocols.SoapParameterStyle.Bare)> _
    <SoapHeader("secHeader", Direction:=SoapHeaderDirection.In)> _
    Public Function
ImportSinglePaymentOrders(<System.Xml.Serialization.XmlArrayAttribute([Namespace]="http://
www.mygemini.com/schemas/mygemini"),
System.Xml.Serialization.XmlArrayItemAttribute("singlePaymentOrder", IsNullable:=False)>
ByVal ImportSinglePaymentOrdersRequestIo() As PaymentOrderIo) As
<System.Xml.Serialization.XmlArrayAttribute("ImportSinglePaymentOrdersResponseIo",
[Namespace]="http://www.mygemini.com/schemas/mygemini"),
```

```
System.Xml.Serialization.XmlArrayItemAttribute("PaymentOrdersResults", IsNullable:=False)>
PaymentOrderResultIo()
    Dim results() As Object = Me.Invoke("ImportSinglePaymentOrders", New Object()
{ImportSinglePaymentOrdersRequestIo})
    Return CType(results(0), PaymentOrderResultIo())
End Function
```

Basically, you can declare the class anywhere in your project - just the instance must be visible for the decorator.

Finally, sample usage of the ImportSinglePaymentOrders:

Module Module1

```
Sub Main()
    Dim ps As New PaymentService
    AddSecurityHeader(ps)
    ps.Url = "https://test.tbconline.ge/dbi/dbiService"
    Dim paymentOrder = CreateSamplePaymentOrder()
    Dim paymentOrders = {paymentOrder}
    ps.ImportSinglePaymentOrders(paymentOrders)
End Sub

Function CreateSamplePaymentOrder() As TransferToOwnAccountPaymentOrderIo
    CreateSamplePaymentOrder = New TransferToOwnAccountPaymentOrderIo
    CreateSamplePaymentOrder.creditAccount = New AccountIdentificationIo()
    CreateSamplePaymentOrder.creditAccount.accountNumber = "GE86TB1144836120100002"
    CreateSamplePaymentOrder.creditAccount.accountCurrencyCode = "USD"
    CreateSamplePaymentOrder.debitAccount = New AccountIdentificationIo()
    CreateSamplePaymentOrder.debitAccount.accountNumber = "GE69TB1144836020100001"
    CreateSamplePaymentOrder.debitAccount.accountCurrencyCode = "GEL"
    CreateSamplePaymentOrder.amount = New MoneyIo()
    CreateSamplePaymentOrder.amount.amount = 1
    CreateSamplePaymentOrder.amount.currency = "USD"
    CreateSamplePaymentOrder.description = ".net test VB"
End Function

Sub AddSecurityHeader(ps As PaymentService)
    ps.secHeader = New PaymentService.Security
    ps.secHeader.UsernameToken = New PaymentService.UsernameToken()
    ps.secHeader.UsernameToken.Username = "USERNAME"
    ps.secHeader.UsernameToken.Password = "PASSWORD"
    ps.secHeader.UsernameToken.Nonce = "1111"
End Sub

End Module
```

## 12.3. Calling DBI WS from Excel

For calling DBI WS from Excel, it is first needed to create DLL in Visual Basic with WS client. This DLL must be then registered on client machine and finally this DLL must be referenced in VBA in Excel sheet with macros.

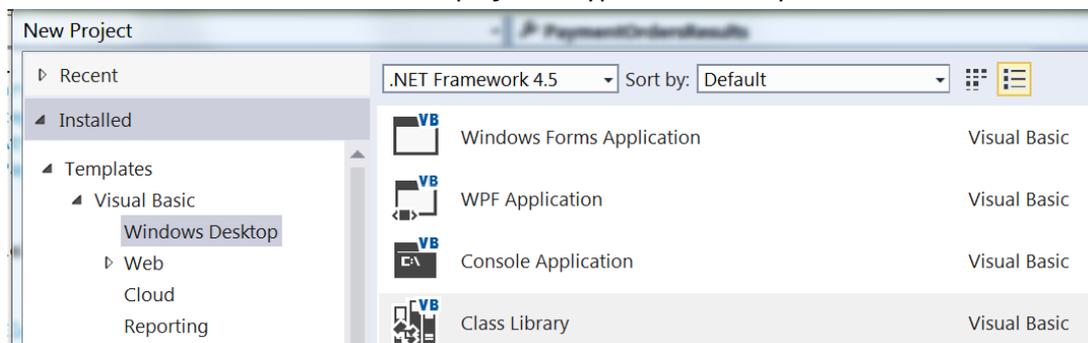
Security protocol TLS 1.2 will be required by TBC DBI server; to fulfill this, these prerequisites must be filled for client machines:

- 1) Clients need to use at least Windows 7 (tested also on Windows 8 and 10)
- 2) Installed Microsoft .NET Framework 4.5 (or newer)

Clients can have 32bit and also 64bit excel, it doesn't matter.

### 12.3.1. Creating WS client DLL in Visual Basic

- 1) In Microsoft Visual Studio create new project of type "Class library"



- 2) Generate WS client from wsdl and xsd:

