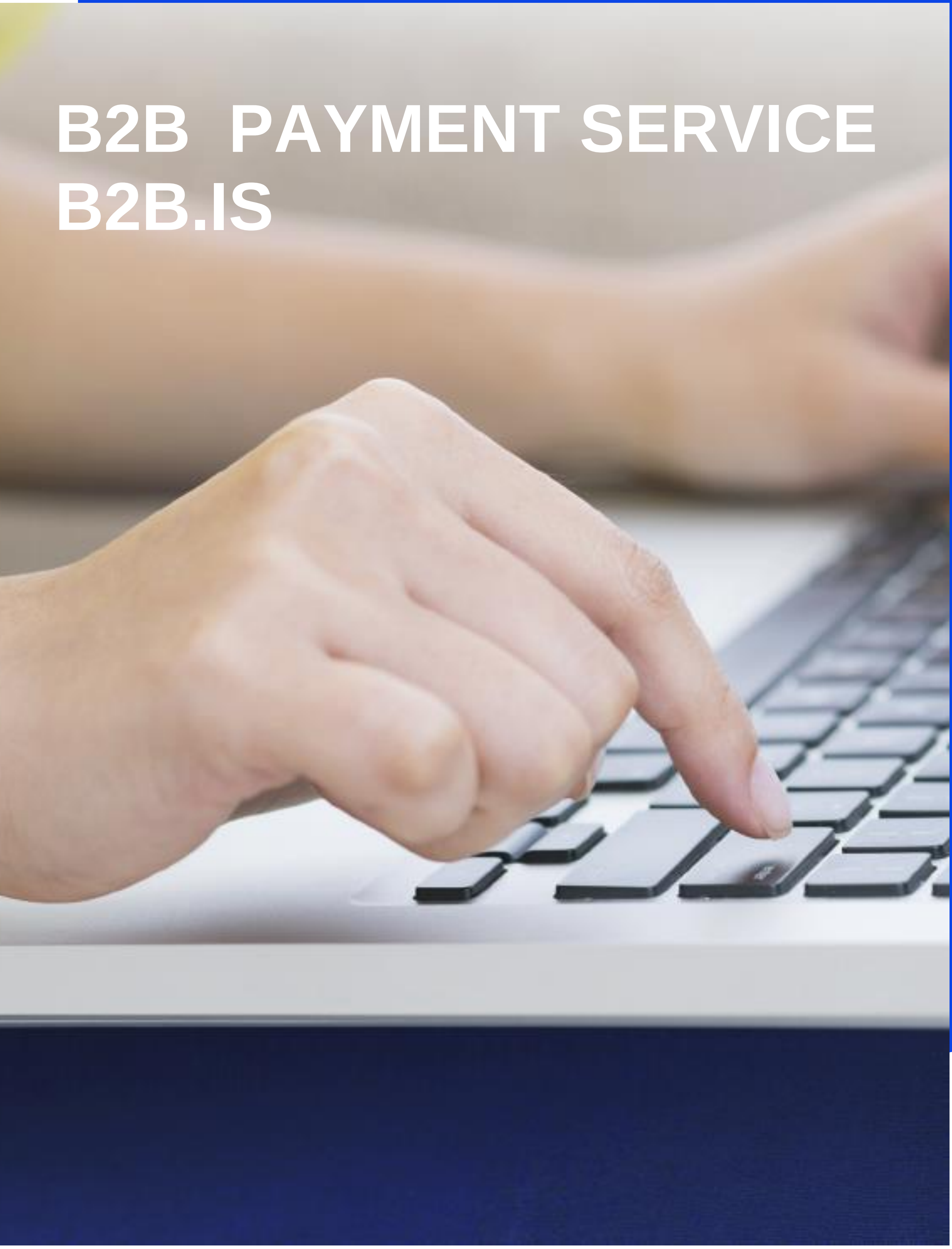
B2B PAYMENT SERVICE B2B.IS



Introduction

B2B is a collection of webservices designed to simplify online banking and give a clear overview of customers’ finances. All major bank transactions can be conducted directly from a customer’s accounting system. B2B is a communication technology for corporate accounting systems that exchanges information to and from the banks. Each company’s accounting system is used without having to use the online bank. B2B is a powerful solution in the field of electronic services. With the introduction of B2B, work processes can be shortened and accounting efficiency increases. B2B saves time and decreases the probability of errors, since only one system is used, i.e., the accounting system of the company in question. Additionally, there is greater opportunity to make informed decisions when a real-time overview of accounting is available.

For further information, please contact Arion bank corporate services: arionbanki@arionbanki.is

Table of contents

1 Common information

This webservice allows you to make money transfers between accounts and pay claims. It is possible to do a single payment, as well as a batch of payments in one call. Batch payments can be done in one of two ways:

- Direct payment from your Account system through the payment service.
- Send batch payments through the service to the Arion online bank, where you are required to log in and confirm the payment batch before it is executed.

By default, batch payments will stop in the Arion online bank until confirmed. To enable straight through payments (STP), you need to contact Arion bank via email arionbanki@arionbanki.is to have the configuration changed.

Users can view a list of executed and unexecuted batch payments in the online bank. For each batch that has been executed, users can download a receipt with detailed information about the payment execution. For example, which records were executed successfully and which ones hit an error.

1.1 Access control

The service has the same Access control as Arion online bank, by login and password. In addition to login, all communication between Arion and your system must be signed by a certificate.

The webservice for Payments is built on a schema interface that all the major banks in Iceland have agreed upon. The service has a WCF endpoint with a Mutual Authentication security model. The information transmitted between systems is encrypted over HTTPS. More information on certificate and installation can be found in the following document: http://b2b.is/services/Docs/UserManual-AuthenticationAndCertificates-english.pdf

1.2 Web paths

The Payment service can be found at:

https://ws.b2b.is/Payments/20131015/PaymentService.svc

1.3 Schema

Schema can be found at:

https://ws.b2b.is/Payments/20131015/PaymentService.svc?wsdl

1.4 Overview

This documentation is intended for programmers who implement solutions to connect directly from their system to Arion Bank's B2B services. This document includes use cases, descriptions, and information on the data format.

2 Functions

The B2B 2013 Payment Service can perform financial operations and requests on both single payments and bulk payments. Single payments include credit transfers and claim payments. See chapter 6.1 for more details on the input and output parameters of the functions.

2.1 Create payment

The function *DoPayment* creates a single transaction payment. The function supports credit transactions and claim payments. After payment creation, the function responds with the status of the payment.

2.2 Create payments

The function *DoPayments* creates a batch payment. The function supports credit transactions and claim payments. The response of the function will be a unique identifier for the batch that can be sent to the query function *GetPaymentsResult* to get information on the status of the batch.

