



Íslandsbanki

Vefþjónusta

Sambankaskema

10. mars 2008

TABLE OF CONTENTS

Introduction	3
Purpose	3
Scope	3
References	3
Design Goals and Limitations	3
Problem statement	3
Design Goals	3
Design decisions	4
Implementation	5
Overview	5
Behaviour	5
Operations	6
Exceptions	6
Timestamps	7
UserNameToken	7
Signing Messages	8
Icelandic Online	9
IcelandicOnlinePayments	9
Payment (DoPayment)	9
Out	10
In	11
ABGiro	12
CGiro	12
PaymentSlip	13
Transfer	14
Receipt	15
PaymentsResult (DoPaymentResponse)	16
Success	17
PaymentSlip (in Success)	18
Errors	19
Error	19
Payments (DoPayments)	220
Payment query (GetPaymentResult)	21
PaymentsResult	21
Payments query (GetPaymentsResult)	22
IcelandicOnlineStatements	23
AccountStatement (GetAccountStatement):	23
Get AccountStatementResponse	24
Transactions	25
CurrencyRateRequest:	27
Currency rate (CurrencyRateResponse):	27
IcelandicOnlineClaims	28
Claim creation/Claim modification(CreateClaims/AlterClaims)	28
ClaimKey	29
NoticeAndPaymentFee	29
DefaultCharge	30
DefaultInterest	30
Currency information	31
Discount	31

Bill Presentment System	32
Printing	33
ClaimOperationResult	34
CancelClaims	35
CreateClaim/AlterClaim	36
CancelClaim	38
GetClaimOperationResult	39
QueryClaims	41
QueryClaim	42
QueryPayments	45
IcelandicOnlineSecondaryCollectionClaims	49
AlterClaims / CancelClaims / AlterClaim / CancelClaim / GetOperationResult /	
QueryClaim	49
QueryClaims	49
QuerySecondaryCollectionPayments	49
QuerySecondaryCollectionClaims	50
SecondaryCollectionReturnClaim	50

Design

Introduction

Purpose

The main purpose of this document is to give an overview of services which make it possible for users connect to a banking institution and perform financial operations.

The intended users of this document:

Designers and programmers at banking institutions which will design and program systems along with unit and integration tests on the services described here.

IT Workers at banking institutions, which will deploy and maintain the services.

Companies, ISV's or independent programmers, creating systems utilizing these services.

Scope

This document attempts to address issues of security, the standards that are adhered to and the main design decisions made about how the services behave. The operations are described in details.

This document does not go into service implementation details which will inevitably be different for different banks. Neither is it the purpose of this document to describe details about client implementation.

References

[WS-Security Specification](#)

[Technical specification of HTTP over TLS \(RFC 2818\)](#)

Design Goals and Limitations

Problem statement

Today all Icelandic banks provide roughly the same financial services over the Internet or closed nets. The services enable users to retrieve bank statements, transfer funds and manipulate claims. The methods and protocols of communication vary from sending text based documents over FTP, XML over HTTP to SOAP message interchanges. Because of this users must customize their system to each banking institution.

By standardizing the interface to these financial services, users can create one client which can communicate with all institutions which implement the services.

Design Goals

The main design goals that were set at the start of the project were:

Communication with the services should be easy and inexpensive.

1. It should be easy for users/programmers to create clients which can be used to interact with the services.
2. Build on known and accepted standards as much as possible.
3. The services should be as secure as possible without affecting usability.
4. Enable the banking institutions to implement these services in a consistent manner.
5. An attempt should be made not to limit services to a specific transfer protocol if possible.

Design decisions

In light of the design goals a decision was made to implement the services as SOAP Web Services, accessible over the Internet through the HTTP secured with SSL (HTTPS) and Web Service Security (WSS) using the Username token and the X509 certificate token profiles.

The Internet is the most common and most inexpensive method of communicating between separate networks and HTTPS has the benefit of being a very common transfer protocol, accessible through most firewalls on port 443 (design goal 1).

The reason for choosing SSL instead of XML Encryption as defined in WS Security is that with regards to the current state of toolkits available to many programmers, the former is probably easier to implement (design goal 2, 3 and 4).

The use of certificates is deemed necessary, despite increased complexity for client implementations, as the level of security measures acceptable to the banking institutions and their clients rules out using username and passwords as the only barrier to accessing sensitive information (design goal 3).

SOAP is a standard for exchanging XML-based messages over a computer network. XML has various benefits as a message exchange format in B2B and B2C scenarios. Knowledge among programmers is widespread, a variety of tools for manipulation is available along with the fact that XML is human readable (design goal 2 and 3).

WSS has been a stable OASIS standard since 2004 and the UsernameToken profile is a relatively straightforward way of sending user authentication information in the context of a message. As the transport is secure and to simplify usage and server side implementation, the password is sent as clear text. Under WSS messages can also be signed in accordance with the XML Signature specification. Tools are available to most platforms that ease the use of WSS, such as WSE 2.0 and 3.0 for Microsoft .NET and the XWS-Security Framework for Java to name two (design goal 2, 3 and 4).

The reason that custom services and schemas were chosen was that no existing standard for financial services would have fit into the Icelandic banking environment without considerable modification. Most implement custom security models, some require specialized tools for usage and the learning curve would have been steep. Using XML schemas and the Web Service Description Language (WSDL) enables automatic tools to bootstrap the code for communicating with the services in a few minutes. The aim was that such code generation would create classes that would be as easy as possible to use, always keeping in mind that XML and object oriented structures do not always map directly (design goals 2).

The services offered are modelled on existing services and functionality that the banking institutions are all able to offer (design goal 5).

Finally it is worth mentioning that although this version of the services only offers HTTP as the application level protocol, it should be possible in the future to use other protocols (design goal 6), should that be deemed necessary.

Implementation

Overview

This following section describes some general issues that affect the use of the services and some code examples are given.

Behaviour

The following is a high level diagram of how a client will communicate with a server which hosts the web service, which then communicates with its back office systems.

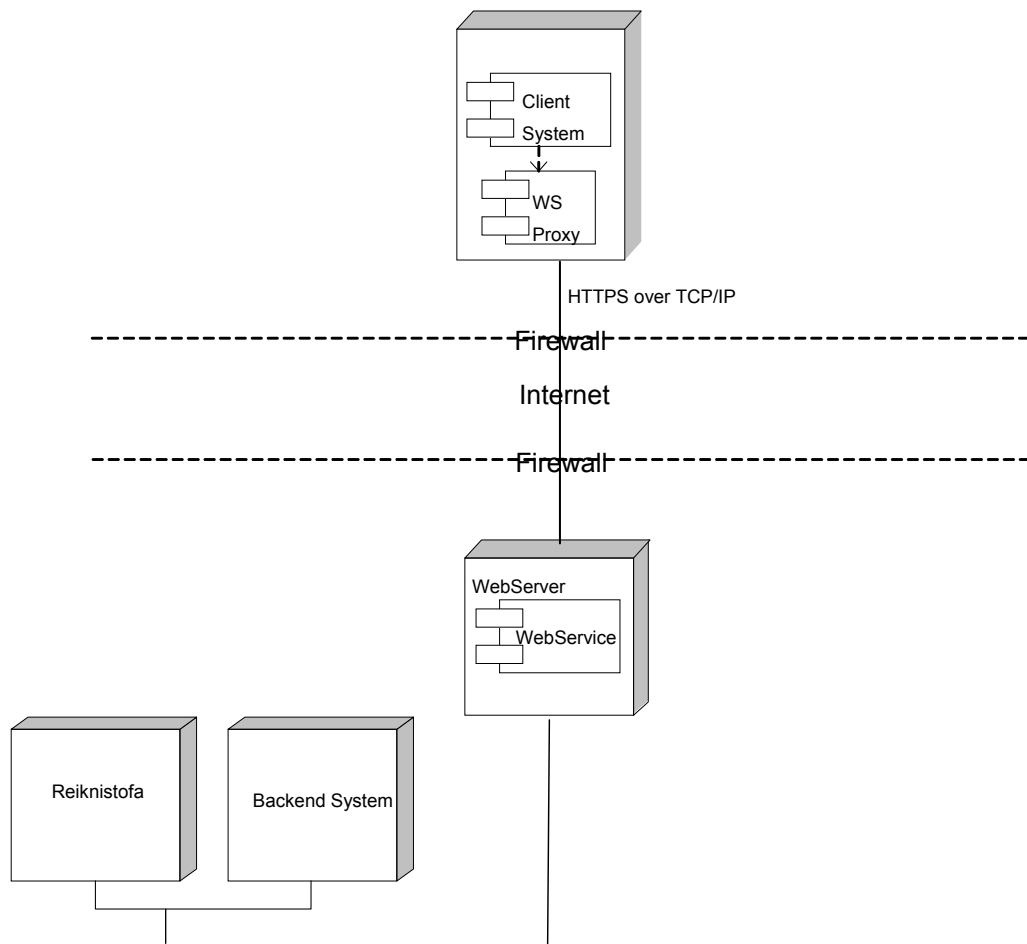


Figure 1. Basic setup.

Operations

The system tries to keep responses to operations are kept as quick and light as possible. It was decided that in most cases a user only wants to know that the operation has succeeded and does not need to receive the data he sent with the response. There are special operations designed to return the results of operations and all related data.

Exceptions

All communication is prone to exceptions and SOAP has a standard mechanism to communicate exceptions. These SOAP Exceptions are only thrown when it is not possible to complete an operation, usually due to faulty data or other technical reasons. Additionally, some circumstances where the input data does not conform to a given criteria can lead to an exception being thrown. When executing a batch, where it is possible for some operations to succeed but not others, other ways of returning error information is preferred.

Special error messages are returned in the details node of a SOAP exception when the error does not deal with SOAP headers. The different nodes returned in the details node are described in the following table:

Code	Type	Details
GeneralErrorCode	xs:string	Common error code across banking institutions.
GeneralErrorText	xs:string	Text to describe the GeneralErrorCode. ex: "Authentication failed", "Data could not be validated" etc.
BanksErrorCode	xs:string	Error code specific to the banking institution and the error instance.
BanksErrorText	xs:string	Text to describe the BanksErrorCode and/or data to resolve or help troubleshoot problems between banking institutions.

The BanksErrorCode can be used by each individual institution to identify individual error occurrences, e.g. to enable tracking. The GeneralErrorCodes are common error codes and indicate which class of error has occurred.

Code	Text	Details
0001	Service is Unavailable.	Implies that the service is closed for some reason.
1000	An error occurred.	A general error if a more detailed description is not available.
1100	Access to the operation is not present.	
1200	Data could not be validated.	The data could not be validated according to the XML schema.
1300	Business logic error.	Business rules were broken, e.g. dates or amounts were not valid.

All documents sent to the service are validated according to schema. Figure 1 shows a sample of an error.

```
<soap:Fault>
  <faultcode>soap:Client</faultcode>
```

```
<faultstring>System.Web.Services.Protocols.SoapException: Payor ID could not
be validated at
KBBanki.Krofulina.WebServices.BankingClaims.CreateClaim(Claim
claim)</faultstring>
<faultactor>http://www.somewebserver.is/WebServices/2005/12/01/Claims.asmx</f
aultactor>
<detail>
  <GeneralErrorCode>1200</GeneralErrorCode>
  <GeneralErrorText>Data could not be validated.</GeneralErrorText>
  <BanksErrorCode>5e80f0c6-479f-4dc2-9ab1-f2abed1e9f71</BanksErrorCode>
  <BanksErrorText>The 'Identifier' element has an invalid value according to its
data type. An error occurred at , (5, 78). BanksErrorText>
</detail>
</soap:Fault>
```

Figure 1. A sample of a SOAP exception.

Timestamps

It is necessary to make sure that the clocks are as synchronized as possible on clients and servers. The reason for this is that SOAP messages include a "Time to live", which is important because the system will not perform operations which do not arrive within a reasonable time. The services in this document use a default time of 900 seconds.

UserNameToken

Each call to the service should include a UserNameToken in accordance with the OASIS WSS UsernameToken Profile 1.0. The token should include the Username and Password tags. The Password@Type attribute references by default the URI „...#PasswordText“ and the password should be sent as clear text.

```
<S11:Envelope xmlns:S11="..." xmlns:wssse="...">
  <S11:Header>
    ...
    <wsse:Security>
      <wsse:UsernameToken>
        <wsse:Username>MyUserName</wsse:Username>
        <wsse:Password>My1ongA$ndDiff9ltP%$$phr$se</wsse:Password>
      </wsse:UsernameToken>
    </wsse:Security>
    ...
  </S11:Header>
  ...
</S11:Envelope>
```

Figure 2. A sample of a security header.

The Nonce and Created tags are optional and their usage will not be enforced server side.

Signing Messages

Digital signature of messages is mandatory. Each banking institution defines its own rules for which types of certificates can be used for the services. This means that certificate which is used at one bank may, or may not, be accepted by other banks.

Icelandic Online

This section contains a manual for users of the web services for the banking institutions. The services described here are valid with all the banks, i.e. the same schemas and objects apply with all the banks.

The following paper describes the operations that can be performed in the first version of this standard. The operations are described in a manner of the way they are performed, i.e. that each operation is described in a way that depicts all the factors that need to be taken into consideration while performing each operation.

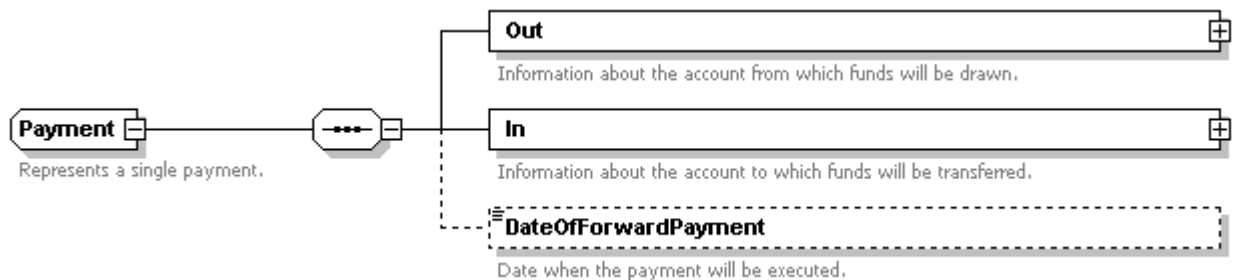
Pictures are used to further explain how objects are connected within each operation. Solid lines in these pictures indicate that the element in question must be entered, but the dotted lines indicate that the element is optional.

