Use Case Considerations for Extending DSS with Local Signature Computations

E.J. Van Nigtevecht
Sonnenglanz Consulting BV

Version June 9, 2011

Goal

- Enable DSS for use cases where the (secure) signature creation device is not part of the DSS service itself.
- Allow the client to specify the hash algorithm to be used.

Assumptions

- The OASIS DSS Core is used.
- A (Secure) Signature Creation Device is connected to a User-Agent or a (separate) User-Device.
- A User-Agent or User-Device may be equipped with a gradual set of signature-creation related functionality. For example ranging between:
  - APDU (ISO 7816);
  - IFD-Client (ISO/IEC 24727 / CEN 15480);
  - Full OASIS DSS(-X) profiles;
- A User-Agent or User-Device may have limited software & performance capabilities and hence may be supported by a Digital Signature Service to handle the complexities of the signature creation if it cannot manipulate the document itself.

Assumptions

- A User-Agent or User-Device will always initiate the transaction and acts as an HTTP-client.
- A document may remain on the client or server side or transferred from one side to the other.
- The default Use Case of DSS will not be shown (DSS req/resp with document as a parameter). Variants of the default Use Case are explored instead.

Some Terminology

- Terminology
  - userID: a way to identify a user;
  - docRef: a reference to a document (url) for retrieval;
    - Currently, DSS only supports a reference to the document inside the request structure. Therefore, a docRef only makes sense if the document is located elsewhere (not on the requesting client or DSS server).
  - docID: a way to identify a document by a user, in a user friendly manner;
  - digest: the hash of the document used for the signature creation (the calculation of the hash value depends on the type of document, for instance XML, PDF or ‘binary’);
  - digestSignature: the ‘raw’ signature of the digest;
  - hashAlg: the hash algorithm to be used (or that has been used);

Use Case 1

- Actor
  - An End-User.
- System
  - A User-Agent (the ‘client’) with a (secure) signature creation device, (S)SCD, connected to a Service (the ‘server’).
- Basic Restriction(s)
  - Communication between the client (the User-Agent) and the server (the Service) is always initiated by the client.
- Goal
  - By using a Digital Signature Service an end-user signs a document (located at the client or at the server) with the (S)SCD at the User-Agent.

Use Case 1

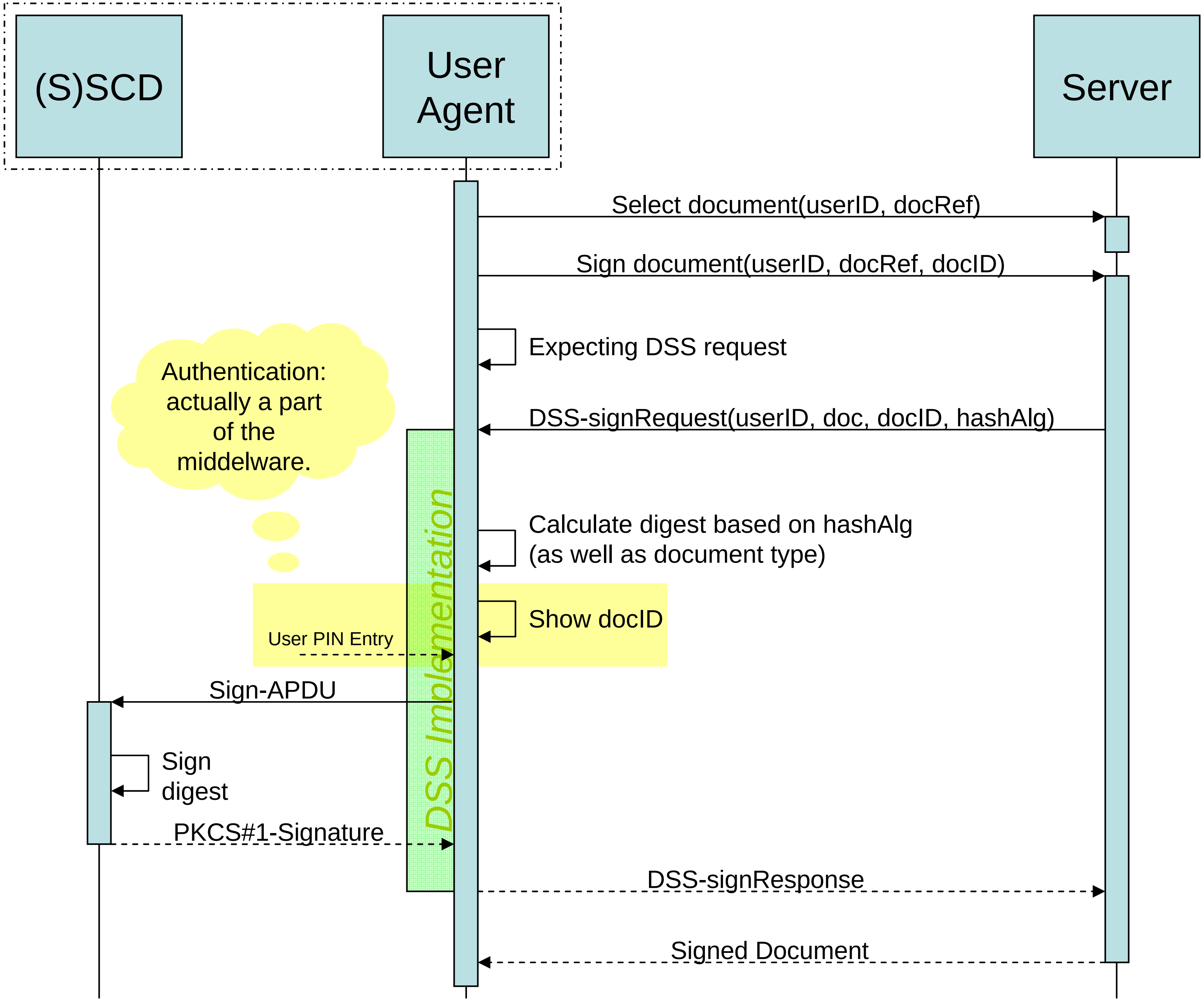
- Basic Flow
  - The End-User calls a Service by means of the User-Agent.
  - The End-User selects a document by means of the Service.
  - The End-User requests a signing operation for the document by means of the Service.
  - The User-Agent asks the user for a PIN or Password.
  - The End-User enters the PIN or Password.
  - The User-Agent creates the signature using the (Secure) Signature Creation Device.
  - The User-Agent shows the signed document.

- Example
  - A user signs a document, opened in a web browser (running at a PC) by means of a web application, using a smartcard/usb-token connected to the PC.
  - A user signs a document with an app on the iPhone using a certificate installed on the SIM-card at the same iPhone.