2.3 Query payment

The function *GetPaymentResult* is not implemented. When creating a payment with the *DoPayment* function, the status of the payment is returned instantly.

2.4 Query payments

The function *GetPaymentsResult* returns the status of requested batch and the status of the payments it holds. It also returns which payments within the batch resulted in an error.

3 Use Cases

3.1 Create a batch payment

Users can create a batch payment in their accounting system and send it to the B2B payment service. The batch is then created in the Arion online bank and is visible on the “Batch Payments” page. The user receives a unique ID for the batch and can go to the Arion online bank to view, delete, or confirm (execute) the payment batch. It is not possible to edit the batch in the online bank, so errors must be corrected in your account system. After correcting the errors, delete the old batch from the online bank and send the new one in via the B2B service. It is possible to book batch payments in the online bank even though it has errors. In that case, you will receive a confirmation for the payments that were booked as well as the payments that resulted in an error. After analyzing the errors, you can correct them in your accounting system and resubmit them, as described above.

3.2 Initiate a batch payment without using the online bank to confirm payments

Users can create a batch payment in their accounting system that does not need to be confirmed in the online bank. To do this, the user creating the batch must be configured as a “straight through” user. To configure a user as a “straight through” user, the company’s power of attorney must contact Arion’s corporate finance division and request to have this functionality configured (see chapter 4.2 for more information). After a batch is created, it is confirmed (executed) automatically with the straight through process. A unique identifier is returned by the function and information on the batch can be fetched using the *GetPaymentsResult* (6.1) function, with the unique identifier as an input.

3.3 Cancel creation of batch if errors are present

Users can create a batch payment in their accounting system that is configured so the batch will not be created if an error is present. To do this, the *RollbackOnError* parameter in the *DoPayments* (6.1) function should be set to *true.* If errors are present and the batch is not created, the first error in the batch will be returned.

3.3.1 RollbackOnError payment process (not using “Straight through”)

The steps below describe the RollbackOnError processes when there are errors present.

RollbackOnError = True

1. User creates a batch payment.
2. Batch is error checked.
3. There are errors present, batch is not created.
4. The first error in the batch is returned.
5. User cannot fetch information on the batch as it was not created.

RollbackOnError = False

1. User creates a batch payment.
2. Batch is error checked.
3. The batch is created, even if it has errors.
4. All the errors are returned.
5. User fetches information on the unconfirmed batch, such as its status and list of errors.
6. User goes to the online bank to confirm the payments that do not have an error.
7. User fetches information on the confirmed batch, along with a list of payments that did not get confirmed due to errors.
8. User fixes the error payments in their accounting system and creates a new batch for those payments.
9. User goes to the online bank to confirm the new batch.

3.3.2 RollbackOnError payment process (using “Straight through”)

The steps below describe the RollbackOnError processes when there are errors present.

RollbackOnError = True

1. User creates a batch payment.
2. Batch is error checked.
3. There are errors present, batch is not created.
4. The first error in the batch is returned.
5. User cannot fetch information on the batch, as it was not created.

RollbackOnError = False

1. User creates a batch payment.
2. Batch is error checked.
3. There are errors present.
4. Payments that do not have an error get confirmed, while payments with errors are not confirmed (executed).
5. User fetches information on the confirmed batch, as well as a list of payments that did not get confirmed due to errors.
6. User fixes the error payments in their accounting system and creates a new batch for those payments.

3.4 Fetch receipts for payments

Users can fetch information on a batch by using the *GetPaymentsResult* (6.1) function. If the batch is confirmed, the user will receive information on confirmed payments, as well as a list of payments that could not be confirmed due to an error. If the batch is not confirmed, the user will receive two lists: one of unconfirmed payments that do not have an error and one of unconfirmed payments that have an error.

3.5 Initiate payment without using the online bank to confirm payments

Users can create a single payment (both credit transfers and claim payments) without needing to confirm it in the online bank. To do this, the user creating the payment must be configured as a “straight through” user (see chapter 3.2 and 4.2 for more information). The payment gets confirmed automatically and information on the payment will be returned.

3.6 Future payment batches

Users can create future batch payments by setting the *DateOfForwardPayment* parameter in the *Payments* (6.3) class as a date in the future. If information is fetched on the batch before the date of payment execution, the date of future payment is returned along with other information. It is also possible to see the date of future payment in the online bank.

3.7 Future payments

Users can create future payments by setting the *DateOfForwardPayment* parameter in the *Payment* (6.26.3) class as a date in the future. It is possible to see the date of future payment in the online bank.

3.8 Approval process

If a user that creates a batch is configured to use the online bank approval process, the batch needs to be approved in the online bank. For more details on the approval process, see chapter 4.4. Note that the approval process is only available for batch payments.

3.9 Claims with incurred expenses

If a batch is created that does not need to be confirmed in the online bank (using straight through processing, see chapters 3.2 and 4.2), and the batch includes a claim that has incurred costs such as interest or other costs, the claim will be paid with all the incurred costs. This means that the total amount of the batch payment might be higher than originally anticipated because of the incurred costs of the claims within. If information on the batch is fetched, a breakdown of all the costs for the claims will be retrieved.

If a batch is created that needs to be confirmed in the online bank and it includes a claim that has incurred costs, the updated amount has not been calculated and the batch will get an error if a user tries to confirm it.

When a single claim payment is created and it has incurred costs when it is confirmed, the total amount withdrawn will be higher than the original claim amount. After creating such a payment, information on it is returned immediately, along with a breakdown of all costs.

3.10 Submitting a batch that is visible to all employees

Batches created by the payment service are by default only visible to the person that created them in the online bank. However, if an asterisk (*) is added to the front of the batch name, then the batch will be visible to all users that have access to the company's online bank. All of those users will be able to view and confirm the batch in the online bank.

4 Relationship with online bank

4.1 Batch processing

The B2B payment webservice allows companies to create payments from their own accounting system. They can be credit transfers or claim payments, with both single payments and batch payments available. There are two ways to confirm the payments: either with the straight through process, where payments are confirmed automatically without using the online bank; or they can be sent to the online bank where they await confirmation. In the latter case, users can access their batch payments on the "Batch list" page, which is accessible from the sitemap on the online bank. From there, it is possible to view, delete, or confirm the batch payments. Note that it is not possible to edit the batch in the online bank, as that would create a discrepancy between the account system and the online bank. It is possible to confirm batches that contain one or more payments with an error. In that case however, only the payments without an error will be confirmed. When this occurs, the user will need to delete the batch in the online bank, solve the errors in their account system, and create a new batch with the resolved error payments.