The schemas themselves will be accessible through other means of publication.

IcelandicOnlinePayments

Payment (DoPayment)

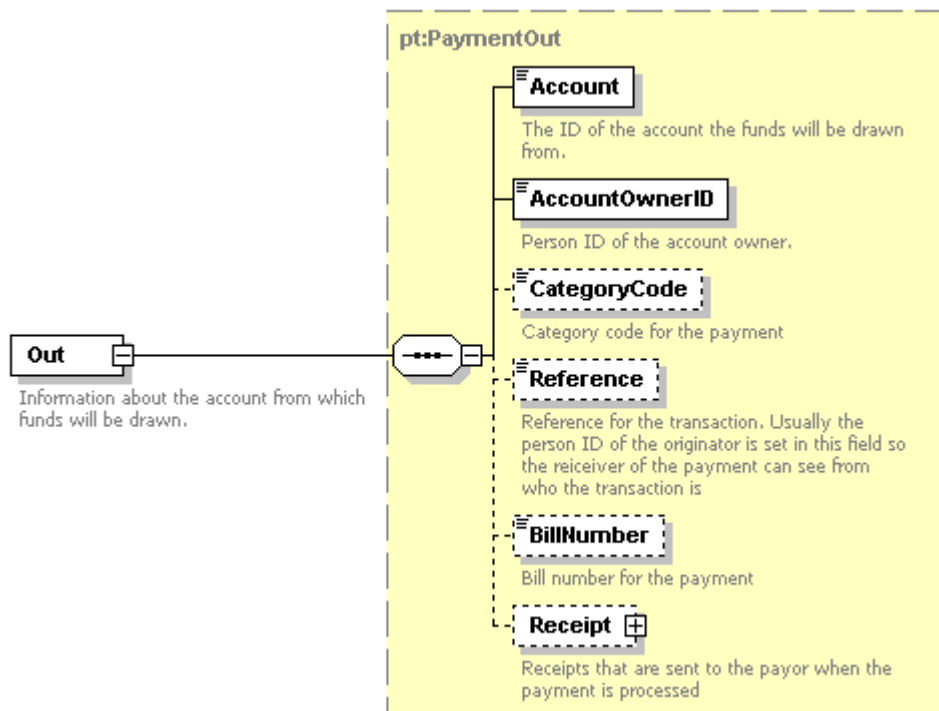
A description of how single payments are created. The object Payment has a list of PaymentOut and PaymentIn, which are the withdrawals and deposits, along with the date the payment shall be made. If a payment date is not entered, it is generally assumed that the payment shall be performed the very same day.



A more detailed description of the sub-items of **Payment** follows:

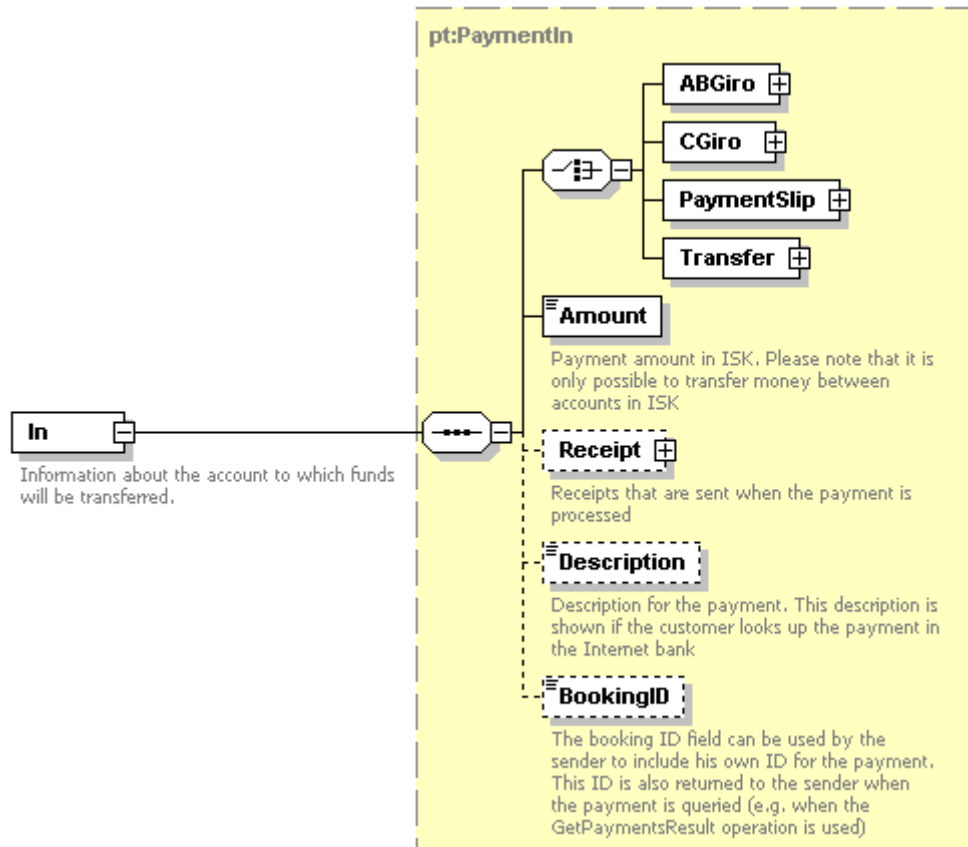
Out

Here we describe the withdrawal that takes place during payment. The only thing that must be entered here is the account number and the ID of the account owner. Category code, reference number and bill number can be entered and that information will be accessible when account statements are viewed. A receipt (intented for the payor) can also be sent.



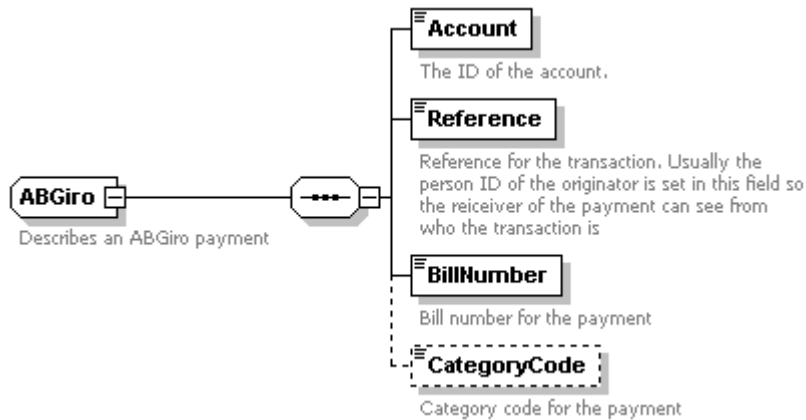
In

A choice is made between 4 types of deposits, AB giro, C giro, Payment bills and standard transfers. One of these must be selected. The amount in question must also be entered, but a receipt and a description of the payment are optional. The BookingId is thought as an supplementary field that the users can use to link payments into their own accounting systems.



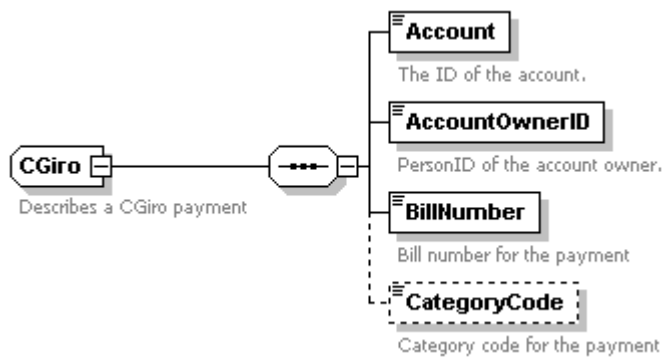
ABGiro

Obligatory fields are the account ID (to which money will be deposited), the reference number for the payment and the bill number of the giro to be paid. An optional field is also available for the category code.



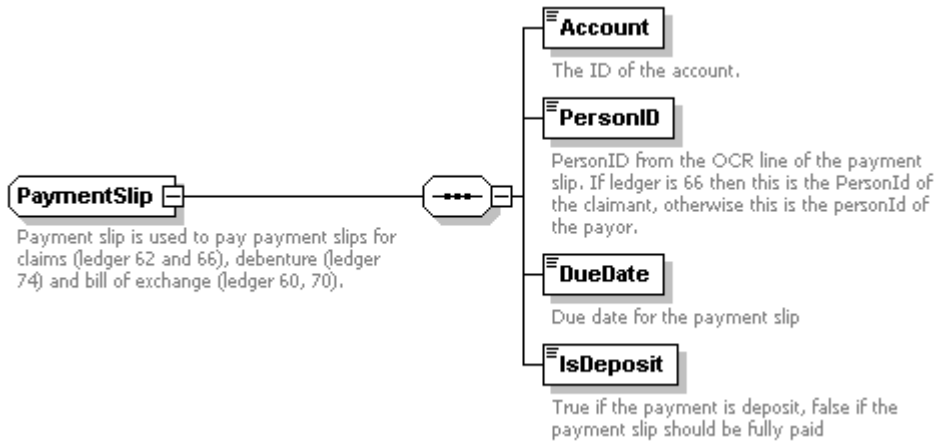
CGiro

Obligatory fields are the account ID (to which money will be deposited), the personal ID of the account owner and the bill number for the giro. The category code field is optional.



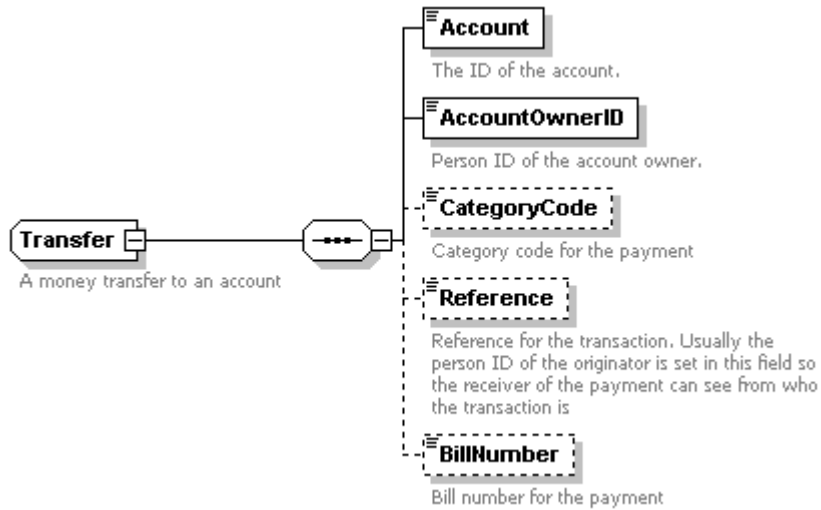
PaymentSlip

All fields are obligatory, the account (to which funds will be deposited), the ID number of the payor or invoicer (depends on the ledger), the due date of the slip and the IsDeposit fields dictates whether this payment is a partial or complete payment of the slip.



Transfer

A standard transfer to an account. Obligatory are the account number and the ID number of the account owner fields (of the account that funds will be deposited to), and optional fields are for the category code, reference number and the bill number.

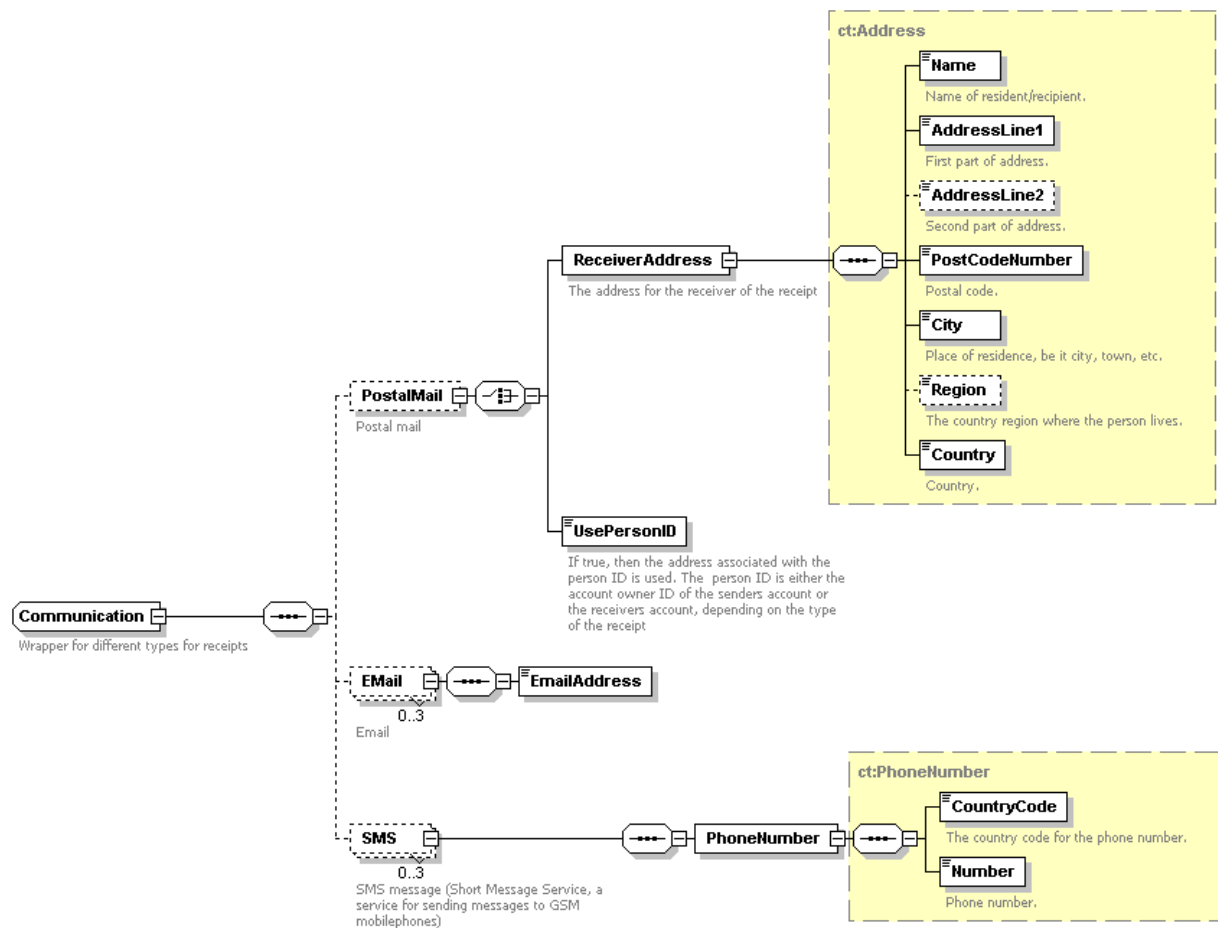


Receipt

Receipts are sent to the payment recipient when it is performed. This is an optional field on both in and out payments. The choice stands between sending PostalMail (a standard letter mail), Email and SMS. The options are sending 1 PostalMail, 3 emails and 3 sms's. If PostalMail is selected, then a recipient must either be entered by using the ReceiverAddress or by setting the UsePersonID field as true, in which case a receipt is sent so the recipients home as listed in the national register. The ReceiverAddress consists of a name, two address lines, the postal code, city, region and country.

