Implementing BankID for Mobile based on BankID WebClient

Innhold

This document describes what must be done to implement BankID for Mobile (BIM) based on an existing BankID 2.x implementation. The document does not describe how to implement the BIM GUI.

Similarities and differences between BIM and WebClient

Similarites

- Uses a BIDFacade to access BankID functions
- Callbacks from BankID have the same format and sequence

Differences

- Clients must build the GUI
- Clients must supply the user's phoneNumber and the user's birthdate (YYMMDD)
- Client must create a unique identifier for the BIM session
- Requests are not sent from the users browser but from BIM Gateway, finding the web application's sessiondata must therefore be based on the unique identifier
- Clients must choose either synchronous or asynchronous mobile action mode when initiating the BankID session in BankID server.
    - For asynchronous mode, clients must either poll their server to check for completion or use websockets to notify browser that BIM is finished
    - For synchronous mode, the BankID Server will block until the user is finished
- The user cannot cancel a started operation, must wait for timeout to retry/reinitialize
- Locale for the phone GUI cannot be set
- Signing is restricted to text mode and at most 118 characters, character set is not exactly ISO-8859-1 (these restrictions are subject to change)
- Signing does not support multidocument signing
- Man in the Middle Configuration must include the callback url's hostname and ip.

Building the GUI

The GUI layout and process flow are defined in the document: ***Brukergrensesnitt BankID Mobil.pdf***
How this is implemented is out of scope for this document.

Preconditions

MITM configuration

The propertyfile for the BankID merchant , *yourmerchant.prop*, file must have a webadress entry to allow callbacks from BIM gateway:

webaddress=myhost.mydomain,123.123.123.123

If the hostname maps to several ip addresses, these must be separated with commas.

Assumptions about existing implementation

Since there are many different ways to implement BankID 2.x functionality, a simple one in Java using session cookies to identify session state for the user's web session is assumed. Clustering, error handling etc. is beyond scope.

To get the user's BankID session state the following code is assumed:

```
BankIDState bankIDState = (BankIDState)
httpRequest.getSession().getAttribute("BankIDState");
```

The BankIDState class is a assumed to hold all state needed for handling BankID WebClient signing or authentication.

Implementing BIM

Holding and finding BankIDState, sid, bankIDStateMap

Since the callbacks from BankID are initiated by the BIM gateway, no session cookie wil exist for these requests. The requests from BIM gateway must anyhow be connected to the user's session. A unique identifier, sid, must be created and used as a key to identify the user's session.

A sid may be created using:

```
String sid = UUID.randomUUID().toString();
```

All data used to hold state for a giver BankID session should be accessible using this sid. A size restricted java Map may be sufficient for most applications. This map may be created by Spring, in a static variable or likewise.

```
Map<String,BankIDState> bankIDStateMap = new SizeRestrictedMap<>(5000);
```

Changes to BankIDState handling

Move BankIDState data stored in the httpSession to the Map keyed by the sid. Store the sid in the httpSession.

When creating the user's instance of BankIDState, a sid should be created and this should be placed in the state map and the httpSession.

Replace

```
httpRequest.getSession().addAttribute("BankIDState", new BankIDState());
```

with

```
String sid = UUID.randomUUID().toString();
bankIDStateMap.put(sid, new BankIDState());
httpRequest.getSession().addAttribute("sid", sid);
```

and replace

```
BankIDState bankIDState =
    (BankIDState) httpRequest.getSession().getAttribute("BankIDState");
```

with

```
String sid = (String) httpRequest.getSession().getAttribute("sid");
BankIDState bankIDState = bankIDStateMap.get(sid);
```

When handling request from BankID WebClient and BankID mobile BankID infrastructure will always send the sid as a request parameter. When handling these requests, replace

```
BankIDState bankIDState = (BankIDState)
httpRequest.getSession().getAttribute("BankIDState");
```

with

```
String sid = (String) httpRequest.getParameter("sid");
BankIDState bankIDState = bankIDStateMap.get(sid);
```

This should be sufficient regarding changes in state handling for BIM and WebClient.

