

Interface specification B2BService

Invoice upload and download web services
provided by PostFinance



E-invoice Customer Service

PostFinance Ltd
Mingerstrasse 20
3030 Berne

www.postfinance.ch/e-bill

Consulting and Sales business customers

Phone 0848 888 900 (max. CHF 0.08/min. in Switzerland)

Integration and Support

Helpdesk E-Invoice

Phone 0800 111 101

E-mail e-bill.help@postfinance.ch

Contents

| | | |
|------------|--|-----------|
| 1. | Introduction | 4 |
| 1.1 | Abbreviations and definitions | 4 |
| 2. | Interface overview | 5 |
| 2.1 | Transport and security | 5 |
| 2.2 | Authentication | 5 |
| 2.3 | Destinations | 5 |
| 2.4 | Service description | 6 |
| 2.5 | General Process for downloading data | 6 |
| 3. | Service methods | 7 |
| 3.1 | ExecutePing | 7 |
| 4. | Service methods for Billers | 8 |
| 4.1 | GetInvoiceListBiller | 8 |
| 4.2 | GetInvoiceBiller | 9 |
| 4.3 | SearchInvoices | 10 |
| 4.4 | GetProcessProtocolList | 12 |
| 4.5 | GetProcessProtocol | 13 |
| 4.6 | GetRegistrationProtocolList | 14 |
| 4.7 | GetRegistrationProtocol | 15 |
| 4.8 | UploadFilesReport | 16 |
| 4.9 | GetEBillRecipientSubscriptionStatus | 17 |
| 4.10 | InitiateEBillRecipientSubscription | 18 |
| 4.11 | ConfirmEBillRecipientSubscription | 19 |
| 5. | Service methods for Payers | 20 |
| 5.1 | GetInvoiceListPayer | 20 |
| 5.2 | GetInvoicePayer | 21 |
| 6. | Comparison between previous and new methods | 22 |
| 7. | Error Codes | 22 |
| 8. | .NET Example (C#) | 23 |
| 8.1 | Username & Password authentication | 23 |
| 8.1.1 | Configuration | 23 |
| 8.1.2 | Code | 23 |
| 8.2 | Certificate authentication | 24 |
| 8.2.1 | Configuration | 24 |
| 8.2.2 | Code | 24 |
| 9. | JAVA Example | 25 |
| 9.1 | Username & Password authentication | 26 |
| 9.1.1 | Configuration for Axis2 | 26 |
| 9.1.2 | Code for Axis2 | 27 |
| 9.1.3 | Configuration for CXF | 29 |
| 9.1.4 | Code for CXF | 29 |
| 9.1.5 | Configuration for JAX-WS | 30 |
| 9.1.6 | Code for JAX-WS | 33 |
| 9.2 | Certificate authentication | 34 |
| 9.2.1 | Configuration for Axis2 | 34 |
| 9.2.2 | Code for Axis2 | 35 |
| 9.2.3 | Configuration for CXF | 38 |
| 9.2.4 | Code for CXF | 39 |
| 9.2.5 | Configuration for JAX-WS | 40 |
| 9.2.6 | Code for JAX-WS | 43 |
| 10. | Disclaimer | 44 |

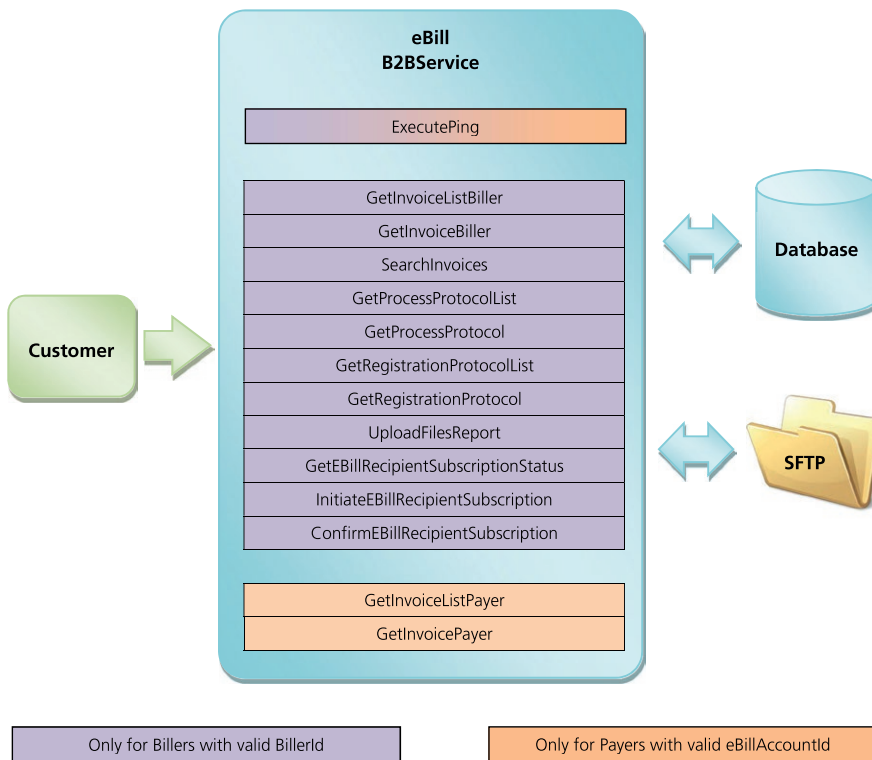
1. Introduction

This document describes the interface and methods of the SOAP web service (B2BService) provided by PostFinance for the invoice upload and download.

1.1 Abbreviations and definitions

| Abbreviation | Description |
|--------------|--|
| B2B | Business-To-Business |
| Biller | Customer who acts as a Biller and sends Invoices to a Payer. Has a BillerId. |
| DB | Database |
| HTTPS | Hypertext Transfer Protocol Secure |
| Payer | Customer who acts as a Payer and receives Invoices from a Biller. Has an eBillAccountId. |
| SFTP | Secure File Transfer Protocol |
| SOAP | Simple Object Access Protocol |
| WCF | Windows Communication Foundation |

2. Interface overview



2.1 Transport and security

The communication to the web service will be established with a SOAP WebService via HTTPS.

2.2 Authentication

For security reasons this web service must be implemented on the basis of the WCF.NET Security Extensibility. In order to communicate with the web service, a valid username and password or a Certificate must be submitted.

See chapter [.NET Example](#) for further reference.

2.3 Destinations

| | | |
|---------------------|---|---|
| Integration* | URL UserName & Password authentication | https://ebill-ki.postfinance.ch/B2BService/B2BService.svc |
| | URL Certificate authentication | https://ebill-ki.postfinance.ch/B2BService/B2BServiceCert.svc |
| Production | URL UserName & Password authentication | https://ebill.postfinance.ch/B2BService/B2BService.svc |
| | URL Certificate authentication | https://ebill.postfinance.ch/B2BService/B2BServiceCert.svc |

* Due to possible maintenance windows, we cannot guarantee 100% availability of the integration platform.

2.4 Service description

You are a biller or a payer and wish to use the invoice upload or download web services provided by PostFinance. You can implement your client in any programming language you wish as long as the XML requests correspond to the SOAP specifications. Because those XMLs are fairly comprehensive there are a variety of frameworks that simplify implementation of the web services considerably. You therefore do not have to create the SOAP XML "by hand". This document describes the client implementation of the B2BService in C# and Java.

In case of an application error, an error code and description will be returned (see *Error codes*).

If a technical error occurred or the caller cannot be authenticated a FaultException will be thrown.

In the following method description, mandatory fields are written in **bold**. All mandatory fields will be checked.