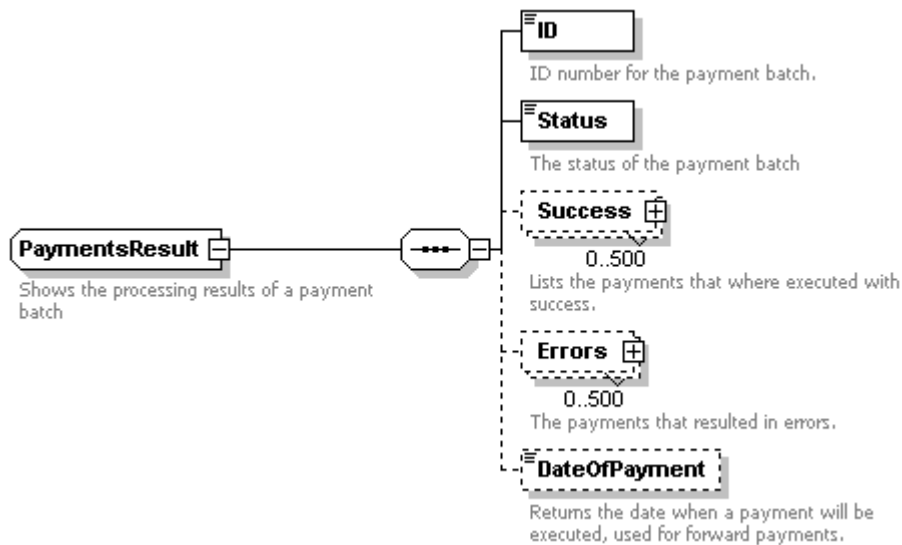
If it is selected to send an email, then only the email address must be entered.

If it is selected to send an SMS, then a country code and phone number must be entered.



PaymentsResult (DoPaymentResponse)

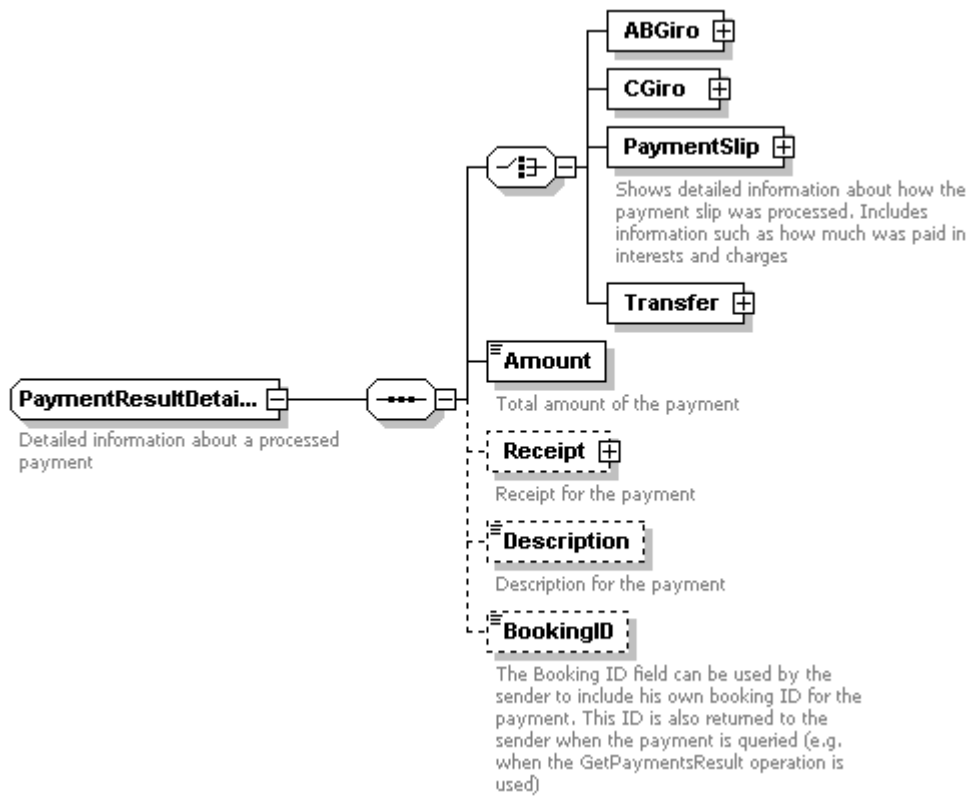
This is a description of the response to a creation of a single payment (the results for a payment batch). ID is the unique identification for the batch that was created and Status is the batch status. Success and Errors are kept optional, because it is always possible that one of those would be completely empty. In Success and Errors the payments just created are being returned. If a future date for payment was selected in the payment creation, then that date is returned here in DateOfPayment, and the status of the batch is put as OnHold.



A further description of the things PaymentsResult consists of:

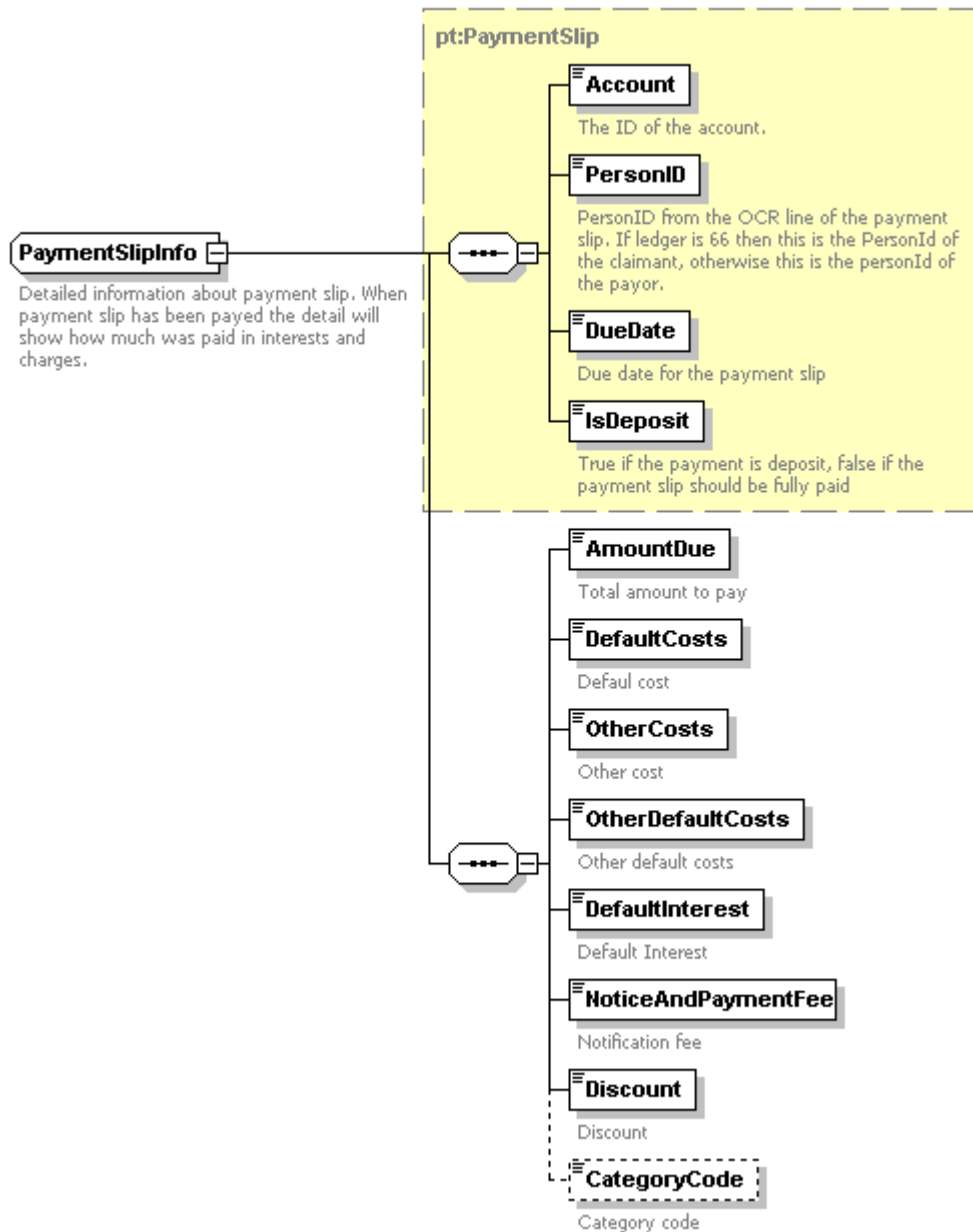
Success

A list of payments that were successfully performed. The amount of the payment is shown. ABGiro, CGiro and Transfer are identical to the actual payment, but PaymentSlip changes in the way that more detailed information about the interests and fees for the payment is given.



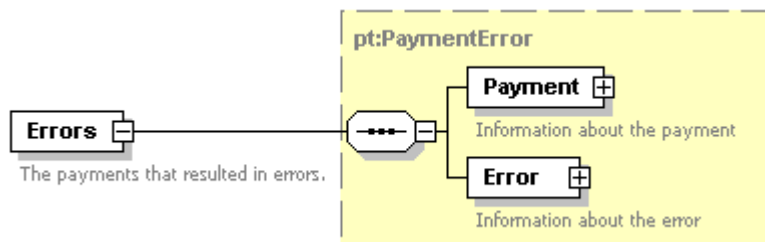
PaymentSlip (in Success)

The key in the PaymentSlip that is a part of the payment, is only a little part of the answer, as a part of PaymentSlipInfo. Added to it are details about the payment slip.



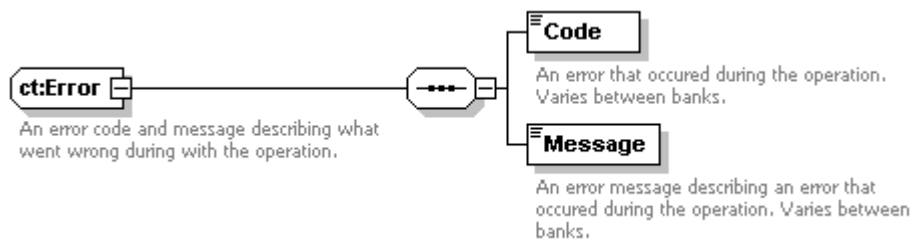
Errors

A list of the payments that an error occurred on and could therefore not be created. The item Payment is identical to the one previously described in this document.



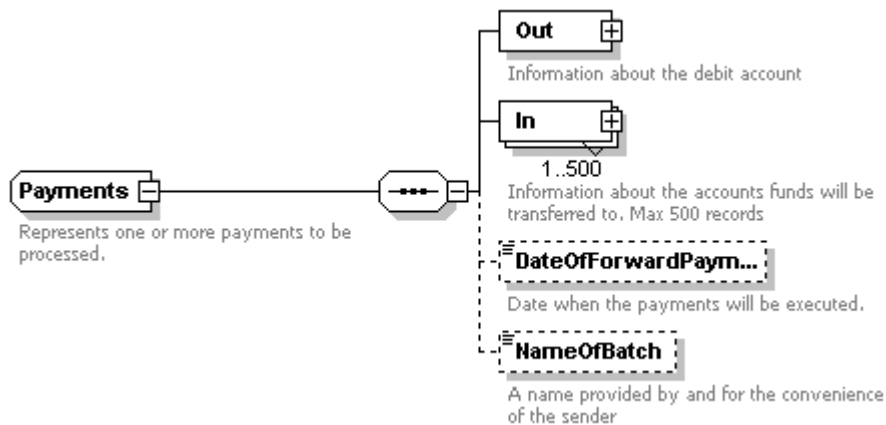
Error

A more detailed description of the error that occurred. Code is the number of the error and Message a description of the error that occurred.



Payments (DoPayments)

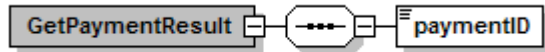
A description of how a list of payments is created. The element Payments has one field for a withdrawal and 1 to 500 possible deposits. Payments also has two attributes, RollbackOnError and IsOneToMany. RollbackOnError means that if any one of the payments fails, then all payments are cancelled. IsOneToMany indicates whether one withdrawal should be made for the entire batch or if one withdrawal should be made per deposit. A date for forward payment and a batch name can also be entered, but those elements are optional. In and Out elements are identical to the ones in creation of a single payment



The response to a batch creation is OperationID which is a string variable that is an identifier for the operation.

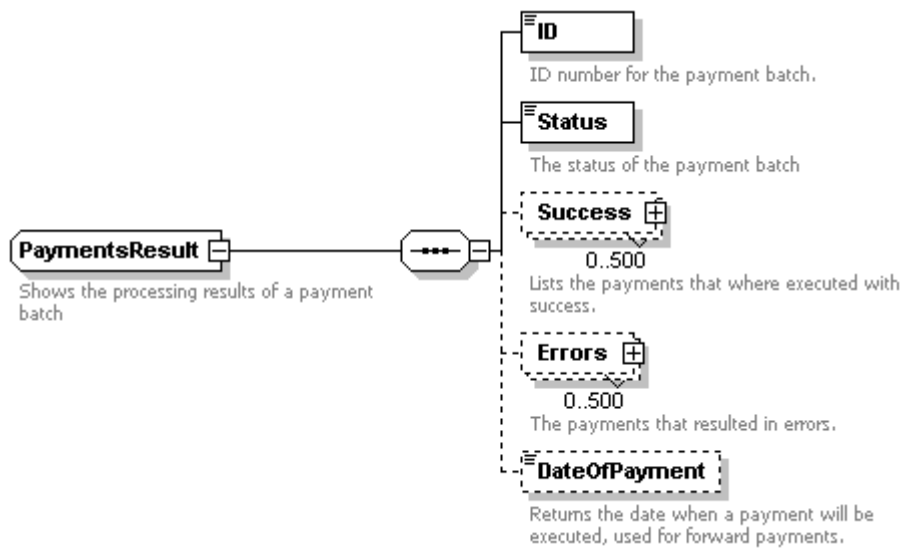
Payment query (GetPaymentResult)

An OperationID (string) is sent for the payment to be fetched.