Preparing the mobile interaction

Get values for and verify format for phonenumber (8 digits) and birthdate (YYMMDD) from the user

Generate the sid for the user's BankID session and the corresponding BankIDState.

```
String sid = UUID.randomUUID().toString();
bankIDStateMap.put(sid, new BankIDState());
httpRequest.getSession().addAttribute("sid", sid);
```

Create a MobileInfo and store it in the BankIDState:

```
MobileInfo mobileInfo = new MobileInfo();
mobileInfo.setDoSynchronousCommunication(false); // for requestMobileAction
mobileInfo.setSid(sid);
mobileInfo.setUrl(callbackUrl);//as 2.x's InitSessionInfo.setMerchantURL

mobileInfo.setPhoneAlias(birthDate);
mobileInfo.setPhoneNumber(phoneNumber);

bankIDState.setMobileInfo(mob);
```

Authentication

Generate the merchant reference (ex: feilfri diamant).

```
String merchantReference = bidFacade.generateMerchantReference("no_NO");

mobileInfo.setMerchantReference(merchantReference);
mobileInfo.setAction("auth");
```

Signing

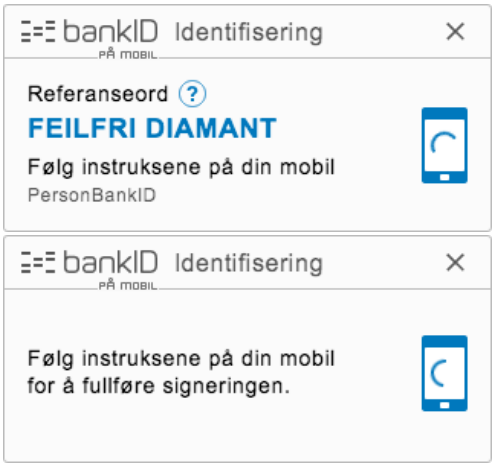
Verify the length and the characters of the data to sign. The length is restricted to 118 characters and the legal characters are (for now):

[0-9] [a-z] [æ] [ø] [å] [A-Z] [Æ] [Ø] [Å] [CR] [LF] [#] [$] [%-&] [(-?] [@]
[¡] [£] [¤] [¥] [§] [¿] [Ä] [Ç] [É] [Ñ] [Ö] [Ü] [ß] [à] [ä] [è] [é] [ì] [ñ] [ò] [ö] [ù]

```
mobileAction.setAction(”sign”);
bankIDStateData.setTextToSign(stringToSign);// used in operation=”initSign”
```

Starting the mobile interaction

Display the waiting dialog, for authentication the merchant reference (feilfri diamant) must be included.



Then start the action on the mobile

```
TransactionAndStatus transactionAndStatus =
        bidFacade.requestMobileAction(mobileInfo); // this is async or sync

if (! ”0”.equals(transactionAndStatus.getStatusCode())) {
    bankIDState.setErrorCode(transactionAndStatus.getStatusCode());
} else {
    // If getTransactionInfo or pushSms needed, the reference is needed.
    bankIDState.setTransactionReference(getTransactionReference());
}
```

Handle callbacks from BankID for mobile

These callbacks does not come from the user’s browser but from BIM gateway.

```
Handle this request almost exactly the same way as the callbacks from
BankID WebClient. Use the request parameter “sid” to get the BankIDState to
work with.
Return the same as for BankID WebClient.

If the BankIDState is not found, i.e. unknown “sid”, then return
“errCode=someNumber”.
```

When receiving a request having operation parameter equal to "verifyAuth","verifySign" or "handleError" the bankIDState must be marked as finished.