2.5 General Process for downloading data

To download the desired data, please notice that you have to call the "List" method first, then call the appropriate "Get" method with the returned values from the "List" method.

| Desired data | 1st Method (List available data) | 2nd Method (Download data) |
|--|-------------------------------------|----------------------------|
| Invoice for Billers | GetInvoice List Biller | GetInvoiceBiller |
| Invoice for Payers | GetInvoice List Payer | GetInvoicePayer |
| ProcessProtocol | GetProcessProtocol List | GetProcessProtocol |
| RegistrationProtocol | GetRegistrationProtocol List | GetRegistrationProtocol |
| Invoice Metadata for Billers | SearchInvoices | – |
| Subscription Status of eBill Recipient (Look-Up) | GetEBillRecipientSubscriptionStatus | – |
| Initiate eBill Subscription | InitiateEBillRecipientSubscription | – |
| Confirm eBill Subscription | ConfirmEBillRecipientSubscription | – |

3. Service methods

3.1 ExecutePing

| | | | |
|------------------|---|-------------|--|
| Method | B2BService.ExecutePing | | |
| Input | Used to check the reachability of this web service and to simulate errors | | |
| Output | Biller | | |
| Input | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | BillerID | String | Either the BillerID OR the eBillAccountID must be provided. |
| | eBillAccountID | String | |
| | ErrorTest | Boolean | If set to true, a unhandled error will be created in the response |
| | ExceptionTest | Boolean | If set to true, a FaultException will be thrown |
| Output | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | <i>Return value</i> | String | BillerID or eBillAccountID |
| Exception | FaultException | | |

4. Service methods for Billers

The following methods provide the biller with several invoice management functions.

A distinction is made between open invoices, protocols and registrations (never downloaded) and archived data (already downloaded). Data already collected remain available in the archive for 40 days after the first download.

4.1 GetInvoiceListBiller

| | | | |
|--------------------|---|--|---|
| Method | B2BService.GetInvoiceListBiller | | |
| Description | Returns a list of available invoices for a Biller | | |
| Role | Biller | | |
| Input | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | BillerID | String | Your BillerID |
| | ArchiveData | Boolean | False = Never downloaded data True = Already downloaded data |
| Output | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | <i>Return value</i> | Array of InvoiceReport | List of Invoices. See the following table for the definition. |
| Exception | FaultException | | |

| InvoiceReport* | Type | Description |
|-----------------------|-------------|------------------------------|
| BillerID | String | BillerID of the Invoice |
| DeliveryDate | DateTime | DeliveryDate of the Invoice |
| FileType | String | RGXMLSIG or EDIFACT |
| TransactionId | String | TransactionId of the Invoice |

* Was called InvoiceReportWithId in previous version

4.2 GetInvoiceBiller

| | | | |
|--------------------|----------------------------------|---------------------------------------|--|
| Method | B2BService.GetInvoiceBiller | | |
| Description | Download an Invoice for a Biller | | |
| Role | Biller | | |
| Input | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | BillerID | String | Your BillerID |
| | TransactionID | String | TransactionID of the desired Invoice returned from the Method GetInvoiceListBiller |
| | BillDetail | Boolean | True = Download BillDetail PDF False = Download Invoice only |
| Output | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | <i>Return value</i> | Array of DownloadFile | The Invoice and (if specified and present) the BillDetail PDF. See the following table for the definition. |
| Exception | FaultException | | |



| DownloadFile | Type | Description |
|---------------------|-------------|---|
| Data | Byte Array | Binary data |
| Filename | String | Filename of the Invoice or BillDetail PDF |

 **Please notice that downloaded Invoices will be marked as delivered in the eBill System**

4.3 SearchInvoices

| | | | |
|--------------------|---|--|--|
| Method | B2BService.SearchInvoices | | |
| Description | Search your own invoices based on various search parameters | | |
| Role | Biller | | |
| Input | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | Parameter | SearchInvoiceParameter | See the following table for the definition. |
| Output | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | <i>Return value</i> | SearchInvoicesResponse | See the table on the next page for the definition. |
| Exception | FaultException | | |

| SearchInvoiceParameter | Type | Description |
|--|-------------|--|
| BillerID | String | Your BillerID |
| TransactionID | String | |
| eBillAccountID | String | |
| AmountFrom | Decimal | |
| AmountTo | Decimal | |
| State | Enumeration | Possible values (Case sensitive): Open, Paid, Rejected, Incomplete, Invalid, Deleted, Processing, Unsigned |
| DeliveryDateFrom | DateTime | Must not be before the actual date -30 days. Default: Actual date -30 days |
| DeliveryDateTo | DateTime | Must not be after the actual date +30 days. Default: Actual date +30 days |
| PaymentDueDateFrom | DateTime | |
| PaymentDueDateTo | DateTime | |

-  **The BillerId has to be specified because it will be used for authentication purposes and for the search. You can only search your own Invoices.**
-  **If no other search parameter is specified, the noted default values are used.**

| SearchInvoicesResponse | Type | Description |
|------------------------|-----------------------|---|
| InvoiceCount | Int | Number of returned Invoices |
| TotalInvoiceCount | Int | Number of found Invoices |
| InvoiceList | List of SearchInvoice | See the following table for the definition. |

i If no Invoices are found, and empty InvoiceList will be returned.
InvoiceCount and TotalInvoiceCount will be 0.

i If more than 200 Invoices were found, only the newest 200 Invoices will be returned and the TotalInvoiceCount will be bigger than 200.
Please restrict your search then.

| SearchInvoice | Type | Description |
|-----------------|-------------|---|
| BillerID | String | Your BillerID |
| TransactionID | String | |
| eBillAccountID | String | |
| Amount | Decimal | |
| State | Enumeration | Possible values: Open, Paid, Rejected, Incomplete, Invalid, Deleted, Processing, Unsigned |
| PaymentType | String | |
| ESRReferenceNbr | String | |
| DeliveryDate | DateTime | |
| PaymentDueDate | DateTime | |
| ReasonCode | String | |
| ReasonText | String | |

4.4 GetProcessProtocolList

| | | | |
|--------------------|--|---|---|
| Method | B2BService.GetProcessProtocolList | | |
| Description | Returns a list of available ProcessProtocols | | |
| Role | Biller | | |
| Input | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | BillerID | String | Your BillerID |
| | ArchiveData | Boolean | False = Never downloaded data True = Already downloaded data |
| Output | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | <i>Return value</i> | Array of ProtocolReport | List of Protocols. See the following table for the definition. |
| Exception | FaultException | | |

| ProtocolReport | Type | Description |
|-----------------------|-------------|--|
| CreateDate | DateTime | Creation date and time of the Protocol |
| FileType | String | P (ProcessProtocol) |

4.5 GetProcessProtocol

| | | | |
|--------------------|--------------------------------------|------------------------------|---|
| Method | B2BService.GetProcessProtocol | | |
| Description | Download the desired ProcessProtocol | | |
| Role | Biller | | |
| Input | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | BillerID | String | Your BillerID |
| | CreateDate | DateTime | CreateDate of the desired ProcessProtocol returned from the Method GetProcessProtocolList |
| | ArchiveData | Boolean | False = Never downloaded data True = Already downloaded data |
| Output | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | <i>Return value</i> | DownloadFile | The ProcessProtocol. See the following table for the definition. |
| Exception | FaultException | | |

| DownloadFile | Type | Description |
|------------------------------|------------|---------------------------------|
| Data | Byte Array | Binary data |
| Filename | String | Filename of the ProcessProtocol |