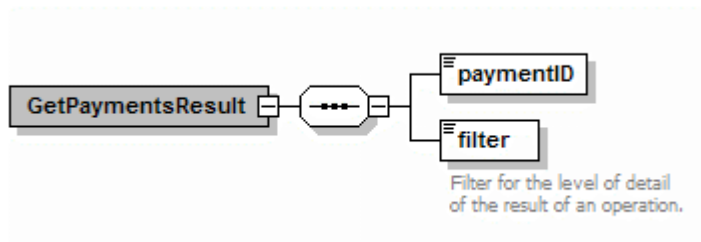
PaymentsResult

The response to the query. Same answer as to the creation of a single payment.



Payments query (GetPaymentsResult)

A query is sent that consists of a paymentID (string), and a filter that contains PaymentStatus. Using the filter, it is possible to get the status of payments (GetStatus), get all payments on errors (GetErrors), get all successful payments (GetOkay) and getting all payments (GetAll).

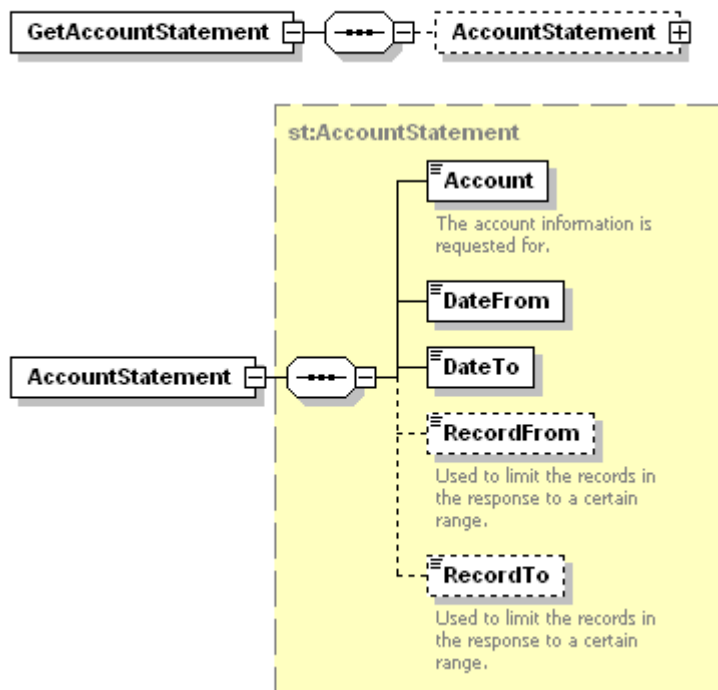


The answer to this query is the same as in `GetPaymentResult`, except that in this case it is a lot more likely that the lists are used more than in the single payment.

IcelandicOnlineStatements

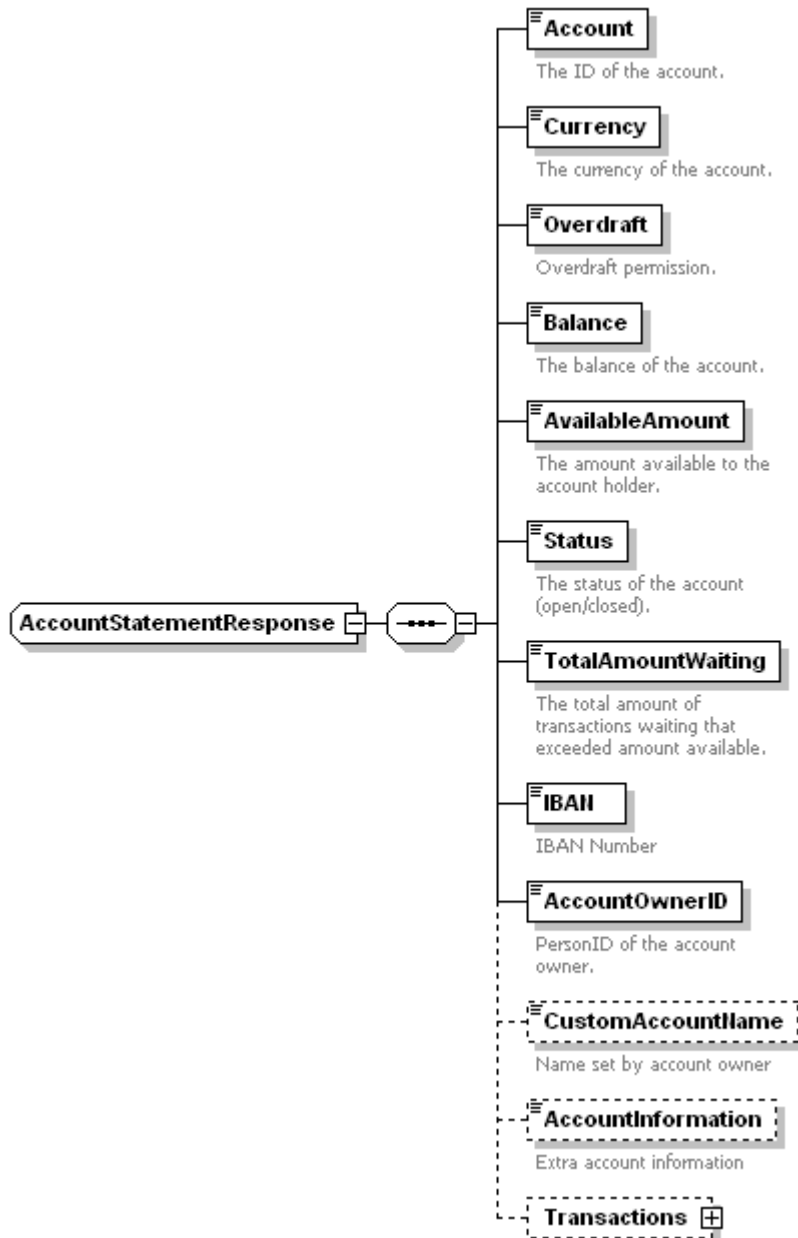
AccountStatement (GetAccountStatement):

GetAccountStatement has one element AccountStatement, which is used to perform a query on an account. The obligatory fields for this query are the account number (Account) and the start and end dates of the statement. It is also possible to select specific records from within the statement. This is added for the user, in case there are very many entries within the same period.



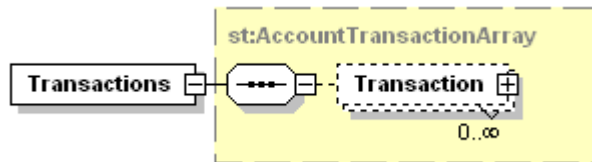
Get AccountStatementResponse

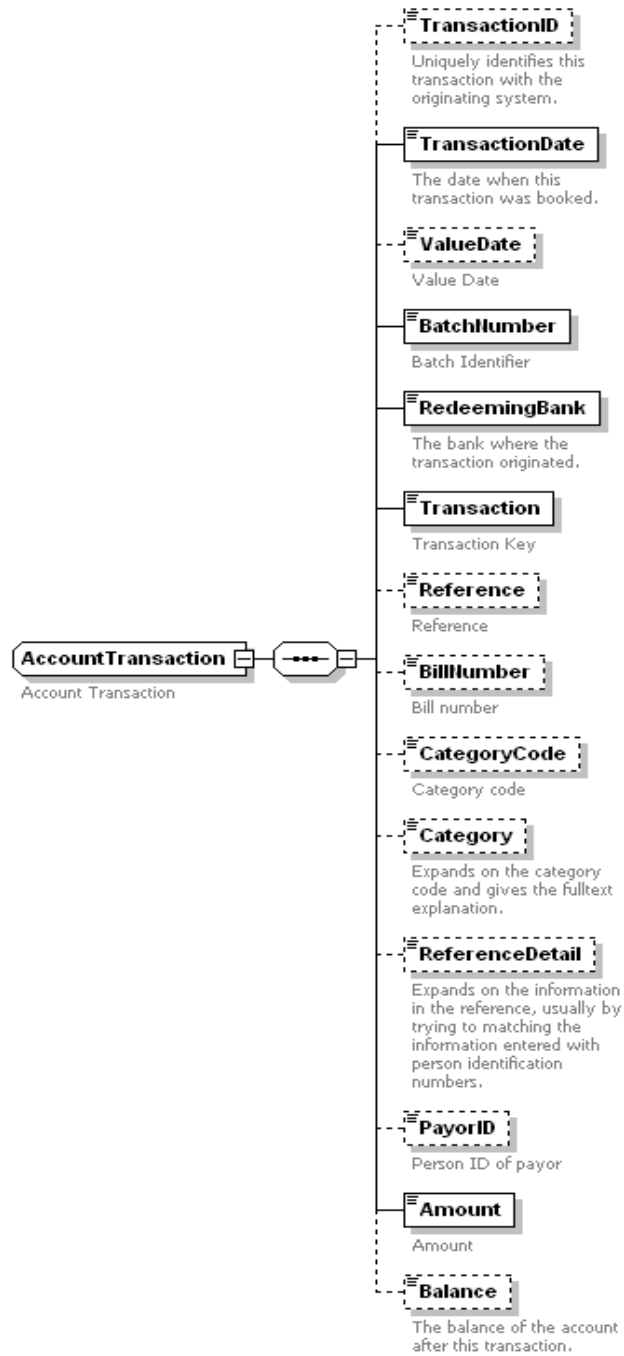
An account statement, contains information about the account itself, as well as all the account entries (Transactions), but that element is not returned if no entries were found.



Transactions

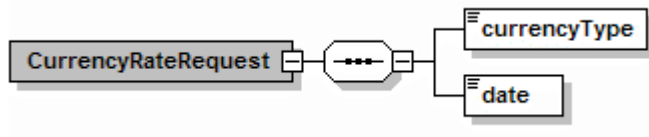
The element Transaction in the account statement contains a list of entries (AccountTransactionArray). Each account transaction has a detailed description of the information regarding an account statement entry.





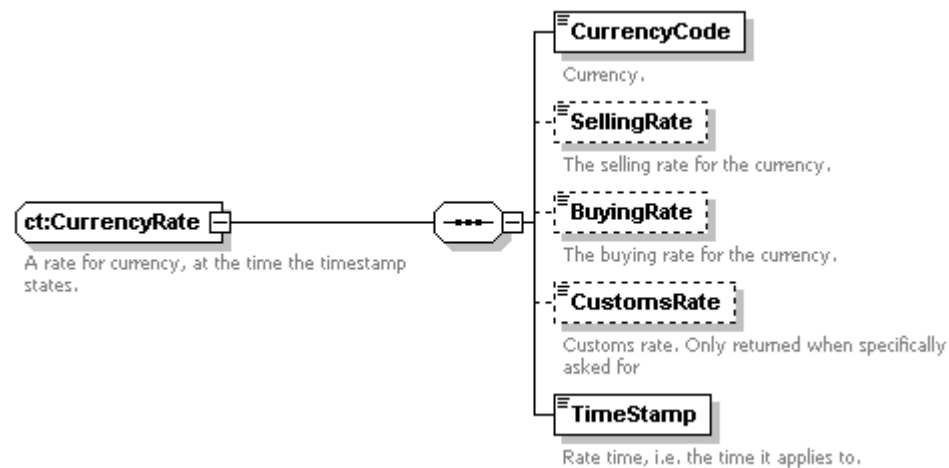
CurrencyRateRequest:

When a query is made regarding currency rate, the date of the rate in question is entered, as well as a CurrencyType element which dictates which type of rate is to be fetched.



Currency rate (CurrencyRateResponse):

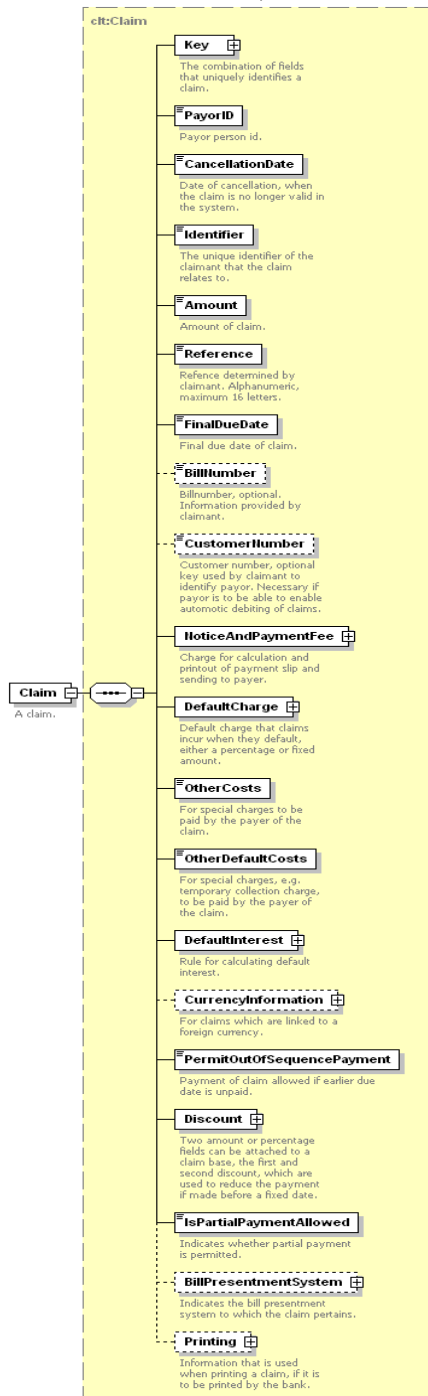
The response returns a list of CurrencyRate elements. The rate elements are made optional because if the query is made for customs rate, then only the customs rate is returned, and not the selling rate or buying rate. It's the same thing when the query is made for note rate or exchange rate.



