UPU Interconnection Platform

Web Service Integration Guide

version 1.08

Last updated: 16 December 2024

Contact

Postal Technology Centre - Universal Postal Union Weltpoststrasse 4, 3015 Bern - Switzerland

Phone: +41 31 350 31 11

Service Desk Portal: https://support.upu.int

This document and the associated software contain proprietary information of the Universal Postal Union (UPU) and are provided under a specific agreement with eligible postal entities stipulating restrictions on use and disclosure. This document and the associated software are protected by law, including, as applicable, copyright laws.

This document and the associated software may not be, partly or as a whole, copied, disassembled, decompiled, modified or reverse-engineered without the express written permission from the UPU.

References in this publication to UPU products, applications, or services do not imply that the UPU intends to, or can make the said software, products, applications, or services, or parts thereof, available in all UPU member countries. Furthermore, no reference to a UPU software, product, application, or service is intended to state or imply that only UPU software, products, applications, or services may be used. Evaluation and verification of operation in conjunction with other products, applications, or services, except those expressly designated by the UPU, are the responsibility of the user.

Any references to particular designations of countries or territories shall not imply any endorsement or judgement by the UPU as to the legal status of such countries or territories, of their authorities and institutions or of the delimitation of their boundaries. Moreover, any references to names of specific companies or products (whether or not indicated as registered) shall not imply any intention to infringe proprietary rights, nor shall it be construed as an endorsement or recommendation on the part of the UPU.

The UPU shall not be liable for any loss or damage arising from, or directly or indirectly connected to, the use of, reference to, or reliance on the associated software or any other UPU product, application, or service, including, but not limited to, any liability arising from negligent misuse, errors, disclosure, undue transfer, loss or destruction of data that may occur.

Any trademarks mentioned or referred to in this document and the associated software are the property of their respective owners.

The information in this document, including uniform resource locators (URLs) and other website references, is subject to change without notice.

Nothing in or relating to this notice shall be deemed or interpreted as a waiver, express or implied, of the privileges and immunities enjoyed by the UPU as an intergovernmental organization and specialized agency of the United Nations.

Copyright © 1996-2024 Universal Postal Union. All rights reserved.

PTC
PostalTechnologyCentre
UNIVERSAL POSTAL UNION

Table of contents

About this document 4
- Intended audience 4
- How to use this manual 4
- Terminology 4

Introduction 5
- Overview 5
- Web Service 5
- Security 5

Web Service Interface 7
- Methods 7

Data Structures 8
- TDateAndPlace class 8
- TMO class 10
- TSearchCriteria class 13
- TPayInfo class 13
- TStatus class 16

Connecting to the UPU-IP's Web Service 22
- Verify the certificates installation 22
- How a certificate is validated 23
- The Web Service's WSDL file 24
- Generating a Web Service proxy client from the WSDL 24
- Generating an API proxy client 24

UPU-IP Client certificate request requirements 28
- Connecting to the UPU-IP API 28
- Requesting a client certificate from a non-Windows-based OS 28

Calling the UPU-IP's Web Service from a Java client 31
- Prerequisite 31
- Getting a client certificate in Java Key Store 31
- Connecting to the UPU-IP web service 32

Integration scenarios 35
- Issue a postal payment 35
- Retrieve and pay a postal payment 36
- Cancel a postal payment 38
- Reimburse a postal payment 41
- Payments to/from external partners 41

About this document

Intended audience

This document is intended for system developers or IT staff of postal organizations and their subsidiaries who are building Web Service client applications to connect their national system to the UPU Interconnection Platform (UPU-IP).

How to use this manual

For information on:

- the methods exposed by the UPU-IP's Web Service, see "Web Service Interface" on page 7.
- how to connect to the UPU-IP's Web Service, see "Connecting to the UPU-IP's Web Service " on page 22
- the various integration scenarios, see "Integration scenarios" on page 35.
- the certificate requirements and how to connect to the UPU-IP API from a non-Windows operating system, see "UPU-IP Client certificate request requirements" on page 28.
- how to call the UPU-IP Web Service from a Java client, see "Calling the UPU-IP's Web Service from a Java client" on page 31.

You may not copy, rewrite or redistribute this document in any form. To do so is a violation of international copyright laws. However, the Postal Technology Centre welcomes your input. For queries or service requests, you can raise them at https://support.upu.int.

Terminology

The following key terms are used throughout this document:

Term	Description
Postal payment	A wide range of payment services and instructions, used instead of "money order"
WS	Web Service
Business partner	Refers to organizations, Designated Operators (DOs), or third-party commercial entities that connect or use UPU-IP to exchange payment and payment-related messages with other business partners. A business partner performs various postal payment transactions: sending, receiving, and others.

Introduction

Overview

The **UPU Interconnection Platform (UPU-IP)** enables near real-time exchange of payment and payment related messages between business partners.

Through the UPU-IP, the following types of services are possible:

- **Urgent service**: As soon as payment is issued to the UPU-IP, the paying organization can retrieve it. The Web Service protocol enables urgent, on demand issuing and payout.
- **Payment anywhere**: Payout can take place at any connected payout agent of the beneficiary's choice such as a post office or an external agent.
- **Connection of external non-DO partners**: Partners outside the UPU network can connect to the UPU-IP for the issuing and payout of postal payments.

A bridge links the UPU-IP to the UPU FTP/EDI network, enabling business partners exchanging via the IFS FTP network or IFS EDI messages network to exchange with business partners connected to the UPU-IP. The resulting service is, however, non-urgent as it is dependent on the Posts' individual FTP upload/download schedules.

Web Service

The UPU-IP has a **Web Service Interface** which is exposed to business partners. Business partners include not just designated postal operators but also permitted remittance service providers external to the UPU network. Using the exposed Web Service (WS), business partners are able to build their own web client applications to exchange payment messages via the UPU-IP. This results in a more open network of remittance services.

Security

Message Level Security

The UPU-IP WS is secured using Message Level Security (MLS). Security credentials and claims are encapsulated with every message providing end-to-end security independent of the transport protocol.

Web service request and response messages to and from the UPU-IP are digitally signed and encrypted. By using x.509 certificates, authenticity is ensured while the use of digital signatures at the application level ensures the non-repudiation of messages. MLS also has the advantage that the message may be routed through intermediary systems without loss of security.

Transport Level Security

Transport Level Security (TLS) is an evolved version of Secure Socket Layer (SSL). Using x.509 certificate, TLS ensures end-to-end security of data sent between applications, avoiding possible eavesdropping or alteration of the content being transported.

Web service requests and responses to and from UPU-IP are authenticated with x.509 certificates and encrypted.

Web Service Interface

The **Web Service Interface** is exposed to business partners to allow them to exchange postal payment related messages via the UPU Interconnection Platform (UPU-IP). **Business partners** are organizations that use a national system and wish to connect this system to the UPU-IP by developing a WS client. External payment networks, non-designated operators (DOs), money transfer operators, etc. can also exchange payment messages through the UPU-IP.

Methods

For a complete list of the methods that the Web Service Interface exposes, see the UPU-IP API documentation at https://upu.api.post/upu-ip_api/api/index.html.

Data Structures

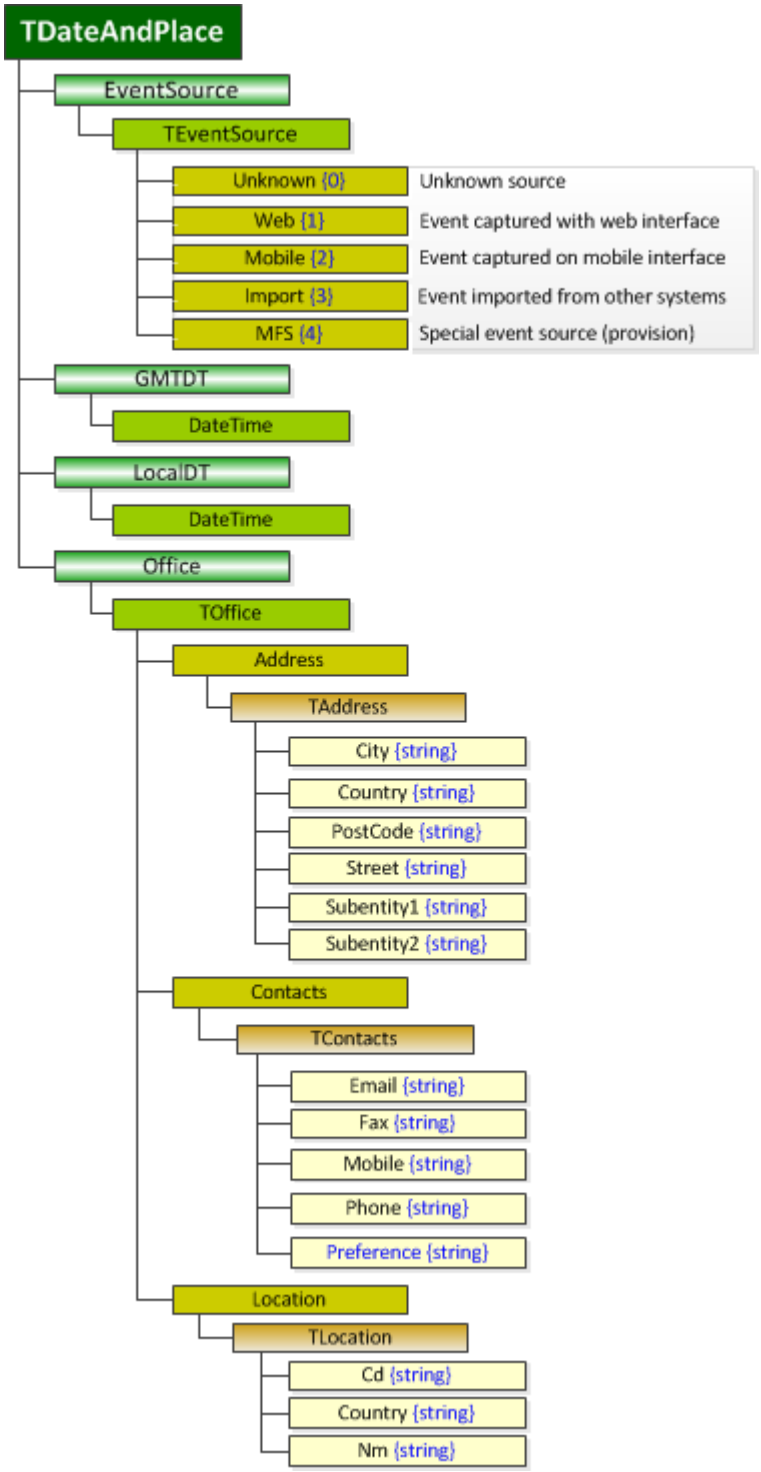
This section uses diagrams to show the data structures passed as arguments to the **Web Service Interface methods** when a business partner's WS client application makes a call to the UPU-IP using the methods described in the previous chapter.

For more information on parameters, refer to the UPU-IP API documentation at https://upu.api.-post/upu-ip_api/api/index.html.

TDateAndPlace class

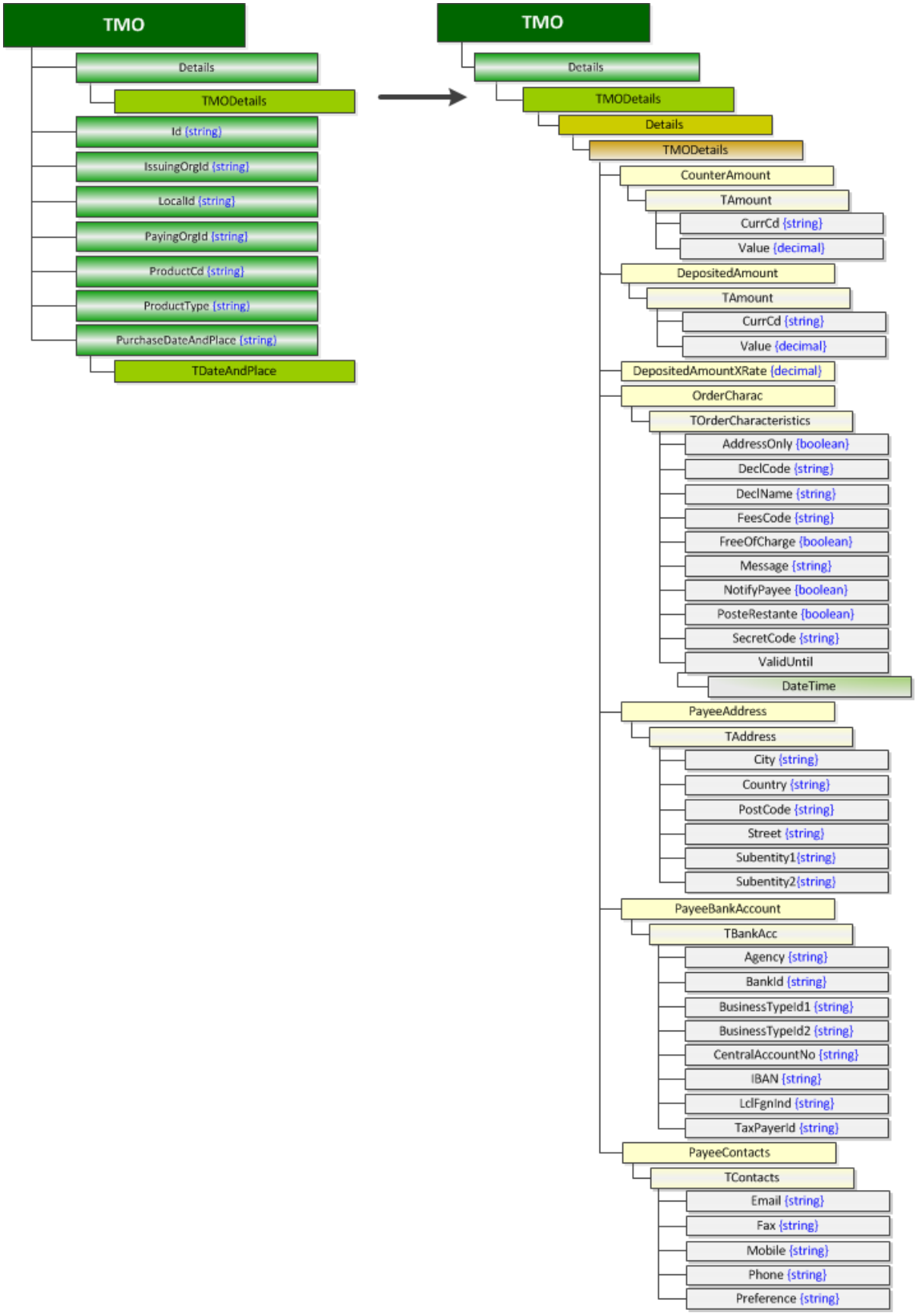
The `TDateAndPlace` class describes the date and place that an event took place. This parameter is used when:

