Verified.Me Data Asset Client(DAC) & DC Integration

High Level Design

Version: 0.5
Date: 2021/09/24

Created by **Russ Profant**
Digital & Contact Center Technology (DCCT)

Revision history

Version	Revision Date	Summary of Changes	Updated By
0.1	Sep 24, 2021	Initial Draft	Russ Profant
0.2	Dec 22, 2021	Added diagrams	Russ Profant
0.3	Mar 22, 2022	Added Batch Estimate	Russ Profant
0.4	Apr 25, 2022	Added DC integration diagram	Russ Profant
0.5	May 9, 2022	Updated initialization flow	Russ Profant

Project Overview and Scope

This document describes high-level design (HLD) for the implementation of a third-party (SecureKey) data service called Verified.Me or V.Me

V.me will be used as an additional data provider to confirm the identity of new customers for on-line banking for AML (AntiMoneyLaundering) purposes.

Three systems are in scope for this project: Compass, eBanking, ECIF

The scope in eBanking covers these current applications CCFA, DFA. Others may be added later.

Additionally, eBanking will create a service that will act as an internal proxy for other internal CIBC clients interested in consuming the V.Me service.

Contributors:

Resource	Role
Russ Profant	eBanking App Consultant

Statement of Business Problem

4

- When new clients apply for a credit card or a deposit account

  they must undergo AML check as mandated by the federal government and if they don't pass their application cannot be fulfilled on-line but they must come to a branch

  V.me is much easier to use for a new client than DIV because it requires the client only to login to their current FI (with browsers typically prefilling the login info) and confirm the login through a one-time code verification.

  The current AML options in eBanking

AML Method	ID Verification	Method	Data source	AML compliance
DIV	Face-to-face	single	2 (pic+document)	full
TransUnion AML	Non-face-to-face	dual	2 credit sources	full
TransUnion AML	Non-face-to-face	dual	1 credit source	partial

- There are other LOBs in the firm that may want to utilize this service for their own needs with minimum time and effort as an internal REST service.

Proposed Business Solution

5

- Offer an easier and faster alternative to new clients for AML verification to the current DIV

  This will be an additional method to compliment the AML methods listed previously and will also act as an alternative to the DIV method. DIV method of client identification is somewhat tedious, and clients sometimes abandon it in the middle of the process.

AML Method	ID Verification	Method	Data source	AML compliance
DIV	Face-to-face	single	2 (pic, document)	full
TransUnion AML	Non-face-to-face	dual	2 credit sources	full
TransUnion AML	Non-face-to-face	dual	1 credit source	partial
VME	Non-face-to-face	dual	1 banking source	partial
TU AML + VME	Non-face-to-face	dual	2 banking sources	full

- eBanking will offer simplified REST API to Verified.Me to internal clients as a shared service

Logical & Technical Solution Overview

PI Data	Provide an entry point for V.me login in DFA & CCFA and collect the data from V.Me
AML	Match the data to the user data and send it to Compass for AML use
Shared Service	Provide V.Me client proxy to internal users as a REST API

VME Proxy for Verified.Me	Create REST API wrapper service for internal LOBs to access Verified.Me 'Account Profile Service'
Cart Integration of VME	Integrate the wrapper service into DC flows based on business requirements

Project Assumptions and Dependencies

	Assumptions
1	Verified.Me is available in the public cloud as an API service
2	The whole transaction between the client and V.Me service is carried out in a modified OpenID Connect authentication and authorization flow
3	Compass directs the AML processing and decides which methods to offer to clients to identify themselves via eBanking
4	ECIF will create a new verification method that will combine TU AML single source with VME single source into “dual source” AML ID verification
5	EBM-TSS will be used initially as the security mechanism for the internal VME shared service

	Dependencies
1	eBanking UI web and mobile apps
2	Content team for screens and messages
3	Verified.Me service by SecureKey
4	FI (Financial Institutions) file in EFT Hub
5	Key pair availability from TEM and AO teams