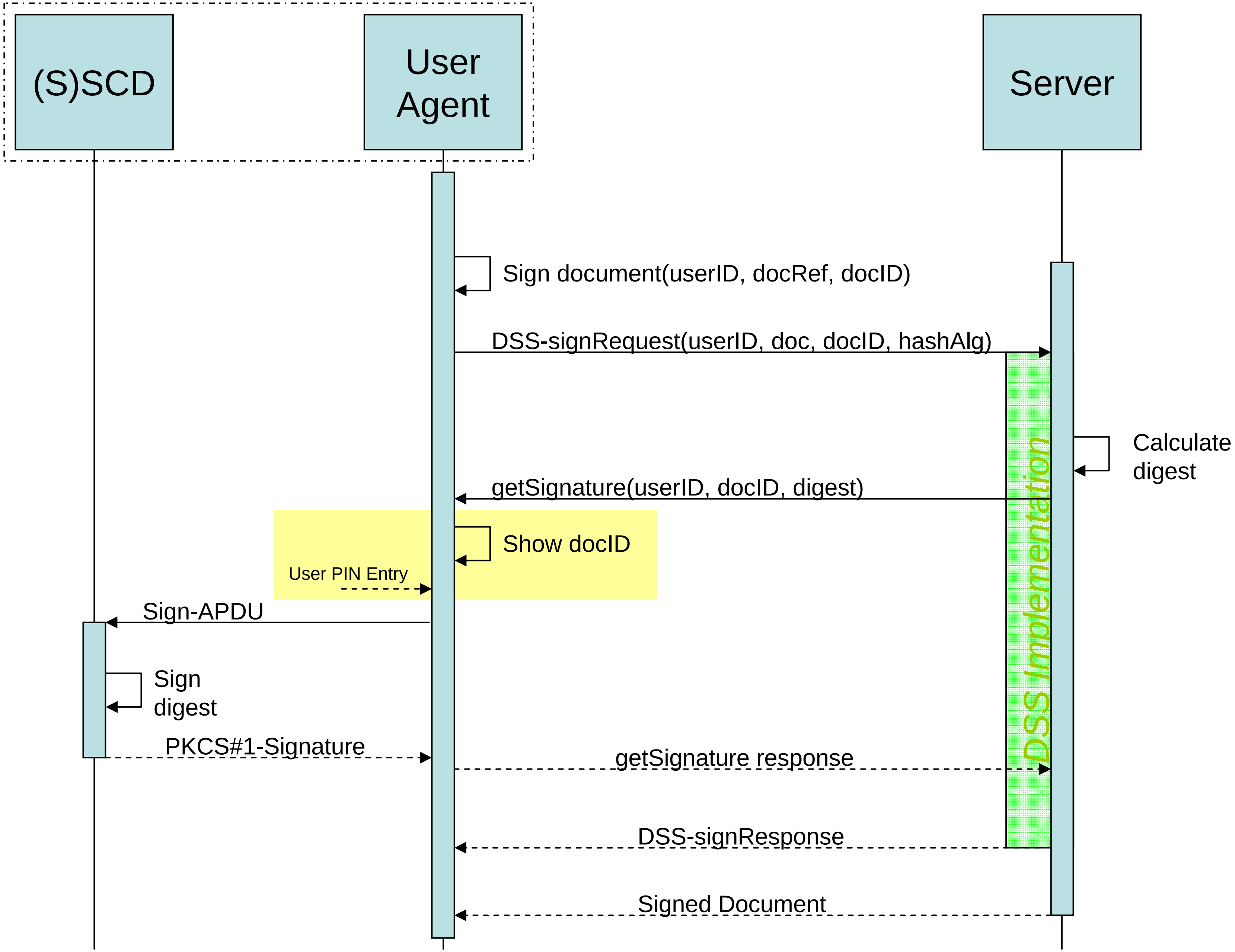
Variants Use Case 1

- Use Case 1a – Smart User-Agent
  - The User-Agent implements a Digital Signature Service.
  - The document is at the server.
- Use Case 1b – Simple User-Agent
  - The User-Agent is NOT capable of implementing a Digital Signature Service. Instead, the User Agent implements an IFD or an APDU interface.
  - A server that is used by the User-Agent (the client) for ‘business’ functionality and for Digital Signature Service functionality.
  - The document is at the client (User-Agent) and is transferred to server.
- Use Case 1c – Simple User-Agent
  - Like Use Case 1b, but the document stays at the client.

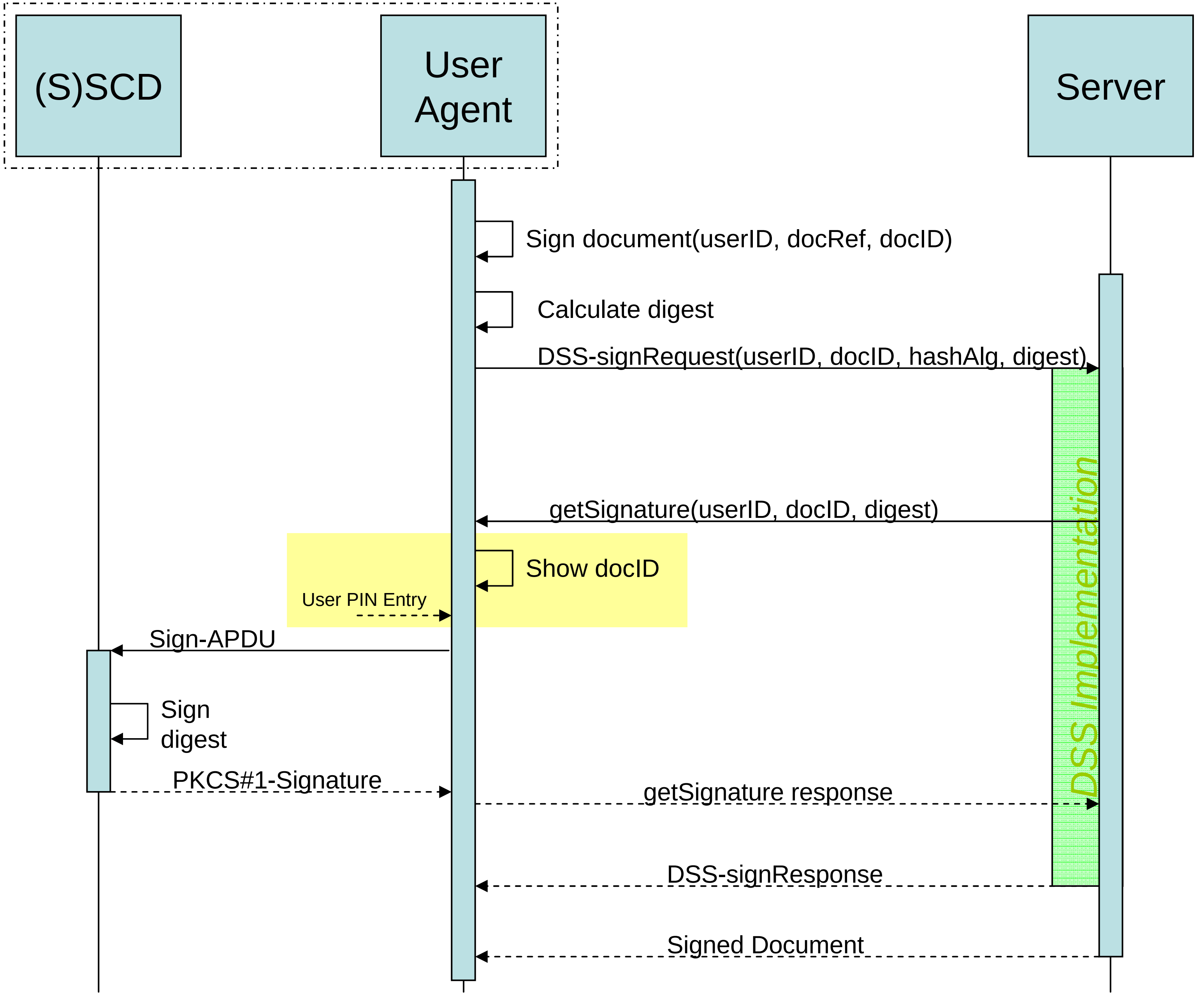
Sequence Diagram Use Case 1a – Smart User-Agent


(S)SCD
User Agent
Server
Select document(userID, docRef)
Sign document(userID, docRef, docID)
Expecting DSS request
DSS-signRequest(userID, doc, docID, hashAlg)
Authentication: actually a part of the middelware.
Calculate digest based on hashAlg (as well as document type)
Show docID
User PIN Entry
DSS Implementation
Sign-APDU
Sign digest
PKCS#1-Signature
DSS-signResponse
Signed Document

Sequence Diagram Use Case 1b – Simple User-Agent (document transfer)



Sequence Diagram Use Case 1c – Simple User-Agent (no document transfer)



(S)SCD

User Agent

Server

Sign document(userID, docRef, docID)

Calculate digest

DSS-signRequest(userID, docID, hashAlg, digest)

getSignature(userID, docID, digest)