IcelandicOnlineClaims

Claim creation/Claim modification(CreateClaims/AlterClaims)

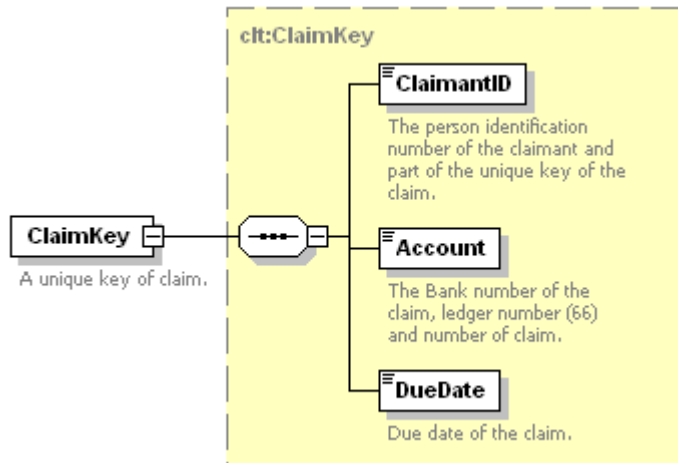
Receives a list claims, that consists of Claim elements.



A more detailed description of the elements that a claim consists of:

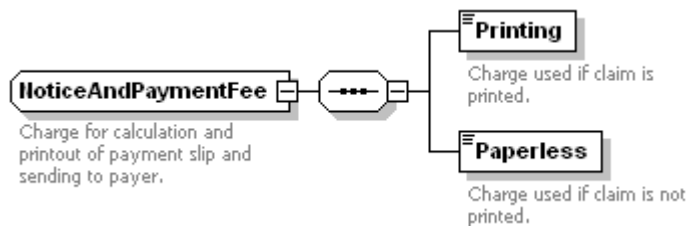
ClaimKey

A unique key for a claim, that consist of the personal ID of the claimant, the bank number of the claim and it's due date.



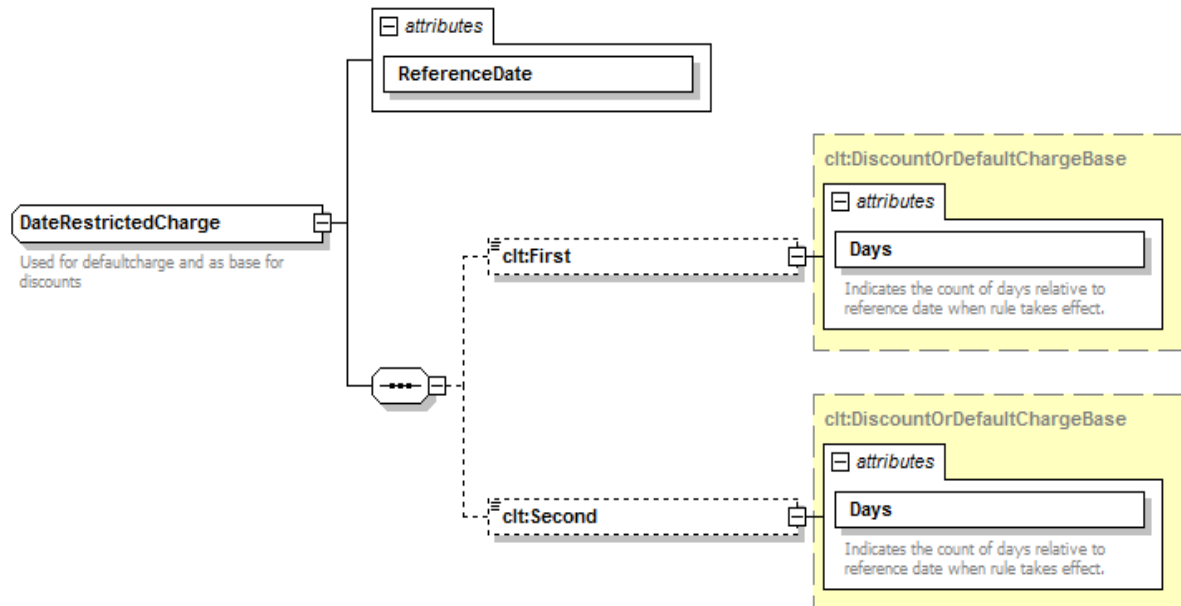
NoticeAndPaymentFee

The fee for sending a notice to the payor, but printing out the claim is optional.



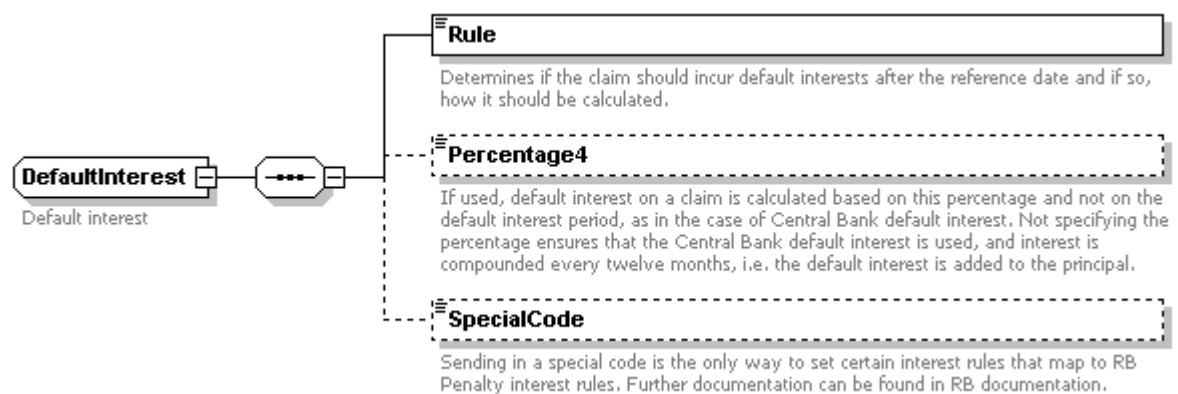
DefaultCharge

A charge that is added to claims once they end up in default. There are 2 different charges, first and second default charge. They both consist of an amount and a percentage.



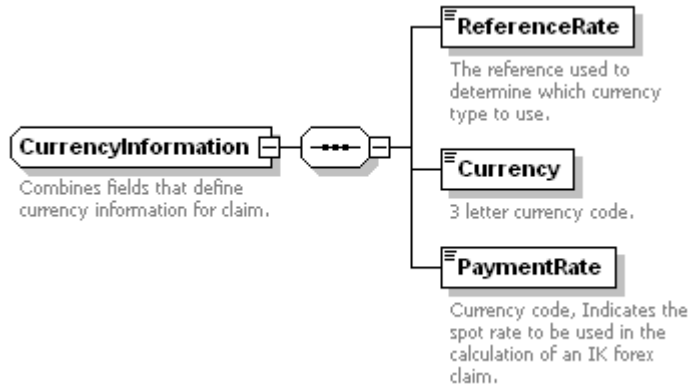
DefaultInterest

Dictates which default interest rule to use if a claim becomes default.



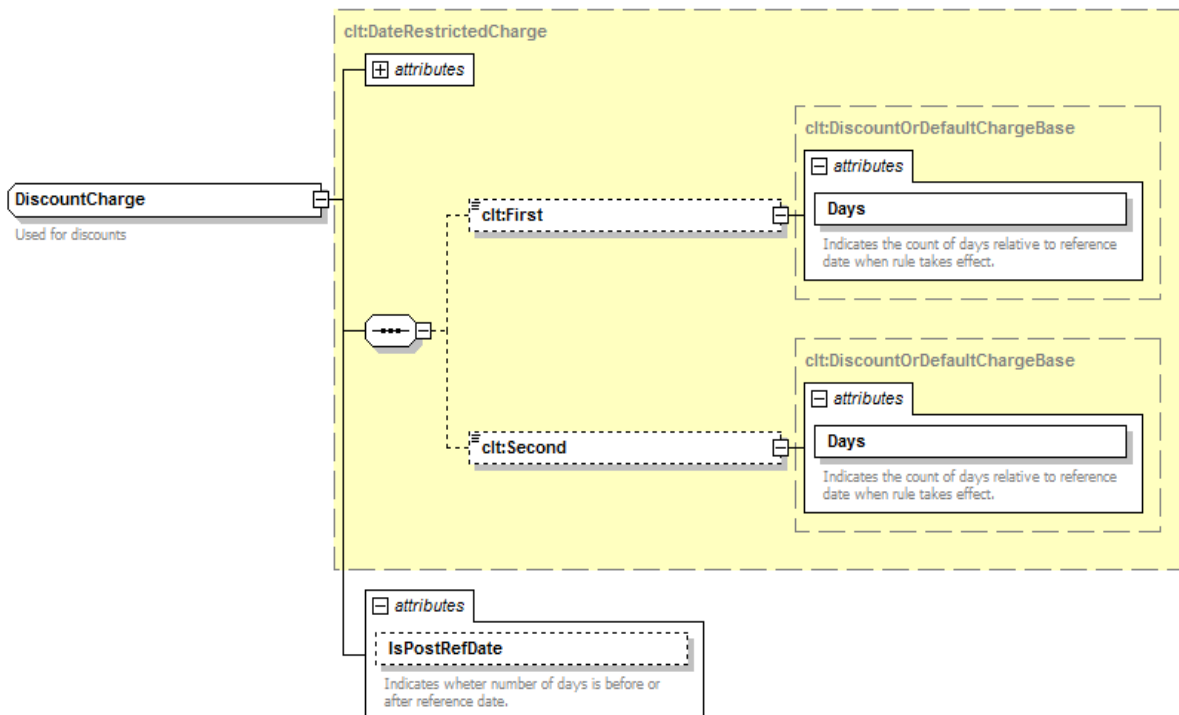
Currency information

An optional element on a claim, but is used for currency claims.



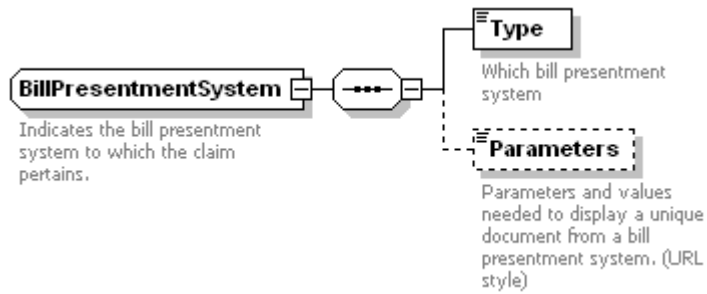
Discount

Which discounts are given on a claim, and look very similar to the default charge.



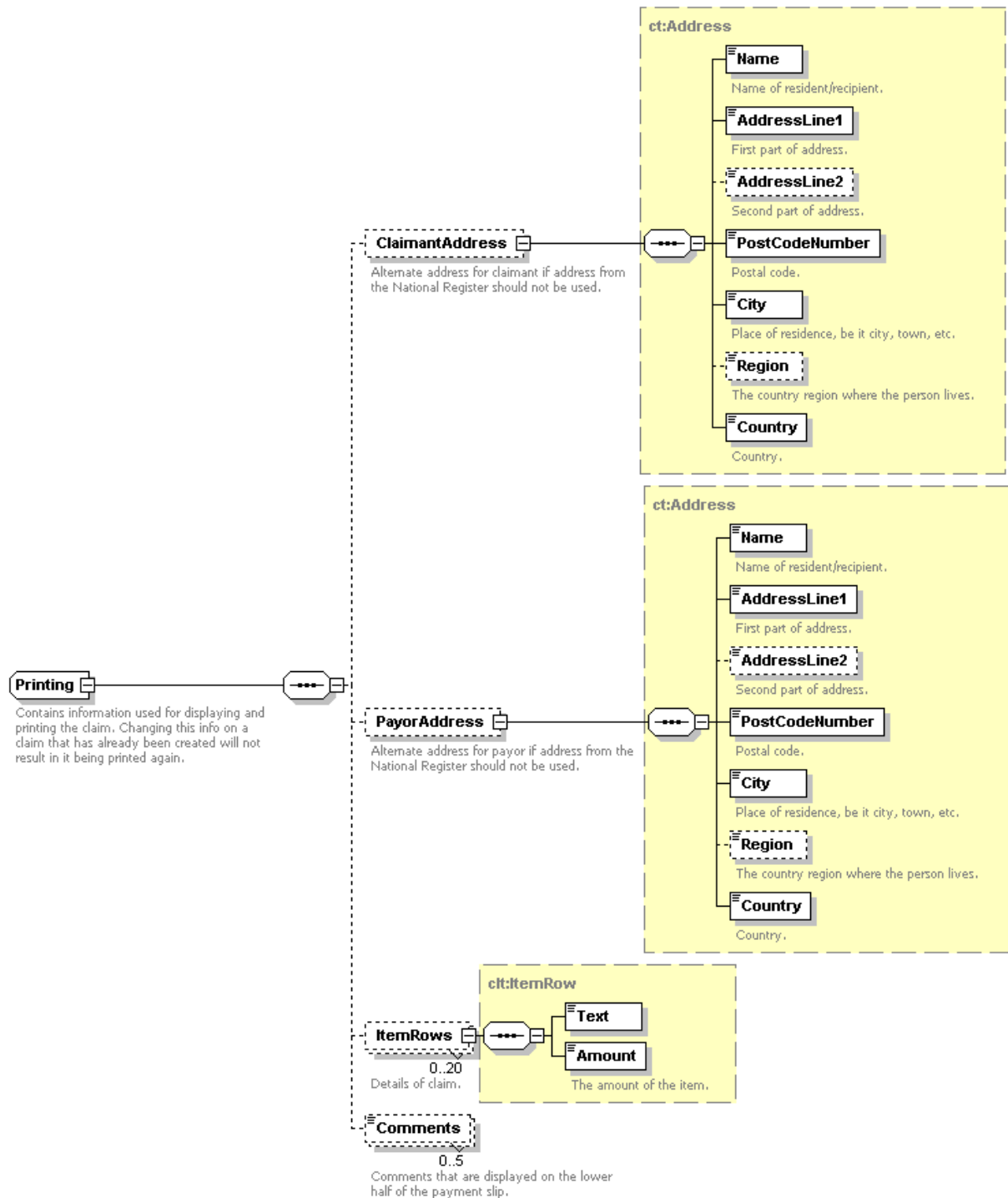
Bill Presentment System

Which presentment system to use, determined by the Type element, and a reference to specific system using parameters.