 **Please notice that downloaded ProcessProtocols will be marked as delivered in the eBill System**

4.6 GetRegistrationProtocolList

| | | | |
|--------------------|---|---|---|
| Method | B2BService.GetRegistrationProtocolList | | |
| Description | Returns a list of available RegistrationProtocols | | |
| Role | Biller | | |
| Input | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | BillerID | String | Your BillerID |
| | ArchiveData | Boolean | False = Never downloaded data True = Already downloaded data |
| Output | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | <i>Return value</i> | Array of ProtocolReport | List of Protocols. See the following table for the definition. |
| Exception | FaultException | | |

| ProtocolReport | Type | Description |
|-----------------------|-------------|-------------------------------|
| CreateDate | DateTime | Creation date of the Protocol |
| FileType | String | R (RegistrationProtocol) |

4.7 GetRegistrationProtocol

| | | | |
|--------------------|---|------------------------------|---|
| Method | B2BService.GetRegistrationProtocol | | |
| Description | Download the desired RegistrationProtocol | | |
| Role | Biller | | |
| Input | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | BillerID | String | Your BillerID |
| | CreateDate | DateTime | CreateDate of the desired RegistrationProtocol returned from the Method GetRegistrationProtocolList |
| | ArchiveData | Boolean | False = Never downloaded data True = Already downloaded data |
| Output | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | <i>Return value</i> | DownloadFile | The RegistrationProtocol. See the following table for the definition. |
| Exception | FaultException | | |

| DownloadFile | Type | Description |
|------------------------------|------------|--------------------------------------|
| Data | Byte Array | Binary data |
| Filename | String | Filename of the RegistrationProtocol |

 **Please notice that downloaded RegistrationProtocols will be marked as delivered in the eBill System**

4.8 UploadFilesReport

This method is intended for billers and provides functionality to upload bill files. Only the following invoice types are supported. The FileType determinate the file extension of the uploaded file. This is important for further processing.

| Invoice Types | FileType (File Extension) |
|---------------------------------------|---------------------------|
| PDF bill detail for presentment | "PDF" |
| yblInvoice format | "XML" |
| Custom XML bill format like SAPiDoc | "EAI.XML" |
| Custom EDIFACT bill format | "EAI.EDI" |
| Structured PDF like QR-bill, Faktur X | "Struct.PDF" |

In the response detailed information about the transmission status of the files is given.

| | | | |
|--------------------|---|---|--|
| Method | B2BService.UploadFilesReport | | |
| Description | Upload your Invoices to SFTP for further processing | | |
| Role | Biller | | |
| Input | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | BillerID | String | Your BillerID |
| | Invoices | Array of Invoice | The Invoices. See the following table for the definition. |
| Output | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | <i>Return value</i> | Array of ProcessedInvoice | Report of the uploaded Invoices. See the following table for the definition. |
| Exception | FaultException | | |

| Invoice | Type | Description |
|-------------------------|------------|----------------------------|
| FileType | String | PDF, XML, eai.xml, eai.PDF |
| TransationID | String | |
| Data | Byte Array | Binary data |

| ProcessedInvoice | Type | Description |
|----------------------------------|----------|---|
| FileType | String | PDF, XML, eai.xml, eai.PDF |
| ProcessingState | String | OK if the invoice was successfully submitted to SFTP, otherwise NOK |
| SubmitDate | DateTime | Submit Date and Time |
| TransactionID | String | Extracted TransactionID from the Invoice |

4.9 GetEBillRecipientSubscriptionStatus

| | | | |
|--------------------|--|----------------|---|
| Method | B2BService.GetEBillRecipientSubscriptionStatus | | |
| Description | Returns the information whether invoices can be sent to a specific eBill recipient | | |
| Role | Biller | | |
| Input | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | BillerID | String (N17) | Your BillerID |
| | RecipientID | String | eBillRecipientID (N17) or eBillRecipient email address (^S+@S+\$) or eBillRecipient UIDHR (CHE[0-9]{9}) |
| | Up to 100 RecipientIDs, comma separated, can be set in one call. | | |
| Output | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | eBillAccountID | String (N17) | ID of eBillRecipient Will be returned when a relation is agreed |
| | emailAddress | String (AN255) | eMailAddress of eBillRecipient Will be returned when a relation is agreed. Only provided if used as a search term. |
| | UIDHR | String (AN12) | enterpriseIdentificationNumber of eBillRecipient Will be returned when a relation is agreed. Only provided if used as a search term. |
| | Message | String | Contains a message in the following cases: – Biller not enabled → Biller is not enabled to deliver invoices to eBill SIX – Subscription not found → Biller to bill recipient relation not agreed – Exception message when an error occurs while calling eBill SIX: %Exception% |
| Exception | FaultException | | |

4.10 InitiateEBillRecipientSubscription

| | | | |
|--------------------|--|---------------|--|
| Method | B2BService.InitiateEBillRecipientSubscription | | |
| Description | Initiates the subscription of an eBill recipient at eBill network. Must be confirmed by B2BService. ConfirmEBillRecipientSubscription. | | |
| Role | Biller | | |
| Input | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | BillerID | String (N17) | YourBillerID |
| | SubscriptionInitiationEmailAddress | String | eMailAdress of eBillRecipient (^S+@S+\$) |
| Output | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | SubscriptionInitationToken | String (AN36) | SubscriptionInitationToken Example: "0dc2ff79-db4c-4635-a4aa-f93f36ab5dbf" |
| | Message | String | A human readable explanation specific of this occurrence of the problem |
| Exception | FaultException | | |

4.11 ConfirmEBillRecipientSubscription

| | | | |
|--------------------|--|---------------------------|---|
| Method | B2BService.ConfirmEBillRecipientSubscription | | |
| Description | Confirms the subscription of an eBill recipient at eBill network by sending the activation code entered by the eBill recipient. Requires B2BService.InitiateEBillRecipientSubscription | | |
| Role | Biller | | |
| Input | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | BillerID | String (N17) | YourBillerID |
| | SubscriptionInitiationToken | String (AN36) | SubscriptionInitiationToken returned in B2BService.InitiateEBillRecipientSubscription |
| | SubscriptionInitiationActivation Code | String (6) | activation code provided by the user |
| Output | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | eBillAccountID | String (N17) | ID of eBillRecipient |
| | eMailAddress | String (AN255) | eMailAddress of eBillRecipient |
| | UIDHR | String (AN12) | enterpriseldentificationNumber of eBillRecipient |
| | Type | Enum | PRIVATE, COMPANY |
| | Language | String (3) | ISO-639-2/B |
| | Party | PartyType | Refer to yblInvoice_V*.xsd |
| | Party/Address/CompanyName | String (AN70) | In case of Type=COMPANY |
| | Party/Address/LastName | String (AN50) | In case of Type=PRIVATE |
| | Party/Address/GivenName | String (AN50) | In case of Type=PRIVATE |
| | Party/Address/Address1 | String (50) | |
| | Party/Address/ZIP | String (AN10) | |
| | Party/Address/City | String (AN50) | |
| | Party/Address/Country | Enum (ISO 3166-1 alpha 2) | |
| | Message | String | A human readable explanation specific of this occurrence of the problem |
| Exception | FaultException | | |

5. Service methods for Payers

The following methods provide the payer with invoice management functions.

A distinction is made between outstanding invoice (never downloaded and thus not yet paid) and archived invoices (already downloaded). Data already collected remain available in the archive for 40 days after the first download.