Show docID

User PIN Entry

Sign-APDU

Sign digest

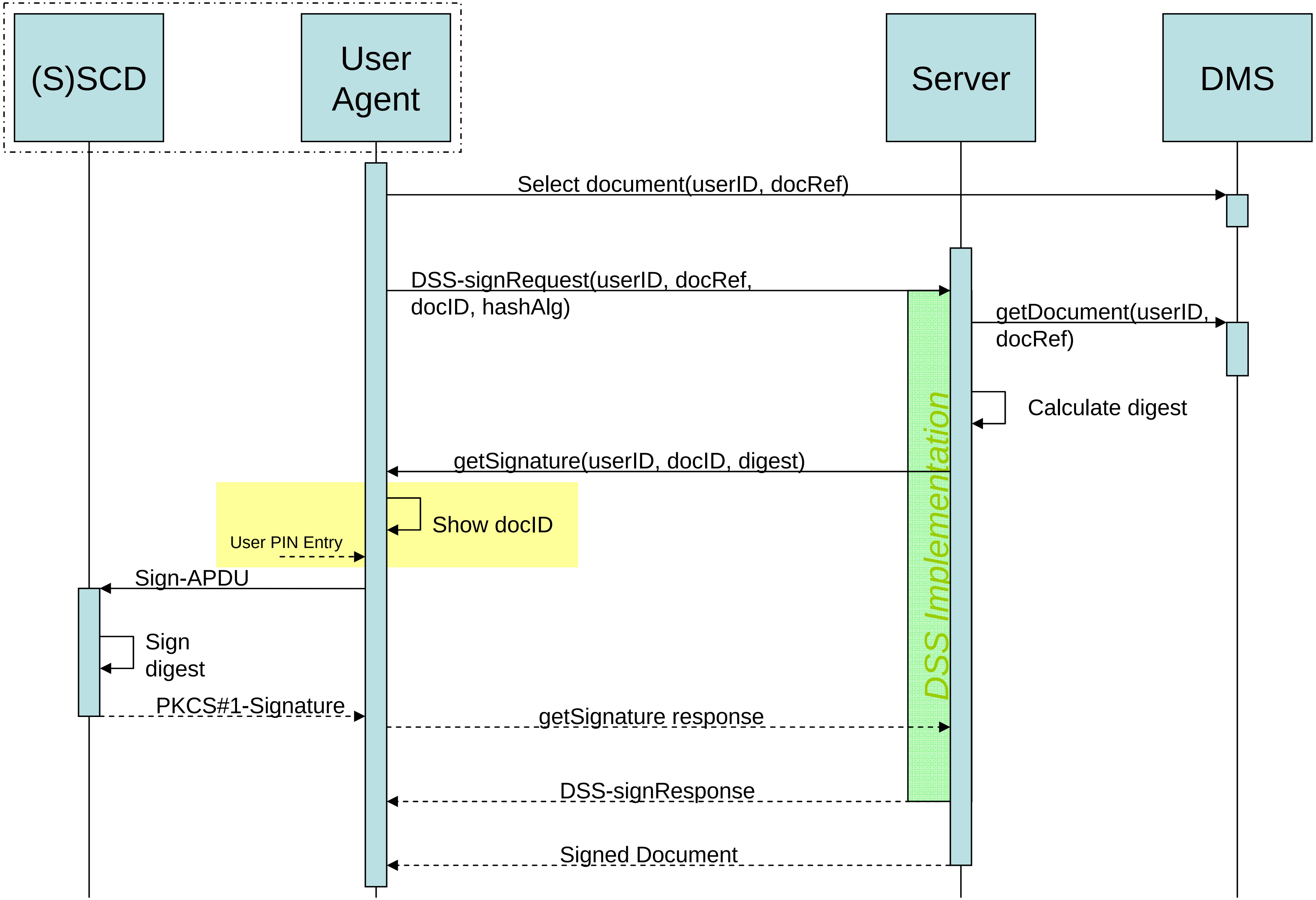
PKCS#1-Signature

getSignature response

DSS-signResponse

Signed Document

Sequence Diagram Use Case 1d – Simple User-Agent (document elsewhere)



Use Case 2

- Actor
  - An End-User.
- System
  - A User-Agent without a signature creation device, connected to a Service (the ‘server’). Another User-Device is used for the (secure) signature creation device.
- Basic Restriction(s)
  - Communication between the client (the User-Agent) and the server (the Service) is always initiated by the client.
- Goal
  - By using a Digital Signature Service an end-user signs a document (located at the client or at the server) with the (S)SCD at the User-Device.

Use Case 2

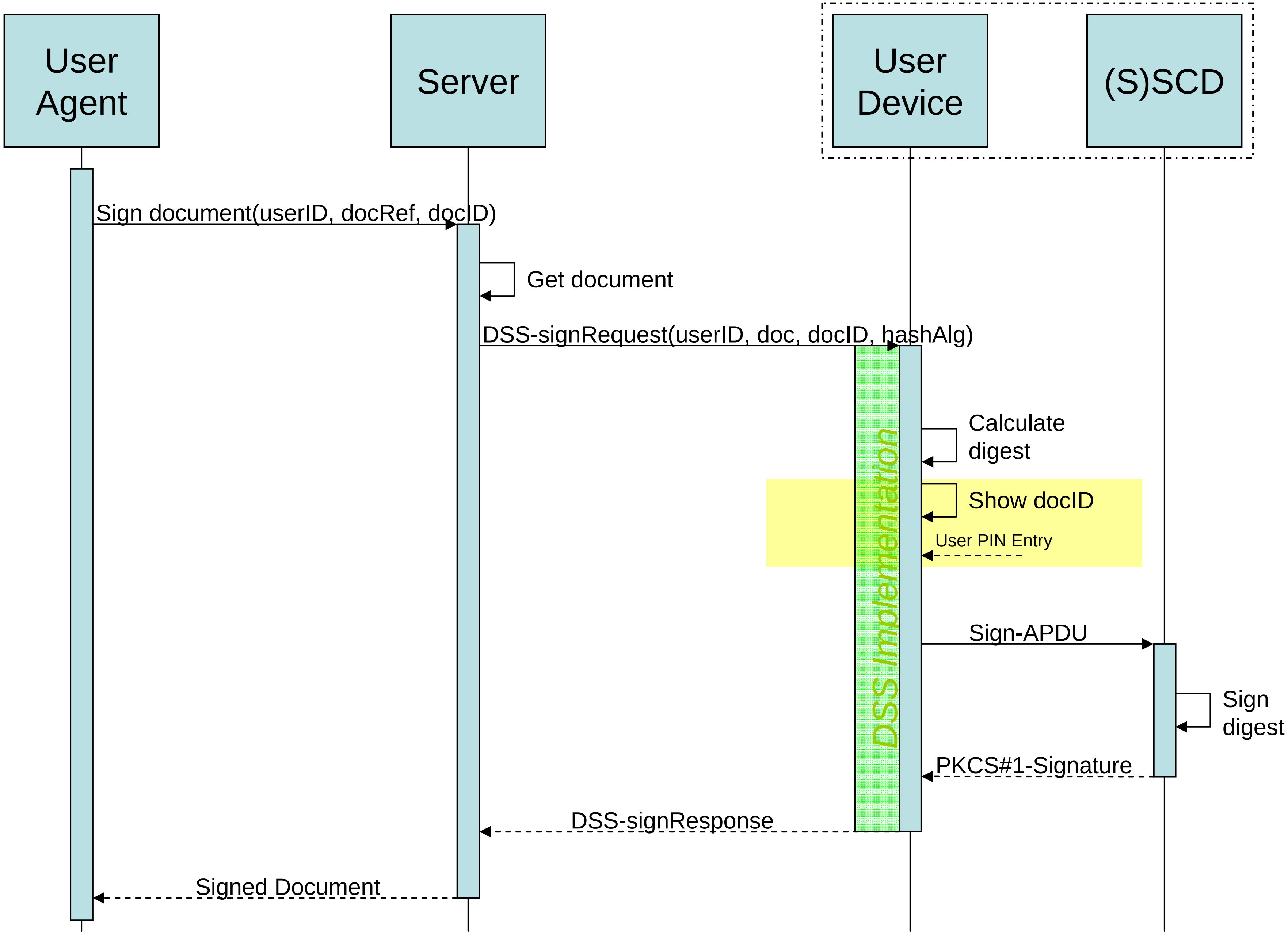
- Basic Flow
  - The End-User calls a Service by means of the User-Agent.
  - The End-User registers his/her User-Device at the Service (specifying device type and address).
  - The End-User selects a document by means of the Service.
  - The End-User requests a signing operation for the document by means of the Service. (The Service requests a signature creation operation at the User-Device.)
  - The User-Device shows an identification of the request and asks the user for a PIN or Password.
  - The End-User verifies if it is the right identification and enters the PIN or Password at the User-Device.
  - The User-Device creates the signature using the (Secure) Signature Creation Device.
  - The End-User views the signed document.

- Example
  - A user initiates a signature operation for a document with an app on his/her iPad, using a certificate installed on the SIM-card at his/her iPhone.