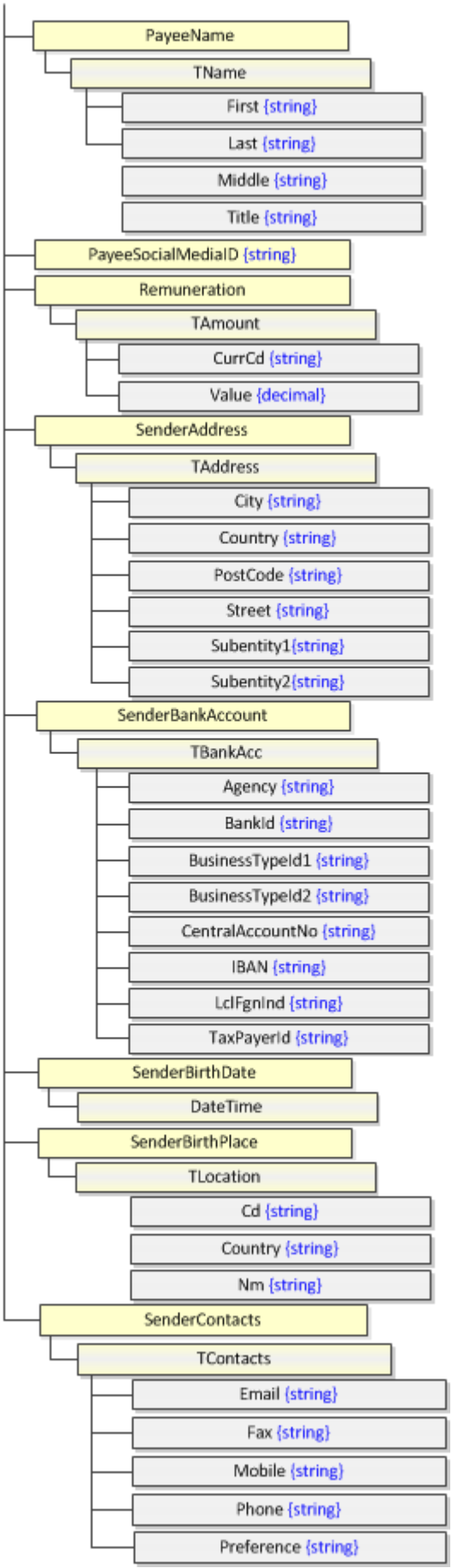
- issuing a postal payment using the Issue() method
- canceling a previously-issued postal payment using the Cancel() method
- paying a postal payment using the Pay() method
- reimbursing a postal payment using the Reimburse() method

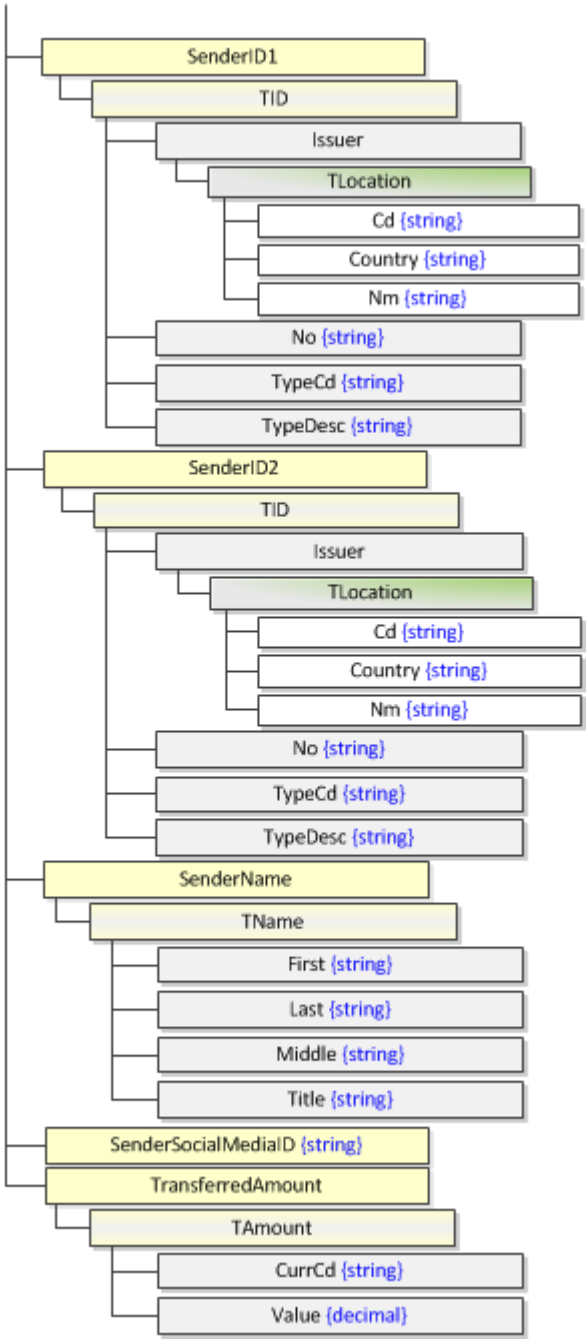

TDateAndPlace
EventSource
TEventSource
Unknown {0}
Unknown source
Web {1}
Event captured with web interface
Mobile {2}
Event captured on mobile interface
Import {3}
Event imported from other systems
MFS {4}
Special event source (provision)
GMTDT
DateTime
LocalDT
DateTime
Office
TOffice
Address
TAddress
City {string}
Country {string}
PostCode {string}
Street {string}
Subentity1 {string}
Subentity2 {string}
Contacts
TContacts
Email {string}
Fax {string}
Mobile {string}
Phone {string}
Preference {string}
Location
TLocation
Cd {string}
Country {string}
Nm {string}

TMO class

The `TMO` class contains information about the postal payment and is used in the Issue() method. The TMO class exposes several properties and also makes use of the `TDateAndPlace` class.



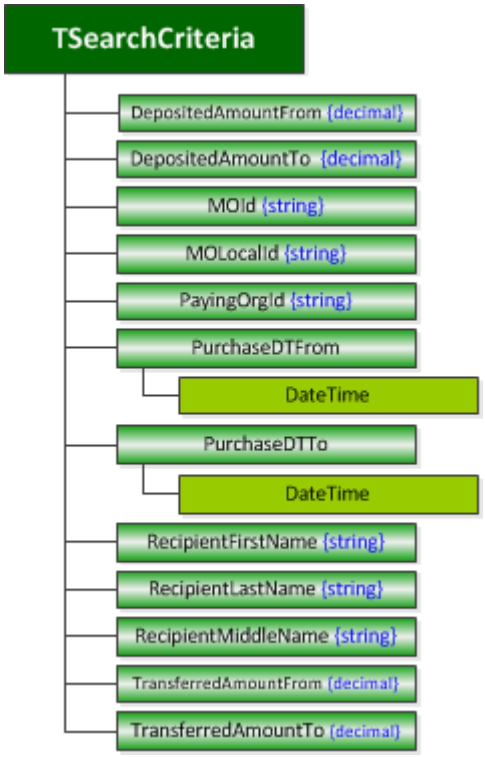

PayeeName
TName
First {string}
Last {string}
Middle {string}
Title {string}
PayeeSocialMediaID {string}
Remuneration
TAmount
CurrCd {string}
Value {decimal}
SenderAddress
TAddress
City {string}
Country {string}
PostCode {string}
Street {string}
Subentity1{string}
Subentity2{string}
SenderBankAccount
TBankAcc
Agency {string}
BankId {string}
BusinessTypeId1 {string}
BusinessTypeId2 {string}
CentralAccountNo {string}
IBAN {string}
LclFgnInd {string}
TaxPayerId {string}
SenderBirthDate
DateTime
SenderBirthPlace
TLocation
Cd {string}
Country {string}
Nm {string}
SenderContacts
TContacts
Email {string}
Fax {string}
Mobile {string}
Phone {string}
Preference {string}



For details on the `TDateAndPlace` class, see "TDateAndPlace class" on page 8.

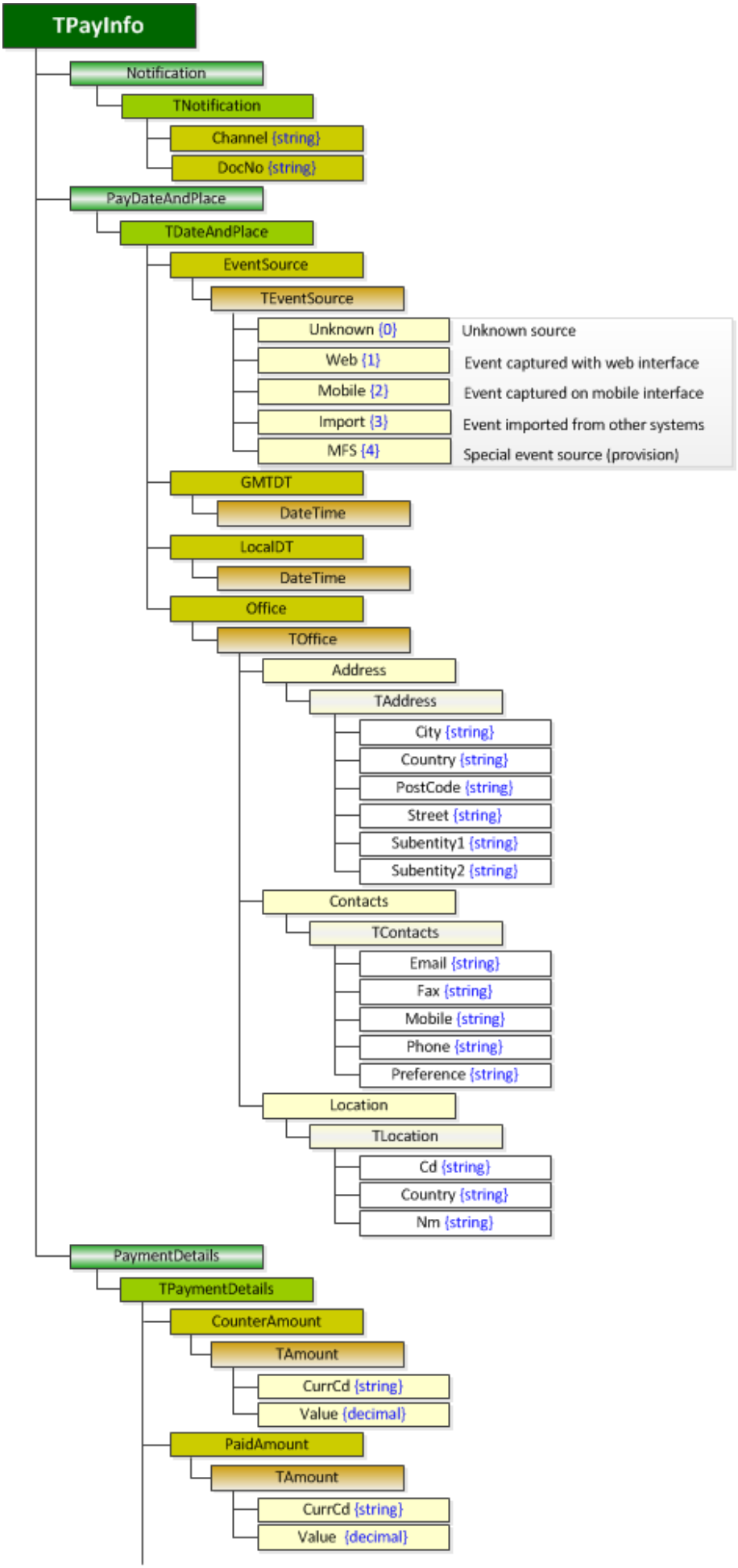
TSearchCriteria class

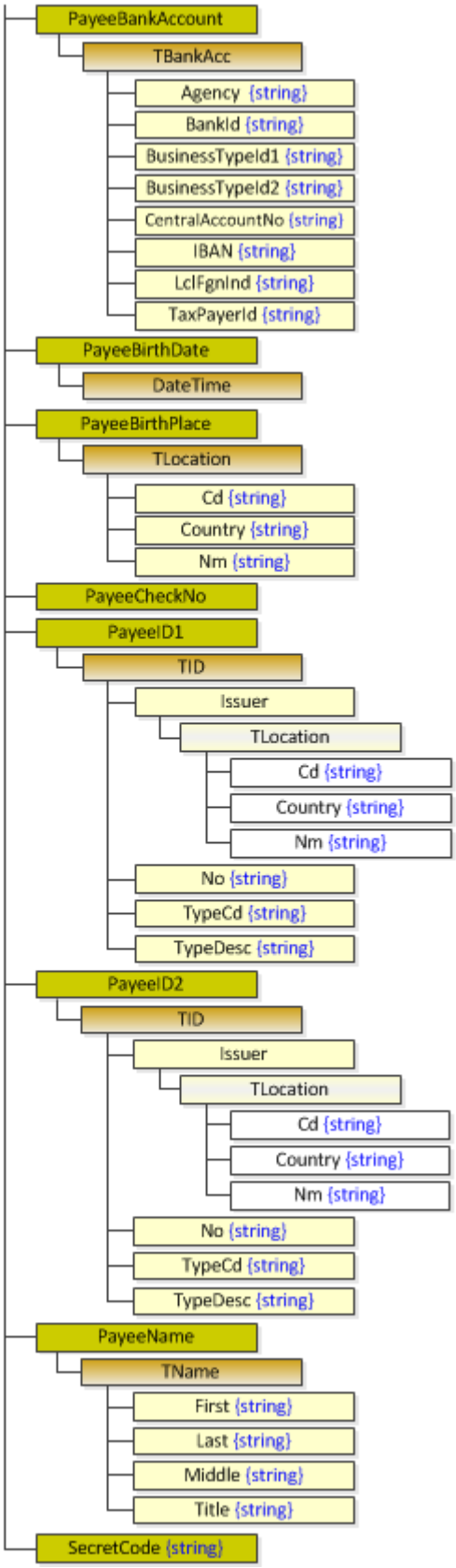
The `TSearchCriteria` is passed as a parameter when the calling party wants to obtain postal payment information using the Get() method.