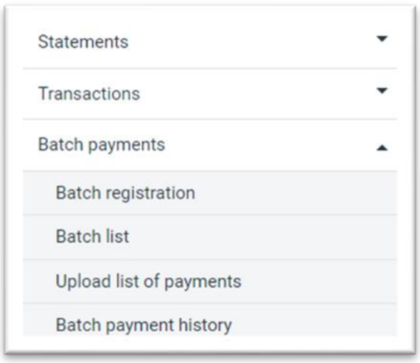
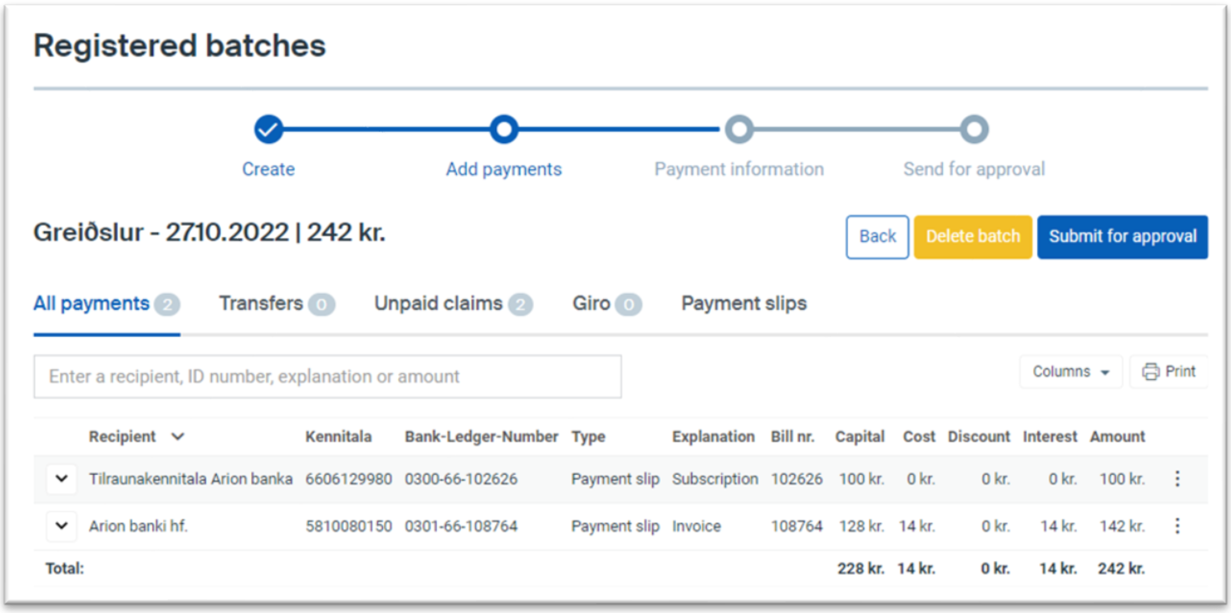


Figure 1: Location of Batch list on sitemap



Greiðslur - 27.10.2022 | 242 kr.

Recipient	Kennitala	Bank-Ledger-Number	Type	Explanation	Bill nr.	Capital	Cost	Discount	Interest	Amount
Tilraunakennitala Arion banka	6606129980	0300-66-102626	Payment slip	Subscription	102626	100 kr.	0 kr.	0 kr.	0 kr.	100 kr.
Arion banki hf.	5810080150	0301-66-108764	Payment slip	Invoice	108764	128 kr.	14 kr.	0 kr.	14 kr.	142 kr.
Total:						**228 kr.**	**14 kr.**	**0 kr.**	**14 kr.**	**242 kr.**

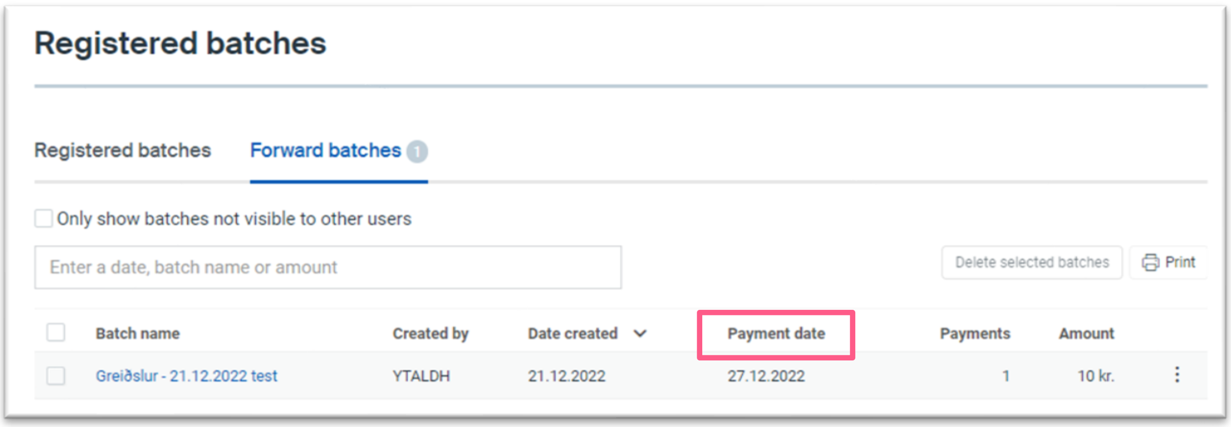
Figure 2: Registered batches (Batch list) in the online bank

4.2 Initiating payments without using the online bank to confirm payments

Companies can request authorization for some employees to be able to confirm payments straight from their account systems, without needing confirmation in the online bank. This functionality is called straight through processing (STP) and is defined on a user level. If companies would like to incorporate this functionality, their power of attorney holder needs to contact Arion's corporate services and request to have this functionality configured.

4.3 Future payments and future batch payments

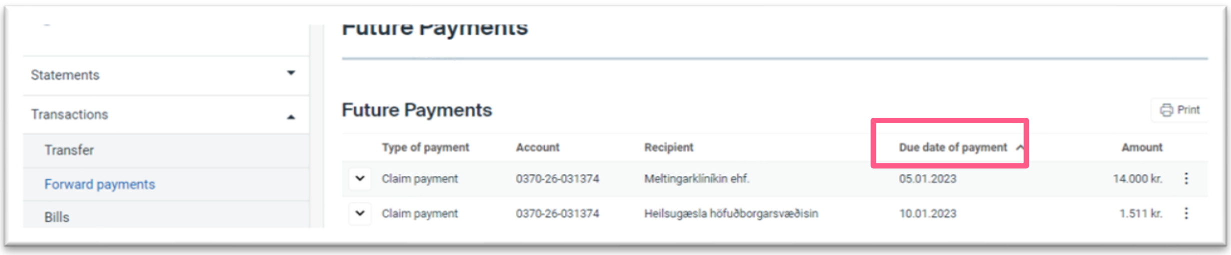
Users can create future payments and future batch payments through the B2B payments service. By fetching information on a batch via the *GetPaymentResults* (6.1) function, users can see the future payment date of the batch. It is also possible to see the future payment date of a batch in the online bank interface in the list of registered batches (see Figure 3):



	Batch name	Created by	Date created	Payment date	Payments	Amount
	Greiðslur - 21.12.2022 test	YTALDH	21.12.2022	27.12.2022	1	10 kr.