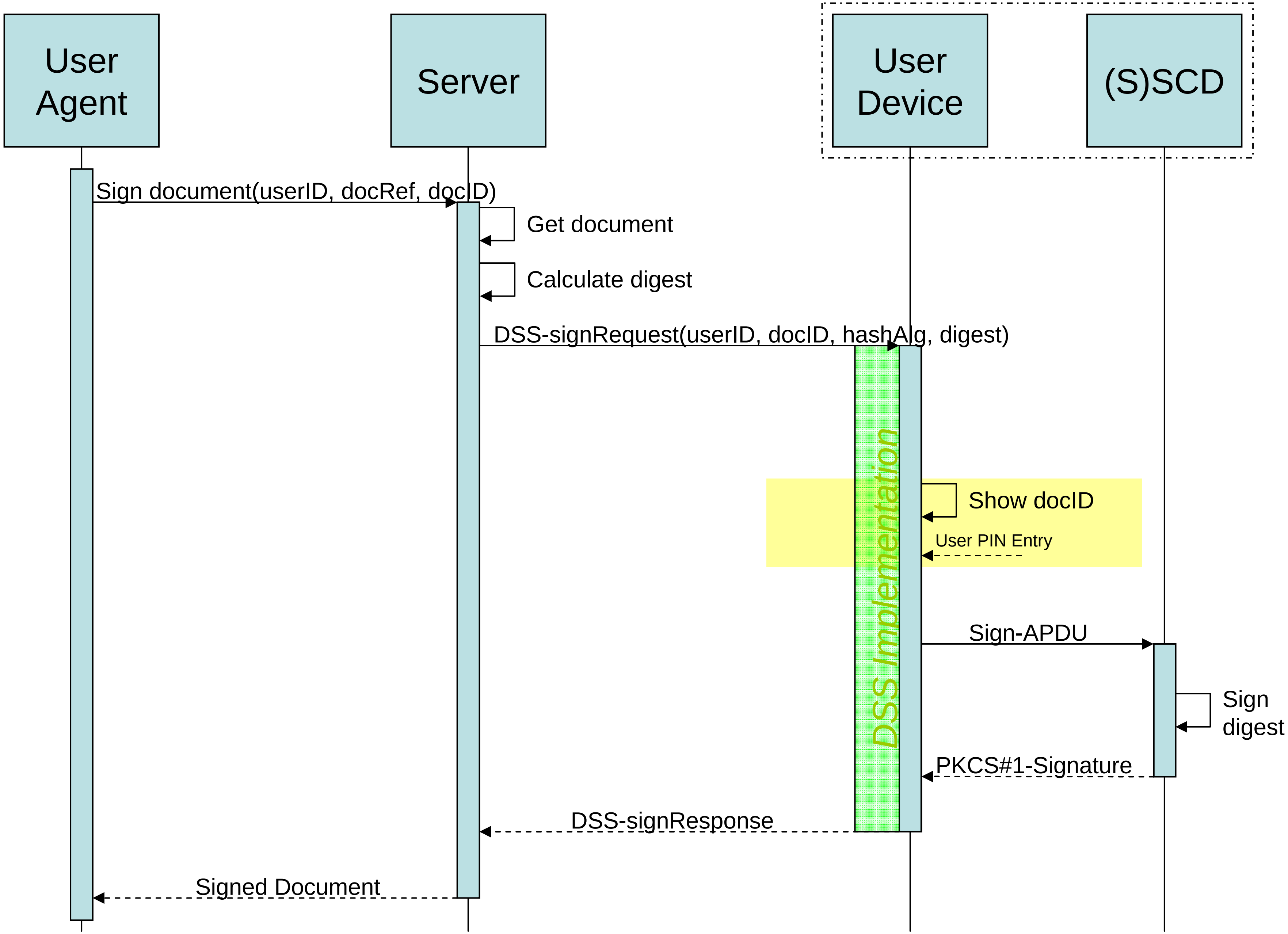
Variants Use Case 2

- Use Case 2a – Smart User-Device
  - The User-Device implements a Digital Signature Service.
  - The document is at the server and transferred to the User-Device.
- Use Case 2b: Like Use Case 2a, but the document stays at the server.
- Use Case 2c – Simple User-Device
  - The User-Device is NOT capable of implementing a Digital Signature Service. Instead, the User-Device implements an IFD or an APDU interface.
  - A server that is used by the User-Agent (the client) for ‘business’ functionality and for Digital Signature Service functionality.
  - The document is at the client (User-Agent) and is transferrer to the server.
- Use Case 2d: Like Use Case 2b, but the document stays at the client.

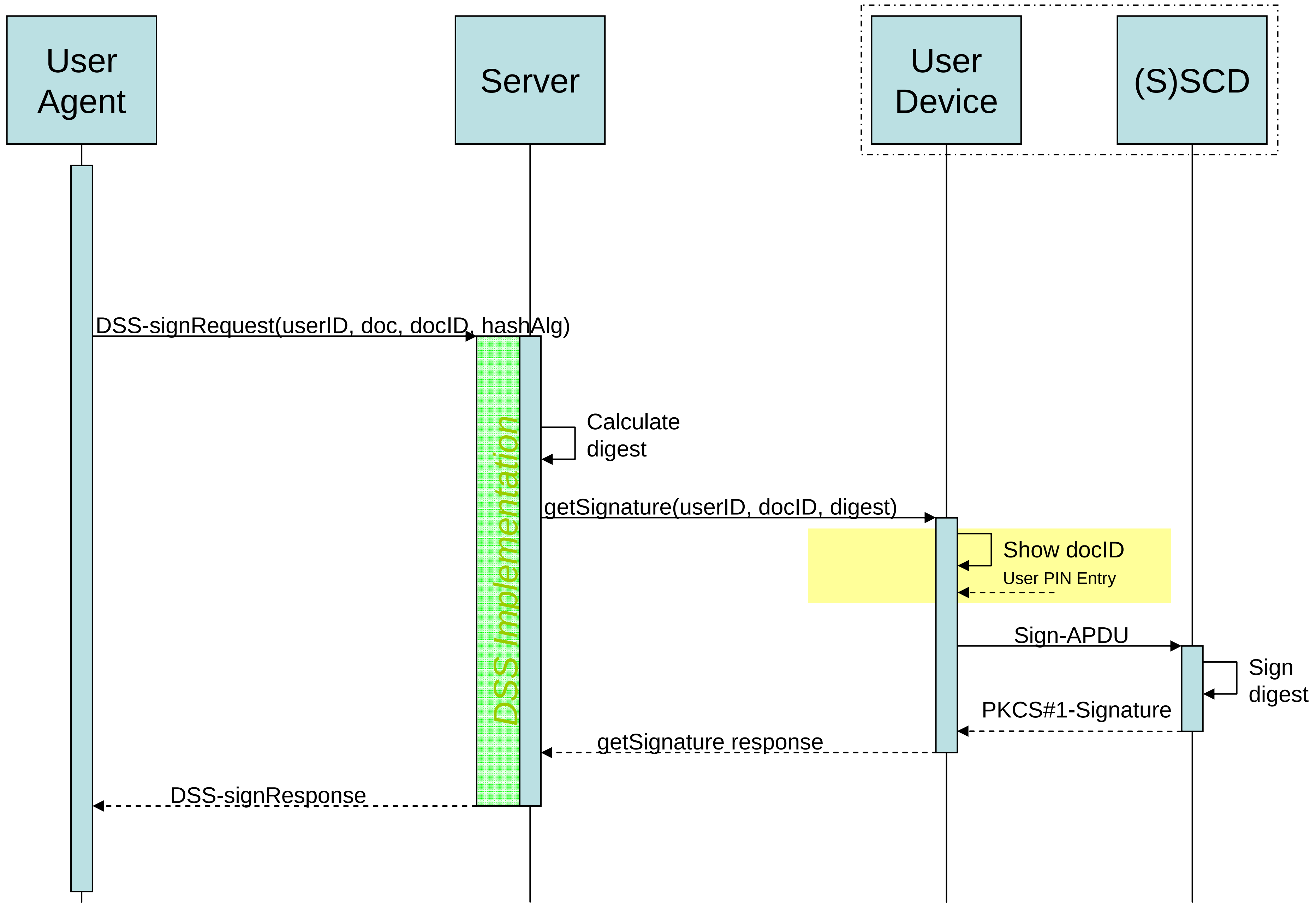
Sequence Diagram Use Case 2a – Smart User-Device (document transfer)