In case of an exception the bankIDState must also be marked as finished.

```
bankIDState.setFinishState("FINISHED");
```

or

```
bankIDState.setFinishState("ERROR");
```

Considerations about "initSign":

Signing in BIM does not support multidocument as introduced in BankID 2.1 for the BIDSessionData class. To set document to sign the following should be used:

```
sessionData.setDataToBeSigned(bankIDState.getTextToSign());
sessionData.setDataToBeSignedMimeType("text/plain");
sessionData.setDataDescription("Some description");
```

Browser side JavaScript modifications, asynchronous mode

There are several ways to handle BIM in the browser. The browser may periodically poll the server for BankIDState using the sessioncookie or the sid. The server may also use some pushtechnology as websockets to update the client when a terminating operation code or an error occurs.

A simple polling function using jQuery looks like.

```
        function performPoll() {
            $.post('https://myapp/poll',
                function (data) {
                    switch (data.state) {
                        default:
                            setTimeout(performPoll, 1000);
                            break;
                        case 'FINISHED':
                            handleBIMFinished();
                            break;
                        case 'ERROR':
                            handleBIMError();
                            break;
                    }
                }, 'json');
        }
        setTimeout(performPoll, 0); // Starts polling
```

A Spring request handler for the action looks like:

```
@RequestMapping(value="poll", method = RequestMethod.POST,
                produces = "application/json")
public String pollBankIDState(Model model, HttpSession session) {
    String sid = (String) session.getAttribute("sid");
    BankIDState bankIDState = bankIDStateMap.get(sid);
    model.addAttribute("nextStep",
       bankIDState == null ? "nextStep" : bankIDState.getBankIDState());
    return "json";
}
```

When the bankID WebClient terminates it either calls a JavaScript callback function or redirects to another page. For BIM this must explicitly be handled by either calling the same JavaScript callback function or setting window.location.href = nextUrl;

Various options

Sid as requestparameter or part of callback url

Instead of using the unique identifier, sid, as a request parameter, the identifier may be used as part of the callback url.

`Unused methods`

`MobileInfo.setCountryCode` must be used if the user's registered BankID phone number is not Norwegian.

`MobileInfo.setPhoneNumber` must not include the countrycode.

`MobileInfo.setLocale` is not supported.

Synchronous or asynchronous mode

The mobile interaction may be started in asynchronous and in synchronous mode. Setting
`mobileInfo.setDoSynchronousCommunication(true)`

makes

`bidFacade.requestMobileAction(mobileInfo)`

block until the user is finished interacting with the mobile. Callbacks from BIM gateway must anyhow be handled. When synchronous mode there will be no need for a polling loop in the browser since the initiating request will not return to the browser until BIM is finished.

If synchronous/blocking mode is acceptable or not is dependent on the application. Synchronous mode blocks a thread in the merchant web application, asynchronous mode blocks a thread in a browser window.

Debugging tip

Finding bugs is ofter easier using the synchronous mode. In asynchronous mode almost all errors will be reported using the callback url. If for some reason this callback cannot be received (error in url, firewalls e.al.) the error will just disappear. In synchronous mode such error will be seen. Therefore it can be recommended to use synchronous mode at least in the initial testing and development phase.

Getting error C221: this error, "Url given by the the merchant is illegal" very often means that

mobile-preprod1.bankid.no

is not accessible. See the troubleshooting chapter in BankID Implementation Guide.