5.1 GetInvoiceListPayer

| | | | |
|--------------------|---|--|---|
| Method | B2BService. GetInvoiceListPayer | | |
| Description | Returns a list of available invoices for a Biller | | |
| Role | Payer | | |
| Input | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | eBillAccountID | String | Your eBillAccountID |
| | ArchiveData | Boolean | False = Never downloaded data True = Already downloaded data |
| Output | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | <i>Return value</i> | Array of InvoiceReport | List of Invoices. See the following table for the definition. |
| Exception | FaultException | | |

| InvoiceReport* | Type | Description |
|-----------------------|-------------|------------------------------|
| BillerID | String | BillerID of the Invoice |
| DeliveryDate | DateTime | DeliveryDate of the Invoice |
| FileType | String | RGXMLSIG, EDIFACT, PDF, ZIP |
| TransactionId | String | TransactionId of the Invoice |

* Was called InvoiceReportWithId in previous version

5.2 GetInvoicePayer

| | | | |
|--------------------|---------------------------------|------------------------------|---|
| Method | B2BService.GetInvoicePayer | | |
| Description | Download an Invoice for a Payer | | |
| Role | Payer | | |
| Input | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | eBillAccountID | String | Your eBillAccountID |
| | BillerID | String | BillerID of the desired Invoice returned from the Method GetInvoiceListPayer |
| | TransactionID | String | TransactionID of the desired Invoice returned from the Method GetInvoiceListPayer |
| | FileType | String | FileType of the desired Invoice returned from the Method GetInvoiceListPayer |
| Output | <i>Parameter</i> | <i>Type</i> | <i>Description</i> |
| | <i>Return value</i> | DownloadFile | The Invoice. See the following table for the definition. |
| Exception | FaultException | | |

| DownloadFile | Type | Description |
|---------------------|-------------|-------------------------|
| Data | Byte Array | Binary data |
| Filename | String | Filename of the Invoice |

 **Please notice that downloaded Invoices will be marked as delivered in the eBill System**

6. Comparison between previous and new methods

If you are already using our Webservices, based on a documentation dated before 2016, you will find in the following table, how the previous methods refer to the new methods.

| Previous method(s) | New method |
|--|-------------------------------------|
| For Billers | |
| GetInvoiceList / GetInvoiceListWithIDs | GetInvoiceListBiller |
| GetInvoice / GetInvoiceByID | GetInvoiceBiller |
| – | SearchInvoices |
| GetProcessProtocolList | GetProcessProtocolList |
| GetProcessProtocol | GetProcessProtocol |
| GetRegistrationList | GetRegistrationProtocolList |
| GetRegistration | GetRegistrationProtocol |
| UploadFiles / UploadFilesReport | UploadFilesReport |
| – | GetEBillRecipientSubscriptionStatus |
| – | InitiateEBillRecipientSubscription |
| – | ConfirmEBillRecipientSubscription |
| For Payers | |
| GetArchiveList / GetOpenList / GetInvoiceListWithIDs | GetInvoiceListPayer |
| GetInvoicesArchive / GetInvoices / GetPDF / GetPDFArchive / GetInvoiceByID | GetInvoicePayer |

7. Error Codes

| Code | Description |
|------|-----------------------|
| 000 | Unspecified Error. |
| 001 | No Customer. |
| 002 | Currently not in use. |
| 003 | Missing Parameter. |
| 004 | No Application. |
| 005 | No Process Service. |
| 006 | Invalid Parameter. |

8. .NET Example (C#)

You can use either Username and Password authentication or Certificate authentication.

Please notice the **marked** differences.

8.1 Username & Password authentication

8.1.1 Configuration

```
<system.serviceModel>
  <bindings>
    <wsHttpBinding>
      <binding name="WSHttpBinding_B2BService">
        <security mode="TransportWithMessageCredential">
          <transport clientCredentialType="None" />
          <message clientCredentialType="Username" establishSecurityContext="false" />
        </security>
      </binding>
    </wsHttpBinding>
  </bindings>
  <client>
    <endpoint address="https://XXX/B2BService/B2BService.svc"
      binding="wsHttpBinding" bindingConfiguration="WSHttpBinding_B2BService"
      contract="B2BServiceCert" name="WSHttpBinding_B2BService" />
  </client>
</system.serviceModel>
```

Replace the endpoint address with an address for Username & Password authentication provided in chapter [destinations](#).

8.1.2 Code

```
B2BServiceClient client = new B2BServiceClient();
client.ClientCredentials.UserName.UserName = "XXX";
client.ClientCredentials.UserName.Password = "YYY";

// Example for Billers
string billerId = "YOUR BILLERID";
string id = client.ExecutePing(billerId, null, false, false);

// Example for Payers
string eBillAccountID = "YOUR eBillAccountID";
string id = client.ExecutePing(null, eBillAccountID, false, false);
```

A valid BillerId (or eBillAccountID), Username and Password and RequestID must be used.

8.2 Certificate authentication

Use the URL for Certificate authentication provided in chapter 2.3.

8.2.1 Configuration

```
<system.serviceModel>
  <bindings>
    <wsHttpBinding>
      <binding name="WSHttpBinding_B2BServiceCert">
        <security mode="TransportWithMessageCredential">
          <transport clientCredentialType="None" />
          <message clientCredentialType="Certificate" establishSecurityContext="false" />
        </security>
      </binding>
    </wsHttpBinding>
  </bindings>
  <client>
    <endpoint address="https://XXX/B2BService/B2BServiceCert.svc"
      binding="wsHttpBinding" bindingConfiguration="WSHttpBinding_B2BServiceCert"
      contract="B2BService" name="WSHttpBinding_B2BServiceCert" />
  </client>
</system.serviceModel>
```

Replace the endpoint address with an address for Certificate authentication provided in chapter [destinations](#).

8.2.2 Code

```
X509Certificate2 certificate = new X509Certificate2("PATH TO CERTIFICATE", "CERTIFICATE PASSWORD");

B2BServiceClient client = new B2BServiceClient();
client.ClientCredentials.ClientCertificate.Certificate = certificate;

// Example for Billers
string billerId = "YOUR BILLERID";
string id = client.ExecutePing(billerId, null, false, false);

// Example for Payers
string eBillAccountID = "YOUR eBillAccountID";
string id = client.ExecutePing(null, eBillAccountID, false, false);
```

A valid BillerId (or eBillAccountID) and Certificate must be used.

9. JAVA Example

You can use either UserName and Password authentication or Certificate authentication.

Please notice the marked differences.

We currently only support the 3 most used Java WebService frameworks on the market.

- Axis2
- CXF
- JAX-WS

Minimum requirement of Java SE Development Kit is version 7.

In the following chapters you can find descriptions, how to use these frameworks, related to this particular WebService (B2BService).

There are also java test projects available for Eclipse IDE. Please don't hesitate to contact us, that we can handover these test projects to you.

9.1 Username & Password authentication

9.1.1 Configuration for Axis2

- Generate service client stub code with wsdl2java tool of Axis2.
- Create axis2-policy.xml file and place it in the resources folder of your java client project:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsp:Policy wsu:Id="B2BService_UserNamePassword_policy"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
  xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:TransportBinding xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
        <wsp:Policy>
          <sp:TransportToken>
            <wsp:Policy>
              <sp:HttpsToken RequireClientCertificate="false"/>
            </wsp:Policy>
          </sp:TransportToken>
          <sp:AlgorithmSuite>
            <wsp:Policy>
              <sp:Basic256/>
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:Layout>
            <wsp:Policy>
              <sp:Strict/>
            </wsp:Policy>
          </sp:Layout>
          <sp:IncludeTimestamp/>
        </wsp:Policy>
      </sp:TransportBinding>
      <sp:SignedSupportingTokens xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
        <wsp:Policy>
          <sp:UsernameToken
            sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
            <wsp:Policy>
              <sp:WssUsernameToken10/>
            </wsp:Policy>
          </sp:UsernameToken>
        </wsp:Policy>
      </sp:SignedSupportingTokens>
      <sp:Wss11 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
        <wsp:Policy/>
      </sp:Wss11>
      <sp:Trust10 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
        <wsp:Policy>
          <sp:MustSupportIssuedTokens/>
          <sp:RequireClientEntropy/>
          <sp:RequireServerEntropy/>
        </wsp:Policy>
      </sp:Trust10>
      <wsaw:UsingAddressing/>
      <ramp:RampartConfig xmlns:ramp="http://ws.apache.org/rampart/policy">
        <ramp:user>userid</ramp:user>

        <ramp:rampartConfigCallbackClass>b2bservice.ebill.swisspost.ch.SecurityConfigCallbackHandler</ramp:rampartConfigCallbackClass>
      </ramp:RampartConfig>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Please change the yellow marked entries according your needs.