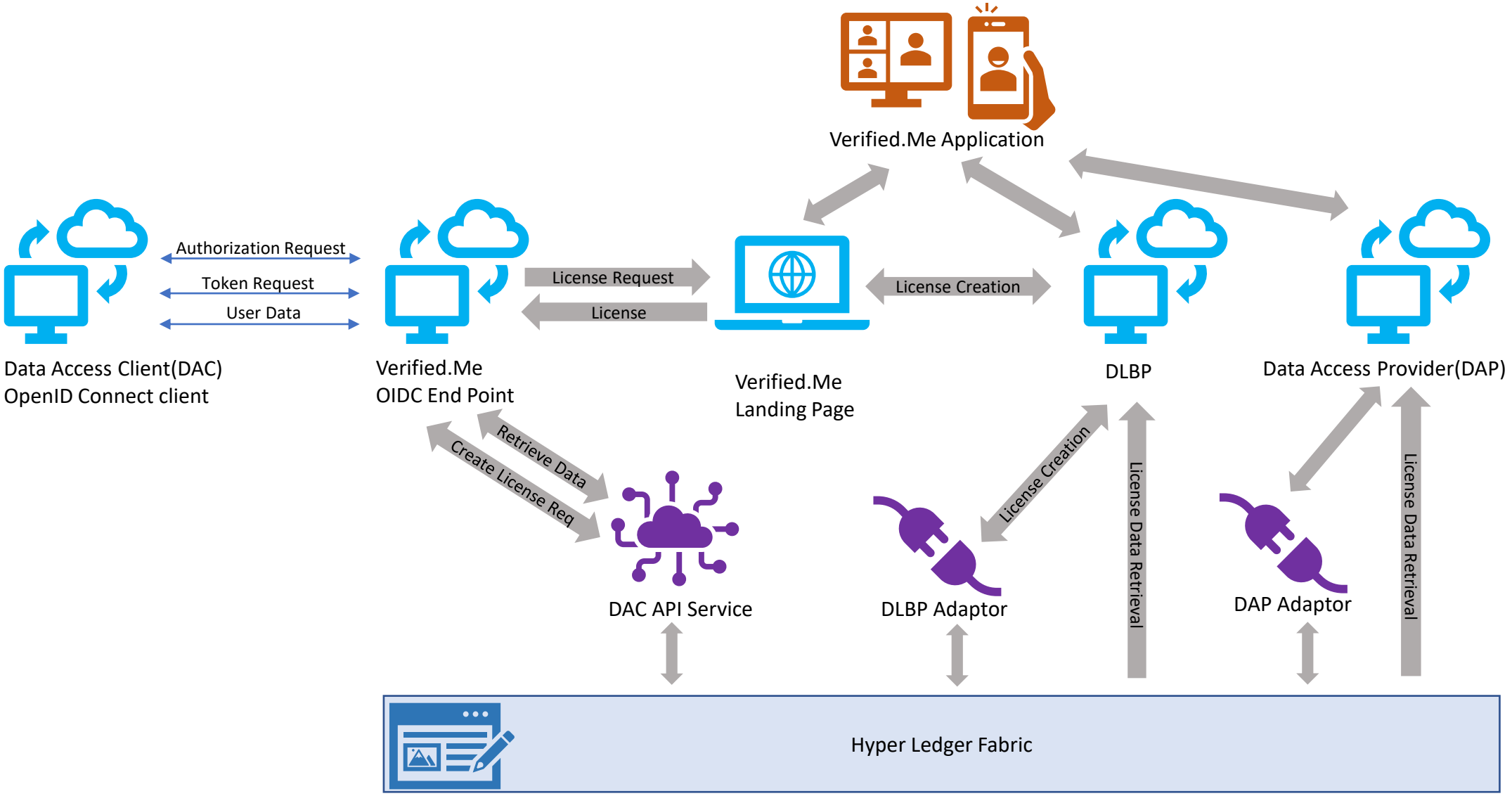
PI Data & Proxy Service - Impacted eBanking Components