TPayInfo class

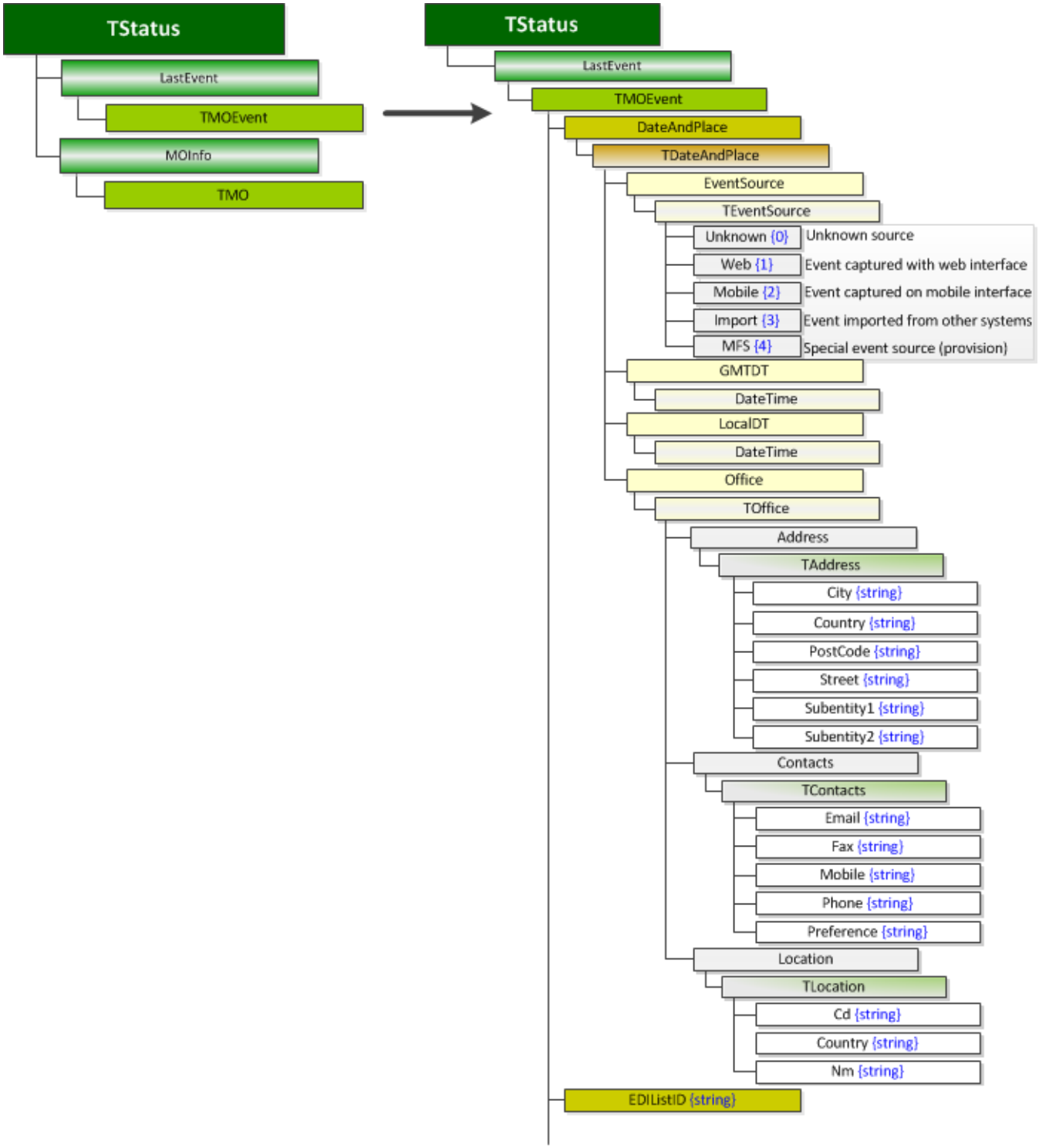
The `TPayInfo` class is passed as a parameter when the Pay() or Reimburse() methods are used for paying out or reimbursing a postal payment.

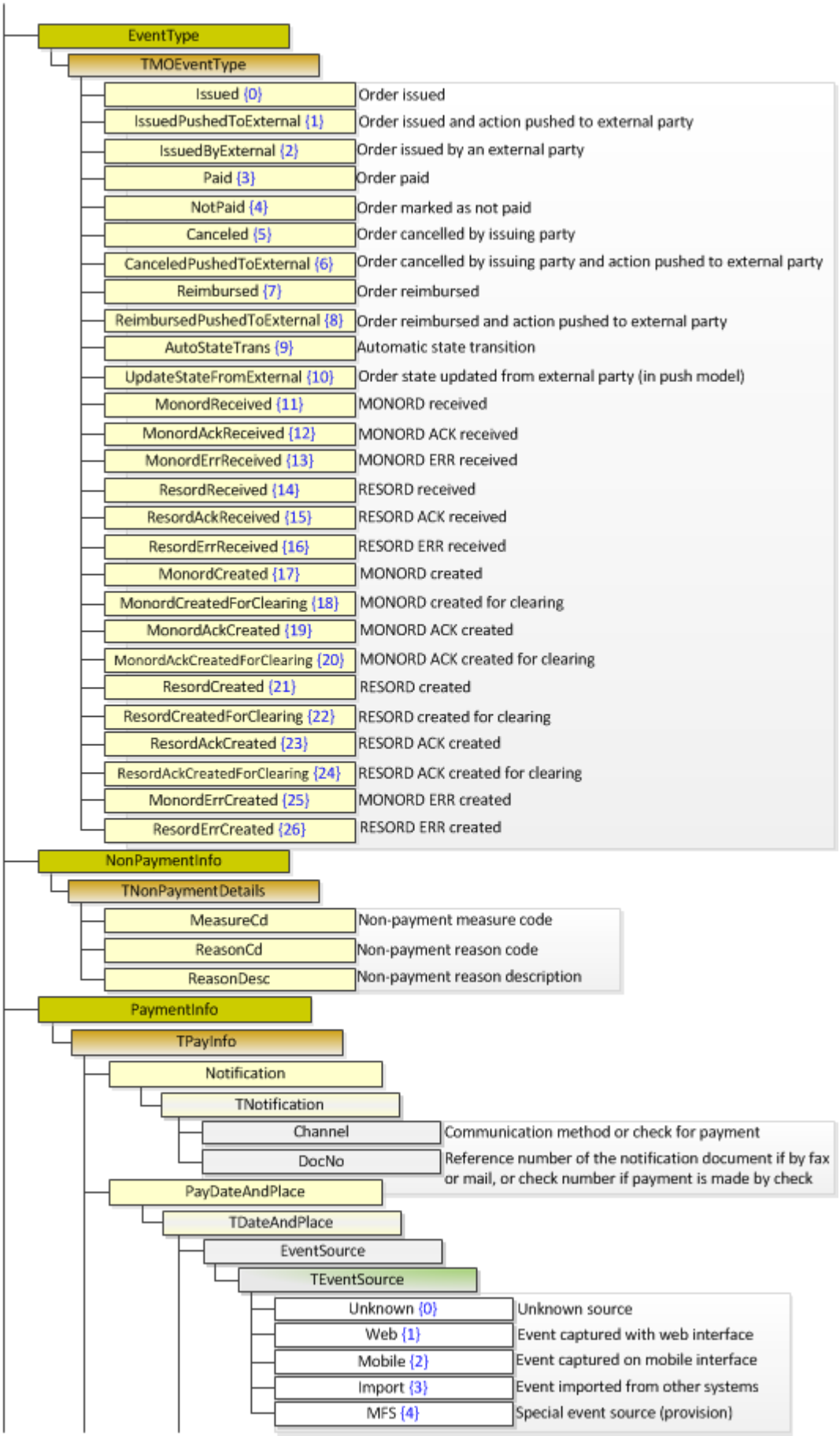

TPayInfo
Notification
TNotification
Channel {string}
DocNo {string}
PayDateAndPlace
TDateAndPlace
EventSource
TEventSource
Unknown {0}
Unknown source
Web {1}
Event captured with web interface
Mobile {2}
Event captured on mobile interface
Import {3}
Event imported from other systems
MFS {4}
Special event source (provision)
GMTDT
DateTime
LocalDT
DateTime
Office
TOffice
Address
TAddress
City {string}
Country {string}
PostCode {string}
Street {string}
Subentity1 {string}
Subentity2 {string}
Contacts
TContacts
Email {string}
Fax {string}
Mobile {string}
Phone {string}
Preference {string}
Location
TLocation
Cd {string}
Country {string}
Nm {string}
PaymentDetails
TPaymentDetails
CounterAmount
TAmount
CurrCd {string}
Value {decimal}
PaidAmount
TAmount
CurrCd {string}
Value {decimal}

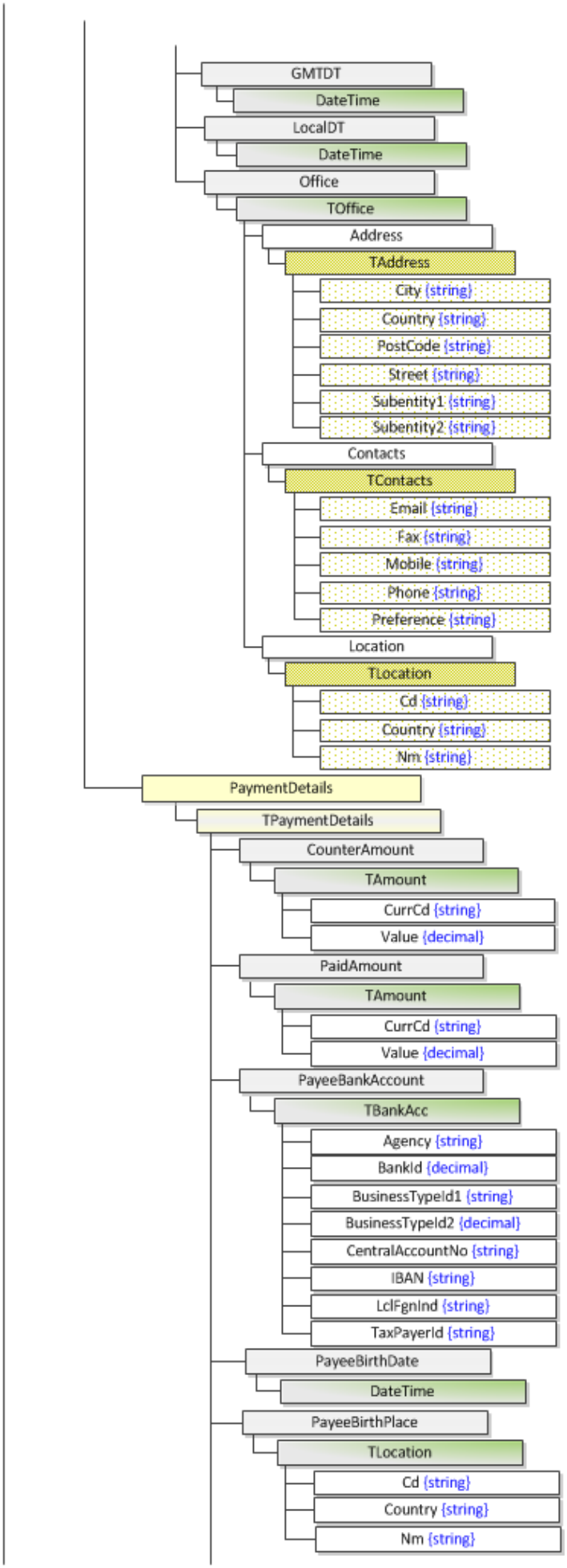

PayeeBankAccount
TBankAcc
Agency {string}
BankId {string}
BusinessTypeId1 {string}
BusinessTypeId2 {string}
CentralAccountNo {string}
IBAN {string}
LclFgnInd {string}
TaxPayerId {string}
PayeeBirthDate
DateTime
PayeeBirthPlace
TLocation
Cd {string}
Country {string}
Nm {string}
PayeeCheckNo
PayeeID1
TID
Issuer
TLocation
Cd {string}
Country {string}
Nm {string}
No {string}
TypeCd {string}
TypeDesc {string}
PayeeID2
TID
Issuer
TLocation
Cd {string}
Country {string}
Nm {string}
No {string}
TypeCd {string}
TypeDesc {string}
PayeeName
TName
First {string}
Last {string}
Middle {string}
Title {string}
SecretCode {string}