Example code: SizeRestrictedMap

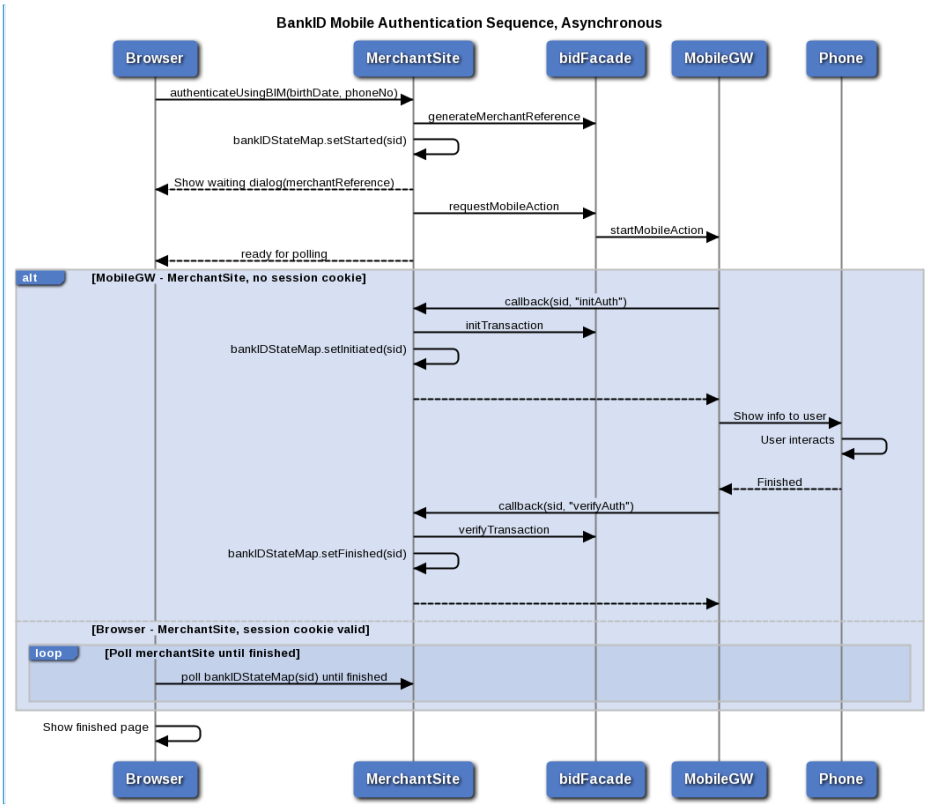
```
public class SizeRestrictedMap<K, V> extends LinkedHashMap<K, V>
{
```

```
    final int maxSize;

    public SizeRestrictedMap(int maxSize)
    {
        super(maxSize, 0.75f, true);
        this.maxSize = maxSize;
    }

    @Override
    protected boolean removeEldestEntry(Map.Entry<K,V> eldest)
    {
        return size() > maxSize;
    }
}
```