Name	High Level Changes	Effort Size
EBM-VME	**Module**	6 sprints DEV 3 sprints DEV QA (setup/support) 6 sprints App Consult
	• Create a new module for this service called “ebm-vme” • Transfer the service security from DIV module which uses EBM-TSS – API key validation	
	API	
	• Create new API for the module - follow closely the ebm-div API model but without DIV data models	
	Workflow	
	• Create API handlers that will perform all the work • Create matching functionality for user data vs V.Me data	
DC UI	• Update AML catalog pages in all in-scope applications with V.Me service option	
Resources	• Update messages and content as necessary for this project	
EBM-TSS	• Register VME module as well as eBanking as a client of VME module	0.5 sprint

AML - Impacted eBanking Components

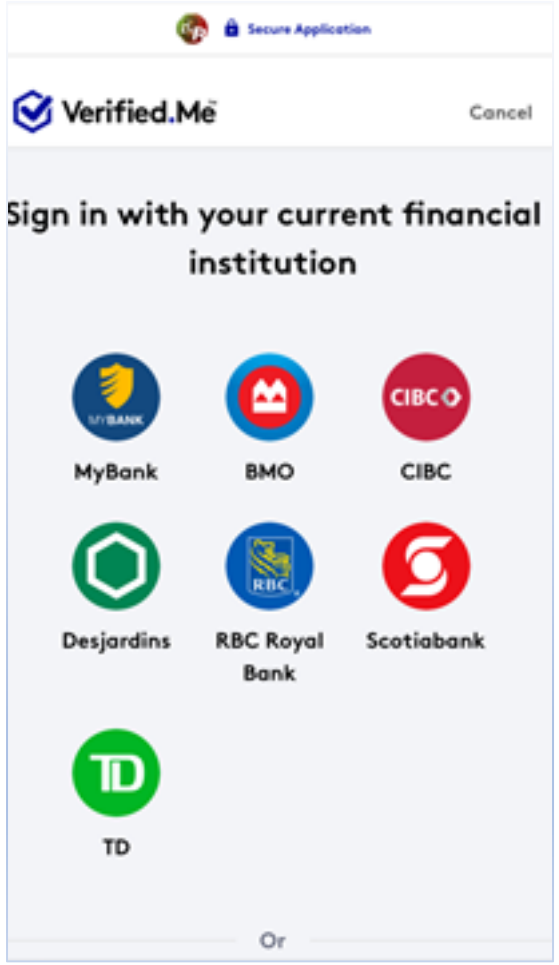
Name	High Level Changes	Effort Size
EBM-DC	**Compass API** • Create new Compass API service to send V.Me data to Compass	2 sprints DEV 1 sprints DEV QA 2 sprints AC
	• Update APIs to include “V.Me” option • Create the service in the module to offer this functionality to all in-scope applications • Update all in-scope applications to perform the V.Me data collection if chosen by the user	3 sprints DEV per app (CCFA, DFA) 1 sprint DEV QA per app
	Batch	2 sprints DEV 1 sprint DEV QA
	• Process FF file from EFT and load it to Gemfire	

E2E Verified.Me Process Diagram

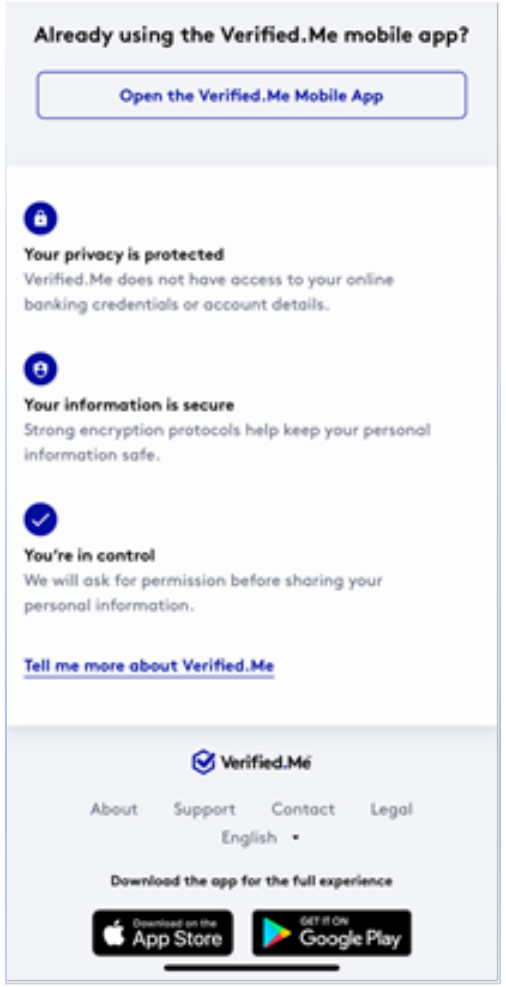


Verified.Me Screen Flow (mobile version)