9.1.2 Code for Axis2

- Create new Java code file, which has the following code:

```
package b2bservice.ebill.swisspost.ch;

import org.apache.rampart.RampartConfigCallbackHandler;
import org.apache.rampart.policy.model.RampartConfig;

public class SecurityConfigCallbackHandler implements
    RampartConfigCallbackHandler {

    @Override
    public void update(RampartConfig rampartConfig)
    {
        rampartConfig.setPwCbClass(PasswordCallbackHandler.class.getName());
    }
}
```

Please change the yellow marked entries according your needs.

- Make sure you put the name incl. package path to `ramp:rampartConfigCallbackClass` element (see chapter 8.1.1).
- Create new Java code file, which has the following code:

```
package b2bservice.ebill.swisspost.ch;

import java.io.IOException;

import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.UnsupportedCallbackException;

import org.apache.ws.security.WSPasswordCallback;

public class PasswordCallbackHandler implements CallbackHandler {

    @Override
    public void handle(Callback[] callbacks) throws IOException,
        UnsupportedCallbackException {

        for (int i = 0; i < callbacks.length; i++) {

            // To use the private key to sign messages, we need to provide
            // the private key password
            WSPasswordCallback pwcb = (WSPasswordCallback) callbacks[i];

            if (pwcb.getIdentifier().equals("userid")) {
                pwcb.setPassword("password");
                return;
            }

        }

    }
}
```

Please change the yellow marked entries according your needs.

- Unit Test code to check that service can be properly accessed.

```
package b2bservice.ebill.swisspost.ch;

import org.apache.axiom.om.impl.builder.StAXOMBuilder;
import org.apache.axis2.client.ServiceClient;
import org.apache.neethi.Policy;
import org.apache.neethi.PolicyEngine;
import org.apache.rampart.RampartMessageData;
import org.apache.xmlbeans.XmlObject;

/**
 * B2BServiceTest Junit test case
 */

public class B2BServiceTest extends junit.framework.TestCase {

    /**
     * Auto generated test method
     */
    public void testExecutePing() throws java.lang.Exception {

        B2BServiceStub stub = new B2BServiceStub("https://ebill-ki.postfinance.ch/B2BService/B2BService.svc");

        ServiceClient sc = stub._getServiceClient();

        sc.engageModule("addressing"); // module to engage WS-Addressing
        sc.engageModule("rampart"); // module to engage WS-Security

        // load axis2 policy file and use it to configure rampart module
        String policyFile = getClass().getClassLoader().getResource("axis2-policy.xml").getFile();
        sc.getOptions().setProperty(RampartMessageData.KEY_RAMPART_POLICY, loadPolicy(policyFile));

        // prepare webservice request
        ExecutePingDocument executePingData = (ExecutePingDocument) getTestObject(ExecutePingDocument.class);

        executePingData.addNewExecutePing().setBillerID("41101000011707505");

        // execute webservice method
        ExecutePingResponseDocument response = stub.executePing(executePingData);

        assertNotNull(response);

        assertEquals("41101000011707505", response.getExecutePingResponse().getExecutePingResult());
    }

    // Create the desired XmlObject and provide it as the test object
    public XmlObject getTestObject(java.lang.Class<?> type)
        throws java.lang.Exception {
        java.lang.reflect.Method creatorMethod = null;
        if (org.apache.xmlbeans.XmlObject.class.isAssignableFrom(type)) {
            Class<?>[] declaredClasses = type.getDeclaredClasses();
            for (int i = 0; i < declaredClasses.length; i++) {
                Class<?> declaredClass = declaredClasses[i];
                if (declaredClass.getName().endsWith("$Factory")) {
                    creatorMethod = declaredClass
                        .getMethod("newInstance", null);
                    break;
                }
            }
        }
        if (creatorMethod != null) {
            return (XmlObject) creatorMethod.invoke(null, null);
        } else {
            throw new java.lang.Exception("Creator not found!");
        }
    }

    private static Policy loadPolicy(String xmlPath) throws Exception {
        StAXOMBuilder builder = new StAXOMBuilder(xmlPath);
        return PolicyEngine.getPolicy(builder.getDocumentElement());
    }
}
```

Please change the yellow marked entries according your needs.

9.1.3 Configuration for CXF

- Generate service client stub code with wsdl2java tool of CXF.
IMPORTANT: Make sure you pass the following parameter:
`-autoNameResolution`
- Create cxf.xml file and place it in the resources folder of your java client project:

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:jaxws="http://cxf.apache.org/jaxws"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://cxf.apache.org/jaxws
    http://cxf.apache.org/schemas/jaxws.xsd">

  <jaxws:client name="{http://ch.swisspost.ebill.b2bservice}UserNamePassword" createdFromAPI="true">
    <!-- Comment below element to use non-WS-SecPol CXF interceptor method -->
    <jaxws:properties>
      <entry key="security.callback-handler" value="b2bservice.ebill.swisspost.ch.PasswordCallbackHandler"/>
      <entry key="security.username" value="userid"/>
    </jaxws:properties>
  </jaxws:client>

</beans>
```

Please change the yellow marked entries according your needs.

9.1.4 Code for CXF

- Create new Java code file, which has the following code:

```
package b2bservice.ebill.swisspost.ch;

import java.io.IOException;

import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.UnsupportedCallbackException;

import org.apache.wss4j.common.ext.WSPasswordCallback;

public class PasswordCallbackHandler implements CallbackHandler {

    public void handle(Callback[] callbacks) throws IOException, UnsupportedCallbackException {
        for (int i = 0; i < callbacks.length; i++) {
            WSPasswordCallback pwcb = (WSPasswordCallback)callbacks[i];

            if (pwcb.getIdentifier().equals("userid")) {
                pwcb.setPassword("password");
                return;
            }
        }
    }
}
```

Please change the yellow marked entries according your needs.

- Make sure you put the name incl. package path to `security.callback-handler` entry element (see chapter 8.1.3).

- Unit Test code to check that service can be properly accessed.

```
package b2bservice.ebill.swisspost.ch;
import static org.junit.Assert.*;

import java.net.MalformedURLException;
import java.net.URL;

import org.junit.Test;

public class B2BServiceTest {

    @Test
    public void testExecutePing() throws MalformedURLException {

        URL wsdlURL = new URL("https://ebill-ki.postfinance.ch/B2BService/B2BService.svc?singleWsdl");

        B2BService_Service service = new B2BService_Service(wsdlURL, B2BService_Service.SERVICE);
        B2BService port = service.getUserNamePassword();

        String result = port.executePing("41101000011707505", null, null, null);

        assertNotNull(result);

        assertEquals("41101000011707505", result);
    }

}
```

Please change the yellow marked entries according your needs.

9.1.5 Configuration for JAX-WS

- Generate service client stub code with wsimport tool of JAX-WS.
IMPORTANT: Make sure you pass the following parameter:
`-extension -B-XautoNameResolution`
- Create wsit-client.xml file and place it in the resources folder of your java client project:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    name="mainclientconfig">
    <import location="B2BService.xml" namespace="http://ch.swisspost.ebill.b2bservice" />
</wsdl:definitions>
```

Please change the yellow marked entries according your needs.