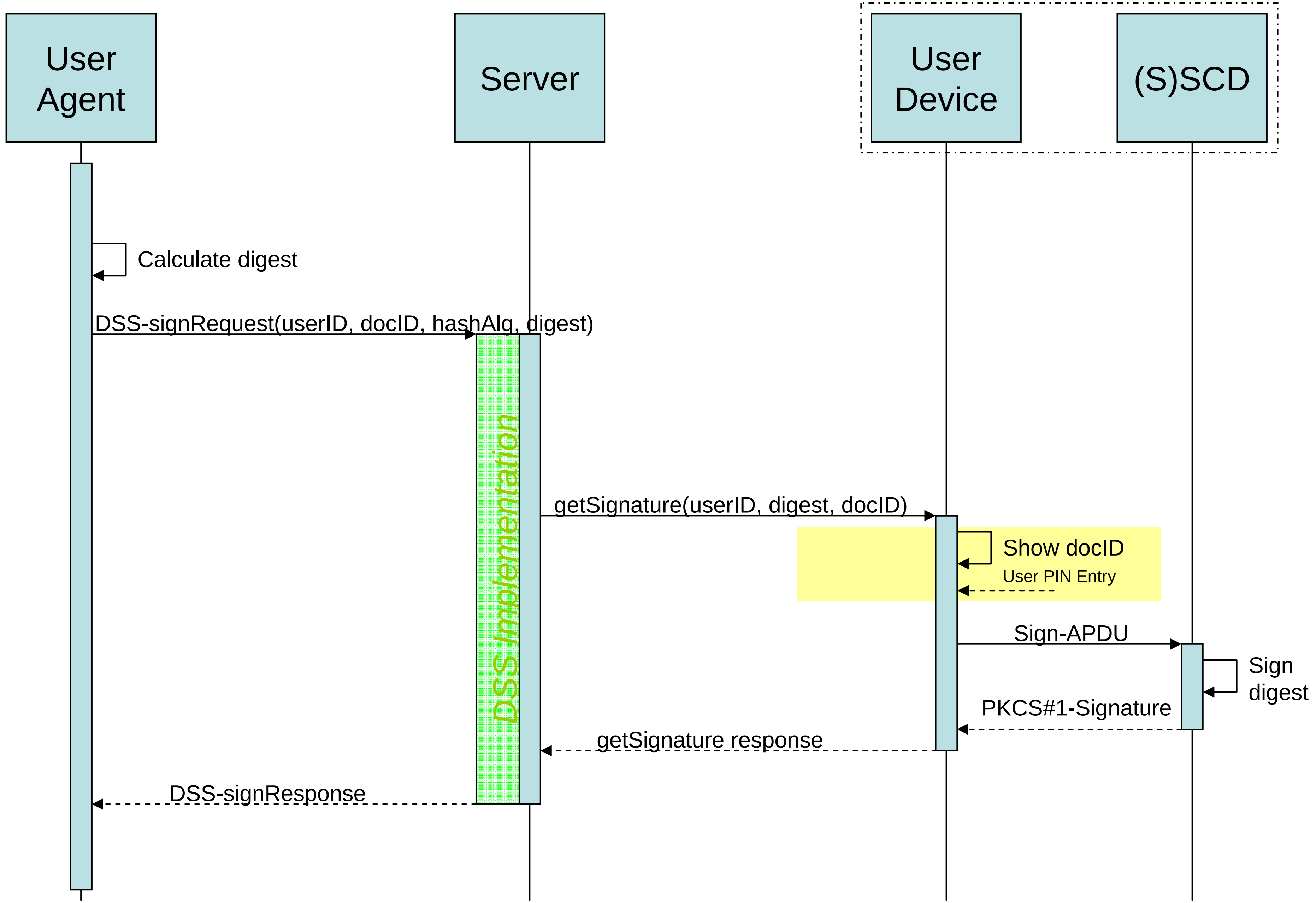
Sequence Diagram Use Case 2b – Smart User-Device (no document transfer)



Sequence Diagram Use Case 2c – Simple User-Device (document transfer)



Sequence Diagram Use Case 2d – Simple User-Device (no document transfer)


User Agent

Server

User Device

(S)SCD

Calculate digest

DSS-signRequest(userID, docID, hashAlg, digest)

DSS Implementation

getSignature(userID, digest, docID)

Show docID

User PIN Entry

Sign-APDU

Sign digest

PKCS#1-Signature

getSignature response

DSS-signResponse

Transport Bindings

- A transport binding is ‘orthogonal’ to the actual DSS protocol.

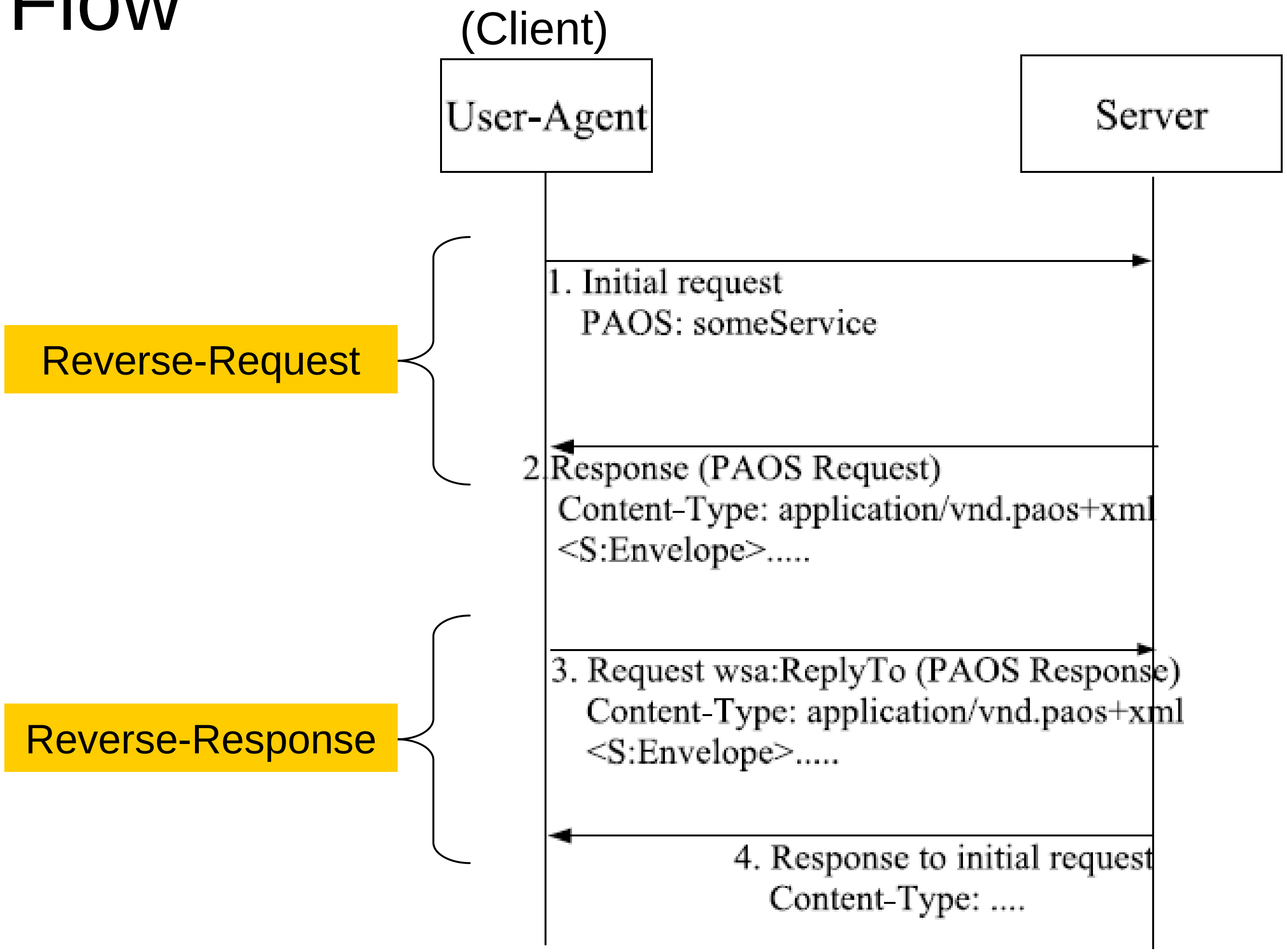
- Point of attention:
  - Handling a request/response from the server to the client.

- Possible Transport Bindings:
  - PAOS, reverse SOAP. Two separate HTTP Req/Res (from client to server) encapsulate a single Req/Resp from the server to the client.
  - AS4 / ebMS v3, using the PULL-mode.
  - REST

- Next slides use the Use Case sequence diagrams, addressing the transport binding.

PAOS

- Basic Flow



(Client)

User-Agent

Server

Reverse-Request

1. Initial request
PAOS: someService

2.Response (PAOS Request)
Content-Type: application/vnd.paos+xml
<S:Envelope>.....