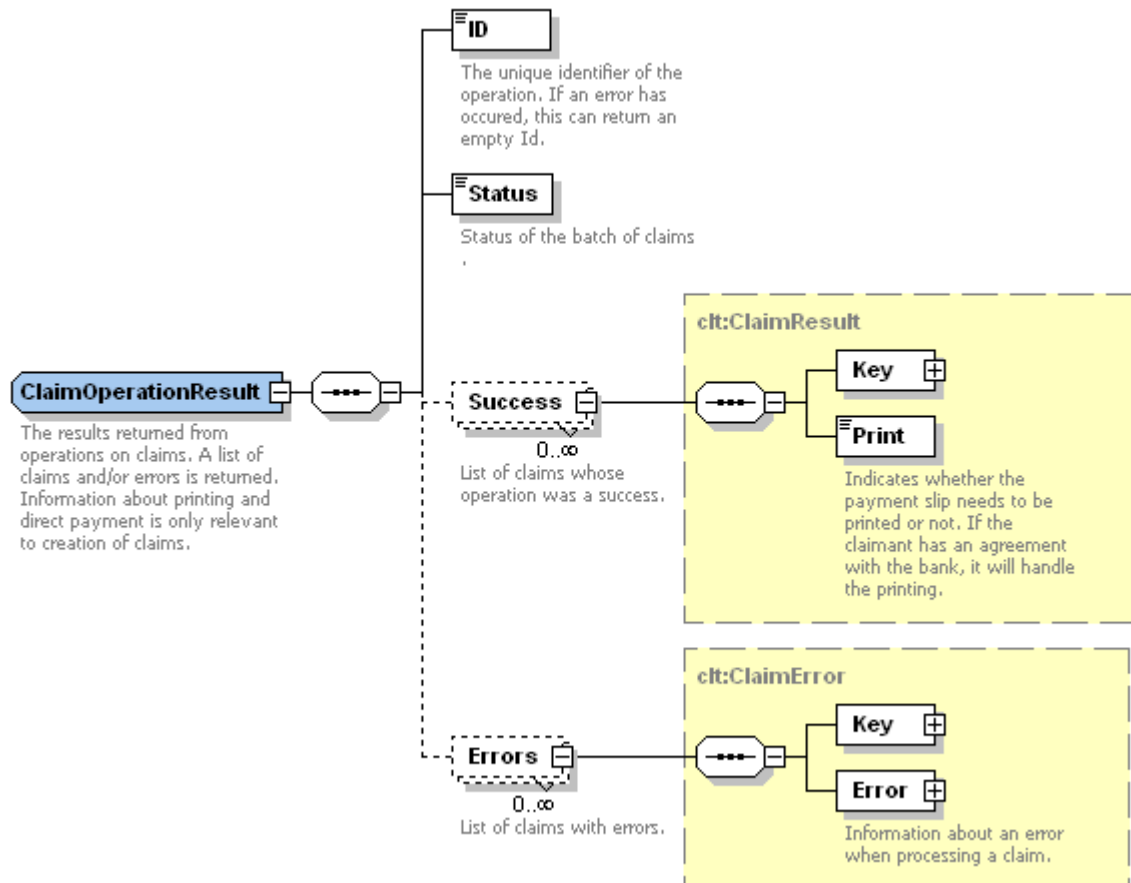
Printing

A description of how a claim is to be printed if done so by a banking institution. All elements in the printing section are optional.



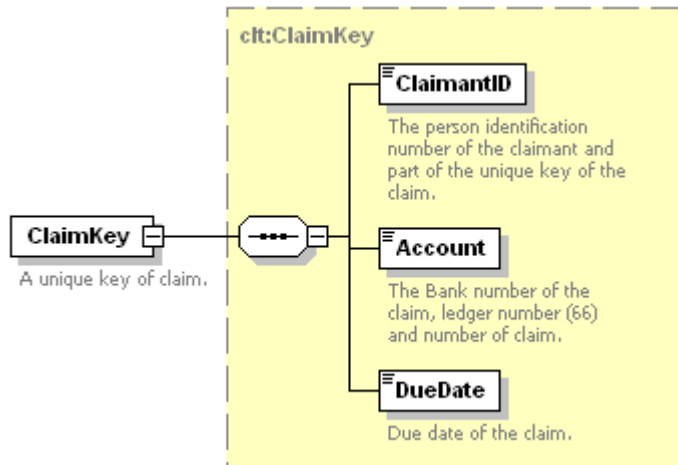
ClaimOperationResult

Information about the result of an operation. A list of claims and/or errors is returned. Information about printing and direct payment only apply when a claim is created.



CancelClaims

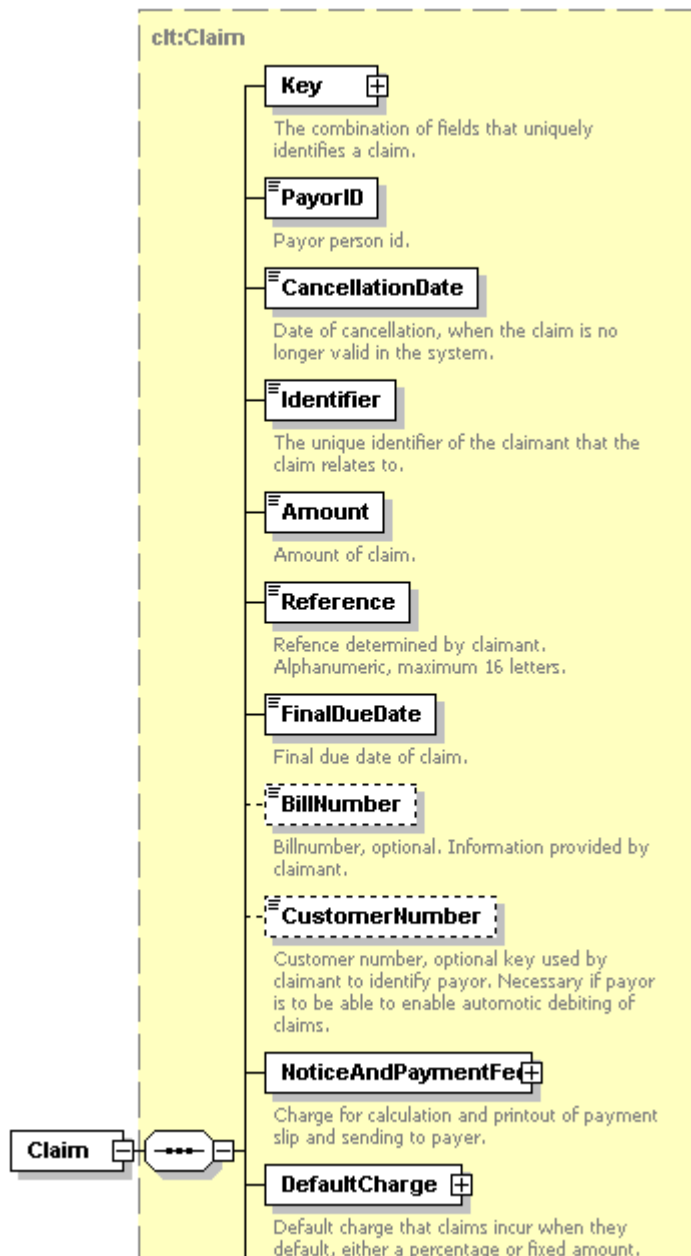
Sends in a list of keys for the claims to be cancelled. The claim key is the same as in the creation/modification of claims.



The response to CancelClaims is the same as to creation/modification, i.e. CancelClaimsResponse that contains the string OperationID.

CreateClaim/AlterClaim

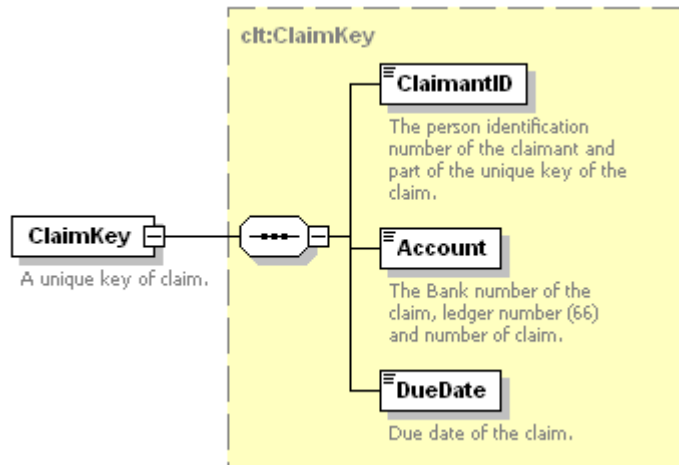
The creation and the modification of a single claim is the same as in Claims, except that here it is always a single claim that is being processed, not a list. The claim itself looks the same, but the answer to creation/modification is a ClaimOperationResult.



OtherCosts	For special charges to be paid by the payer of the claim.
OtherDefaultCosts	For special charges, e.g. temporary collection charge, to be paid by the payer of the claim.
DefaultInterest +	Rule for calculating default interest.
CurrencyInformation +	For claims which are linked to a foreign currency.
PermitOutOfSequen...	Payment of claim allowed if earlier due date is unpaid.
Discount +	Two amount or percentage fields can be attached to a claim base, the first and second discount, which are used to reduce the payment if made before a fixed date.
IsPartialPaymentAllo...	Indicates whether partial payment is permitted.
BillPresentmentSyst... +	Indicates the bill presentment system to which the claim pertains.
Printing +	Information that is used when printing a claim, if it is to be printed by the bank.

CancelClaim

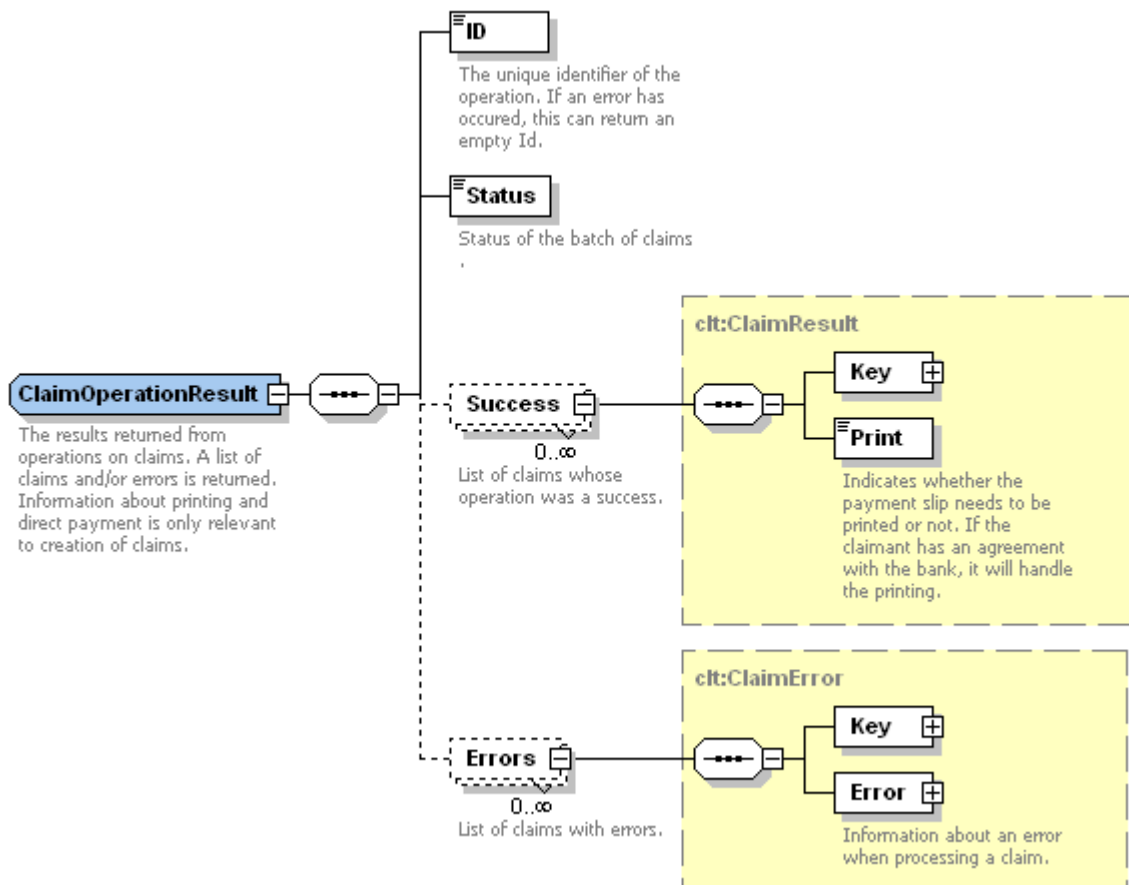
The cancellation of a claim is the same as in Claims, i.e. the key for the claim to be cancelled is sent, but here it is always a single claim that is being processed.



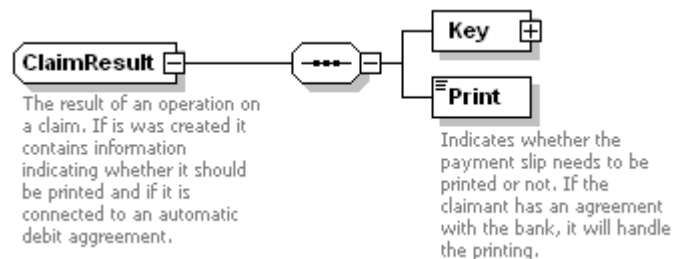
The answer to the cancellation is the same as in the create/alter operation, i.e. `ClaimOperationResult`.

GetClaimOperationResult

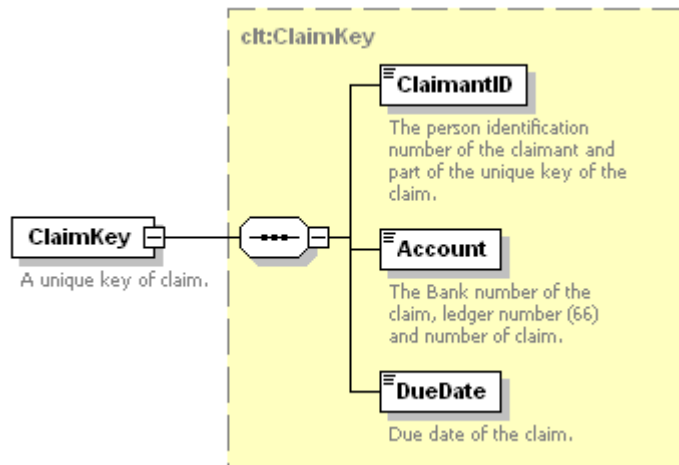
A OperationID (string) is sent for the operation that information is to be collected about.
The answer: **GetClaimOperationResultResponse** which contains **ClaimOperationResult**.