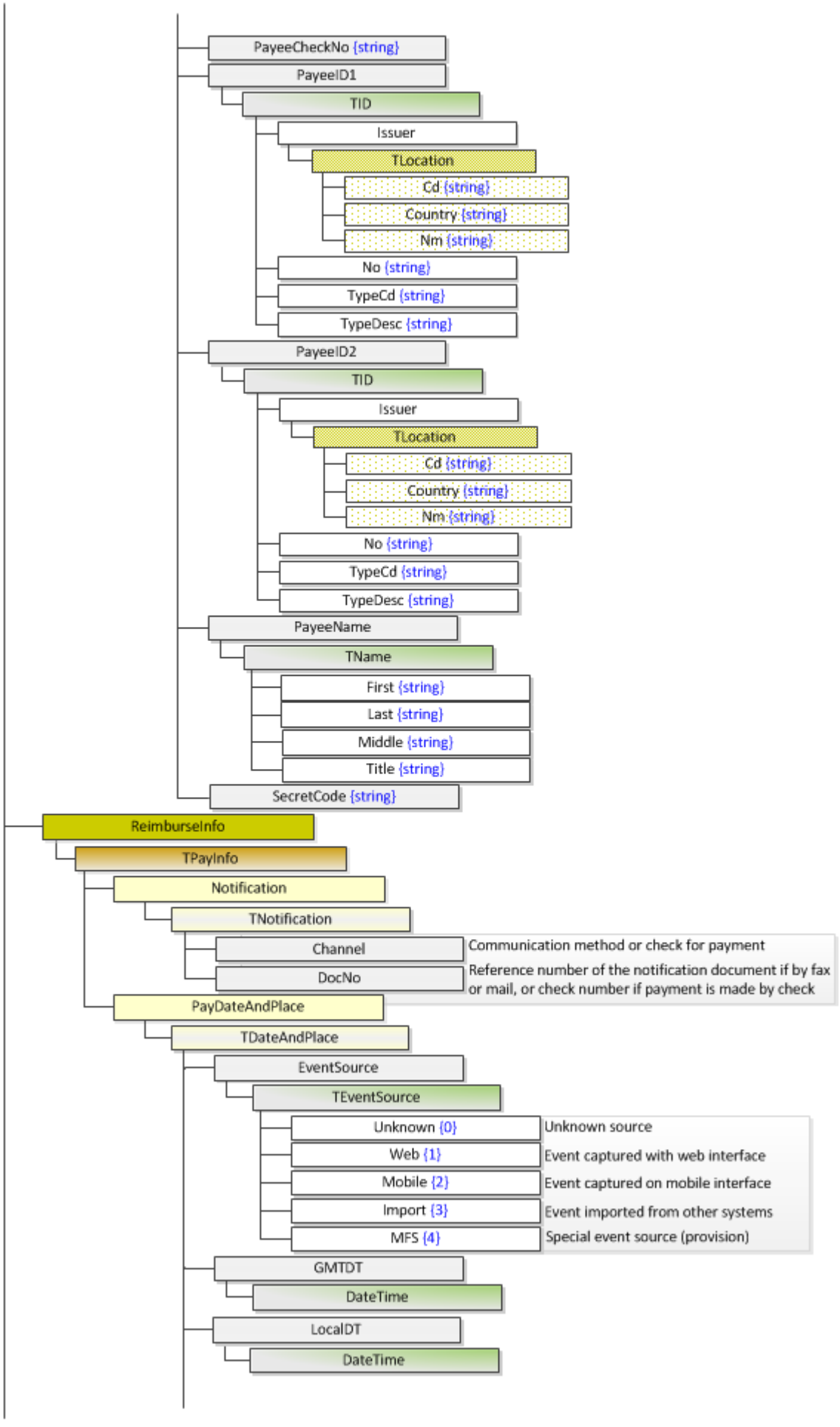
TStatus class

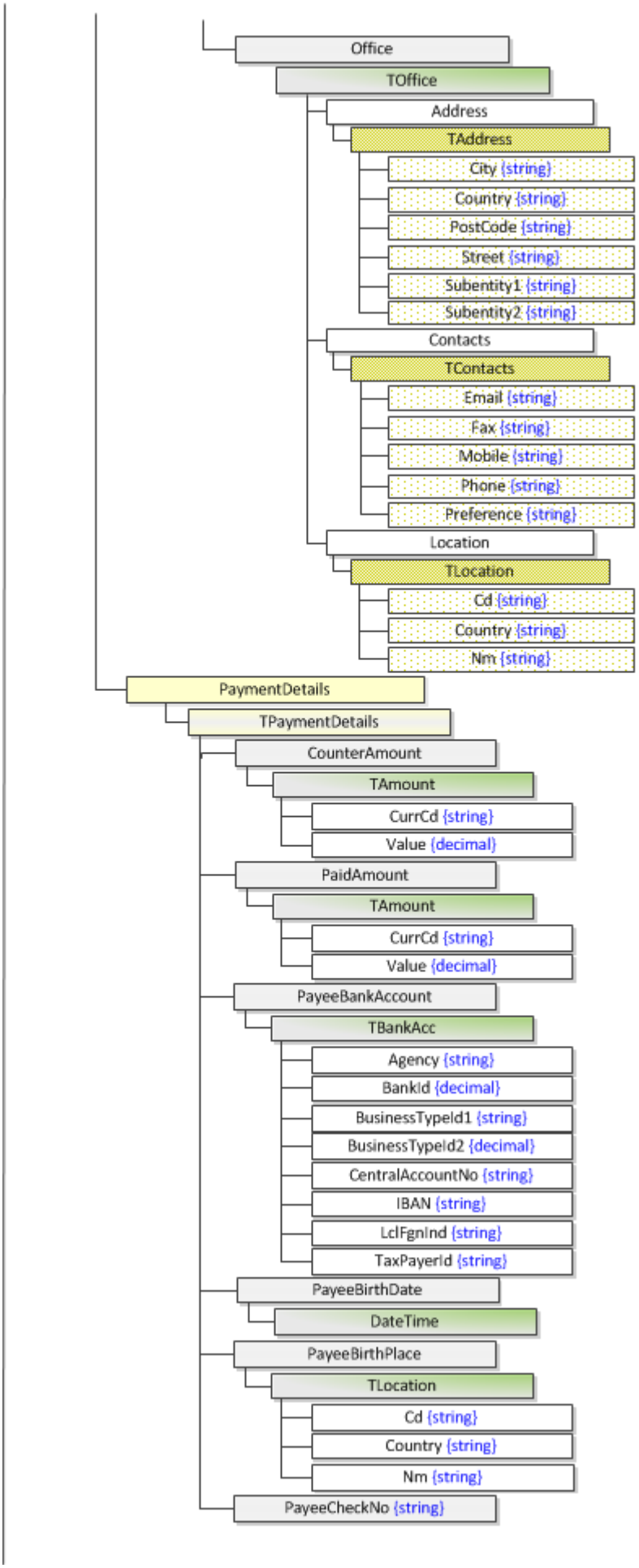
The `TStatus` class contains postal payment status information and is called after the CheckStatus() method. `TStatus` exposes the `LastEvent` and `MOInfo` properties. `MOInfo` makes use of the `TMO` class which is described in "TMO class" on page 10.

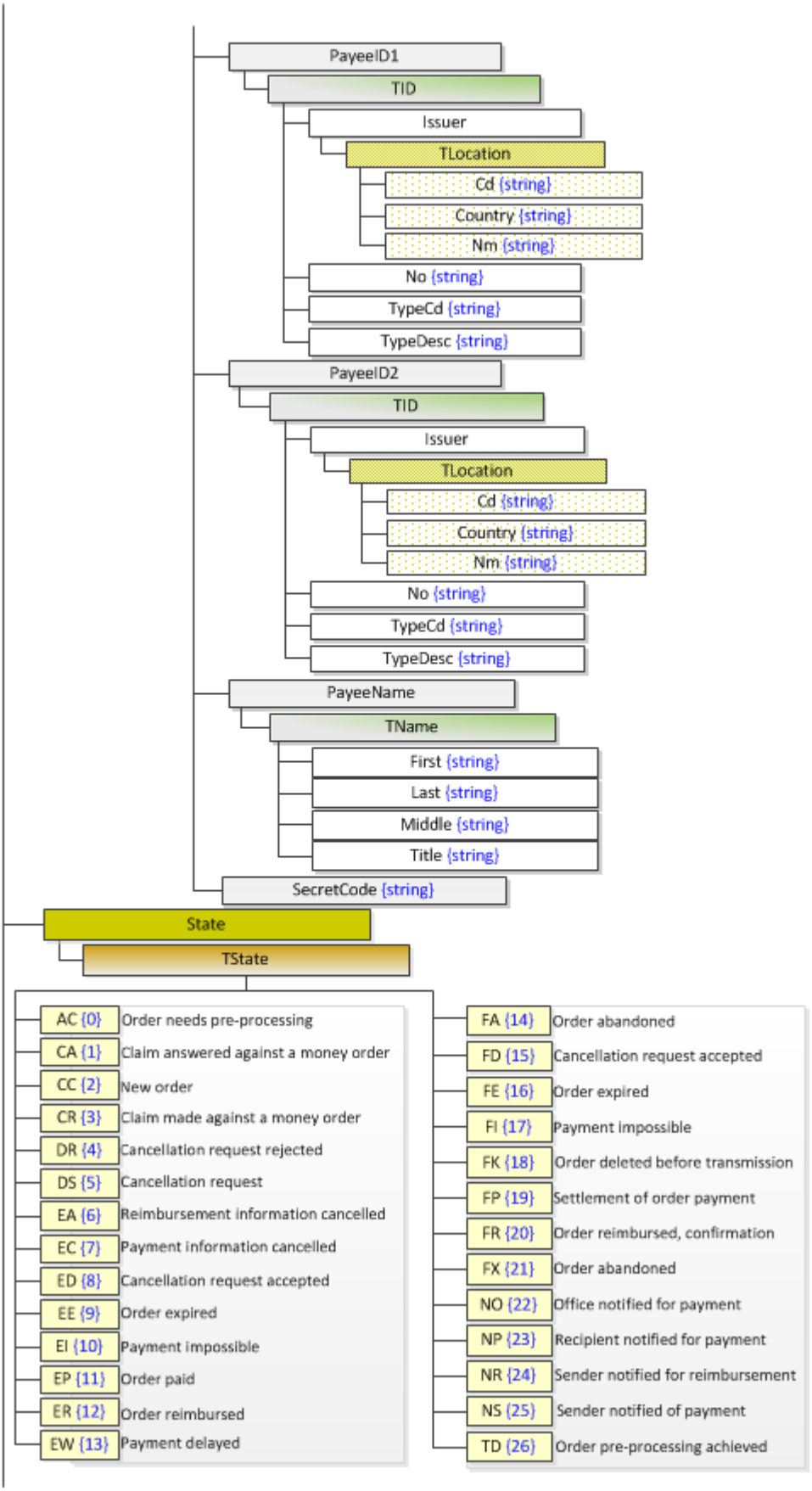



EventType
TMOEventType
Issued {0}
Order issued
IssuedPushedToExternal {1}
Order issued and action pushed to external party
IssuedByExternal {2}
Order issued by an external party
Paid {3}
Order paid
NotPaid {4}
Order marked as not paid
Canceled {5}
Order cancelled by issuing party
CanceledPushedToExternal {6}
Order cancelled by issuing party and action pushed to external party
Reimbursed {7}
Order reimbursed
ReimbursedPushedToExternal {8}
Order reimbursed and action pushed to external party
AutoStateTrans {9}
Automatic state transition
UpdateStateFromExternal {10}
Order state updated from external party (in push model)
MonordReceived {11}
MONORD received
MonordAckReceived {12}
MONORD ACK received
MonordErrReceived {13}
MONORD ERR received
ResordReceived {14}
RESORD received
ResordAckReceived {15}
RESORD ACK received
ResordErrReceived {16}
RESORD ERR received
MonordCreated {17}
MONORD created
MonordCreatedForClearing {18}
MONORD created for clearing
MonordAckCreated {19}
MONORD ACK created
MonordAckCreatedForClearing {20}
MONORD ACK created for clearing
ResordCreated {21}
RESORD created
ResordCreatedForClearing {22}
RESORD created for clearing
ResordAckCreated {23}
RESORD ACK created
ResordAckCreatedForClearing {24}
RESORD ACK created for clearing
MonordErrCreated {25}
MONORD ERR created
ResordErrCreated {26}
RESORD ERR created
NonPaymentInfo
TNonPaymentDetails
MeasureCd
Non-payment measure code
ReasonCd
Non-payment reason code
ReasonDesc
Non-payment reason description
PaymentInfo
TPayInfo
Notification
TNotification
Channel
Communication method or check for payment
DocNo
Reference number of the notification document if by fax or mail, or check number if payment is made by check
PayDateAndPlace
TDateAndPlace
EventSource
TEventSource
Unknown {0}
Unknown source
Web {1}
Event captured with web interface
Mobile {2}
Event captured on mobile interface
Import {3}
Event imported from other systems
MFS {4}
Special event source (provision)


GMTDT
DateTime
LocalDT
DateTime
Office
TOffice
Address
TAddress
City {string}
Country {string}
PostCode {string}
Street {string}
Subentity1 {string}
Subentity2 {string}
Contacts
TContacts
Email {string}
Fax {string}
Mobile {string}
Phone {string}
Preference {string}
Location
TLocation
Cd {string}
Country {string}
Nm {string}
PaymentDetails
TPaymentDetails
CounterAmount
TAmount
CurrCd {string}
Value {decimal}
PaidAmount
TAmount
CurrCd {string}
Value {decimal}
PayeeBankAccount
TBankAcc
Agency {string}
BankId {decimal}
BusinessTypeId1 {string}
BusinessTypeId2 {decimal}
CentralAccountNo {string}
IBAN {string}
LclFgnInd {string}
TaxPayerId {string}
PayeeBirthDate
DateTime
PayeeBirthPlace
TLocation
Cd {string}
Country {string}
Nm {string}


PayeeCheckNo {string}
PayeeID1
TID
Issuer
TLocation
Cd {string}
Country {string}
Nm {string}
No {string}
TypeCd {string}
TypeDesc {string}
PayeeID2
TID
Issuer
TLocation
Cd {string}
Country {string}
Nm {string}
No {string}
TypeCd {string}
TypeDesc {string}
PayeeName
TName
First {string}
Last {string}
Middle {string}
Title {string}
SecretCode {string}
ReimburseInfo
TPayInfo
Notification
TNotification
Channel
Communication method or check for payment
DocNo
Reference number of the notification document if by fax or mail, or check number if payment is made by check
PayDateAndPlace
TDateAndPlace
EventSource
TEventSource
Unknown {0}
Unknown source
Web {1}
Event captured with web interface
Mobile {2}
Event captured on mobile interface
Import {3}
Event imported from other systems
MFS {4}
Special event source (provision)
GMTDT
DateTime
LocalDT
DateTime


Office
TOffice
Address
TAddress
City {string}
Country {string}
PostCode {string}
Street {string}
Subentity1 {string}
Subentity2 {string}
Contacts
TContacts
Email {string}
Fax {string}
Mobile {string}
Phone {string}
Preference {string}
Location
TLocation
Cd {string}
Country {string}
Nm {string}
PaymentDetails
TPaymentDetails
CounterAmount
TAmount
CurrCd {string}
Value {decimal}
PaidAmount
TAmount
CurrCd {string}
Value {decimal}
PayeeBankAccount
TBankAcc
Agency {string}
BankId {decimal}
BusinessTypeId1 {string}
BusinessTypeId2 {decimal}
CentralAccountNo {string}
IBAN {string}
LclFgnInd {string}
TaxPayerId {string}
PayeeBirthDate
DateTime
PayeeBirthPlace
TLocation
Cd {string}
Country {string}
Nm {string}
PayeeCheckNo {string}


PayeeID1
TID
Issuer
TLocation
Cd {string}
Country {string}
Nm {string}
No {string}
TypeCd {string}
TypeDesc {string}
PayeeID2
TID
Issuer
TLocation
Cd {string}
Country {string}
Nm {string}
No {string}
TypeCd {string}
TypeDesc {string}
PayeeName
TName
First {string}
Last {string}
Middle {string}
Title {string}
SecretCode {string}
State
TState
AC {0} Order needs pre-processing
CA {1} Claim answered against a money order
CC {2} New order
CR {3} Claim made against a money order
DR {4} Cancellation request rejected
DS {5} Cancellation request
EA {6} Reimbursement information cancelled
EC {7} Payment information cancelled
ED {8} Cancellation request accepted
EE {9} Order expired
EI {10} Payment impossible
EP {11} Order paid
ER {12} Order reimbursed
EW {13} Payment delayed
FA {14} Order abandoned
FD {15} Cancellation request accepted
FE {16} Order expired
FI {17} Payment impossible
FK {18} Order deleted before transmission
FP {19} Settlement of order payment
FR {20} Order reimbursed, confirmation
FX {21} Order abandoned
NO {22} Office notified for payment
NP {23} Recipient notified for payment
NR {24} Sender notified for reimbursement
NS {25} Sender notified of payment
TD {26} Order pre-processing achieved

Connecting to the UPU-IP's Web Service

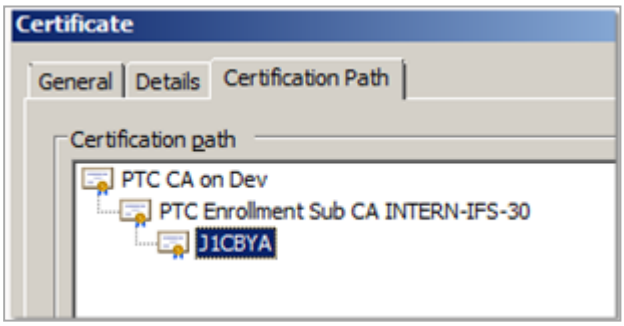
Before access to the **Web Service Interface** is granted, business partners must first generate a Certificate Signing Request (CSR) on the machine hosting the WS client. The CSR must be sent to the UPU for it to be signed then returned to business partners. In addition to the signed certificate, the PTC root Certificate Authority (CA) and sub-CA certificates provided by the PTC must be installed to ensure the chain of trust with the PTC root CA. Once the certificates are installed, business partners can access the UPU-IP's Web Service.