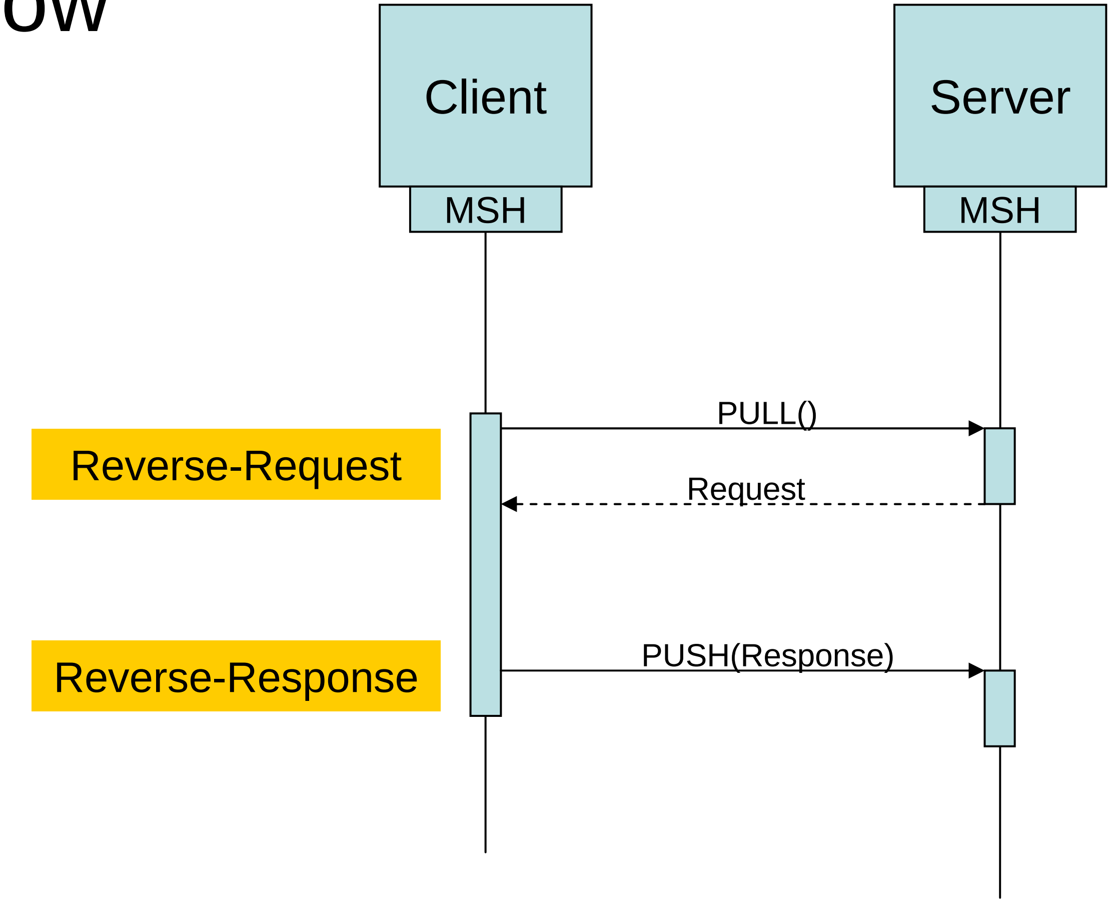
Reverse-Response

3. Request wsa:ReplyTo (PAOS Response)
Content-Type: application/vnd.paos+xml
<S:Envelope>.....

4. Response to initial request
Content-Type: ....

ebMS “PULL”