Figure 3: Future payment date highlighted

It is also possible to see the date of a future single payment by navigating to the Transactions/Forward payments page on the sidebar (see Figure 4). Note that it is not possible to edit a future payment in the online bank, as that would create a discrepancy between the online bank and the accounting system. To change the due date, a user must delete the payment from the online bank and create it again with a new due date.



Type of payment	Account	Recipient	Due date of payment	Amount
Claim payment	0370-26-031374	Meltingarklíníkin ehf.	05.01.2023	14.000 kr.
Claim payment	0370-26-031374	Heilsugæsla höfuðborgarsvæðisin	10.01.2023	1.511 kr.

Figure 4: Due date of future payment highlighted

4.4 Approval process

If a user that creates a batch payment that is configured to use the approval process, the batch will follow the approval process flow. There are two types of users in the approval process, defined as A-users and B-users:

A – User:

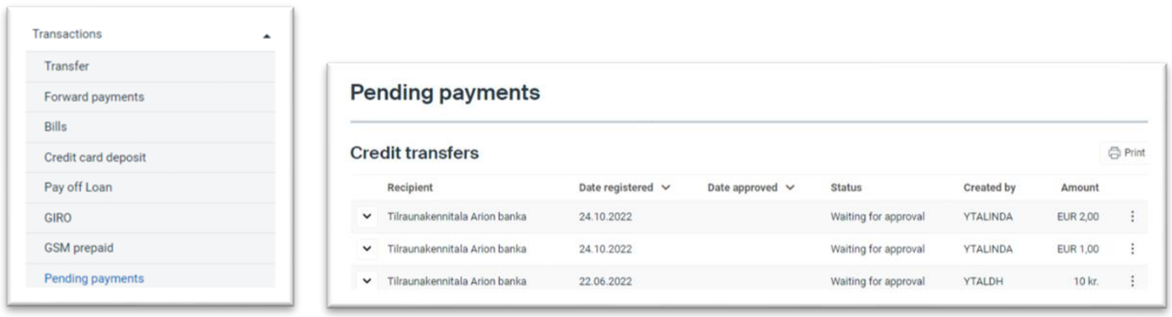
- Can create payments.
- Can pay approved payments from B users.
- Cannot approve payments from other users (A or B).
- Can pay payments that they have made themselves if a B user has approved it.

B – User (has more rights):

- Can create payments.
- Can approve payments from other users (A or B).
- Cannot approve payments that they have made themselves (B) but can approve payments from another B user.
- Can pay payments that they have made themselves (B) if another B user has approved.

- Can pay approved payments from A users.

It is possible to view all pending payments in the online bank by navigating to the Transactions/Pending payments page (see Figure 5).



Pending payments

Credit transfers

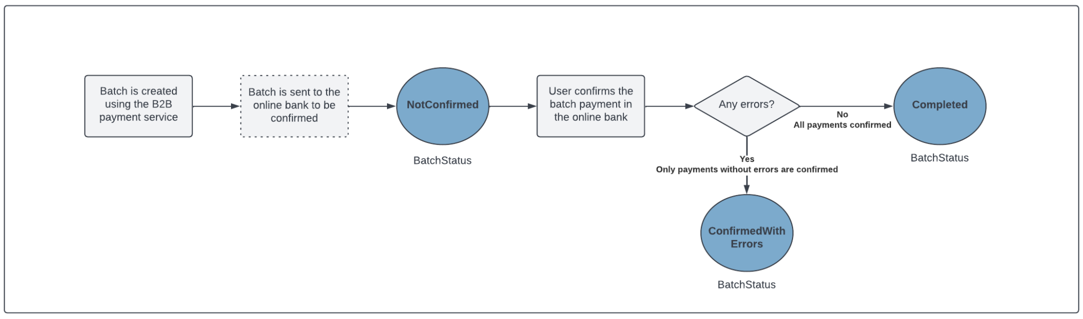
Recipient	Date registered	Date approved	Status	Created by	Amount
Tilraunakennitala Arion banka	24.10.2022		Waiting for approval	YTALINDA	EUR 2,00
Tilraunakennitala Arion banka	24.10.2022		Waiting for approval	YTALINDA	EUR 1,00
Tilraunakennitala Arion banka	22.06.2022		Waiting for approval	YTALDH	10 kr.

Figure 5: Pending payments can be viewed in the online bank

5 Batch states

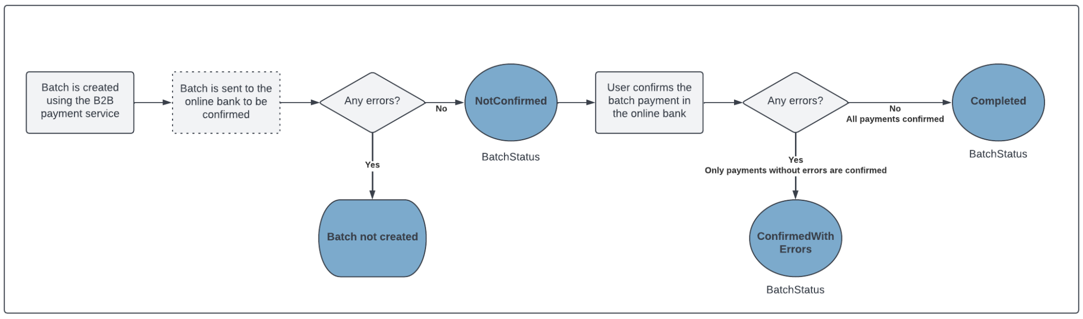
Status	Description
InProgress	Batch does not exist, or an unexpected error occurred on batch creation.
Completed	Processing of batch is completed; all payments have been executed.
CompletedWithErrors	Batch has been processed but one or more payment raised an error.
Cancelled	A NotConfirmed batch has been cancelled, e.g., by a user in the online bank UI.
OnHold	A future batch payment will have the status OnHold until the date of future payment.
NotConfirmed	Batch has been created and is awaiting confirmation (execution) in the online bank. The batch can also have this status if straight through processing (STP) is being used and all payments received an error (see chapters 3.2 and 4.2 for more details on straight through processing).
NotConfirmedWithErrors	Batch has been created but could not be confirmed, as it contained errors. In this case, the batch will not be confirmed unless a user logs in to the online bank and confirms it manually. (See more information on *RollbackOnError* in chapters 3.3 and in the chapter 5 subchapters below).