- Create service XML file and name it as you like. Place it in the resources folder of your java client project. Make sure you put the name to **location** attribute of import element (see chapter 8.1.5).

```
<?xml version="1.0" encoding="utf-8"?>

<wsdl:definitions
    name="B2BService"
    targetNamespace="http://ch.swisspost.ebill.b2bservice"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
    xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex"
    xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/addressing/policy"
    xmlns:msc="http://schemas.microsoft.com/ws/2005/12/wsdl/contract"
    xmlns:i0="http://ch.swisspost.ebill.B2BService"
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tns="http://ch.swisspost.ebill.b2bservice"
    xmlns:wsa10="http://www.w3.org/2005/08/addressing"
    xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:sc1="http://schemas.sun.com/2006/03/wss/client"
    xmlns:wsp1="http://java.sun.com/xml/ns/wsdl/policy">
    <wsp:Policy wsu:Id="UserNamePassword_policy">
        <wsp:ExactlyOne>
            <wsp>All>
                <sp:TransportBinding xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
                    <wsp:Policy>
                        <sp:TransportToken>
                            <wsp:Policy>
                                <sp:HttpsToken RequireClientCertificate="false"/>
                            </wsp:Policy>
                        </sp:TransportToken>
                        <sp:AlgorithmSuite>
                            <wsp:Policy>
                                <sp:Basic256/>
                            </wsp:Policy>
                        </sp:AlgorithmSuite>
                        <sp:Layout>
                            <wsp:Policy>
                                <sp:Strict/>
                            </wsp:Policy>
                        </sp:Layout>
                        <sp:IncludeTimestamp/>
                    </wsp:Policy>
                </sp:TransportBinding>
                <sp:SignedSupportingTokens xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
                    <wsp:Policy>
                        <sp:UsernameToken
                            sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
                                <wsp:Policy>
                                    <sp:WssUsernameToken10/>
                                </wsp:Policy>
                            </sp:UsernameToken>
                        </wsp:Policy>
                    </sp:SignedSupportingTokens>
                    <sp:Wss11 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
                        <wsp:Policy/>
                    </sp:Wss11>
                    <sp:Trust10 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
                        <wsp:Policy>
                            <sp:MustSupportIssuedTokens/>
                            <sp:RequireClientEntropy/>
                            <sp:RequireServerEntropy/>
                        </wsp:Policy>
                    </sp:Trust10>
                    <wsaw:UsingAddressing/>
                    <sc1:CallbackHandlerConfiguration wsp:visibility="private">
                        <sc1:CallbackHandler name="usernameHandler" default="userid" />
                        <sc1:CallbackHandler name="passwordHandler" default="password"/>
                    </sc1:CallbackHandlerConfiguration>
                </wsp>All>
            </wsp:ExactlyOne>
        </wsp:Policy>
    </wsdl:types>
    <xs:schema elementFormDefault="qualified" targetNamespace="http://ch.swisspost.ebill.b2bservice"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
```

```

        <xs:import namespace="http://swisspost_ch.ebs.ebill.b2bservice"/>
        <xs:element name="%List according to xs:element elements in wsdl%">

            </xs:element>
        </xs:schema>
    </wsdl:types>
    <wsdl:message name="%List according to wsdl:message elements in wsdl%">

    </wsdl:message>
    <wsdl:portType name="B2BService">
        <wsdl:operation name="%List according to wsdl:operation elements in wsdl%">

        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="UserNamePassword" type="tns:B2BService">
        <wsp:PolicyReference URI="#UserNamePassword_policy"/>
        <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="%List according to wsdl:operation elements in wsdl%">
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="B2BService">
        <wsdl:port name="UserNamePassword" binding="tns:UserNamePassword">
            <soap12:address location="https://ebill-ki.postfinance.ch/B2BService/B2BService.svc"/>
            <wsa10:EndpointReference>
                <wsa10:Address>https://ebill-ki.postfinance.ch/B2BService/B2BService.svc</wsa10:Address>
            </wsa10:EndpointReference>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>

```

Please change the yellow marked entries according your needs.

9.1.6 Code for JAX-WS

- Unit Test code to check that service can be properly accessed.

```
package b2bservice.ebill.swisspost.ch;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNotNull;

import java.net.MalformedURLException;
import java.net.URL;

import org.junit.Test;

public class B2BServiceTest {

    @Test
    public void testExecutePing() throws MalformedURLException {

        URL wsdlURL = new URL("https://ebill-ki.postfinance.ch/B2BService/B2BService.svc?singleWsdl");

        B2BService_Service service = new B2BService_Service(wsdlURL);
        B2BService port = service.getUserNamePassword();

        String result = port.executePing("41101000011707505", null, null, null);

        assertNotNull(result);

        assertEquals("41101000011707505", result);
    }
}
```

Please change the yellow marked entries according your needs.

9.2 Certificate authentication

9.2.1 Configuration for Axis2

- Generate service client stub code with wsdl2java tool of Axis2.
- Create axis2-policy.xml file and place it in the resources folder of your java client project:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsp:Policy wsu:Id="B2BService_UserCertificate_policy" xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
  <wsp:ExactlyOne>
    <wsp:All>
      <sp:TransportBinding xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
        <wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
          <sp:TransportToken>
            <wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
              <sp:HttpsToken RequireClientCertificate="false"/>
            </wsp:Policy>
          </sp:TransportToken>
          <sp:AlgorithmSuite>
            <wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
              <sp:Basic256/>
            </wsp:Policy>
          </sp:AlgorithmSuite>
          <sp:Layout>
            <wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
              <sp:Strict/>
            </wsp:Policy>
          </sp:Layout>
          <sp:IncludeTimestamp/>
        </wsp:Policy>
      </sp:TransportBinding>
      <sp:EndorsingSupportingTokens xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
        <wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
          <sp:X509Token>
            <wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
              <sp:RequireThumbprintReference/>
              <sp:WssX509V3Token10/>
            </wsp:Policy>
          </sp:X509Token>
          <sp:SignedParts>
            <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
          </sp:SignedParts>
        </wsp:Policy>
      </sp:EndorsingSupportingTokens>
      <sp:Wss11 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
        <wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
          <sp:MustSupportRefThumbprint/>
        </wsp:Policy>
      </sp:Wss11>
      <sp:Trust10 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
        <wsp:Policy xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
          <sp:MustSupportIssuedTokens/>
          <sp:RequireClientEntropy/>
          <sp:RequireServerEntropy/>
        </wsp:Policy>
      </sp:Trust10>
      <wsaw:UsingAddressing xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"/>
      <ramp:RampartConfig xmlns:ramp="http://ws.apache.org/rampart/policy">
        <ramp:rampartConfigCallbackClass>b2bservice.ebill.swisspost.ch.SecurityConfigCallbackHandler</ramp:rampartConfigCallbackClass>
        <ramp:signatureCrypto>
          <ramp:crypto provider="org.apache.ws.security.components.crypto.Merlin" enableCryptoCaching="false">
            <ramp:property name="org.apache.ws.security.crypto.merlin.keystore.type">PKCS12</ramp:property>
            <ramp:property name="org.apache.ws.security.crypto.merlin.file">Certificate.p12</ramp:property>
            <ramp:property name="org.apache.ws.security.crypto.merlin.keystore.password">Certificate password</ramp:property>
          </ramp:crypto>
        </ramp:signatureCrypto>
      </ramp:RampartConfig>
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Please change the yellow marked entries according your needs.