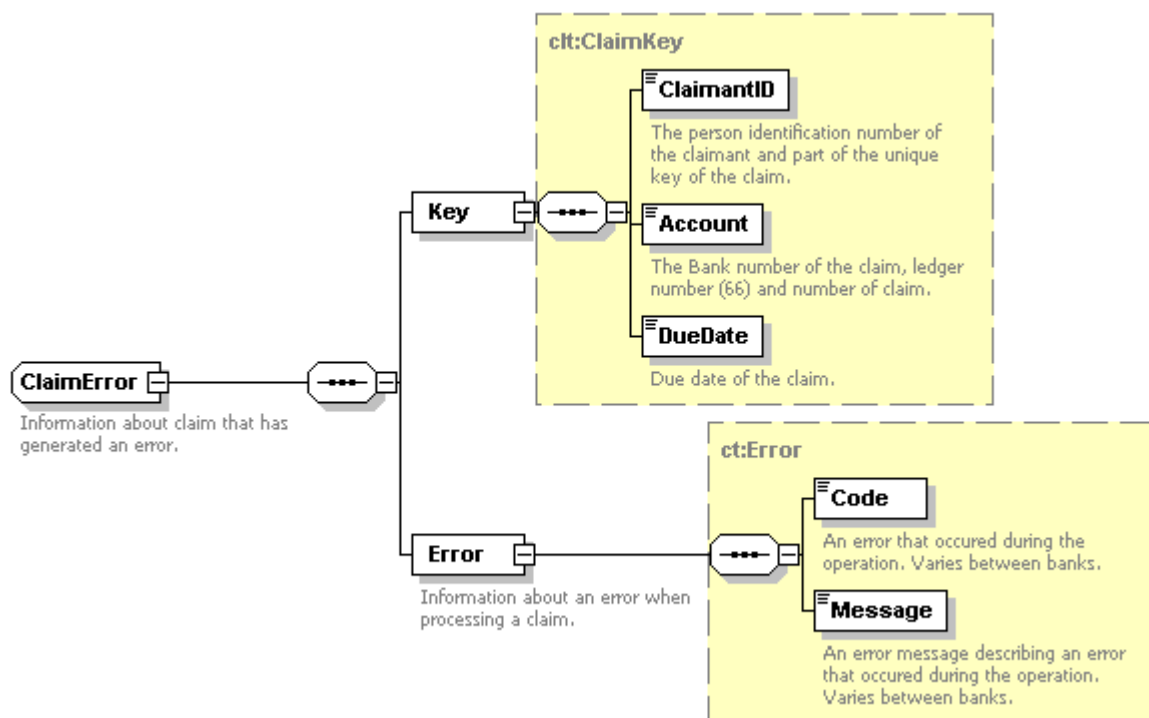
Claims is a list of the claims that were successfully created, i.e. the claim key and information on whether it is to be printed or not.



The claim key is its unique identifier. All elements are obligatory.

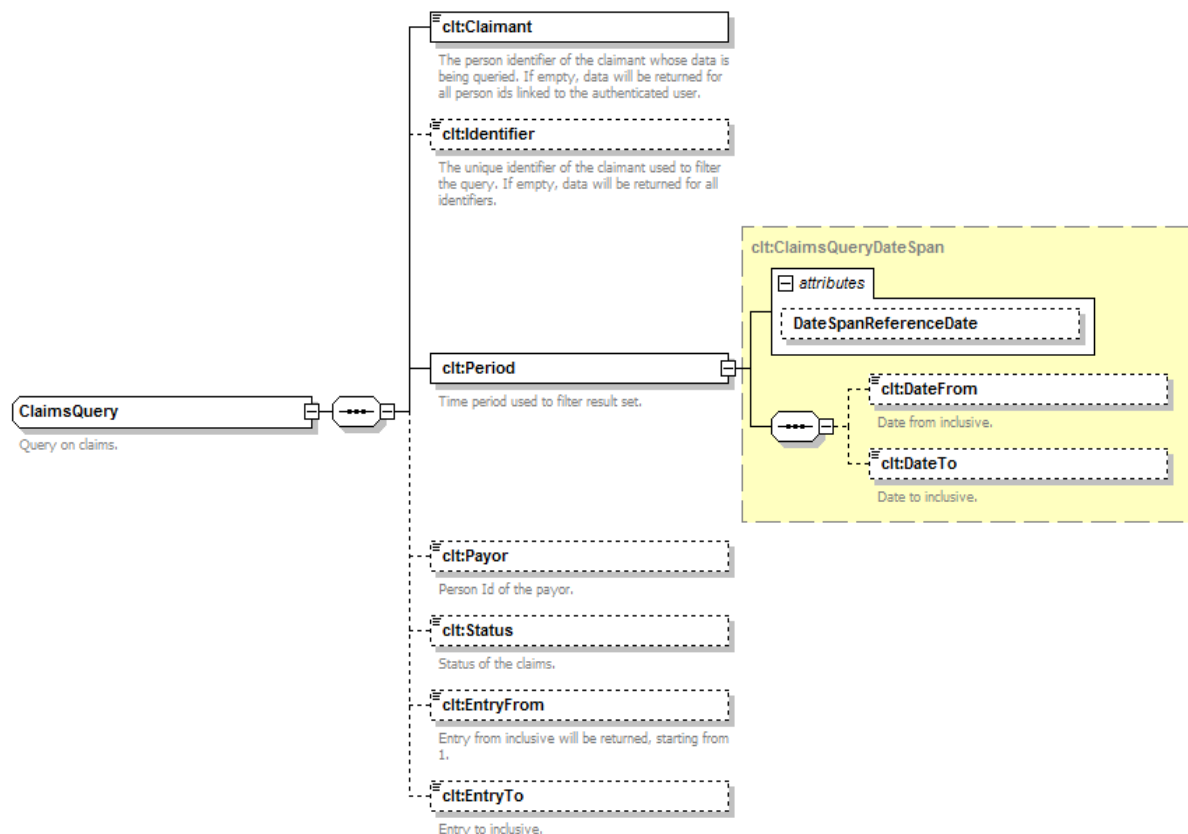


The list of errors in ClaimsResult is the same type as previously shown, e.g. in payments.

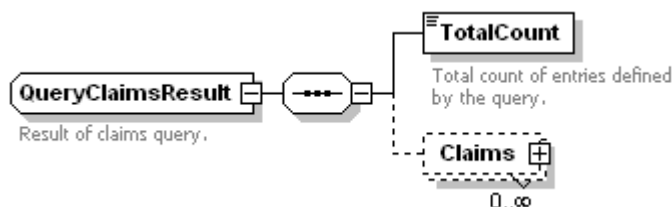


QueryClaims

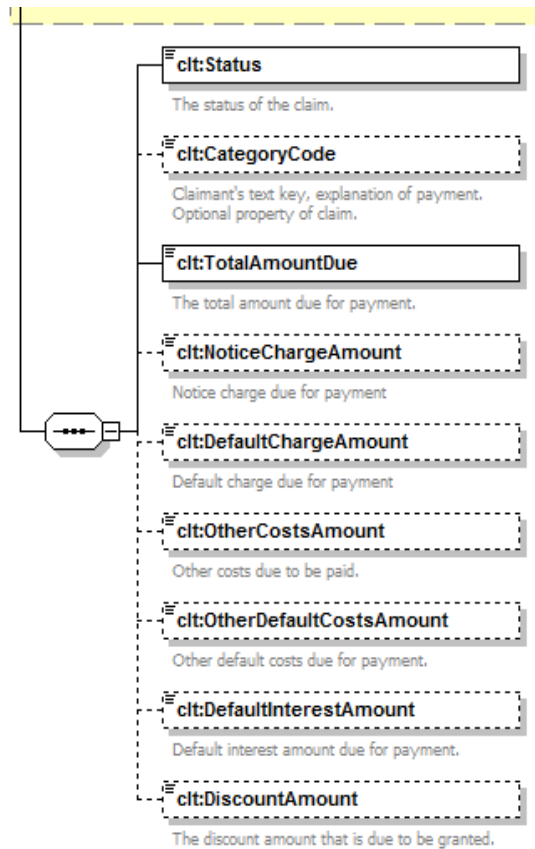
A query on claim status, which uses the ClaimsQuery type. The only obligatory element is Claimant, which is the claim owner. Other elements are mostly self-explanatory, except that it should be noted that when the result set is large only a certain number is returned and paging through the rest is called for, e.g. by specifying entries 501 through 1000 in the next query.



The answer to QueryClaims is QueryClaimsResponse, which contains QueryClaimsResult. The Claims element will only contain a subset of the claims when the result set is large and TotalCount indicates how many remain to be retrieved.

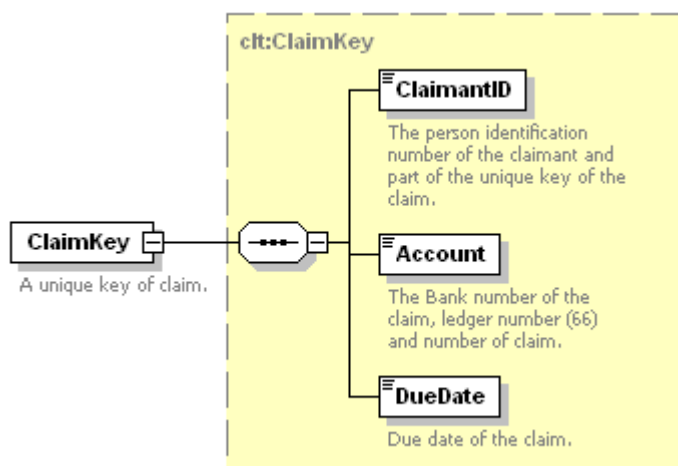


The claims in the list Claims are of the type ClaimInfo, but basically they are the same as the type Claim which has previously been described, with a few added elements. It includes more details about the costs that apply to the claim, e.g. default charge and discount. These additional elements are depicted below.