5.1 Payment process, not STP, RollbackOnError = false


Batch is created using the B2B payment service
Batch is sent to the online bank to be confirmed
NotConfirmed
BatchStatus
User confirms the batch payment in the online bank
Any errors?
No
All payments confirmed
Completed
BatchStatus
Yes
Only payments without errors are confirmed
ConfirmedWith Errors
BatchStatus

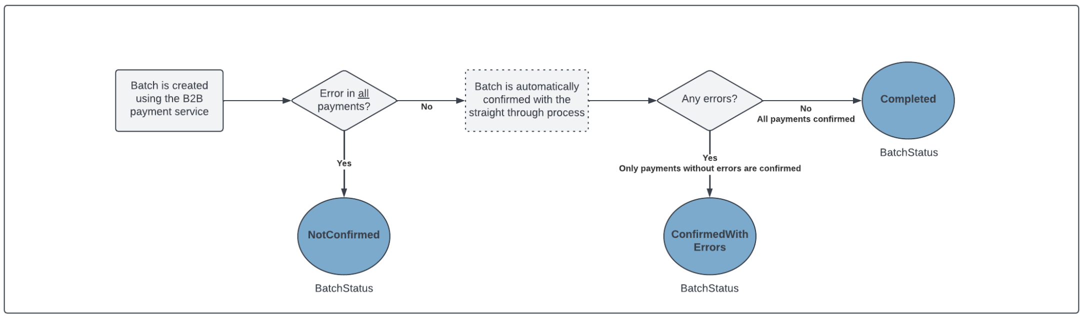
The diagram above shows the payment process for a batch that should be confirmed in the online bank and should not be rolled back if errors occur. After creating a batch, it will get the BatchStatus = NotConfirmed regardless of whether or not it has any errors. The user can then confirm (execute) the batch in the online bank and the status of the batch becomes either Completed or ConfirmedWithErrors, depending on if it had errors. If any errors are present, the payments without an error will still get confirmed. Note that it is not possible to correct the errors in the online bank, as it would create a discrepancy between the account system and the online bank. Therefore, the user needs to delete the batch in the online bank, solve the errors in their account system, and create a new batch with the resolved error payments. It is possible to fetch the status of a batch, along with a list of payments that raised an error, with the *GetPaymentsResult* (6.1) function. For more details on straight through processing (STP) and RollbackOnError, see chapters 3.2, 3.3 and 4.2.

5.2 Payment process, not STP, RollbackOnError = true


Batch is created using the B2B payment service
Batch is sent to the online bank to be confirmed
Any errors?
No
Yes
Batch not created
NotConfirmed
BatchStatus
User confirms the batch payment in the online bank
Any errors?
No
All payments confirmed
Completed
BatchStatus
Yes
Only payments without errors are confirmed
ConfirmedWith Errors
BatchStatus

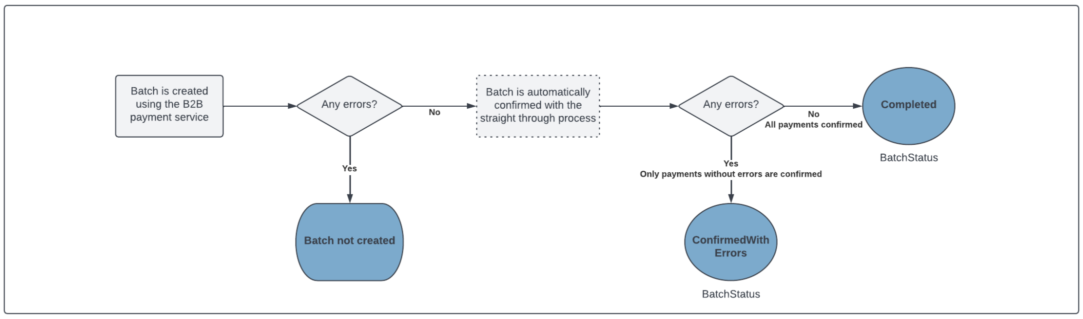
The diagram above shows the payment process for a batch that should be confirmed in the online bank but rolled back if errors are present after creation. If any errors are present after creation, the batch will not be created and the first error in the batch will be returned. Note that it will not have any status since it was not created. If no errors are present, the batch will get the status BatchStatus = NotConfirmed and await confirmation in the online bank. Once the user confirms the batch in the online bank, the status becomes either Completed or ConfirmedWithErrors, depending on whether or not it had any errors (between the time of creation and confirmation, the batch can accumulate errors, e.g., when a withdrawal account suddenly doesn't have sufficient funds for the payment). If any errors are present, the payments without an error will still get confirmed. Note that it is not possible to correct the errors in the online bank, as it would create a discrepancy between the account system and the online bank. Therefore, the user will need to delete the batch in the online bank, solve the errors in their account system, and create a new batch with the resolved error payments. It is possible to fetch the status of a batch, along with a list of payments that raised an error, with the *GetPaymentsResult* (6.1) function. For more details on straight through processing (STP) and RollbackOnError, see chapters 3.2, 3.3 and 4.2.

5.3 Payment process, STP, RollbackOnError = false


Batch is created using the B2B payment service
Error in all payments?
No
Yes
NotConfirmed
BatchStatus
Batch is automatically confirmed with the straight through process
Any errors?
No
All payments confirmed
Completed
BatchStatus
Yes
Only payments without errors are confirmed
ConfirmedWith Errors
BatchStatus

The diagram above shows the payment process for a batch that should be confirmed automatically, without involvement of the online bank, and not be rolled back if an error occurs after creation. If an error occurs on <u>all</u> payments in the batch, it will still be created but it will not be confirmed. It will be visible in the online bank, where it gets the status BatchStatus = NotConfirmed. This is done so that the errors will not be lost. If one or more payments do not have an error, the batch will be automatically confirmed through the straight through process (STP) and the status becomes either Completed or ConfirmedWithErrors, depending on whether or not it had any errors. Note that it is not possible to correct the errors in the online bank, as it would create a discrepancy between the account system and the online bank. Therefore, the user will need to delete the batch in the online bank, solve the errors in their account system, and create a new batch with the resolved error payments. It is possible to fetch the status of a batch, along with a list of payments that raised an error, with the *GetPaymentsResult* (6.1) function. For more details on straight through processing (STP) and RollbackOnError, see chapters 3.2, 3.3 and 4.2.