```
c:\Program Files (x86)\Microsoft SDKs\Windows\v8.1A\bin\NETFX 4.5.1
Tools\wsdl.exe /1:VB PaymentEIService.wsdl PaymentIOTypes.xsd
PaymentIOMessages.xsd
```

- This generates paymentService.vb with VB code. If in VB classes these are missing: ImportSinglePaymentOrdersRequestIo and ImportSinglePaymentOrdersResponseIo, then modify PaymentIOMessages.xsd to generate them:

#### original:

```
<xsd:element name="ImportSinglePaymentOrdersRequestIo">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="singlePaymentOrder" type="PaymentOrderIo" minOccurs="1" maxOccurs="unbounded" >
        <xsd:annotation>
          <xsd:documentation>List of all single payment orders</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="ImportSinglePaymentOrdersResponseIo">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="PaymentOrdersResults" type="PaymentOrderResultIo" minOccurs="0" maxOccurs="unbounded">
        <xsd:annotation>
          <xsd:documentation>List of results for each single payment order</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

#### modified:

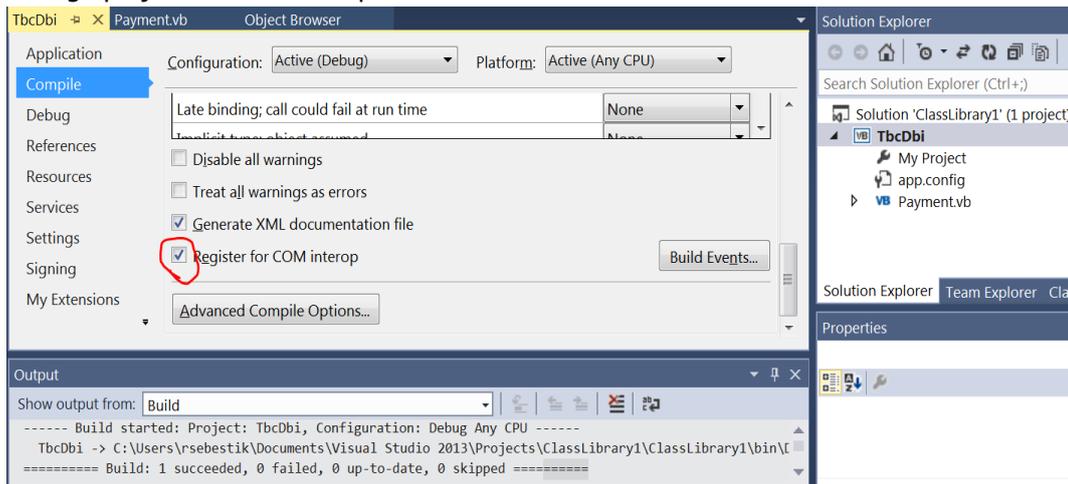
```
<xsd:element name="ImportSinglePaymentOrdersRequestIo">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="AbstractIo">
        <xsd:sequence>
          <xsd:element name="singlePaymentOrder" type="PaymentOrderIo" minOccurs="1" maxOccurs="unbounded" >
            <xsd:annotation>
              <xsd:documentation>List of all single payment orders</xsd:documentation>
            </xsd:annotation>
          </xsd:element>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