Refer to the PTC Enroll Client tool and its documentation to perform the following tasks: create the certificate request and submit it to the PTC, install CA and sub-CA certificates, retrieve signed certificate for the machine and install it, and test the SSL connection.

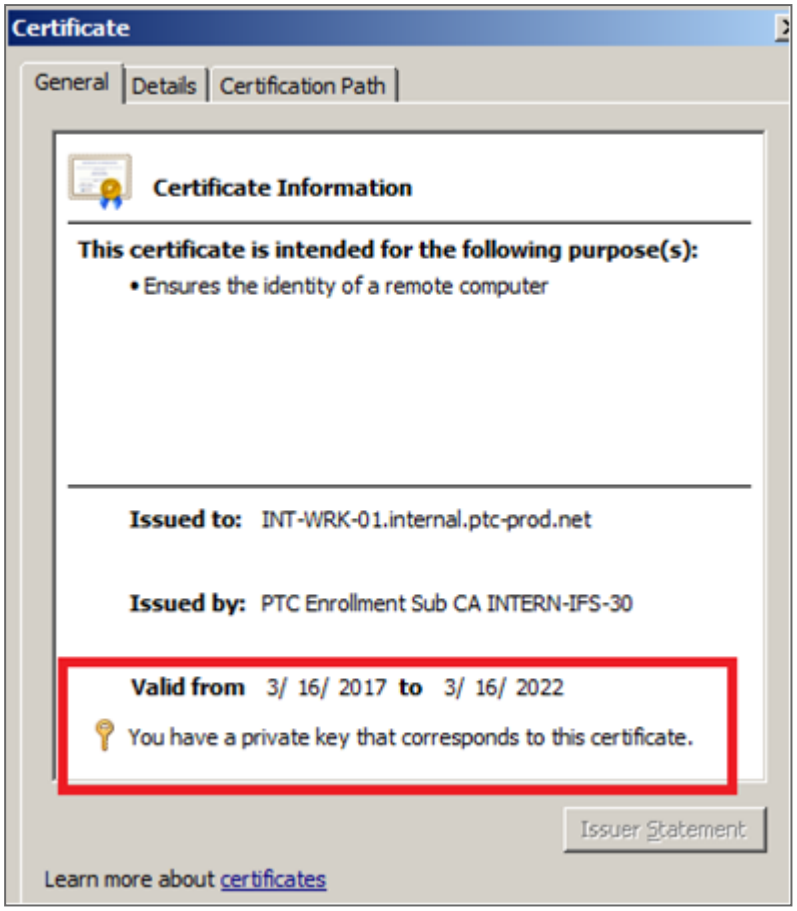
To get a copy of the PTC Enroll Client tool, you can enter your service request at https://support.upu.int.

Verify the certificates installation

1. Open the **Microsoft Management Console (MMC)** by clicking **Start** then typing mmc in the **Search** field. The MMC window opens.
2. Open the signed certificate you have installed and check the **Certification Path** tab. If all certificates are installed properly, your window will show the correct installation paths, as in the example below.

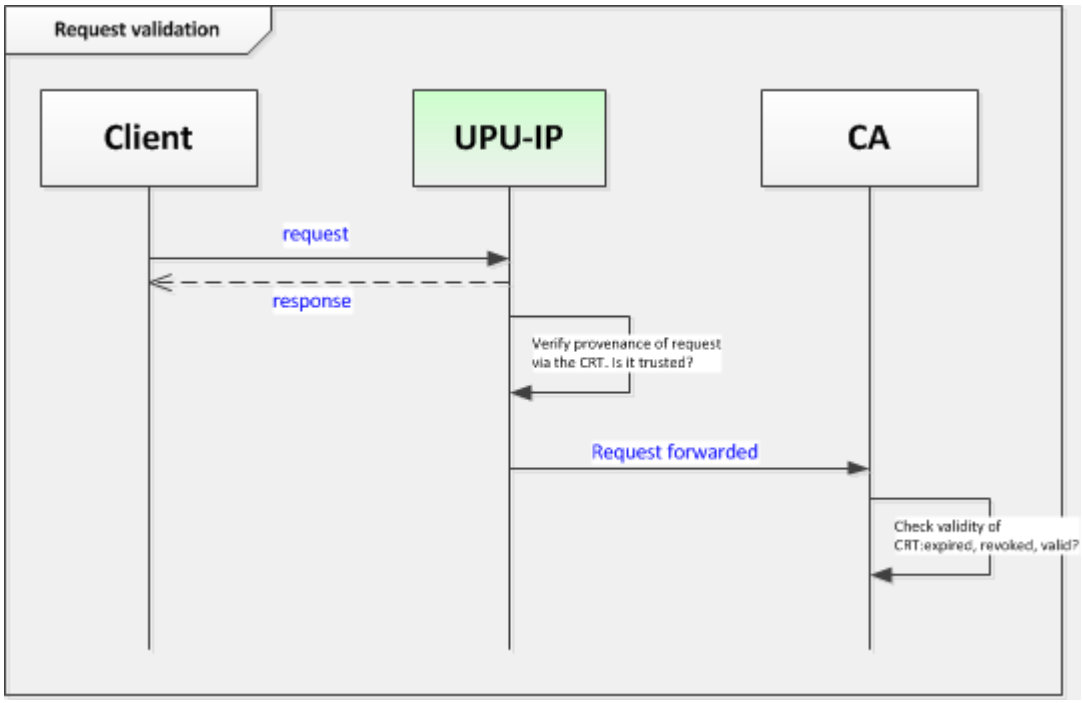


You can also verify whether the certificate you have installed has a private key. Open the certificate then click the **General** tab.



How a certificate is validated

Each time a business partner's WS client application makes a call to the UPU-IP's Web Service, the certificate is validated against the UPU-IP's PTC root CA for authenticity and trustworthiness as illustrated in the diagram below.



The Web Service's WSDL file

The Web Services Description Language (WSDL) is an XML-based definition describing the functions of a Web Service. Also referred to as a .wsdl file, it defines the behavior of an XML Web Service and provides instructions as to how WS clients can interact with the Web Service (what parameters the service expects and what data structures the service returns). The WSDL file for the UPU-IP's Web Service contains this same information. This file forms the basis of how the WS client application integrates with the **UPU-IP's Web Service Interface**.

Generating a Web Service proxy client from the WSDL

To build your WS proxy client application, follow the steps below.

1. Access the UPU-IP's WS endpoint at the URL: http://hubws.ptc.post/ExternalService.svc. Contact the PTC Service Desk at https://support.upu.int if you encounter difficulties connecting.

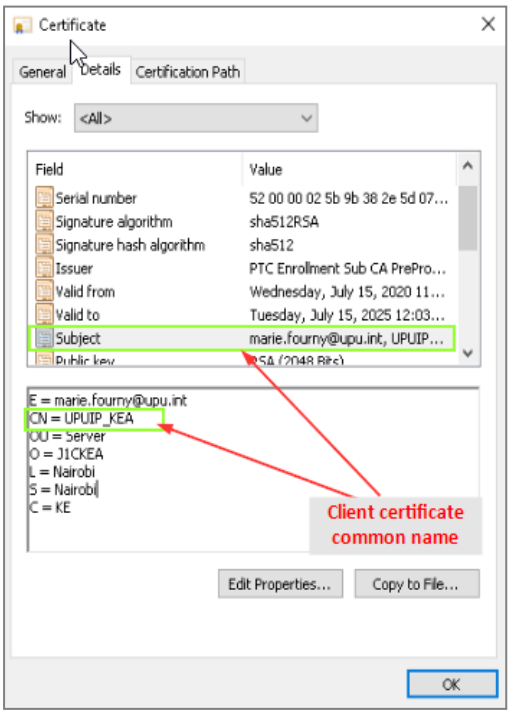
   The URL above is the current production environment endpoint. The URL for test environment endpoint is: http://hubws.preprod.upu.org/ExternalService.svc.

2. Generate a proxy client for calling the UPU-IP's Web Service. You can do this through the command line tools (wsdl.exe, wsimport, etc.) or within your development environment, depending on your platform.

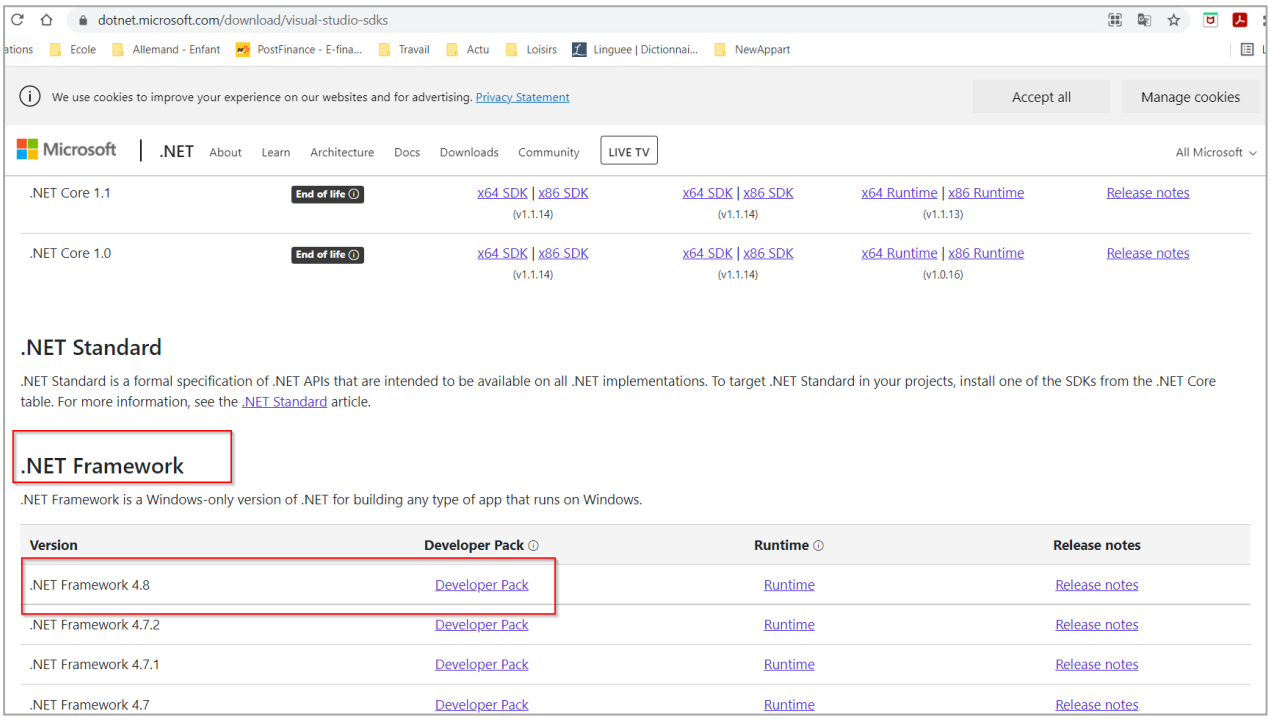
3. Write code to connect the business logic of your national system to the WS proxy client created in step 2.

Generating an API proxy client

1. Verify the certificate installation, see "Verify the certificates installation" on page 22 and check the common name of your client certificate.



2. Collect the UPU-IP API service definition by saving the WSDL available at http://hubws.preprod.upu.org/ExternalService.svc (pre-production):
    - Open your browser
    - Access http://hubws.preprod.upu.org/ExternalService.svc?singleWsdl
    - Save as **UPUIP_ExternalService.wsdl** in the folder where you want to generate the proxy (e.g. C:\UPU-IP_Client)
3. Download and install **Microsoft SDKs for Visual Studio** (.Net Framework 4.8 Developer Pack) from https://dotnet.microsoft.com/download/visual-studio-sdks



    - After completing the installation, the **ServiceModel Metadata Utility Tool** (Svcutil.exe) will be available on your computer. E.g. C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.8 Tools\x64\
    - Documentation about the tool is available at https://docs.microsoft.com/en-us/dotnet/framework/wcf/servicemodel-metadata-utility-tool-svcutil-exe
4. Generate the proxy/client code for UPU-IP API:
    - Open your command line tool
    - Set the current folder to the folder where the proxy will be generated and include the WSDL file (e.g. C:\UPU-IP_Client)
    - Run the following command: `svcutil.exe C:\UPU-IP_Client\UPUIP_ExternalService.wsdl`
    - If the path to **svcutil.exe** was not registered during the installation, you may have to add it manually:

```
C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.8 Tools\x64\svcutil.exe C:\UPU-IP_Client\UPUIP_ExternalService.wsdl
```

This command generates the **ExternalService.cs** and **output.config** files. The **ExternalService.cs** contains the code (C#) to be included in the proxy system you want to develop to expose the UPU-IP service to your system.

The **output.config** file contains the configuration settings to specify how the generated proxy will access the UPU-IP API.

5. Configure the proxy link to access the UPU-IP API
   - The generated proxy includes default configuration settings to access the UPU-IP API in the **output.config** file. For insantance, it may look as below:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.serviceModel>
    <bindings>
      <wsHttpBinding>
        <binding name="ExternalServiceSSL">
          <security>
            <message clientCredentialType="Certificate" />
          </security>
        </binding>
      </wsHttpBinding>
    </bindings>
    <client>
      <endpoint address="http://hubws.preprod.upu.org/ExternalService.svc"
        binding="wsHttpBinding" bindingConfiguration="ExternalServiceSSL"
        contract="IExternalService" name="ExternalServiceSSL">
        <identity>
          <certificate encodedValue="AwAAAAEAAAAUAAAA2EmWmV1DyQ9pExDCYeCAgCovh4AgAAAAAQAAAPoFA.....
/J87WJO5W7VH6Ig=" />
        </identity>
      </endpoint>
    </client>
  </system.serviceModel>
</configuration>
```

   - Edit the configuration file to include reference to your client certificate by adding the behavior element as presented below with reference to the common name of your client certificate.