5.4 Payment process, STP, RollbackOnError = true



Batch is created using the B2B payment service
Any errors?
No
Batch is automatically confirmed with the straight through process
Any errors?
No
All payments confirmed
Completed
BatchStatus
Yes
Yes
Only payments without errors are confirmed
Batch not created
ConfirmedWith Errors
BatchStatus

The diagram above shows the payment process for a batch that should be confirmed automatically, without involvement of the online bank, but rolled back if an error occurs after creation. If any errors are present after creation, the batch will not be created and the first error in the batch will be returned. Note that it will not have any status since it was not created. If no errors are present, the batch will be automatically confirmed through the straight through process (STP) and the batch status becomes either Completed or ConfirmedWithErrors, depending on whether or not it had any errors. Note that it is not possible to correct the errors in the online bank, as it would create a discrepancy between the account system and the online bank. Therefore, the user will need to delete the batch in the online bank, solve the errors in their account system, and create a new batch with the resolved error payments. It is possible to fetch the status of a batch, along with a list of payments that raised an error, with the *GetPaymentsResult* (6.1) function. For more details on straight through processing (STP) and RollbackOnError, see chapters 3.2, 3.3 and 4.2.

6 Fields

This chapter contains information on the input and output fields of the Payment Service.

6.1 Service functions

List of functions within the Payment Service:

Function	Input	Output	Description
DoPayment	Payment (6.2)	PaymentsResult (6.7)	Creates a single payment
DoPayments	Payments (6.2)	PaymentId (6.5)	Creates a payment batch
GetPaymentResult	-	-	Not Implemented.
GetPaymentsResult	PaymentsResultQuery (6.8)	PaymentsResult (6.7)	Fetches information on a payment batch.

6.2 *Payment*

The Class *Payment* is an input to the function *DoPayment*. The fields *Out* and *In* are mandatory, while the *DateOfForwardPayment* field is optional and only used for future payments.

Parameter	Description
Out (6.3.1)	Contains information on the withdrawal of the payment
In (6.3.2)	Contains information on the deposit of the payment.
DateOfForwardPayment	Optional field. Date of future payment. Only used if the payment is to be executed in the future. If this field is empty, then the payment is executed straight away. Note: Date needs to be in the future. It is not possible to create multiple, identical payments on the same day between the same payment account and receiving account. Future payments are not available to and from account types (ledgers) 36 and 38.

6.3 *Payments*

The Class *Payments* is an input to the function *DoPayments*.

Parameter	Description
Out (6.3.1)	Contains information on the withdrawal of the payment
In (6.3.2)	Contains information on the deposit of the payment.
DateOfForwardPayment	Optional field. Date of future payment of batch. Only used if the batch payment is to be executed in the future. If this field is empty, the batch is sent to the online bank straight away or executed straight away if the user creating the payment is defined as a straight through user (see chapters 3.2 and 4.2). Note: Date needs to be in the future. Future payments are not available to and from account types (ledgers) 36 and 38.
RollbackOnError	If RollbackOnError = true and an error occurs, the payment batch is deleted and an error is returned to the user (see chapter 3.3). If RollbackOnError = false and an error occurs, the batch is created in the online bank even though there are errors in the batch. If RollbackOnError = false and an error occurs, and the user creating the payment is defined as a straight through user (see chapters 3.2 and 4.2), the payments that do not have an error get confirmed (executed), and the payments that have an error are not. The batch will get the status “CompletedWithErrors” and details on the errors can be found in the *Errors* class (6.7.5) within the *PaymentsResult* class (6.7). Further details on these scenarios can be found in chapter 5.
IsOneToMany	This functionality is not fully implemented, so we recommend using IsOneToMany = true. If true, then one payment will be withdrawn from the payment account for the whole batch. If false, then one payment will be withdrawn from the payment account for each of the payments in the batch. (This will work if there are no errors, but if errors occur then they will not be visible in the response.)
NameOfBatch	Name of the batch that will be visible on the “Batch Payments” page in the online bank. If an asterisk (*) is placed at the beginning of the name, the batch will be visible to all employees in the online bank that have permission to confirm payments.

6.3.1 *PaymentOut*

The class *PaymentOut* is part of the *Payment/Payments* inputs for the *DoPayment/DoPayments* functions. Account and AccountOwnedID are mandatory. This class is also an output of the *PaymentsResult* (6.7) class.

Parameter	Description
Account	Account number of the withdrawal account (e.g., 030126123456).
AccountOwnerID	Kennitala (national registry ID number) of withdrawal account owner.
CategoryCode	Not used. (The *CategoryCode* from the *Item* parameter in the *PaymentIn* class will be used for both deposit and withdrawal accounts.)
Reference	Not used. (The *Reference* from the *Item* parameter in the *PaymentIn* class will be used for both deposit and withdrawal accounts.)
Billnumber	Not used. (The *Billnumber* from the *Item* parameter in the *PaymentIn* class will be used for both deposit and withdrawal accounts.)
Receipt	Contains information on the payment for the payee or recipient. Receipt is of the class type *Communication* (0). Only used for batch payments, the receipt is sent for the whole batch. (Note that in the *PaymentIn* (6.3.2) class, the Receipt field is used for both single payments and individual payments within a batch.)
SecurityCode	Pin number for the withdrawal account.

6.3.2 *PaymentIn*

The class *PaymentIn* is part of the *Payment/Payments* parameters for the *DoPayment/DoPayments* functions. Item and Amount are mandatory fields, while Description, BookingID, and Receipt are optional.

Parameter	Description
Item	Specifies what kind of payment. Possible payment types are: • *Claim* (6.3.5) • *Transfer* (6.3.6) Note: Payment types *ABGiro, CGiro,* and *Bond* are no longer supported.
Description	A description that is used in the online bank and in communication.
BookingID	Not implemented
Receipt	Contains information on the payment for the payee or recipient. Receipts can be sent for single payments as well as for each individual payment within a batch. Receipt is of the class type *Communication* (0).
Amount	The amount to be paid in Icelandic krónas (ISK).

6.3.3 *ABGiro*

Retired functionality.

6.3.4 *CGiro*

Retired functionality.

6.3.5 *Claim*

The class *Claim* is used for the item parameter in the *PaymentIn* (*6.3.2*) class and contains information about the claim to be paid. All fields are mandatory.