- Copy your Certificate.p12 file to the resources folder of your java client project

9.2.2 Code for Axis2

- Create new Java code file, which has the following code:

```
package b2bservice.ebill.swisspost.ch;

import org.apache.rampart.RampartConfigCallbackHandler;
import org.apache.rampart.policy.model.RampartConfig;

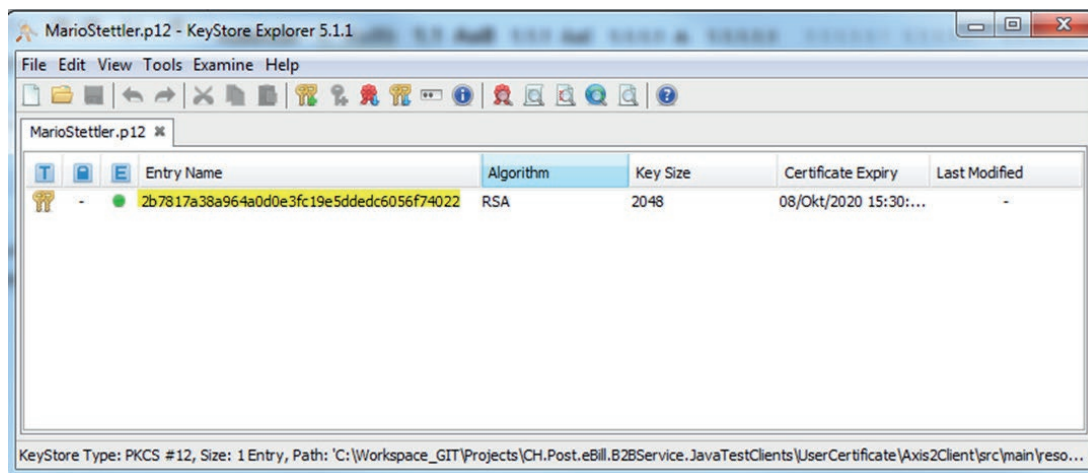
public class SecurityConfigCallbackHandler implements
    RampartConfigCallbackHandler {

    @Override
    public void update(RampartConfig rampartConfig)
    {
        rampartConfig.setUserCertAlias("Alias withing Certificat.p12");
        rampartConfig.setPwCbClass>PasswordCallbackHandler.class.getName());
    }
}
```

Please change the yellow marked entries according your needs.

Alias can be easily figured out with tool "Keystore Explorer":

<http://keystore-explorer.sourceforge.net/index.php>



Screenshot of Keystore Explorer: Alias is mark in yellow.

- Make sure you put the name incl. package path to `ramp:rampartConfigCallbackClass` element (see chapter 8.1.1).

- Create new Java code file, which has the following code:

```
package b2bservice.ebill.swisspost.ch;

import java.io.IOException;

import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.UnsupportedCallbackException;

import org.apache.ws.security.WSPasswordCallback;

public class PasswordCallbackHandler implements CallbackHandler {

    @Override
    public void handle(Callback[] callbacks) throws IOException,
        UnsupportedCallbackException {

        for (int i = 0; i < callbacks.length; i++) {

            // To use the private key to sign messages, we need to provide
            // the private key password
            WSPasswordCallback pwcb = (WSPasswordCallback) callbacks[i];

            if (pwcb.getIdentifier().equals("Alias withing Certificat.p12")) {
                pwcb.setPassword("Certificate password");
                return;
            }

        }

    }
}
```

Please change the yellow marked entries according your needs.

- Unit Test code to check that service can be properly accessed.

```
package b2bservice.ebill.swisspost.ch;

import org.apache.axiom.om.impl.builder.StAXOMBuilder;
import org.apache.axis2.client.ServiceClient;
import org.apache.neethi.Policy;
import org.apache.neethi.PolicyEngine;
import org.apache.rampart.RampartMessageData;
import org.apache.xmlbeans.XmlObject;

/**
 * B2BServiceTest Junit test case
 */

public class B2BServiceTest extends junit.framework.TestCase {

    /**
     * Auto generated test method
     */
    public void testExecutePing() throws java.lang.Exception {

        B2BServiceStub stub = new B2BServiceStub("https://ebill-ki.postfinance.ch/B2BService/B2BServiceCert.svc");

        ServiceClient sc = stub._getServiceClient();

        sc.engageModule("addressing"); // module to engage WS-Addressing
        sc.engageModule("rampart"); // module to engage WS-Security

        // load axis2 policy file and use it to configure rampart module
        String policyFile = getClass().getClassLoader().getResource("axis2-policy.xml").getFile();
        sc.getOptions().setProperty(RampartMessageData.KEY_RAMPART_POLICY, loadPolicy(policyFile));

        // prepare webservice request
        ExecutePingDocument executePingData = (ExecutePingDocument) getTestObject(ExecutePingDocument.class);

        executePingData.addNewExecutePing().setBillerID("41101000011707505");

        // execute webservice method
        ExecutePingResponseDocument response = stub.executePing(executePingData);

        assertNotNull(response);

        assertEquals("41101000011707505", response.getExecutePingResponse().getExecutePingResult());
    }

    // Create the desired XmlObject and provide it as the test object
    public XmlObject getTestObject(java.lang.Class<?> type)
        throws java.lang.Exception {
        java.lang.reflect.Method creatorMethod = null;
        if (org.apache.xmlbeans.XmlObject.class.isAssignableFrom(type)) {
            Class<?>[] declaredClasses = type.getDeclaredClasses();
            for (int i = 0; i < declaredClasses.length; i++) {
                Class<?> declaredClass = declaredClasses[i];
                if (declaredClass.getName().endsWith("$Factory")) {
                    creatorMethod = declaredClass
                        .getMethod("newInstance", null);
                    break;
                }
            }
        }
        if (creatorMethod != null) {
            return (XmlObject) creatorMethod.invoke(null, null);
        } else {
            throw new java.lang.Exception("Creator not found!");
        }
    }

    private static Policy loadPolicy(String xmlPath) throws Exception {
        StAXOMBuilder builder = new StAXOMBuilder(xmlPath);
        return PolicyEngine.getPolicy(builder.getDocumentElement());
    }
}
```

Please change the yellow marked entries according your needs.

9.2.3 Configuration for CXF

- Generate service client stub code with wsdl2java tool of CXF.
IMPORTANT: Make sure you pass the following parameter:
`-autoNameResolution`
- Create cxf.xml file and place it in the resources folder of your java client project:

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:jaxws="http://cxf.apache.org/jaxws"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://cxf.apache.org/jaxws
    http://cxf.apache.org/schemas/jaxws.xsd">

  <jaxws:client name="{http://ch.swisspost.ebill.b2bservice}UserCertificate" createdFromAPI="true">
    <!-- Comment below element to use non-WS-SecPol CXF interceptor method -->
    <jaxws:properties>
      <entry key="security.callback-handler" value="b2bservice.ebill.swisspost.ch.PasswordCallbackHandler"/>
      <entry key="security.signature.properties" value="clientKeystore.properties"/>
      <entry key="security.signature.username" value="Alias within Certificate.p12"/>
    </jaxws:properties>
  </jaxws:client>

</beans>
```

Please change the yellow marked entries according your needs.

- Copy your Certificate.p12 file to the resources folder of your java client project
- Create clientKeystore.properties file and place it in the resources folder of your java client project:

```
org.apache.ws.security.crypto.merlin.keystore.file=Certificate.p12
org.apache.ws.security.crypto.merlin.keystore.password=Certificate password
org.apache.ws.security.crypto.merlin.keystore.type=PKCS12
```

Please change the yellow marked entries according your needs.

9.2.4 Code for CXF

- Create new Java code file, which has the following code:

```
package b2bservice.ebill.swisspost.ch;

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

import javax.xml.security.auth.callback.Callback;
import javax.xml.security.auth.callback.CallbackHandler;
import javax.xml.security.auth.callback.UnsupportedCallbackException;

import org.apache.wss4j.common.ext.WSPasswordCallback;

public class PasswordCallbackHandler implements CallbackHandler {

    private Map<String, String> passwords =
        new HashMap<String, String>();

    public PasswordCallbackHandler() {
        passwords.put("Alias withing Certificate.p12", "Certificate password");
    }

    public void handle(Callback[] callbacks) throws IOException, UnsupportedCallbackException {
        for (int i = 0; i < callbacks.length; i++) {
            WSPasswordCallback pc = (WSPasswordCallback)callbacks[i];

            String pass = passwords.get(pc.getIdentifier());
            if (pass != null) {
                pc.setPassword(pass);
                return;
            }
        }
    }
}
```

Please change the yellow marked entries according your needs.

- Make sure you put the name incl. package path to `security.callback-handler` entry element (see chapter 8.1.3).

- Unit Test code to check that service can be properly accessed.