```

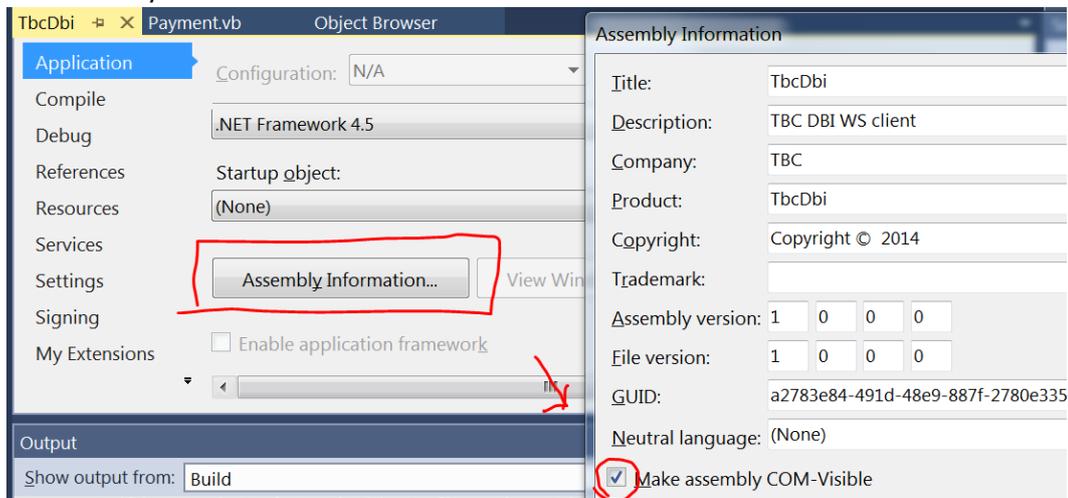
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
</xsd:element>
<xsd:element name="ImportSinglePaymentOrdersResponse" >
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="AbstractIo" >
        <xsd:sequence>
          <xsd:element name="PaymentOrdersResults" type="PaymentOrderResult" minOccurs="0" maxOccurs="unbounded">
            <xsd:annotation>
              <xsd:documentation>List of results for each single payment order</xsd:documentation>
            </xsd:annotation>
          </xsd:element>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>

```

### 3) Change project to COM interop



### 4) Set assembly to COM-Visible



### 5) Add this directive and constants to Payment Service (constants GUIDs are manually generated):

```

System.Xml.Serialization.XmlIncludeAttribute(GetType(AbstractIo)),
ComClass(PaymentService.ClassId, PaymentService.InterfaceId, PaymentService.EventsId)
>
Public Class PaymentService
  Inherits System.Web.Services.Protocols.SoapHttpClientProtocol
  Public Const ClassId As String = "F6BF6479-793C-4125-A4C1-03AB419C8599"
  Public Const InterfaceId As String = "E3ECC247-2E03-4CB0-A5C4-10C42F6A5808"
  Public Const EventsId As String = "84B15DB3-70A4-433B-A4C8-C5846E5514DA"

```

### 6) Enable TLS 1.2 protocol in constructor:

```

Public Sub New()
  MyBase.New()

  'set TLS 1.2 protocol

```

```
Net.ServicePointManager.SecurityProtocol = Net.SecurityProtocolType.Tls12
```

```
Me.Url = "http://localhost:8080/dbi/dbiService"
End Sub
```

7) Add inner classes Security and UsernameToken:

```
'BEGIN manual addition for Security header
<XmlRoot(Namespace:="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd")> _
```

```
Partial Public Class Security
    Inherits SoapHeader
```

```
Public Property UsernameToken As UsernameToken
End Class
```

```
Partial Public Class UsernameToken
    Public Property Username As String
    Public Property Password As String
    Public Property Nonce As String
End Class
```

```
Public Property secHeader As Security
```

```
Public Sub SetUrl(ByVal ServiceUrl As String)
    Me.Url = ServiceUrl
End Sub
```

```
Public Sub SetUsernameToken(ByVal Username As String, ByVal Password As String, ByVal Nonce As String)
    secHeader = New Security
    secHeader.UsernameToken = New UsernameToken
    secHeader.UsernameToken.Username = Username
    secHeader.UsernameToken.Password = Password
    secHeader.UsernameToken.Nonce = Nonce
End Sub
```

```
'END manual addition for Security header
```

8) Add directive of using security header to each WS method, that should be authenticated:

```
<System.Web.Services.Protocols.SoapDocumentMethodAttribute("http://www.mygemini.com/schemas/mygemini/ImportSinglePaymentOrders", Use:=System.Web.Services.Description.SoapBindingUse.Literal, ParameterStyle:=System.Web.Services.Protocols.SoapParameterStyle.Bare)> _
<SoapHeader("secHeader", Direction:=SoapHeaderDirection.In)>
```