Parameter	Description
Account	Claim identifier containing branch ID (first 4 digits), account type (next 2 digits), and the claim number (last 6 digits). E.g., 030126001234.
PayorID	Kennitala (national registry ID number) of the payor.
DueDate	Due date of the claim.
IsDeposit	If true, the payment will act as a partial payment of the claim. If false, the claim will be fully paid. Note that this is only implemented for single payments, not batch payments.
Claimant	Kennitala (national registry ID number) of the claimant.

6.3.6 *Transfer*

The class *Transfer* is used for the item parameter in the *PaymentIn* (*6.3.2*) and contains information on account-to-account transfers. The Account and AccountOwnerId fields are mandatory, while the CategoryCode, Reference, and Billnumber fields are optional. The *Transfer* class is also for the output parameter Item in the *PaymentResultDetails* (6.7.3) class.

Parameter	Description
Account	Account number of recipient, containing branch ID (first 4 digits), account type (next 2 digits), and the account number (last 6 digits). E.g., 030126001234.
AccountOwnerId	Kennitala (national registry ID number) of the recipient.
CategoryCode	Code used to categorize payments in account systems (e.g., 03 = transfer, 04 = salary). Only used for batch payments.
Reference	16-character reference field, usually kennitala (national registry ID number) of payor or recipient. The reference will be visible on both the payor and recipient accounts.
Billnumber	7-character text that is intended as an ID, short message, or description that will be communicated across different banks. Often, “Seðilnúmer” is used.

6.3.7 *Bond*

Retired functionality.

6.4 *Communication*

The class *Communication* contains information that can be used to send notifications to recipient or payor. The notification can be an email or an SMS message.

Parameter	Description
PostalMail (6.4.1)	Retired functionality (not implemented).
Email (6.4.2)	Information used to send an email notification.
SMS (6.4.3)	Information used to send an SMS message.

6.4.1 *CommunicationPostalMail*

Retired functionality.

6.4.2 *CommunicationEmail*

The class *CommunicationEmail* contains information on the recipient of the email. It is only possible to send one email per payment in a batch. For a single payment, it is possible to send two emails. For the batch as a whole, it is possible to send two emails. The EmailAddress field is mandatory, while the Language field is optional.

Parameter	Description
EmailAddress	Email address of email recipient.
Language	Language of email. Possible options are: • ISL (Icelandic) • ENG (English) • POL (Polish) • DAN (Danish) If field is empty, the email will be in Icelandic. The ISO 639 language standard is used, see http://www.loc.gov/standards/iso639-2/iso639jac.html for more details.

6.4.3 *CommunicationSMS*

The class *CommunicationSMS* contains information about the recipient of the text message. The PhoneNumber field is mandatory, while the Language field is optional.

Parameter	Description
PhoneNumber (6.4.4)	Information on the phone number to receive the receipt.
Language	Language of the text message. Possible options are: • ISL (Icelandic) • ENG (English) • POL (Polish) If field is empty, the email will be in Icelandic. The ISO 639 language standard is used, see http://www.loc.gov/standards/iso639-2/iso639jac.html for more details.

6.4.4 *PhoneNumber*

The class *PhoneNumber* contains information on the phone number that will receive a receipt from the *CommunicationSMS* (6.4.3) class.

Parameter	Description
Number	Phone number of recipient
CountryCode	Not implemented. Only text messages to Icelandic numbers are supported.

6.4.5 *Address*

Retired functionality. Was used for postal mails, which are no longer offered.

6.5 *PaymentId*

Unique identifier for a batch payment.

6.6 *PaymentStatus*

The type *PaymentStatus* is used to filter which payments to fetch when using the *PaymentsResultQuery* (6.8) class in the *GetPaymentsResult* (6.1) function.

Parameter	Description
GetStatus	Fetches the status of the batch.
GetErrors	Fetches the status of the batch and a list of payments that could not be executed.
GetSuccess	Fetches the status of the batch and a list of payments that were executed.
GetAll	Fetches the status of the batch and a list of all the payments within the batch (both executed payments and those where an error occurred).

6.7 *PaymentsResult*

The class *PaymentsResult* contains information on the result of single payments and batch payments.

Parameter	Description
ID	Unique identifier of batch payment (6.5). For single payments, a processing number from our system is returned which is not unique.
Status	For batch payments: Status of batch (6.7.1) For single payments: Status of single payment (6.7.2)
Success (6.7.3)	List of payments that were successfully executed.
Errors (6.7.5)	List of payments that were not executed due to an error.
DateOfPayment	Date of payment for batch or single payment. If the batch is not executed, the date “01.01.0001” is returned.

DateOfForwardPayment	Date of future payment only applies to future payments. If payment is not a future payment, the date “01.01.0001” is returned.
Out (6.3.1)	Information on the withdrawal account and account owner. The response is of the class type *PaymentOut* (6.3.1).

6.7.1 Status (*BatchStatus*) for a batch payment

The type *BatchStatus* contains information on the status of a batch payment.

Status	Description
InProgress	Batch does not exist, or an unexpected error occurred on batch creation.
Completed	Processing of batch is complete, and all payments have been executed.
CompletedWithErrors	Batch has been processed, but one or more payments raised an error.
Cancelled	A NotConfirmed batch has been cancelled, e.g., by a user in the online bank UI.
OnHold	A future batch payment will have the status OnHold until the date of future payment.
NotConfirmed	Batch has been created and is waiting to be confirmed (executed) in the online bank. The batch can also get this status if straight through processing (STP) is being used and all payments received an error (see chapters 3.2 and 4.2 for more details on straight through processing).
NotConfirmedWithErrors	Batch has been created but could not be confirmed because it contained errors. In this case, the batch will not be confirmed unless a user logs on to the online bank and confirms it manually. (See more information on *RollbackOnError* in chapters 3.3 and 5.)

6.7.2 Status (*BatchStatus*) for a single payment

The class *BatchStatus* can also be used for information on the status of a single payment. In this case, some fields are not applicable.

Status	Description
InProgress	Not applicable
Completed	Payment is confirmed.
CompletedWithErrors	Not applicable
Cancelled	Not applicable
OnHold	A future payment will have the status OnHold until the date of future payment.
NotConfirmed	Payment is not confirmed.

6.7.3 Success (*PaymentResultDetails*)

The class *PaymentResultDetails* contains details on a payment that has been confirmed (executed).

Parameter	Description
Item	Type of payment. Possible payment types are: • *Claim* (6.3.5) • *Transfer* (6.3.6) Note: Payment types *ABGiro, CGiro,* and *Bond* are no longer supported.
Amount	Total amount that was paid. For claims, this amount will include any interests and incurred costs, if applicable, so the amount might be higher than the original claim amount. (See more details on claims with incurred costs in chapter 3.9.)
Receipt (0)	Receipt with information on the payment.
Description	A description that is used in the online bank and in communication.
BookingID	Not implemented

6.7.4 *ClaimInfo*

The class *ClaimInfo* contains information on a claim.