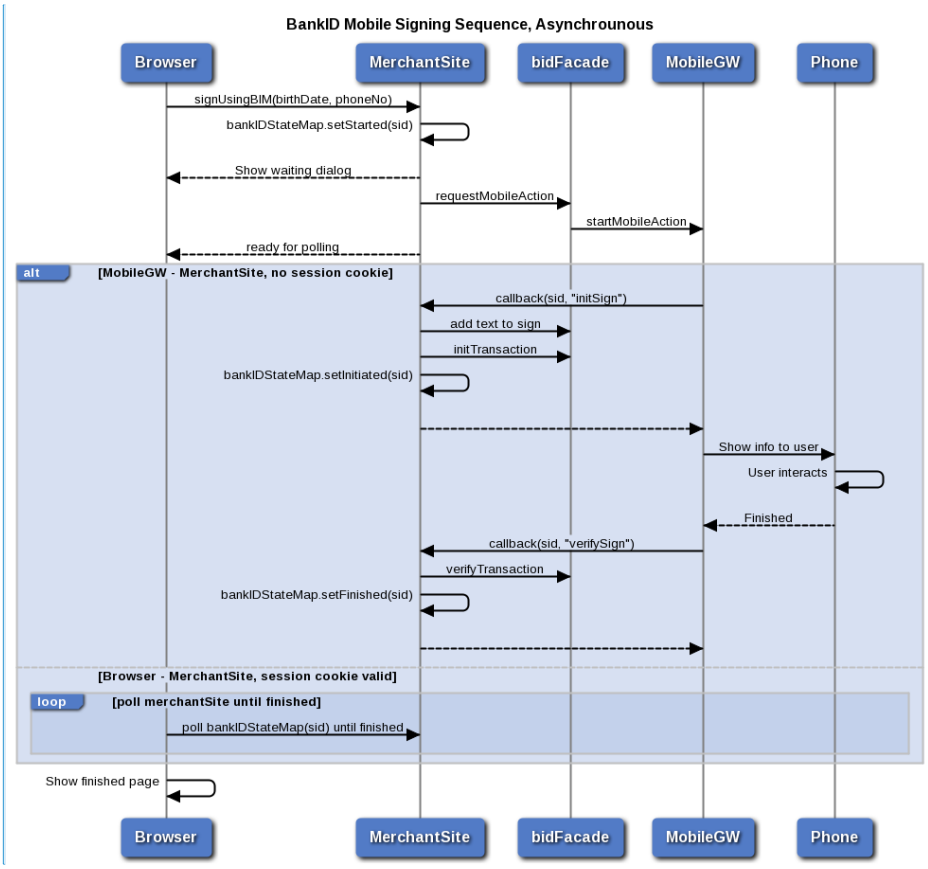
Sequence diagram, Authentication



Kommentert [s1]: WebSequenceDiagram source:
title BankID Mobile Authentication Sequence, Asynchronous
Browser -> MerchantSite: authenticateUsingBIM(birthDate, phoneNo)
MerchantSite -> bidFacade: generateMerchantReference
MerchantSite -> MerchantSite: bankIDStateMap.setStarted(sid)
MerchantSite --> Browser: Show waiting dialog(merchantReference)
MerchantSite -> bidFacade: requestMobileAction
bidFacade -> MobileGW: startMobileAction
MerchantSite --> Browser: ready for polling

alt MobileGW - MerchantSite, no session cookie
  MobileGW -> MerchantSite: callback(sid, "initAuth")
  MerchantSite -> bidFacade: initTransaction
  MerchantSite -> MerchantSite: bankIDStateMap.setInitiated(sid)
  MerchantSite --> MobileGW:
  MobileGW -> Phone: Show info to user
  Phone -> Phone: User interacts
  Phone --> MobileGW: Finished
  MobileGW -> MerchantSite: callback(sid, "verifyAuth")
  MerchantSite -> bidFacade: verifyTransaction
  MerchantSite -> MerchantSite: bankIDStateMap.setFinished(sid)
  MerchantSite --> MobileGW:
else Browser - MerchantSite, session cookie valid
loop Poll merchantSite until finished
Browser -> MerchantSite: poll bankIDStateMap(sid) until finished
end
end
Browser -> Browser: Show finished page

Sequence diagram, Signing



Kommentert [s2]: WebSequenceDiagram source:

```
title BankID Mobile Signing Sequence, Asynchrounous
Browser -> MerchantSite: signUsingBIM(birthDate, phoneNo)
MerchantSite -> MerchantSite: bankIDStateMap.setStarted(sid)
MerchantSite --> Browser: Show waiting dialog
MerchantSite -> bidFacade: requestMobileAction
bidFacade -> MobileGW: startMobileAction
MerchantSite --> Browser: ready for polling

alt MobileGW - MerchantSite, no session cookie
   MobileGW -> MerchantSite: callback(sid, "initSign")
   MerchantSite -> bidFacade: add text to sign
   MerchantSite -> bidFacade: initTransaction
   MerchantSite -> MerchantSite:
bankIDStateMap.setInitiated(sid)
   MerchantSite --> MobileGW:
   MobileGW -> Phone: Show info to user
   Phone -> Phone: User interacts
   Phone --> MobileGW: Finished
   MobileGW -> MerchantSite: callback(sid, "verifySign")
   MerchantSite -> bidFacade: verifyTransaction
   MerchantSite -> MerchantSite:
bankIDStateMap.setFinished(sid)
   MerchantSite --> MobileGW:
else Browser - MerchantSite, session cookie valid
loop poll merchantSite until finished
Browser -> MerchantSite: poll bankIDStateMap(sid) until finished
end
end
Browser -> Browser: Show finished page
```