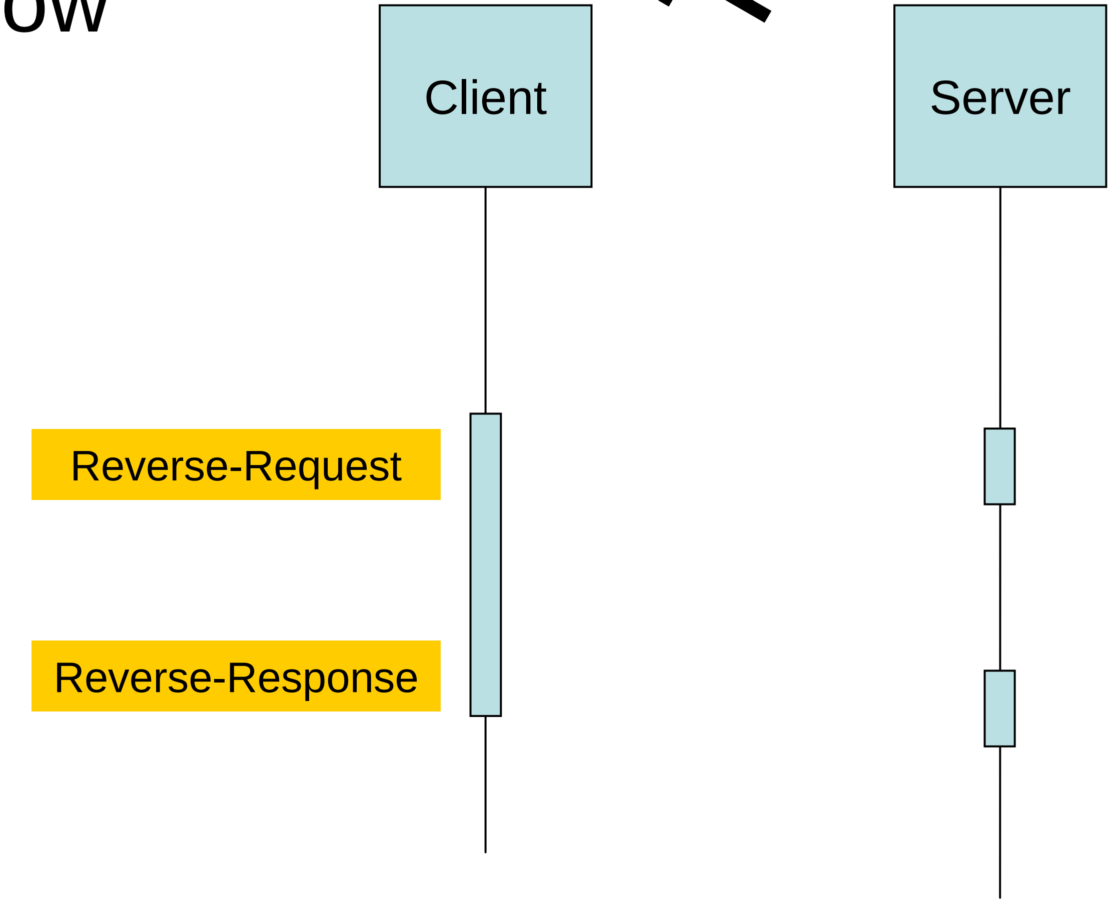
- Basic Flow



REST

TO DO

- Basic Flow

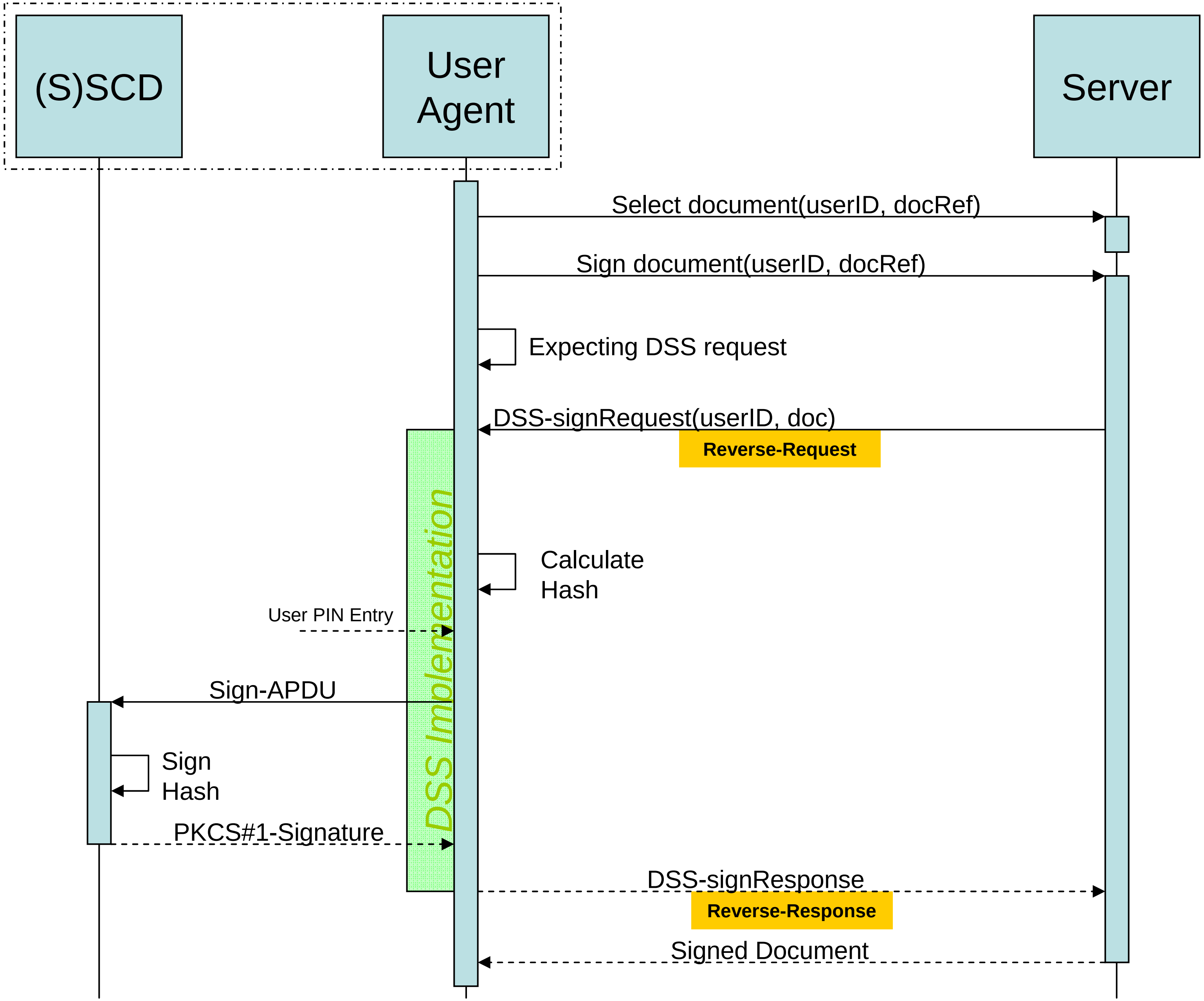


TO DO

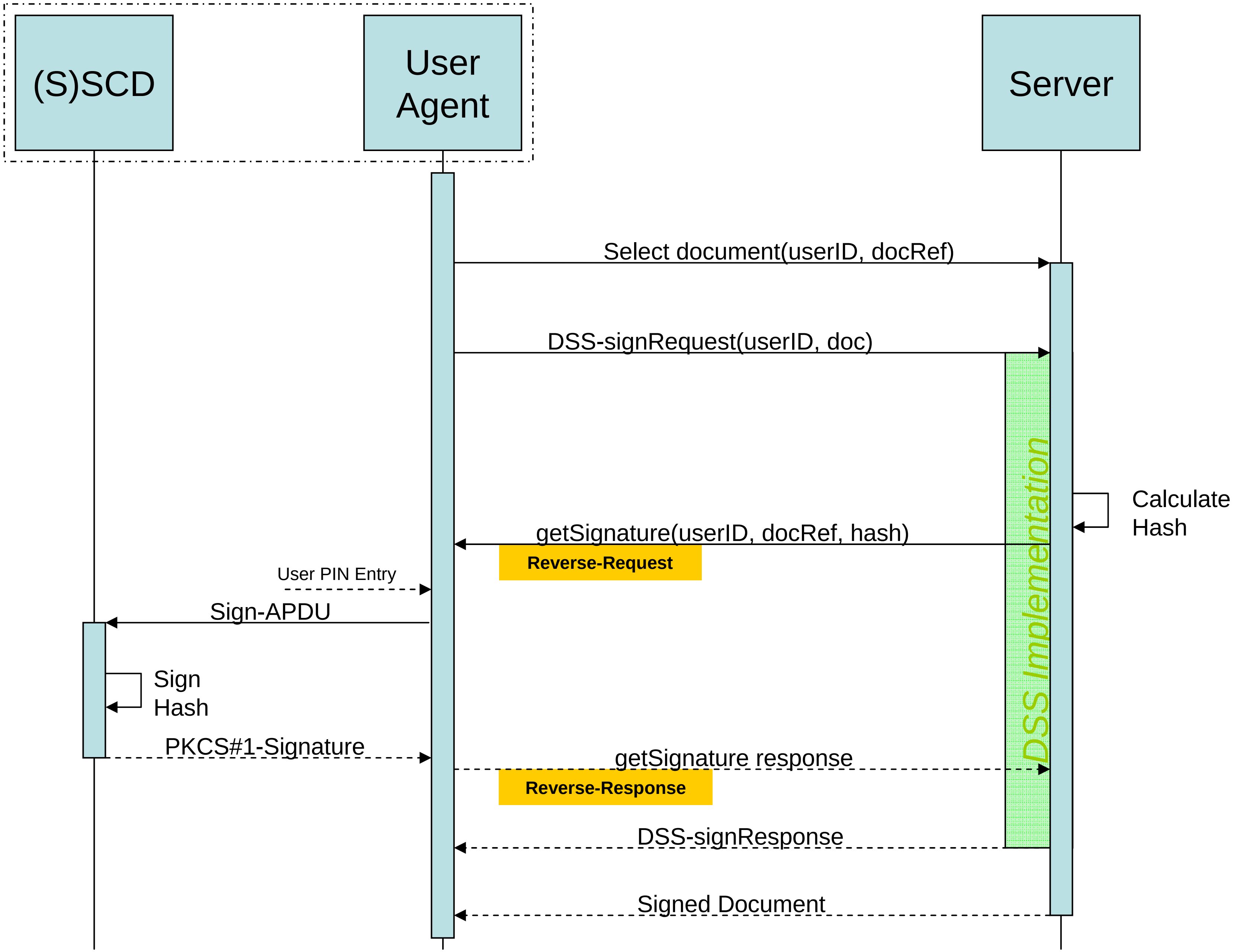
Use of PAOS / ebMSv3

- Both PAOS and ebMSv3 enable the use of reverse req/resp between client and server.
- The next slides indicate the location of these ‘reverse’ request/response (being PAOS or ebMSv3).

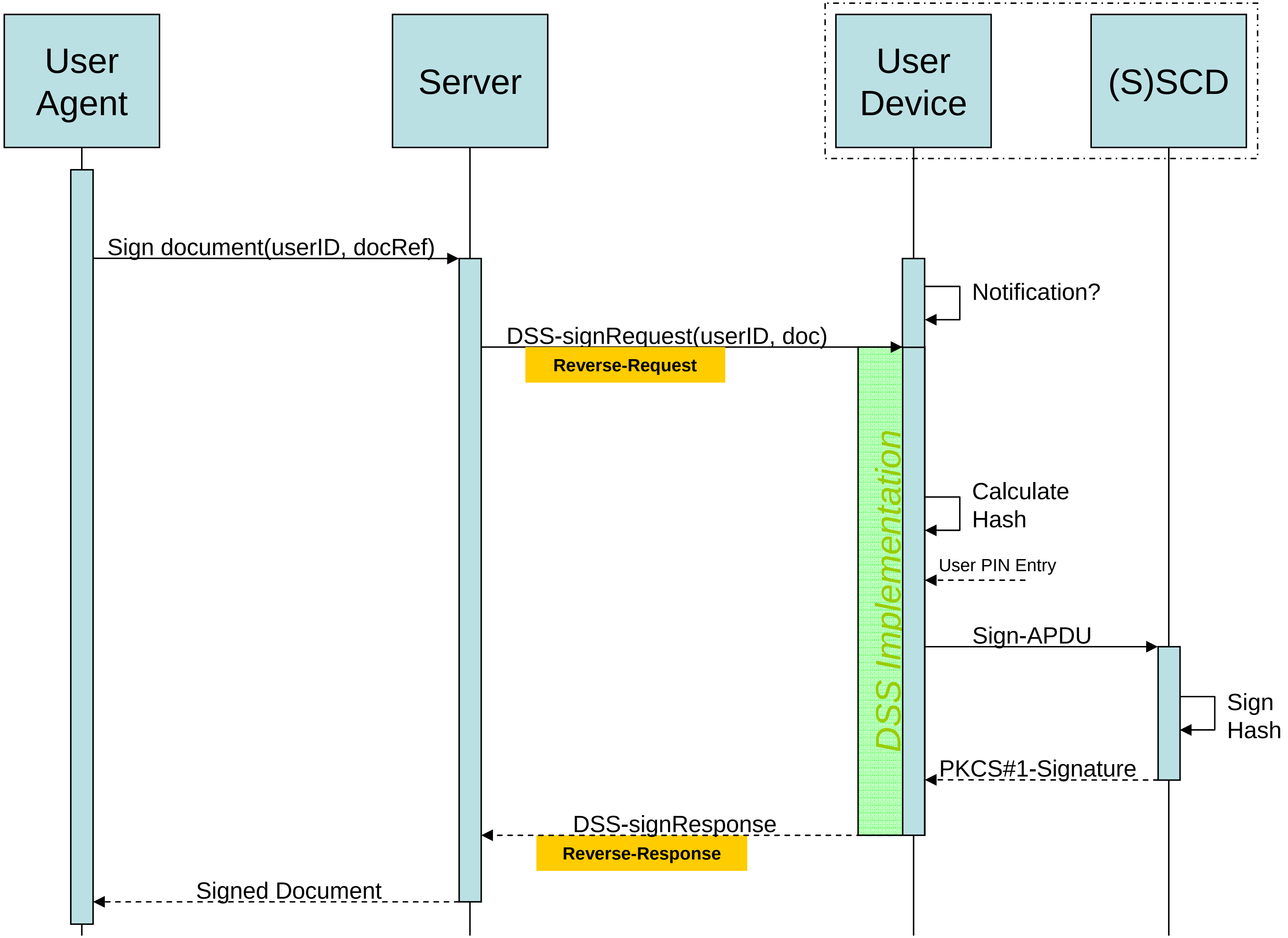
Sequence Diagram Use Case 1a – Smart User-Agent