Parameter	Description
Account	Claim identifier containing branch ID (first 4 digits), account type (next 2 digits), and the claim number (last 6 digits). E.g., 030126001234.
Claimant	Kennitala (national registry ID number) of the claimant.
PayorID	Kennitala (national registry ID number) of the payor.
DueDate	Due date of the claim.
IsDeposit	If true, the payment will act as a partial payment of the claim. If false, the claim will be fully paid. Note that this is only implemented for single payments, not batch payments.
AmountDue	Amount due for the claim in Icelandic krónas, excluding any incurred costs (see more details on claims with incurred costs in chapter 3.9).
DefaultCost	Default cost.
OtherCost	Other cost.
OtherDefaultCost	Other default cost.
DefaultInterest	Default interest.
NoticeAndPaymentFee	Notice and payment fee.
Discount	Discount.
CategoryCode	Code used to categorize payments for accounting systems (e.g., 03 = credit transaction, 04 = salary).

6.7.5 Errors (*PaymentError*)

The class *PaymentError* contains information on a payment that has raised an error.

Parameter	Description
Payment (6.3.2)	Information on the payment. The *PaymentIn* class is returned.
Error (6.7.6)	Information on the error.

6.7.6 *Error*

The class *Error* contains an error code and an error message that describes what went wrong when trying to execute a payment.

Parameter	Description
Code	Error code for the resulting error. Can be different across banks.
Message	Error message for the resulting error. Can be different across banks.

6.8 *PaymentsResultQuery*

The class *PaymentsResultQuery* is used as an input for the *GetPaymentsResult* function (6.1).

Parameter	Description
PaymentId (6.5)	Unique identifier for the batch payment.
PaymentStatus (6.6)	A parameter used to filter which payments to receive from the batch.

6.9 *IOBSFault*

The Class *IOBSFault* has information on errors that can be raised.

Parameter	Description
GeneralErrorCode	General error code.
GeneralErrorText	General error text.
SourceErrorCode	Error code from underlying system. Not Implemented.
SourceErrorText	Error text from underlying system. Not Implemented.

GeneralSourceCode	Technical error code from underlying system.
GeneralSourceText	Technical error text from underlying system.

GeneralErrorCode	GeneralErrorText	Description
0001	Service is unavailable	The service is unavailable.
1000	An error occurred	General error message with possible details.
1100	Access to the operation is not present	The function is restricted, access is needed.
1200	Data could not be validated	Input data could not be validated according to the XML Schema.
1300	Business logic error	General error in business logic.

An example of an error:

```
<Fault>
<Action>http://IcelandicOnlineBanking/2013/10/15/GetPaymentResult
</Action>
      <Code>System.ServiceModel.FaultCode</Code>
      <Reason> The method or operation is not implemented.</Reason>
      <Detail>
            <IOBSFault>
                  <GeneralErrorText> An error occurred.</GeneralErrorText>
                  <GeneralErrorCode>1000</GeneralErrorCode>
                  <BanksErrorText> The method or operation is not implemented.
                  </BanksErrorText>
                  <BanksErrorCode>14000</BanksErrorCode>
                  <GeneralSourceText></GeneralSourceText>
                  <GeneralSourceCode></GeneralSourceCode>
            </IOBSFault>
      </Detail>
</Fault>
```

7 Known issues and unavailable functionality

1. It is not possible to send more than one SMS message or email per payment in batch.
2. The approval process is implemented for batch payments, but not for single payments. (For more information on the approval process, see chapters 3.8 and 4.4.)
3. Foreign payments are not implemented. It is not possible to transfer between currency accounts (account types 36 and 38), nor is it possible to pay claims in foreign currency. This includes single payments and batch payments.
4. In batch payments, the functionality of withdrawing one payment for each individual payment in a batch (using IsOneToMany = false in the *Payments* (6.3) class) is not fully implemented. This will work if there are no errors, but if errors occur then they will not be visible in the response. Therefore, we recommend using IsOneToMany = true for all batch payments.

5. It is not possible to do a partial payment of a claim when using batch payments (using IsDeposit = true in the *claims* (6.3.5) class. However, it is implemented for single payments.
6. The function *GetPaymentResult* (6.1) for single payments is not implemented. However, the payment result is always returned when doing single payments with the *DoPayment* (6.1) function.
7. For batch receipts, the due date is not returned for claim payments (DueDate value in the *ClaimInfo* (6.7.4) class.

8 Error examples

1. **Villuskilaboð: FLYK HBOK**
   This error occurs when trying to create a batch where the transaction key does not match the account type (ledger). This usually occurs when trying to do a partial payment of a claim within a batch payment.
2. **4402 Notandi á ekki reikningsnúmerið**
   This error occurs if the withdrawal account does not match the user that is authenticated and using the service.

Abbreviations	Description
Bank	Branch ID
Hbok	Account type/ledger (Höfuðbók)
Rnum	Account number or claim number (Reikningsnúmer)
Flyk	Transaction key (Færslulykill)
Ktal	National registry ID number (Kennitala)
Vald	Valuedate
Tlyk	Category code (Textalykill), e.g., 04 = salary
Slnr	Bill number (Seðilnúmer)
Upph	Amount (Upphæð)
Tilv	Reference number (Tilvísunarnúmer)
Gjdg	Due date (Gjalddagi)

9 Changes from previous schema

The following changes have been made from the 2005 B2B schema:

Chapter	Description
6.3.1	New parameter *SecurityCode* in the class *PaymentOut.*
6.3.5	Class name changed from *PaymentSlip* to *Claim.* Two new parameters: *Claimant* and *PayorID*. One parameter removed: *PersonID.*
6.3.7	New class *Bond*
6.7	Class *PaymentsResult* has two new parameters: *DateOfForwardPayment* and *Out.*
6.7.1	New status *NotConfirmedWithErrors* in Enum *BatchStatus.*
6.7.2	New status *NotConfirmedWithErrors* in Enum *BatchStatus*.
6.7.2, 6.7.3	Change of possible inputs for the *Item* parameter.
6.7.4	*PaymentSlipInfo* is now called *ClaimInfo.* Two new parameters: *Claimant* and *PayorID*. One parameter removed: *PersonID.*
6.9	Error class name changed from *BankErrorFault* to *IOBSFault.* Two new parameters added: *GeneralSourceText* and *GeneralSourceCode.*