```
<behaviors>
    <endpointBehaviors>
     <behavior name="ClientCertificateBehavior">
      <clientCredentials>
       <clientCertificate findValue="ClientCertificate_CommonName" storeLocation="LocalMachine"
storeName="My" x509FindType="FindBySubjectName" />
       <serviceCertificate>
        <authentication certificateValidationMode="PeerOrChainTrust" revocationMode="NoCheck" />
       </serviceCertificate>
      </clientCredentials>
     </behavior>
    </endpointBehaviors>
  </behaviors>
```

   - Refer to this new behavior in the endpoint by adding the behavior configuration attribute in the endpoint element

```
<client>
  <endpoint address="http://hubws.preprod.upu.org/ExternalService.svc"
    behaviorConfiguration="ClientCertificateBehavior"
     binding="wsHttpBinding" bindingConfiguration="ExternalServiceSSL"
    contract="IExternalService" name="ExternalServiceSSL">
  </endpoint>
 </client>
```

- Remove the identity element if present in the configuration file. The configuration file will look as below.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.serviceModel>
    <bindings>
      <wsHttpBinding>
        <binding name="ExternalServiceSSL">
          <security>
            <message clientCredentialType="Certificate" />
          </security>
        </binding>
      </wsHttpBinding>
    </bindings>
   <behaviors>
    <endpointBehaviors>
     <behavior name="ClientCertificateBehavior">
      <clientCredentials>
       <clientCertificate findValue="ClientCertificate_CommonName" storeLocation="LocalMachine"
storeName="My" x509FindType="FindBySubjectName" />
        <serviceCertificate>
         <authentication certificateValidationMode="PeerOrChainTrust" revocationMode="NoCheck" />
        </serviceCertificate>
      </clientCredentials>
     </behavior>
    </endpointBehaviors>
   </behaviors>
    <client>
      <endpoint address="http://hubws.preprod.upu.org/ExternalService.svc"
        behaviorConfiguration="ClientCertificateBehavior"
         binding="wsHttpBinding" bindingConfiguration="ExternalServiceSSL"
        contract="IExternalService" name="ExternalServiceSSL">
      </endpoint>
    </client>
  </system.serviceModel>
</configuration>
```

6. Run the proxy client.

   Installing the Visual studio tool allows you to run the **ExternalService.cs** file and run the **ExternalServiceClient** class to test calls to UPU-IP API.

UPU-IP Client certificate request requirements

Connecting to the UPU-IP API

PTC's Enroll Client tool is based on the Windows solution and may not be compatible with non-Windows based servers. In such situation, the partner may have to generate a standard client certificate and submit a service request to the UPU via our ticketing system to get a certificate.

This document describes the process and the requirements to generate and retrieve a valid request for a client certificate connecting to UPU-IP API.

Requesting a client certificate from a non-Windows-based OS

Follow the procedure below to obtain a client certificate to connect to the UPU-IP API from a non-Windows-based operating system:

1. Collect the certificate files for the PTC Certificate Authority and PTC Enrollment subordinate certificate authority, which may have to be installed on the non-Windows server.

Certificate	Platform	URL
PTC Certificate Authority	Production	n/a yet, provided as attachment for the moment
PTC Certificate Authority	Pre-production	n/a yet, provided as attachment for the moment
PTC Enrollment Sub CA	Production	n/a yet, provided as attachment for the moment
PTC Enrollment Sub CA	Pre-production	n/a yet, provided as attachment for the moment

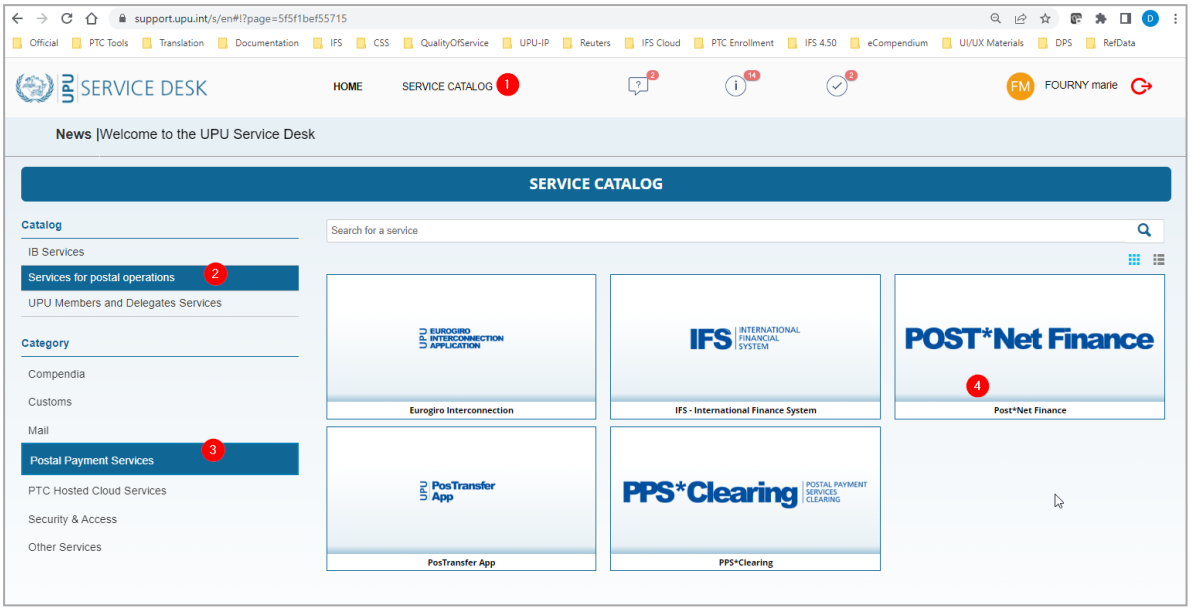
2. Generate a standard PKCS #10 client certificate request that meets the following requirements:

Request property	Expectation
Request format	PKCS #10
Friendly name	The title given to a certificate
CN -Common name attribute in Subject	The public URL of the server, for example, yoursite.com
SAN - Subject Alternative Name	The domain names and IP addresses of the server requesting the certificate. This is an optional attribute. However, since the development of the security protocols requires SAN spe-

Request property	Expectation
	cifications, it is recommended to specify at least the public URL of the server as done in the common name attribute of the subject. E.g. DNS Name=yoursite.com
E- Email attribute in Subject	The email address of the contact handling certificate request and renewal for this certificate
L - Locality attribute in Subject	The locality in which the certificate to be generated will reside based on this certificate request
S State attribute in Subject	The state or province in which the certificate to be generated will reside based on this certificate request
C - Country attribute in Subject	The two-letter code of the country where your organization is located
O - Organization attribute in Subject	The organization code for which the certificate will be produced, such as the Spanish organization code J1CESA. This information is essential since it can be used to confirm validate that the UPU-IP API caller is authorized to retrieve data related to this organization
OU - Organization Unit attribute in Subject	Server
Key usage	Data encipherment Digital signature Key encipherment
Extended Key Usage (application policies)	Client Authentication
Key size	2048
Hash algorithm	Sha256
File export format of the certificate request file	Base 64 file

3. Raise a service request:
    - Log into UPU's ticketing system https://support.upu.int/.
    - Go to the **Service Catalog**, click on **Services for Postal Operations** and, under the **Postal Payment Services** category, raise a new request for Post*Net Finance. Attach

the client certificate request file to the ticket and indicate whether the request is for a pre-production or production environment.



- While handling the service request, PTC will provide the issued certificate as an attachment to the ticket. The provided file will be a .cer file Base 64 exported file.

4. Install the client certificate file on the server (converted to Java Key store format if implemented on the Java platform).

Calling the UPU-IP's Web Service from a Java client

Calling the UPU-IP web service from a Java client requires using Transport Layer Security with certificate authentication.

Prerequisite

- Java JDK installed on the machine

Getting a client certificate in Java Key Store

Both the PTC and the designated operator (DO) must participate in this process. Each step's relevant performer is identified appropriately.

1. (DO) Generate a client certificate key by going to the path where **openssl** is and run the following command:

   ```
   <full_path>\openssl genrsa -out "<full_path>\<DO_SERVER_COMMON_NAME>.key"
   ```

   This will produce an RSA private key at the desired output location. For example,

   J1CESB.key is generated at C:\Program Files\OpenSSL-Win64\bin

2. (DO) Create a certificate request.

   - The file **create_requests.bat** is updated with DO settings. Make sure the folder structure specified in the file exists.
   - Edit the **create_requests.bat** file and examine lines number 10 to 32 to ensure the values are correct. At lines number 12, 13 and 32, type your common name, which is the fully qualified server name of the DO. Open the command prompt as an administrator and run the **create_request.bat** file with arguments (1 1). A sample command is:

     ```
     wine cmd < create_requests.bat 1 1
     ```

     This will create a certificate request (CSR file) in the output folder as specified.

3. (DO) Provide the above CSR file to PTC contacts either through email or through a support ticket created using http://support.upu.int.
4. (PTC) Submit the certificate request for certificate issuance to the UPU-PTC CA authority: using the PTC Enroll tool on a Windows server.
5. (PTC) Once the certificate is issued, retrieve the certificate and test it. Client certificates include the full CA chain.
6. (PTC) Concatenate the three certificates in base64.CER format: root CA, subordinate CA, and the certificate produced in step 4 above, and pass these to the DO.
7. (DO) With the key generated in step 1, create a p12 certificate key and key store from the client certificate using the command below:

Generate p12

Run the command line as an admin and run:

```
type_here_full_path_of_openssl.exe pkcs12 -export -inkey
type_here_the_name_of_your_key -in 1.cer -out identity.p12 -name "mykey"
```

When prompted for a password, type your password, for example, Pass1234.

The file called **identity.p12** is created.

Generate jks

Convert the above generated **identity.p12** to **identity.jks** by running the following as an admin:

```
type_here_full_path_of_java_keytool.exe -importkeystore -destkeystore iden-
tity.jks -deststorepass Pass1234 -srckeystore identity.p12 -srcstoretype
PKCS12 -srcstorepass Pass1234
```

A file called **identity.jks**is created. This file is ready to be used in a SOAPUI project created in the TLS of the PTC.

Connecting to the UPU-IP web service

Use the **jks** file created in step 7 above in a SOAPUI project.

1. (DO) Connect to the relevant TLS endpoint:

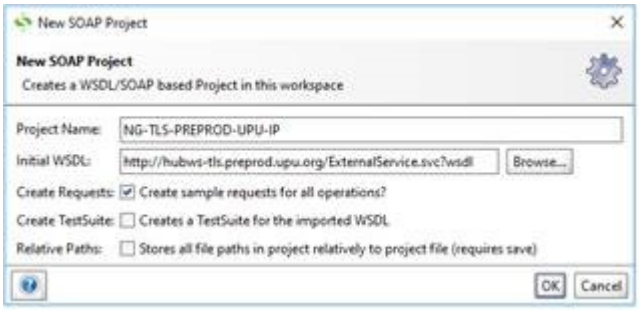
   Pre-production: https://hubws-tls.preprod.upu.org/ExternalService.svc

   Production: https://hubws-tls.ptc.post/ExternalService.svc

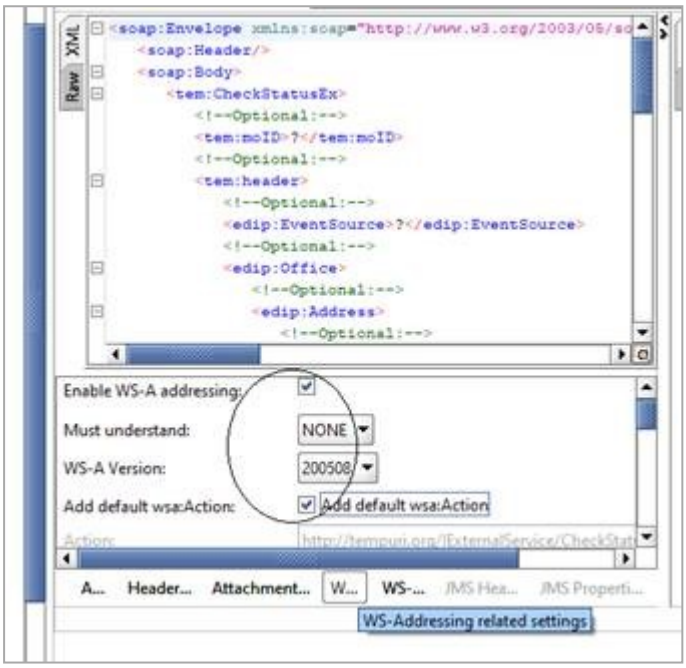
   How to connect

   1. Map your **jks** file to SOAPUI and create a SOAP project with the wsdl of the PTC TLS endpoint in the respective environment: pre-production, and later, production.

      http://hubws-tls.preprod.upu.org/ExternalService.svc?wsdl



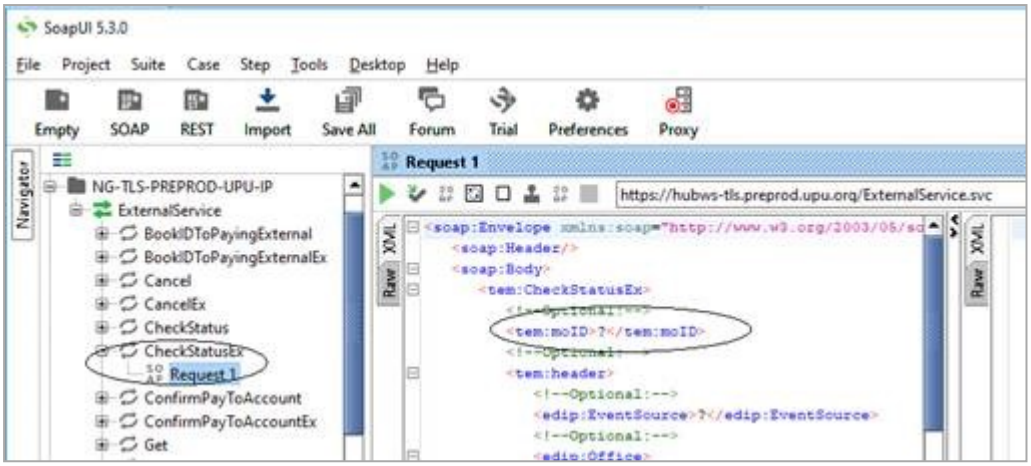
2. Ensure the **Enable WS-A addressing** check box is checked before you run an operation.



3. In SOAPUI, go to Preferences > SSL Settings, and map your Java Key Store (identity.jks) produced above with its password, as shown in the example below.

UPU-IP shows you all methods or operations of the UPU-IP API.

4. Test to confirm if your organization's *nix server is able to communicate with the PTC server by calling the CheckStatusEx() service. As the call is purely for testing connectivity, you can use a random or incorrect moID value.