```
package b2bservice.ebill.swisspost.ch;
import static org.junit.Assert.*;

import java.net.MalformedURLException;
import java.net.URL;

import org.junit.Test;

public class B2BServiceTest {

    @Test
    public void testExecutePing() throws MalformedURLException {

        URL wsdlURL = new URL("https://ebill-ki.postfinance.ch/B2BService/B2BServiceCert.svc?singleWsd1");

        B2BService_Service service = new B2BService_Service(wsdlURL, B2BService_Service.SERVICE);
        B2BService port = service.getUserCertificate();

        String result = port.executePing("41101000011707505", null, null, null);

        assertNotNull(result);

        assertEquals("41101000011707505", result);
    }

}
```

Please change the yellow marked entries according your needs.

9.2.5 Configuration for JAX-WS

- Generate service client stub code with wsimport tool of JAX-WS.
IMPORTANT: Make sure you pass the following parameter:
`-extension -B-XautoNameResolution`
- Create wsit-client.xml file and place it in the resources folder of your java client project:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    name="mainclientconfig">
    <import location="B2BService.xml" namespace="http://ch.swisspost.ebill.b2bservice" />
</wsdl:definitions>
```

Please change the yellow marked entries according your needs.

- Copy your Certificate.p12 file to the resources folder of your java client project.

- Create service XML file and name it as you like. Place it in the resources folder of your java client project. Make sure you put the name to **location** attribute of import element (see chapter 8.1.5).

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions
    name="B2BService"
    targetNamespace="http://ch.swisspost.ebill.b2bservice"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
    xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex"
    xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/addressing/policy"
    xmlns:msc="http://schemas.microsoft.com/ws/2005/12/wsdl/contract"
    xmlns:i0="http://ch.swisspost.ebill.B2BService"
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tns="http://ch.swisspost.ebill.b2bservice"
    xmlns:wsa10="http://www.w3.org/2005/08/addressing"
    xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:sc1="http://schemas.sun.com/2006/03/wss/client"
    xmlns:wsp1="http://java.sun.com/xml/ns/wsdl/policy">
    <wsp:Policy wsu:Id="UserCertificate_policy">
        <wsp:ExactlyOne>
            <wsp:All>
                <sp:TransportBinding xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
                    <wsp:Policy>
                        <sp:TransportToken>
                            <wsp:Policy>
                                <sp:HttpsToken RequireClientCertificate="false"/>
                            </wsp:Policy>
                        </sp:TransportToken>
                        <sp:AlgorithmSuite>
                            <wsp:Policy>
                                <sp:Basic256/>
                            </wsp:Policy>
                        </sp:AlgorithmSuite>
                        <sp:Layout>
                            <wsp:Policy>
                                <sp:Strict/>
                            </wsp:Policy>
                        </sp:Layout>
                        <sp:IncludeTimestamp/>
                    </wsp:Policy>
                </sp:TransportBinding>
                <sp:EndorsingSupportingTokens xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
                    <wsp:Policy>
                        <sp:X509Token
                            sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToRecipient">
                            <wsp:Policy>
                                <sp:RequireThumbprintReference/>
                                <sp:WssX509V3Token10/>
                            </wsp:Policy>
                        </sp:X509Token>
                        <sp:SignedParts>
                            <sp:Header Name="To" Namespace="http://www.w3.org/2005/08/addressing"/>
                        </sp:SignedParts>
                    </wsp:Policy>
                </sp:EndorsingSupportingTokens>
                <sp:Wss11 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
                    <wsp:Policy>
                        <sp:MustSupportRefThumbprint/>
                    </wsp:Policy>
                </sp:Wss11>
                <sp:Trust10 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
                    <wsp:Policy>
                        <sp:MustSupportIssuedTokens/>
                        <sp:RequireClientEntropy/>
                        <sp:RequireServerEntropy/>
                    </wsp:Policy>
                </sp:Trust10>
            </wsp:All>
        </wsp:ExactlyOne>
    </wsp:Policy>

```

```

        </sp:Trust10>
        <wsaw:UsingAddressing/>
        <scl:KeyStore wssp:visibility="private" keypass="Certificate password"
            storepass="Certificate password" type="PKCS12"
            location="Certificate.p12" />
    </wsp:All>
</wsp:ExactlyOne>
</wsp:Policy>
<wsdl:types>
    <xs:schema elementFormDefault="qualified" targetNamespace="http://ch.swisspost.ebill.b2bservice"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:import namespace="http://swisspost_ch.ebs.ebill.b2bservice"/>
        <xs:element name="%List according to xs:element elements in wsdl%">
        </xs:element>
    </xs:schema>
</wsdl:types>
<wsdl:message name="%List according to wsdl:message elements in wsdl%">
</wsdl:message>
<wsdl:portType name="B2BService">
    <wsdl:operation name="%List according to wsdl:operation elements in wsdl%">
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="UserNamePassword" type="tns:B2BService">
    <wsp:PolicyReference URI="#UserNamePassword_policy"/>
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="%List according to wsdl:operation elements in wsdl%">
    </wsdl:operation>
</wsdl:binding>
<wsdl:service name="B2BService">
    <wsdl:port name="UserCertificate" binding="tns:UserCertificate">
        <soap12:address location="https://ebill-ki.postfinance.ch/B2BService/B2BServiceCert.svc"/>
        <wsa10:EndpointReference>
            <wsa10:Address>https://ebill-ki.postfinance.ch/B2BService/B2BServiceCert.svc</wsa10:Address>
        </wsa10:EndpointReference>
    </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

Please change the yellow marked entries according your needs.

9.2.6 Code for JAX-WS

- Unit Test code to check that service can be properly accessed.

```
package b2bservice.ebill.swisspost.ch;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNotNull;

import java.net.MalformedURLException;
import java.net.URL;

import org.junit.Test;

public class B2BServiceTest {

    @Test
    public void testExecutePing() throws MalformedURLException {

        URL wsdlURL = new URL("https://ebill-ki.postfinance.ch/B2BService/B2BServiceCert.svc?singlewsdl");

        B2BService_Service service = new B2BService_Service(wsdlURL);
        B2BService port = service.getUserCertificate();

        String result = port.executePing("41101000011707505", null, null, null);

        assertNotNull(result);

        assertEquals("41101000011707505", result);
    }
}
```

Please change the yellow marked entries according your needs.

10. Disclaimer

We created this specification with great care. Nevertheless we cannot guarantee that there are no errors and no inaccuracy.

Thank you for your information when you believe to have found such an error.

All information in this interface specification can be changed anytime without prior notice.