```
Public Function
ImportSinglePaymentOrders(<System.Xml.Serialization.XmlElementAttribute([Namespace]:="http://www.mygemini.com/schemas/mygemini")> ByVal ImportSinglePaymentOrdersRequestIo As ImportSinglePaymentOrdersRequestIo As
<System.Xml.Serialization.XmlElementAttribute("ImportSinglePaymentOrdersResponseIo", [Namespace]:="http://www.mygemini.com/schemas/mygemini")> ImportSinglePaymentOrdersResponseIo
    Dim results() As Object = Me.Invoke("ImportSinglePaymentOrders", New Object()
{ImportSinglePaymentOrdersRequestIo})
    Return CType(results(0), ImportSinglePaymentOrdersResponseIo)
End Function
```

9) Override ToString() method on VB classes where needed. This is because in Excel VBA example this additional info is used as output. For this excel example these 2 classes has ToString implemented:

```
Partial Public Class PaymentOrderResultIo Inherits AbstractIo
    ...
```

```
'BEGIN manual addition for ToString override
Public Overloads Function ToString() As String
    Dim builder As New System.Text.StringBuilder
    builder.Append("PaymentOrderResult[")
    builder.Append("position: ").Append(positionField)
    builder.Append(", paymentId: ").Append(paymentId)
    builder.Append("]")
    Return builder.ToString
End Function
'END manual addition for ToString override
```

```
End Class
```

```
...
Partial Public Class ImportSinglePaymentOrdersResponseIo Inherits AbstractIo
```

```

...
'BEGIN manual addition for ToString override
Public Overloads Function ToString() As String
    Dim builder As New System.Text.StringBuilder
    builder.Append("PaymentOrderResults[")
    If paymentOrdersResultsField IsNot Nothing Then
        For Each item As PaymentOrderResultIo In paymentOrdersResultsField
            builder.Append(item.ToString).Append(" ")
        Next
    Else
        builder.Append("<empty>")
    End If
    builder.Append("]")
    Return builder.ToString
End Function
'END manual addition for ToString override
End Class

```

10) Build DLL – for 32bit and also make a version for 64bit Windows

### 12.3.2. Register DLL on client machine

- 1) Open command line as administrator in directory with DLL
- 2) Register DLL in system, use command which applies for you (if you are not sure, use 64bit command and if error occurs, use 32bit then)
  - a. For 32 bit Excel run this command:
 

```
> c:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe TbcDbi.dll /codebase /tlb:TbcDbi.tlb
```
  - b. For 64 bit Excel run this command:
 

```
> c:\Windows\Microsoft.NET\Framework64\v4.0.30319\RegAsm.exe TbcDbi64.dll /codebase /tlb:TbcDbi.tlb
```

It should end with successful registration.

### 12.3.3. Use registered DLL in VBA

- 1) Open example Excel file with macros (VBA\_withVB\_wsCall.xlsm)
- 2) Enable macros in Excel if needed
- 3) Go to VBA editor (alt+f11)
- 4) In VBA menu add reference to registered DLL (Tools -> References -> choose "TBC DBI WS client") and press OK. Close VBA window.
- 5) Restart of Excel might be needed. So reopen xlsm file.

### 12.3.4. Use Excel to call DBI via Visual Basic DLL

- 1) In Excel cells set DBI WS url in cell B1 and credentials in cells B3-B5.
- 2) Call WS via buttons (example contains prepared calls for getPaymentOrderStatus and ImportSinglePaymentOrders).
- 3) You can change transaction id next to order status button, so you can call ws for different transaction.
- 4) Other variables in this example (for importSinglePaymentOrders request) are hardcoded in VBA macros, but can be easily extended and configured also from excel cells.
- 5) Results (or errors) from WS call will be shown next to the button.
- 6) Way to extend example excel: Add another macro in VBA and assign this macro to new button in excel.
- 7) VBA macros used in example:

Option Explicit

```
Public ps As PaymentService
Public resultCell As String
```

```
'method for calling GetPaymentOrderStatus endpoint
'it takes paymentId from excel sheet and shows the result of the service in excel sheet
Sub GetPaymentOrderStatus_singleId()
    On Error GoTo NO_SERVICE
    Set ps = PaymentService
    On Error GoTo PROC_ERR
    init ("e9") 'init result cell

    Dim OrderStatus As GetPaymentOrderStatusRequestIo: Set OrderStatus = PrepareSinglePaymentOrderStatusRequest(Range("B9").value)