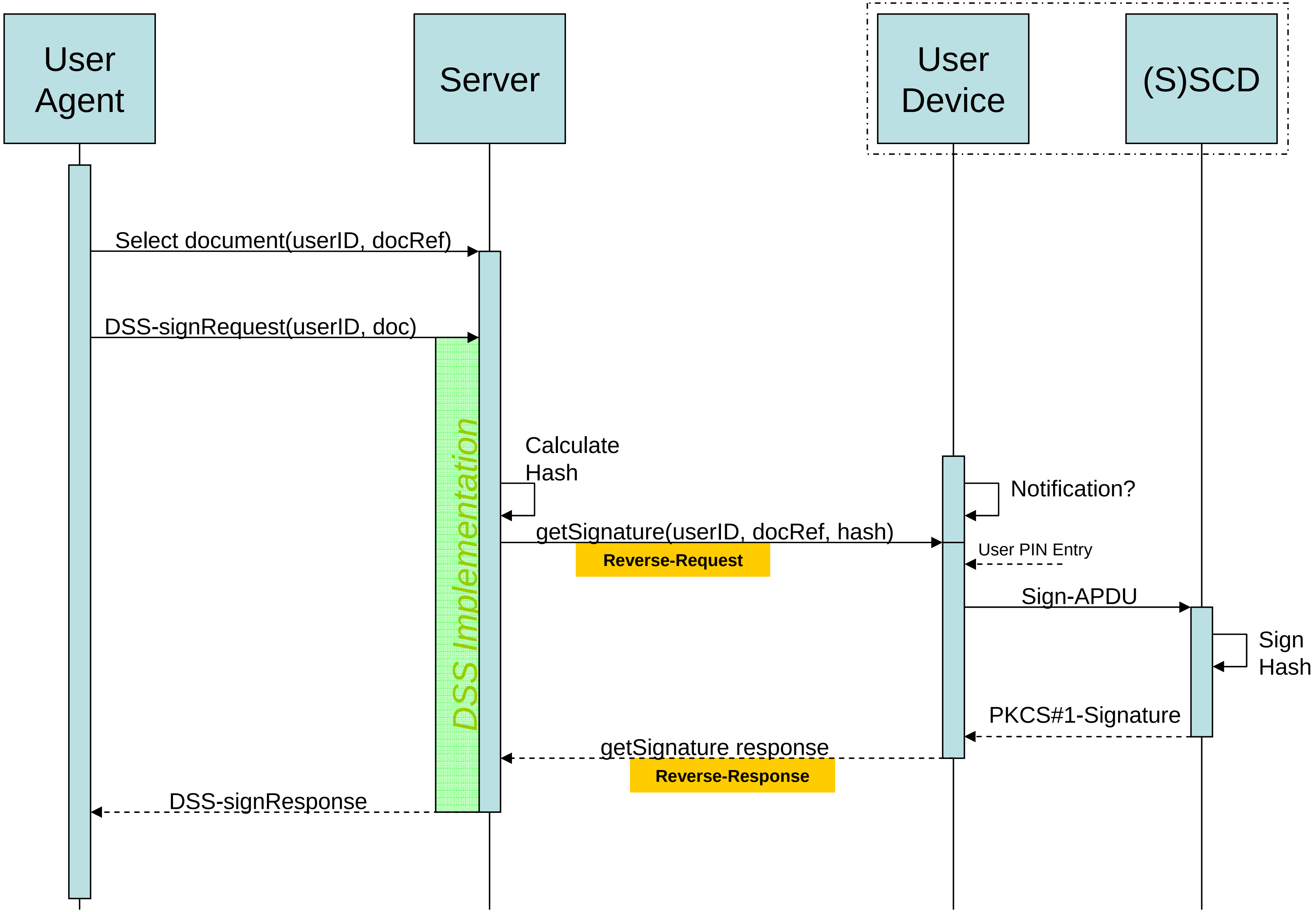
Sequence Diagram Use Case 1b – Simple User-Agent



Sequence Diagram Use Case 2a – Smart User-Device



Sequence Diagram Use Case 2b – Simple User-Device



Finding 1

- In case of a full DSS implementation at the client-side (user agent or user device) a reverse DSS request/response binding is required in case the signature creation is initiated from the server-side.
  - Bindings: PAOS, ebMSv3, REST.
  - Use Cases 1a, 2a, 2b.

➔ Should the reverse binding become part of the DSS profiles?

  - PAOS: Yes/No
  - ebMSv3: Yes/No
  - REST: Yes/No

Note:

If the whole process is initiated (from the client-side) by means of a *blocking* http req/resp, the client must be able to handle the reverse req/resp in parallel.

Finding 2

- In case of a full DSS implementation at the server-side a reverse request/response binding is required for the signature creation request to the User-Agent or User-Device.
  - The signature creation request is not a DSS request; see the ‘getSignature’ in the use cases.
  - The reverse binding is not part of the DSS req/respbinding; it is used by the DSS implementation.
    - ***Does DSS know where to get the signature from?***
  - Bindings: PAOS, ebMSv3, REST.
  - Use Cases 1b, 1c, 2c, 2d.

  Therefore, can be left ‘out of scope’ regarding the DSS protocol. But there is a need to specify how to get the signature.

➔ Should the DSS sign request be extended to specify a location for the signature creation device?

- Yes/No

Finding 3

- The Use Cases specify a number of arguments, not yet part of the DSS sign request (such as hashAlg and digest).

➔ Should the following parameters be added to the DSS core as part of the sign request (response)?

- hashAlg
  → Yes/No
- digest
  → Yes/No
- signatureValue
  → Request: Yes/No; Response: Yes/No
- docID
  →Yes/No
- docRef
  →Yes/No

Finding 4

- The Use Cases show the use of the ‘getSignature’ functionality. This can be any proprietary or already standardized protocol, such as:
  - ISO/IEC 24727 / CEN 15480 (based on DSS!)
  - ISO/IEC 7816

➔ Should the DSS (core) be extended to standardize the ‘getSignature’ functionality?
  - Yes/No

Note:

If the DSS req/resp is extended especially with the signatureValue in the response, it will standardise the ‘getSignature’ functionality...

Finding 5

- The Use Cases show the use of the ‘getSignature’ functionality. This can be any proprietary or already standardized protocol, such as:
  - ISO/IEC 24727 / CEN 15480 (based on DSS!)
  - ISO/IEC 7816

➔ Should the DSS (core) be extended to standardize the ‘getSignature’ functionality?

- Yes/No

Note:

If the DSS req/resp is extended especially with the signatureValue in the response, it will standardise the ‘getSignature’ functionality...