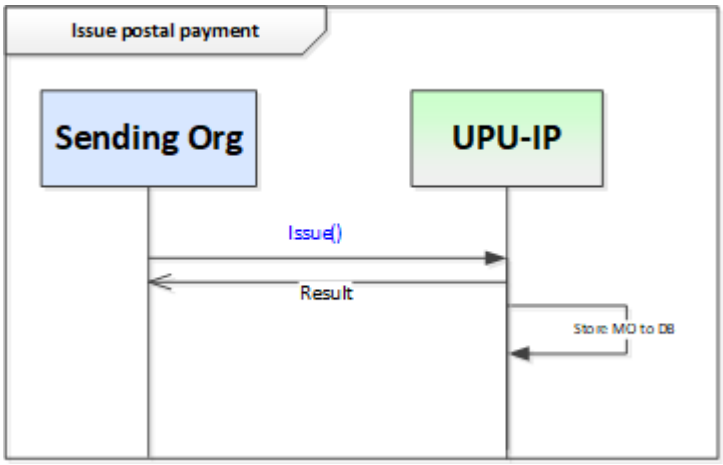

SoapUI 5.3.0
File Project Suite Case Step Tools Desktop Help
Empty SOAP REST Import Save All Forum Trial Preferences Proxy
Navigator
NG-TLS-PREPROD-UPU-IP
ExternalService
BookIDToPayingExternal
BookIDToPayingExternalEx
Cancel
CancelEx
CheckStatus
CheckStatusEx
Request 1
ConfirmPayToAccount
ConfirmPayToAccountEx
Get
Request 1
https://hubws-tls.preprod.upu.org/ExternalService.svc
XML
Raw
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/so
<soap:Header/>
<soap:Body>
<tem:CheckStatusEx>
<tem:moID>?</tem:moID>
<tem:header>
<!--Optional:-->
<edip:EventSource>?</edip:EventSource>
<!--Optional:-->
<edip:Office>

Integration scenarios

This chapter describes the main operations that can be carried out via calls to the WS methods discussed in "Web Service Interface" on page 7.

Issue a postal payment

To issue a postal payment, the business partner makes a WS call to the Issue() method in the UPU-IP's WS to transmit the payment information. The UPU-IP then stores the payment information in its database. The payment is then available for pay out by the paying business partner.



What are the methods used?

To issue the postal payment information to the UPU-IP, the business partner uses the Issue() method with the required parameters indicated on 7.

On receipt of the message, the UPU-IP stores the payment information in its database. The response to the WS call contains the result of the success of the issue process.

Retrieve and pay a postal payment

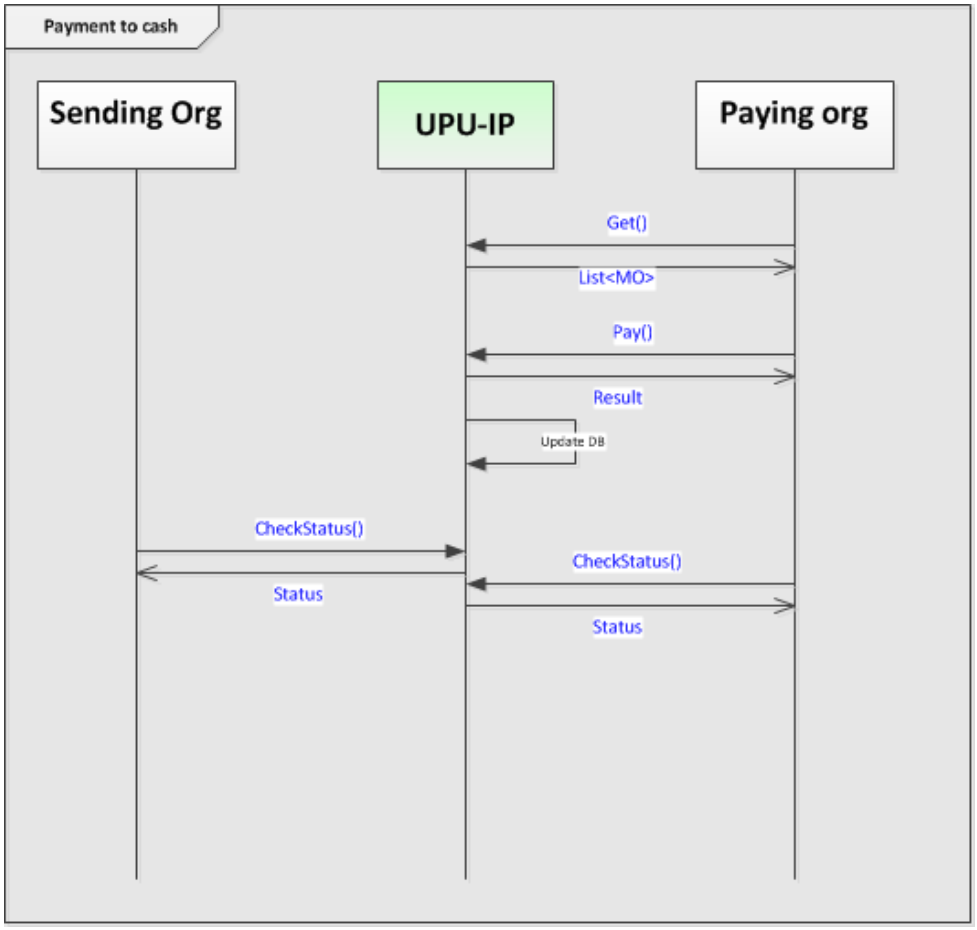
To retrieve a postal payment, the business partner makes a WS call to the Get() or ReceiveForAccount () methods. Any number of the following fields relating to the payment can be used to retrieve the information:

- IFS International money order ID
- Local identifier
- Recipient name
- Deposited amount (lower limit, upper limit)
- Transferred amount (lower limit, upper limit)
- Purchase date (Date from, Date to)

What are the methods used?

The method to use in calling the UPU-IP's Web Service depends on whether you are retrieving information for a cash payment product or an account payment product.

Pay to cash



Use the Get() method to call the UPU-IP's Web Service if the postal payment information to be retrieved is for a cash payment product, of which there are two types:

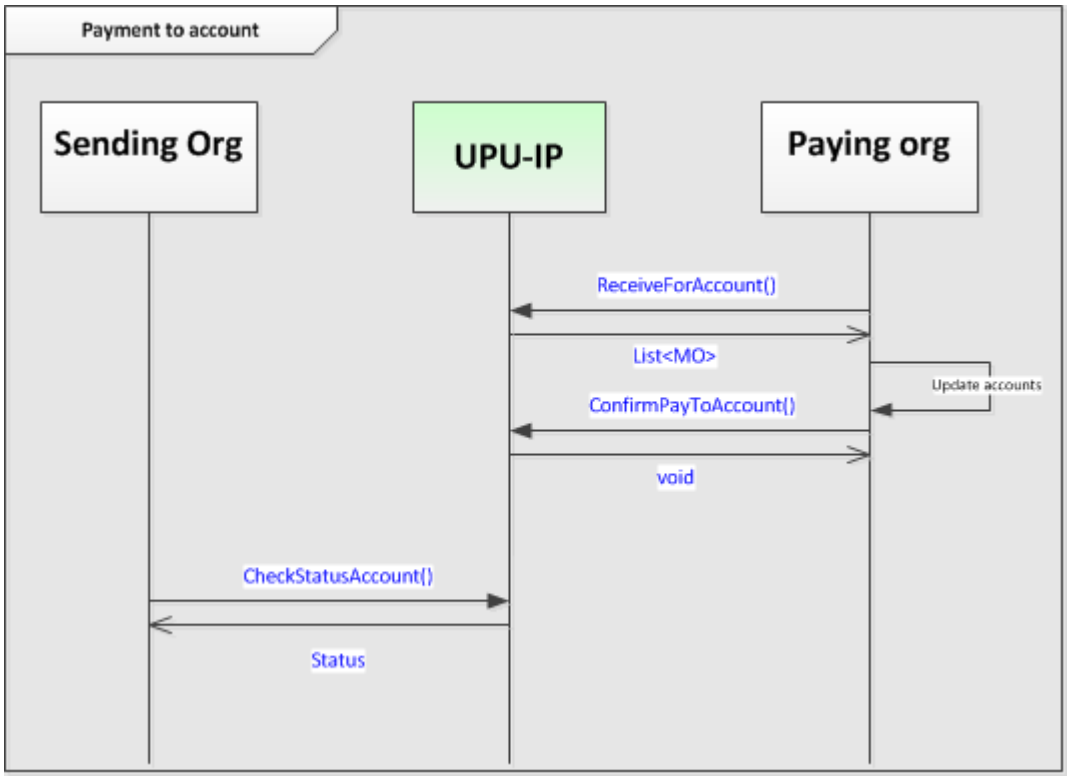
- cash-to-cash (OR)
- account-to-cash (OT)

Upon successful retrieval of the postal payment, the payment operation can take place at the paying business partner. If payout is possible at the business partner, the business partner calls the Pay() method to attempt to record the payment as paid. If the postal payment is still payable, then the UPU records the postal payment as paid.

Where the payout is unsuccessful, the status of the postal payment in the UPU-IP remains unchanged. The UPU-IP returns this information to the paying business partner in the response to the Pay() web service call.

At any point after the postal payment has been issued, both the sending and receiving business partners can check the status of the postal payment sent using the CheckStatus() method. The UPU-IP returns the status of the postal payment in the response to the WS call. In particular, the sending business partner can check the status of an issued postal payment, and where it has been paid out, it can record this state change in its national system.

Pay to account



Use the ReceiveForAccount() WS method to call the UPU-IP's Web Service if the postal payment information to be retrieved is for an account payment product, of which there are two types:

- cash-to-account (IN)
- account-to-account (GT)

The UPU-IP returns a list of all payable postal payments to account that are destined for the receiving business partner between the `From/To date` parameters specified. The receiving business partner then updates the UPU-IP on the status of the payment to account by making a WS call to the UPU-IP using the ConfirmPayToAccount() method.

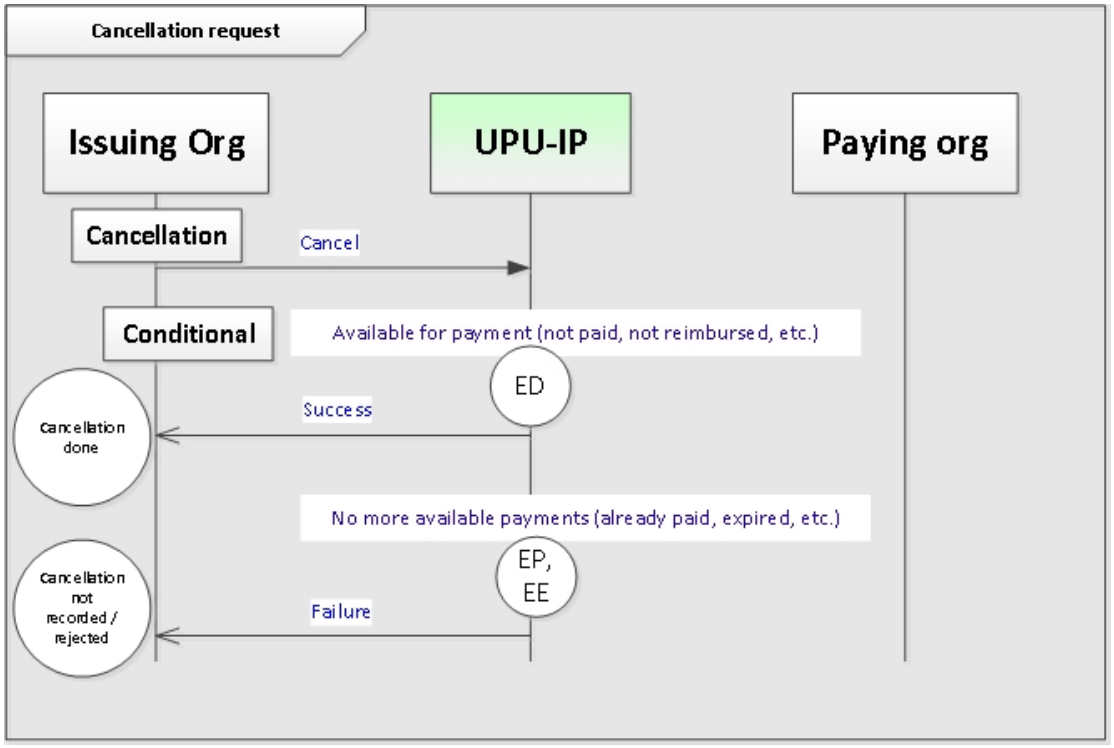
At any point, the sending business partner can check the status of postal payments sent using the CheckStatus() method.

The UPU-IP returns a list of postal payments and their corresponding statuses.

Cancel a postal payment

The sending business partner can immediately cancel postal payments which are exchanged (issued and to be paid) by a partner connecting to the UPU-IP if the payment is in a cancellable state, meaning it has not yet been paid out, canceled or expired. Cancellation is only applicable to cash payment products: cash-to-cash (OR), and account-to-cash (OT). Payable-to account postal payment products (IN and GT) cannot be canceled.

When the postal payment issued to the UPU-IP is to be paid by an EDI partner (IFS or STEFI), the sending business partner cannot proceed with its immediate cancellation. However, they can submit a cancellation request to the EDI partner and wait for them to accept it or reject it.



What are the methods used?

The sending business partner makes a call to the Cancel() method for an immediate cancellation.

The UPU-IP checks the status of the postal payment in the database. If the postal payment has not been paid and is in a cancellable state, the UPU-IP returns a "success" message to the sending business partner in the WS response. If the postal payment has been paid, canceled, or has already expired, the UPU-IP returns an "unsuccessful" response.

If the cancellation is successful, the UPU-IP changes the status of the postal payment in the database to "canceled". The sending business partner can then proceed to request a reimbursement.

If a postal payment is to be paid by an EDI partner, the sending business partner makes a call to the RequestEdiCancellation() method to submit a cancellation request.

The UPU-IP then checks the status of the postal payment in the database. If the postal payment has not been paid and is in a cancellable state, the UPU-IP forwards the cancellation request in an EDI file to the EDI partner. This returns a "success" message to the sending business partner in the WS response. If the postal payment has been paid, has been canceled, or has already expired, the UPU-IP returns an "unsuccessful" response.