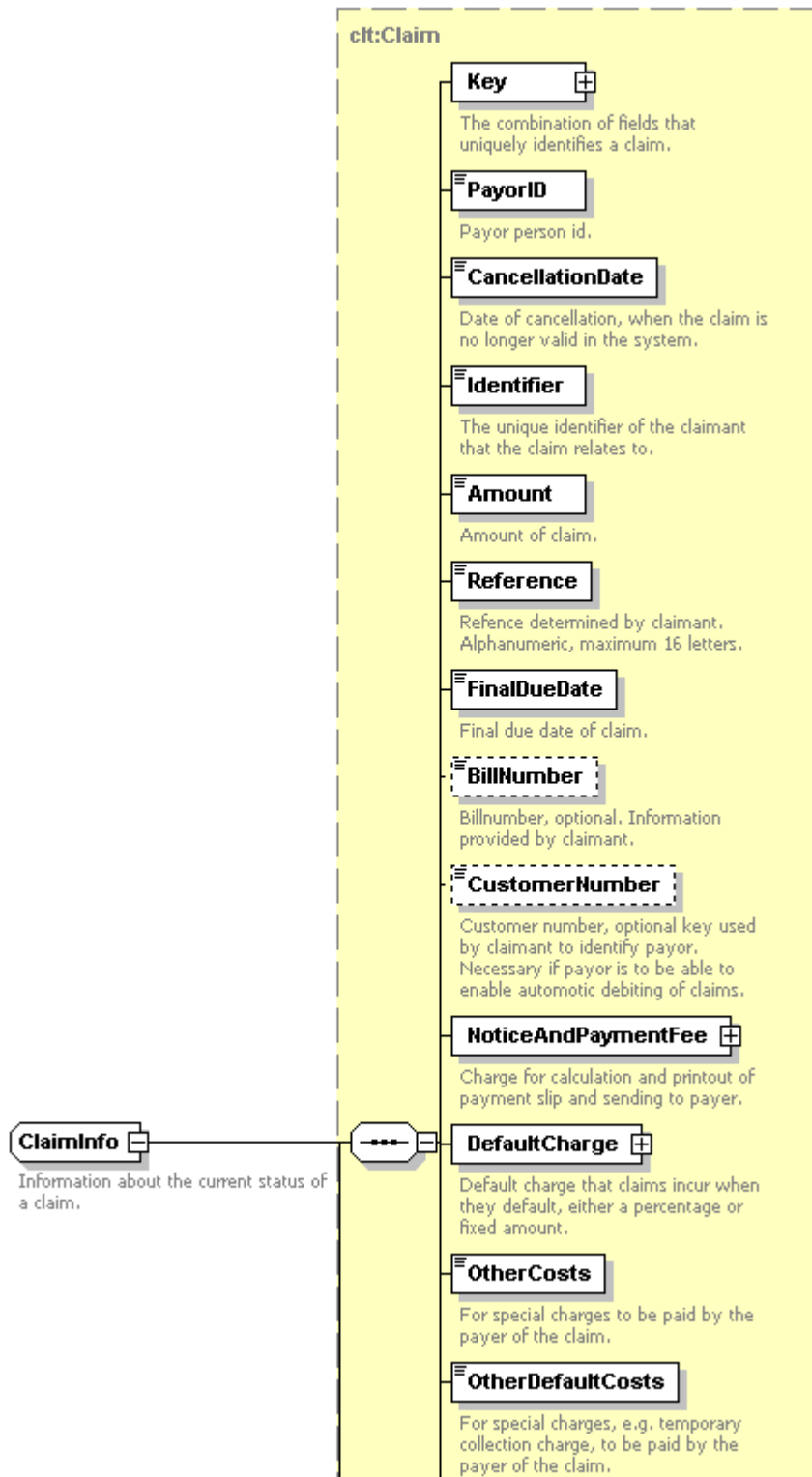


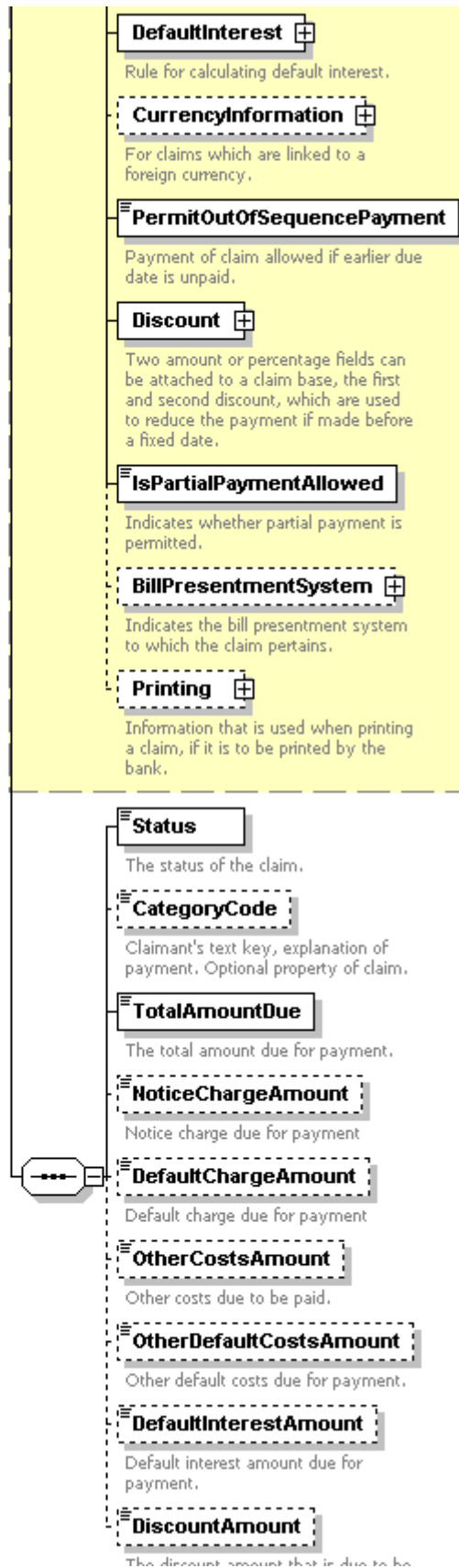
QueryClaim

A query on a single claim. Uses the claim key.



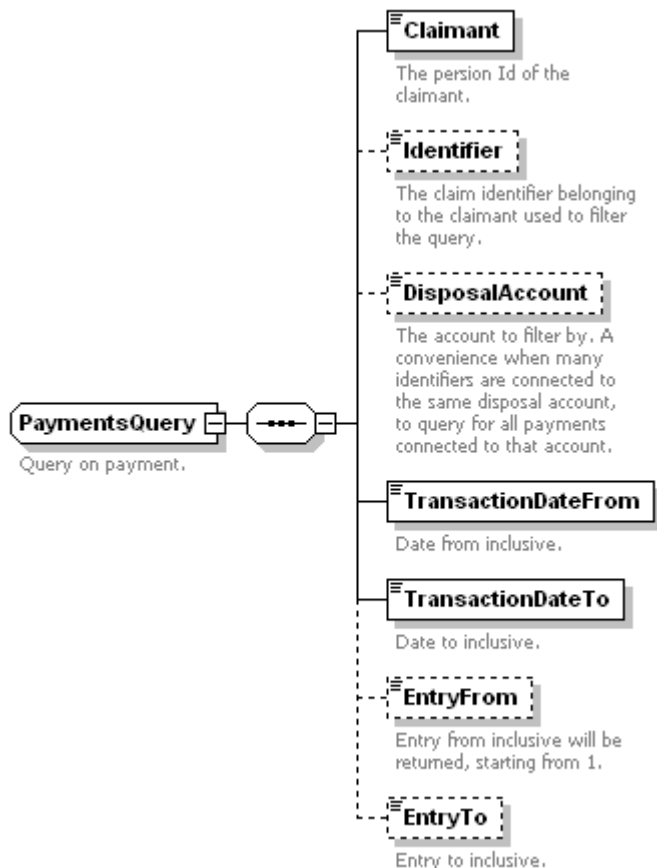
The answer to QueryClaim is QueryClaimResponse, which contains QueryClaimResult which is the type ClaimInfo, the same type as returned in QueryClaimsResult.



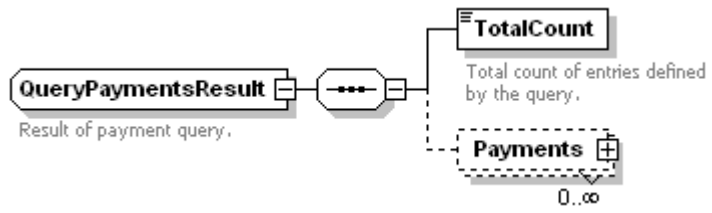


QueryPayments

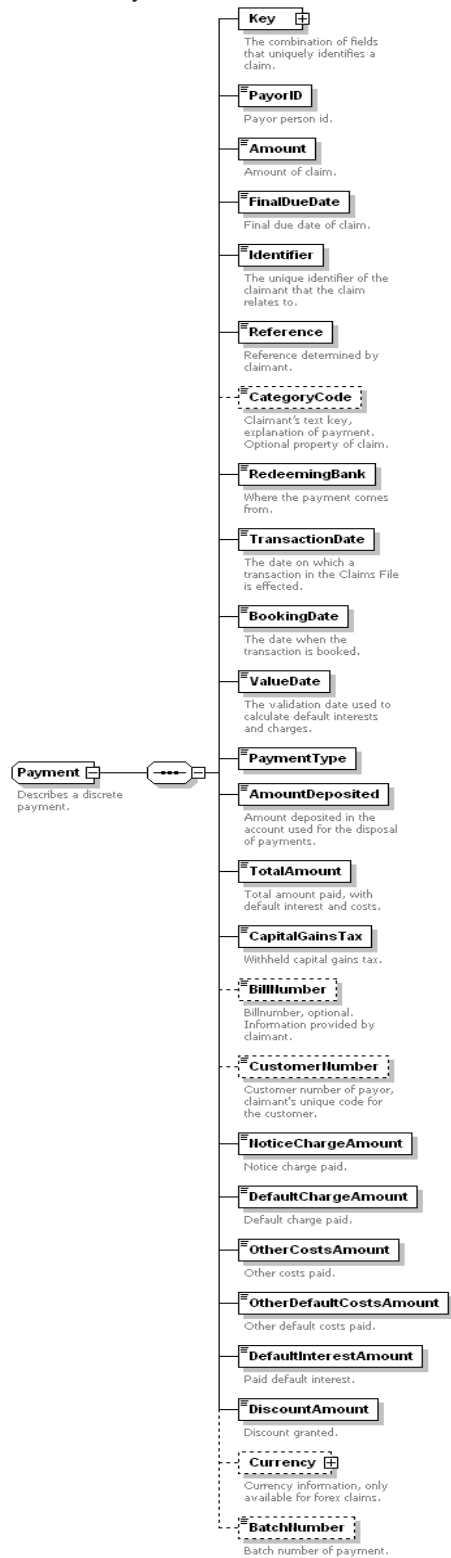
Uses the element query which is of type PaymentsQuery. It is possible to page through the result set, as previously done when querying Claims.



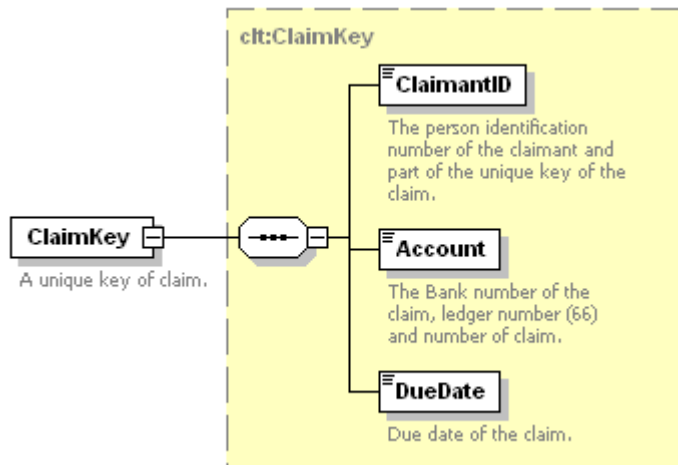
The mandatory time period set in the query is used for retrieving payments according to the transaction date (is. *hreyfingardagur*). As claim transactions are processed on the closing of each bank business day, the claimant will have to make sure to retrieve data for preceding weekends and bank holidays the next day it is accessible. Currently data for e.g. for weekends becomes available on Thursdays, the period from 21:00 on Friday to 21:00 the next Monday being processed together. The transaction date periods are normalized for the query to simplify retrieval, so transactions that occur after 21:00 are considered to belong to the next day. For example, payments occurring after 21:00 on the evening of Friday the 13th will be returned if TransactionDateFrom is set to the 14th. The answer to the QueryPayments query is QueryPaymentsResponse which has the element QueryPaymentsResult which is type QueryPaymentsResult. That contains a list of payments, as well as the total number of payments returned.



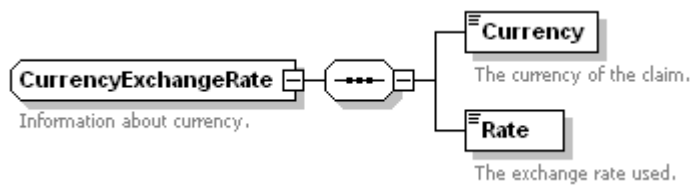
The list Payments contains a list of Payment.



The key in Payment is the same as previously shown, i.e. the claim key.



In addition, currency information is available for currency claims, but that type (**CurrencyExchangeRate**) only contains information about the currency and its rate.



IcelandicOnlineSecondaryCollectionClaims

Secondary collection agencies generally have access to the same operations on the claims being collected as original claimants, with the exception of claim creation. In addition, several specific operations are added for secondary collection companies as well as minor modifications of the queries.

AlterClaims / CancelClaims / AlterClaim / CancelClaim / GetOperationResult / QueryClaim

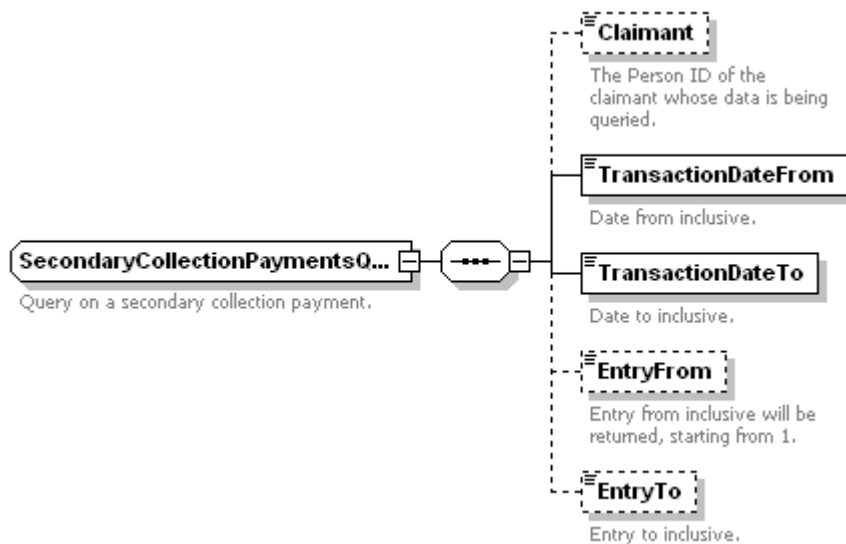
These operations differ from those in the main claim service only by the permissions the collection agency has to manipulate and query claims belonging to other claimants but having been assigned to the agency for collection.

QueryClaims

The query returns information about claims the secondary collection agency is responsible for collecting. It uses the same schema as the QueryClaims operation in the main claim service. Additionally collection agencies can either leave the Claimant field empty in order to query all collection claims or filter by a specific claimant. The collection agency uses the main claim service to query its own claims.

QuerySecondaryCollectionPayments

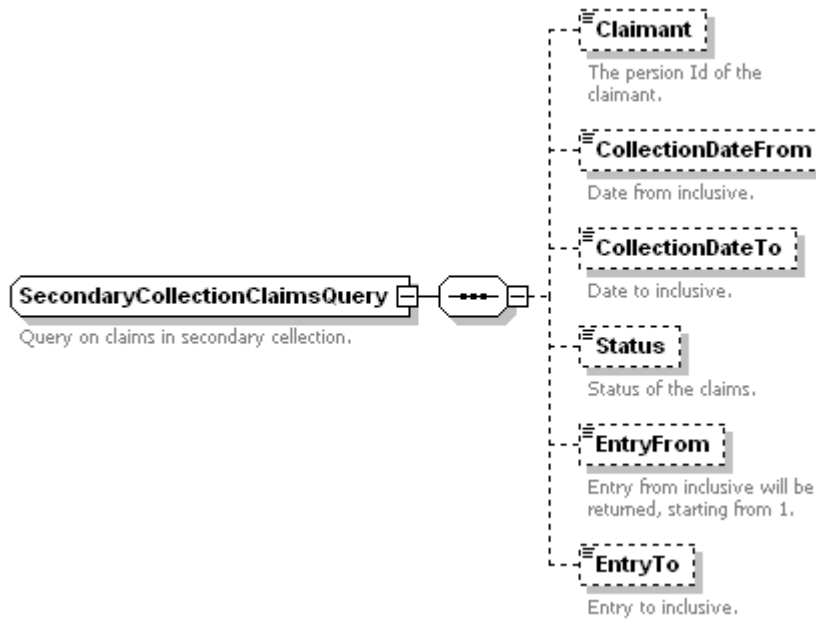
A method for secondary collection agencies to query about payments for claims they are collecting. Fetches the payments that have been processed, where it is possible to retrieve claims filtered by claimants. If no claimant is specified, all payments within the given time period are fetched.



QuerySecondaryCollectionClaims

A query that returns claims that have been assigned to the secondary collection agency during a certain time period. Used mainly for retrieving information about new claims being assigned to the agency for collection.

All elements in the query are optional, if none are used, all claims that have come into collection for this company will be returned. As with the payment query, it is possible to narrow the search down to individual claimants.



SecondaryCollectionReturnClaim

Secondary collection companies can return claims that have reached the secondary collection status. It uses a list of claim keys for the claims that are to be returned.