'service call
Dim OrderStatusResult As GetPaymentOrderStatusResponseIo: Set OrderStatusResult = ps.GetPaymentOrderStatus(OrderStatus)

    setResult ("Status: " & OrderStatusResult.Status)

Exit Sub
PROC_ERR:
    setError
Exit Sub
NO_SERVICE:
    MsgBox "Error creating PaymentService, is TbcDbi DLL properly registered and referenced from VBA?"
End Sub

'method for calling ImportSinglePaymentOrders endpoint
'it creates one TransferToOwnAccountPaymentOrder and shows the result of the service in excel sheet
Sub ImportSinglePaymentOrders()
    On Error GoTo NO_SERVICE
    Set ps = PaymentService
    On Error GoTo PROC_ERR
    init ("e15") 'init result cell

    Dim Orders As ImportSinglePaymentOrdersRequestIo: Set Orders = New ImportSinglePaymentOrdersRequestIo
    Dim OrderArr(0) As TransferToOwnAccountPaymentOrderIo: Set OrderArr(0) = CreateSamplePaymentOrder
    Orders.singlePaymentOrder = OrderArr

'service call
Dim OrderResult As ImportSinglePaymentOrdersResponseIo: Set OrderResult = ps.ImportSinglePaymentOrders(Orders)

    setResult ("Order results: " & vbCrLf & OrderResult)

Exit Sub
PROC_ERR:
    setError
Exit Sub
NO_SERVICE:
    MsgBox "Error creating PaymentService, is TbcDbi DLL properly registered and referenced from VBA?"
End Sub

'===== HELPER METHODS for preparing requests

'prepare request for GetPaymentOrderStatus endpoint
'it takes paymentId for status lookup as param
Private Function PrepareSinglePaymentOrderStatusRequest(paymentId As Long) As GetPaymentOrderStatusRequestIo
    Set PrepareSinglePaymentOrderStatusRequest = New GetPaymentOrderStatusRequestIo
    PrepareSinglePaymentOrderStatusRequest.singlePaymentId = paymentId
    PrepareSinglePaymentOrderStatusRequest.singlePaymentIdSpecified = True
End Function

'prepare request for ImportSinglePaymentOrders endpoint
'it prepares some random valid data for $1 payment order
```

```

Private Function CreateSamplePaymentOrder() As TransferToOwnAccountPaymentOrderIo
    Set CreateSamplePaymentOrder = New TransferToOwnAccountPaymentOrderIo

    CreateSamplePaymentOrder.creditAccount = New AccountIdentificationIo
    CreateSamplePaymentOrder.creditAccount.accountNumber = "GE86TB1144836120100002"
    CreateSamplePaymentOrder.creditAccount.accountCurrencyCode = "USD"

    CreateSamplePaymentOrder.debitAccount = New AccountIdentificationIo
    CreateSamplePaymentOrder.debitAccount.accountNumber = "GE69TB1144836020100001"
    CreateSamplePaymentOrder.debitAccount.accountCurrencyCode = "GEL"

    CreateSamplePaymentOrder.amount = New MoneyIo
    CreateSamplePaymentOrder.amount.amount = 1
    CreateSamplePaymentOrder.amount.currency = "USD"
    CreateSamplePaymentOrder.Description = "VBA test"
End Function

'===== COMMON HELPER METHODS

'init paymentService and resultCell
Private Sub init(resultCellParam As String)
    resultCell = resultCellParam
    setResult ("")
End Sub

'set error from exception to resultCell
Private Sub setError()
    Dim errorResult As String: errorResult = "Error: (" & Err.Number & ") " & Err.Description
    setResult (errorResult)
End Sub

'set result text to the result cell
Private Sub setResult(result As String)
    Range(resultCell).value = result
End Sub

'init paymentService with authentication
Private Function PaymentService()
    'create service client
    Set PaymentService = New PaymentService

    'set DBI URL
    'ps.SetUrl ("https://test.tbconline.ge/dbi/dbiService")
    'ps.SetUrl ("http://s28was7.bsccpraha.cz/tbcibs-dbi/dbiService")
    PaymentService.SetUrl (Range("B1").value)

    'set security username token to WS header
    Call PaymentService.SetUsernameToken(Range("B3").value, Range("B4").value, Range("B5").value)
End Function

```

See complete example for DBI call from Excel VBA in the file attached herein:



DBI\_example\_excel\_vba.zip