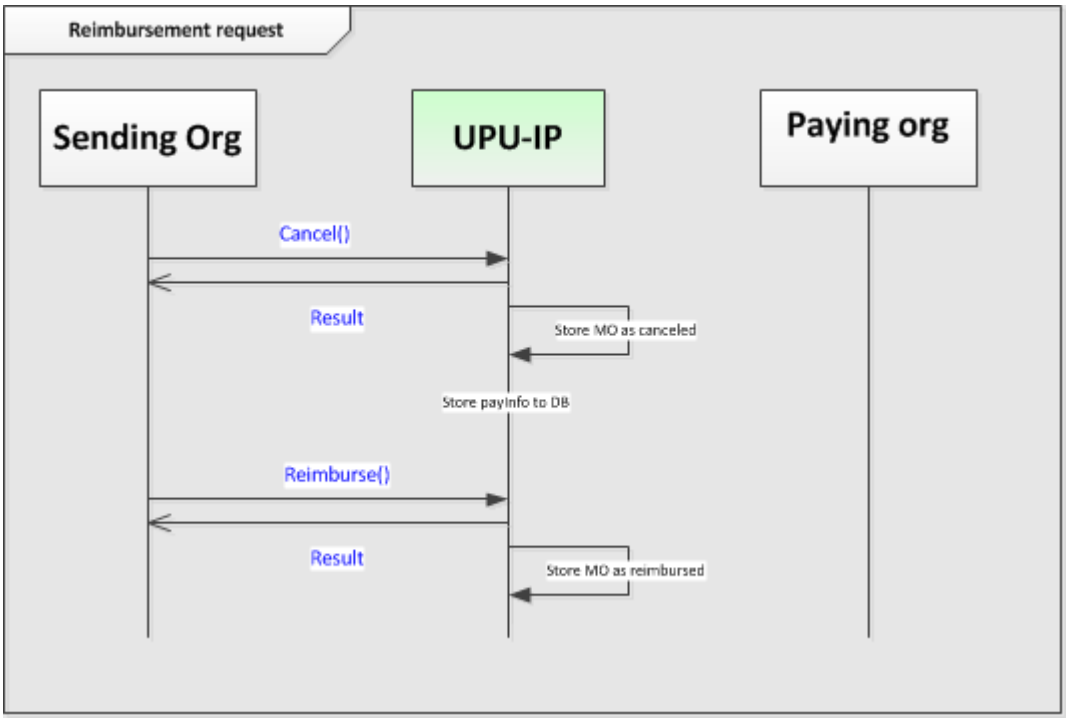
If the EDI cancellation request is successful, the UPU-IP changes the status of the postal payment in the database to "CancellationRequest". The sending business partner shall now regularly call the CheckStatus method to collect EDI partner feedback. Only when the paying EDI partner updates the postal payment with acceptance of the cancellation request (or expiration or payment impossible declaration), can the sending business partner consider the cancellation effective and proceed to request a reimbursement.

The following table lists the various EDI partner responses you may receive after submitting a cancellation request:

UPU-IP Current status	EDI cancellation request status	Possible next step
EP, FP	Rejected	Cancellation and reimbursement impossible
DR	Rejected	Cancellation and reimbursement impossible
EE, EI, ED	Accepted	Reimbursement possible
ER	Accepted	Reimbursement already done, no further evolution of the postal payment
CC	Cancellation request does not exist	No previous cancellation request, cancellation request can be submitted if needed
Other	Still pending	Wait for new EDI partner feedback

Reimburse a postal payment

The sending business partner can request a reimbursement of canceled postal payments.



What are the methods used?

In calling the UPU-IP's Web Service, the sending business partner uses the Reimburse() method.

If the postal payment is in a reimbursable state, the WS returns a "success" message to the sending business partner in the WS response. Otherwise, the WS returns an "unsuccessful" message with details of the error.

If reimbursement is successful, the UPU-IP updates the status of the postal payment in the database.

Payments to/from external partners

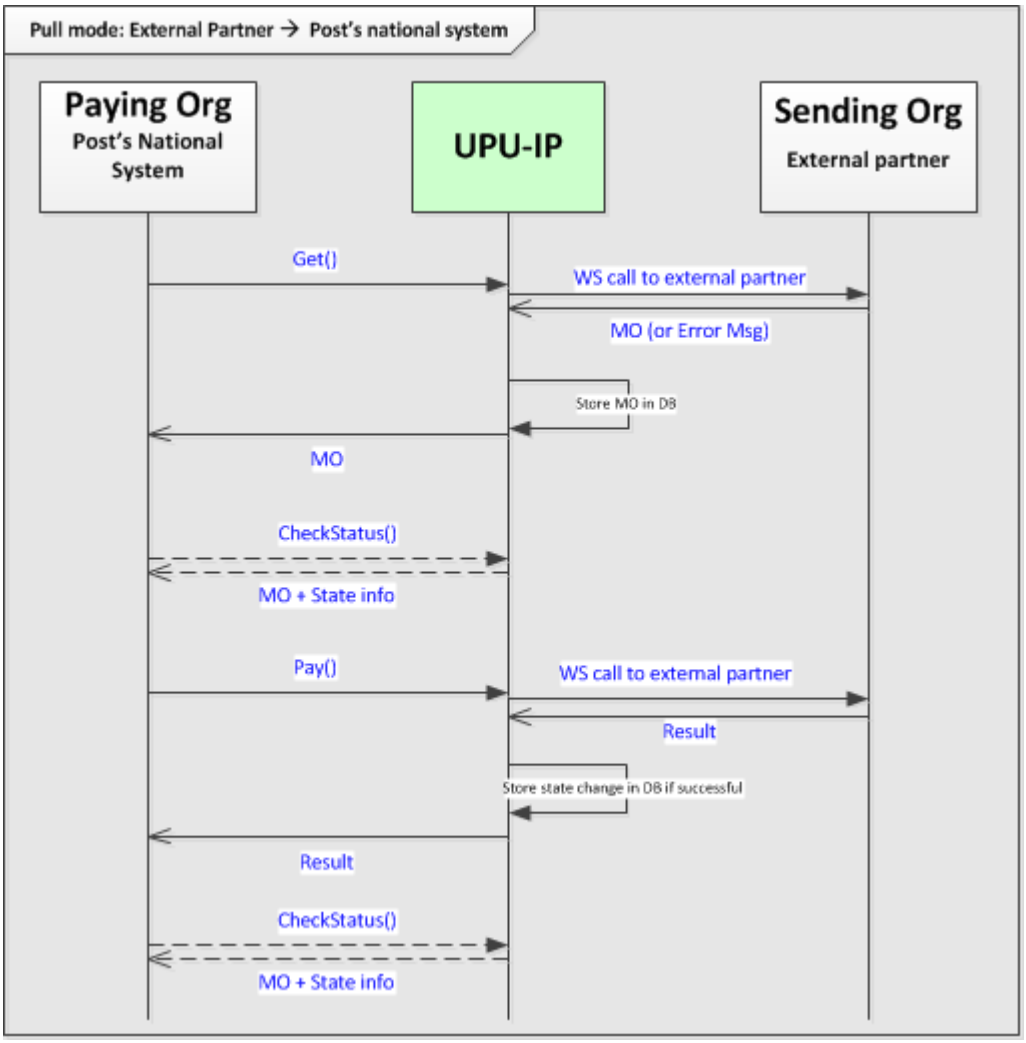
Where the sending or paying business partner is an external partner, then the external partner's system becomes the system of reference for postal payments, rather than the UPU-IP. Therefore, requests to the WS methods on the UPU-IP are typically forwarded to the external partner or system of reference.

External partner as the sending business partner

In this scenario, postal payments are retrieved from the external partner system by the paying business partner via the UPU-IP (pull mode). Also, payouts are recorded in the external sending partner's system.

External partners offer two functions to postal paying partners:

- a function to retrieve payable postal payments
- a function to record payouts



What are the methods used?

The paying business partner makes a call to the Get() method on the UPU-IP to retrieve the details of any payable postal payments from the external sending partner. The UPU-IP forwards this request to the external sending business partner. If successful, the external partner returns the details of the postal payment to the UPU-IP which stores those details in its database then returns those details back to the paying business partner. If the operation was unsuccessful i.e. no payable postal payments could be returned, an error message is returned to the UPU-IP and back to the paying business partner.

In this scenario, a call to the Get() method is only forwarded once by the UPU-IP to the sending external partner if the postal payment was already successfully retrieved. Further calls to Get() are superfluous i.e. the payment information is already present in the UPU-IP database. The Pay() method determines whether the payment is still payable in the external sending partner's system and payout may take place.

On payout at the paying business partner, the paying business partner calls the Pay() method on the UPU-IP in an attempt to record the payment as "paid" in the external sending partner's system. The

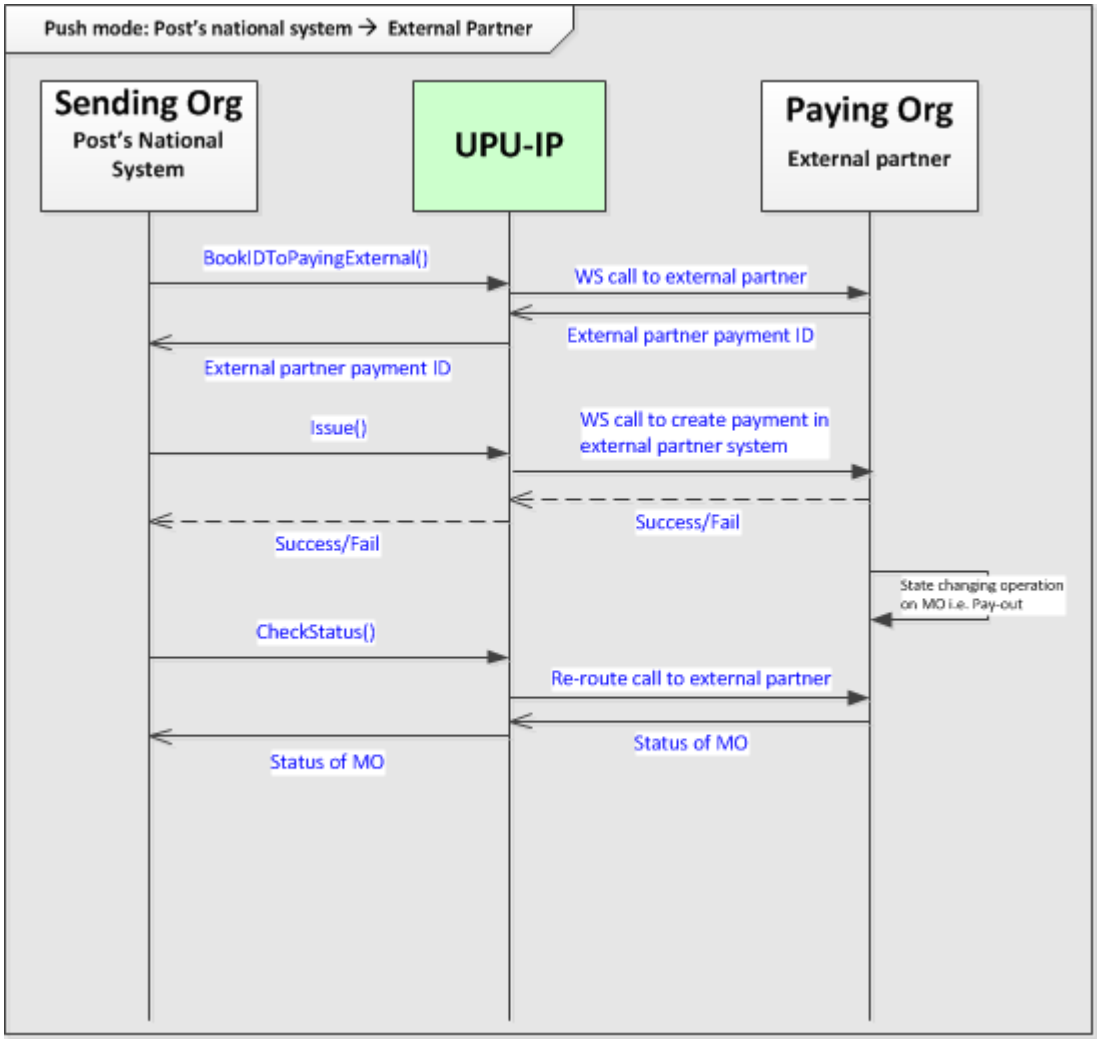
UPU-IP forwards the call to Pay() to the external sending partner. Where the payment is still payable and payout can proceed, as dictated by the external partner's system, the external partner system marks the payment as "paid" in its system and returns a success message back to the UPU-IP. The UPU-IP records this state change in its database and forwards this success message back to the paying business partner. The paying business partner then marks this payment as "paid" in its own national system. Where the postal payment was not payable, as dictated by the external sending partner's system (it was already paid out, canceled, expired etc.), an error message is returned to the UPU-IP and forwarded back to the paying business partner.

Any number of attempts to call Pay() by the paying business partner are forwarded to the external sending partner via the UPU-IP.

In this scenario, calls to the CheckStatus() method by the paying business partner are not forwarded to the external sending partner. Results of calls to CheckStatus() return the status of the payment in the UPU-IP database. This is because external sending partners, at least in this scenario, do not tend to expose track and trace functions to paying business partners. The CheckStatus() method plays no part in the retrieval and payout of money orders from external sending partners. Only calls to the Get() and Pay() methods are pertinent and forwarded via the UPU-IP to the external sending partner.

External partner as the paying business partner

In this scenario, the external partner is the paying business partner and the postal partner is the sending business partner. Payments are issued (or pushed) from the sending business partner to the external paying partner's system via the UPU-IP (push mode). All WS calls to the UPU-IP from the sending partner are forwarded to the external paying partner via the UPU-IP. This is for the reason that, since the paying external partner, in this scenario, does not return any postal payment status information back to the UPU-IP, the sending and paying partner systems need to be synchronized regarding the status of any postal payment.



What are the methods used?

The sending business partner makes a call to the BookIDToPayingExternal() method to reserve a postal payment ID in the external paying partner system. The UPU-IP relays this request to the external paying business partner. The external paying business partner then returns the external payment ID to the UPU-IP which in turn returns this ID to the sending business partner. Where this fails, an error message is returned to the UPU-IP and sending partner from the external sending business partner.

Using the retrieved external partner payment ID, the sending business partner makes a call to the Issue()WS method to send the postal payment details to the external paying partner. The UPU-IP relays this request to the external paying partner to create the payment in its system. The external paying business partner returns a “Success" or "Failure" message which the UPU-IP subsequently returns to the sending business partner.

At any point, the sending partner may call the CheckStatus() method to verify the status of any postal payments issued to the external partner. In this scenario, all requests to CheckStatus() are forwarded to the external paying partner via the UPU-IP. This is the method by which the sending partner is informed of any state change regarding the payment in the external paying partner’s system. For example, where payout took place successfully in the external paying partner’s system, a call to CheckStatus() is forwarded via the UPU-IP to the external paying partner’s system and the new status

of “paid” is returned to the sending partner via the UPU-IP. The sending partner then updates the status of that payment in its national system.

In this scenario, calls to Cancel() and Reimburse() may also be forwarded to the external paying partner’s system via the UPU-IP.