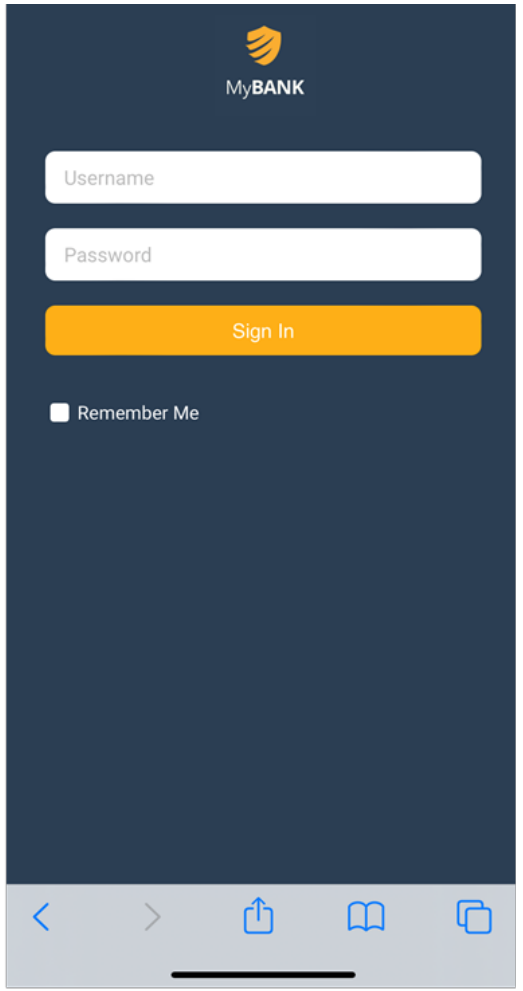
1 Select Financial Institution



2 Select App (mobile only)

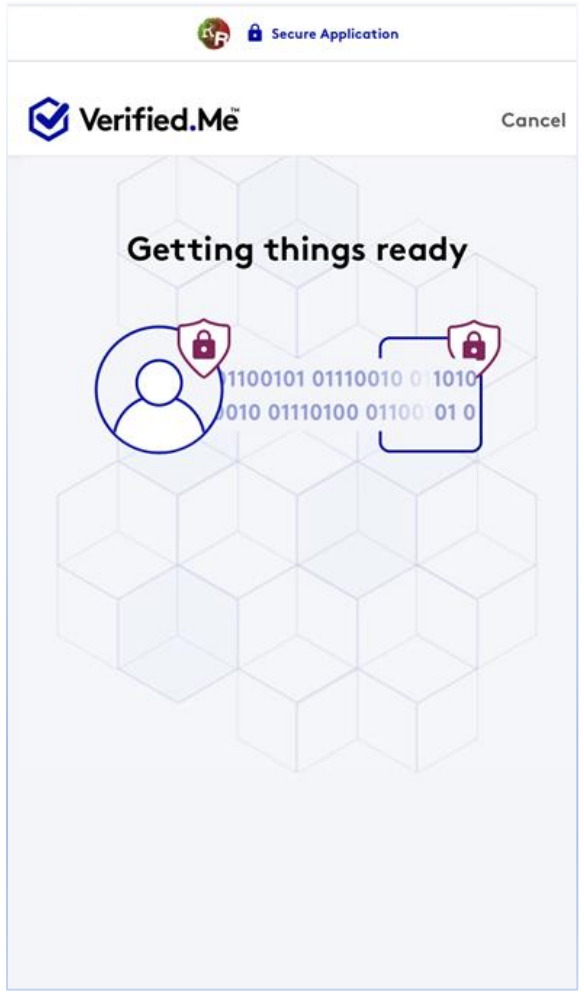


3 Login to FI

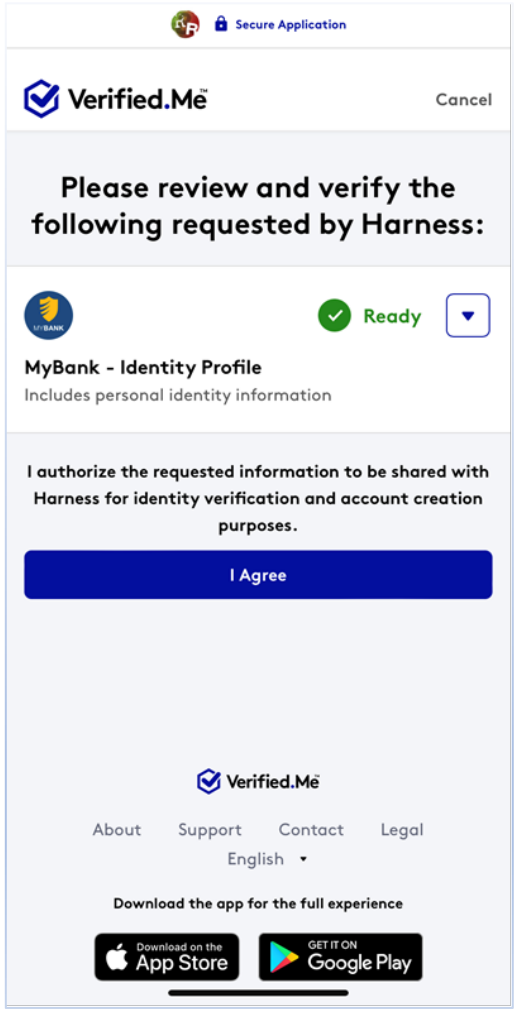


Verified.Me Screen Flow cont. (mobile version)

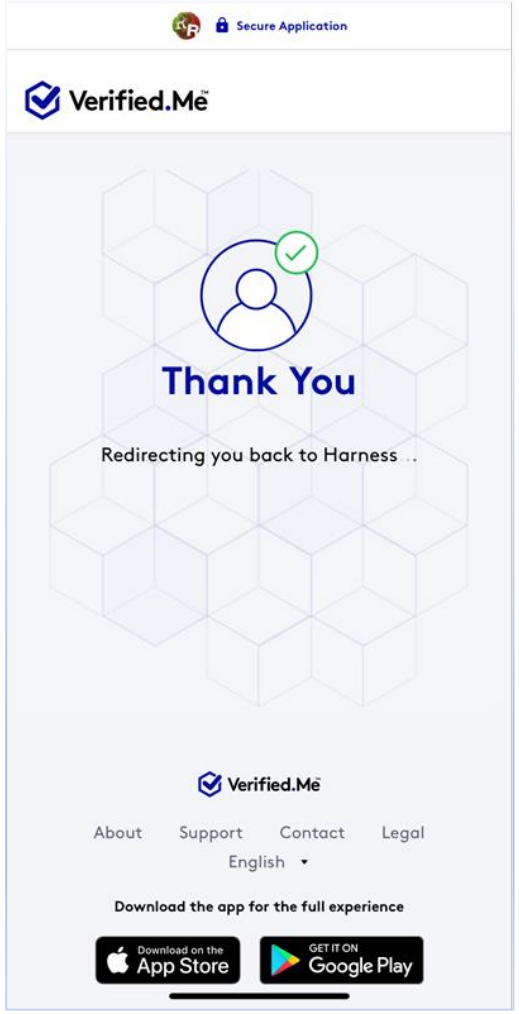
4 User Data Retrieval



5 User Data Authorization



6 End



EBM-VME Design Outline - Approach

DIV code re-use

- Both services work based on OIDC authorization code grant
- EBM-VME will reuse EBM-DIV design including all the code but in a new module
- Changes will be made only where it is necessary

Main technical differences between DIV and VME

- The fundamental difference is the control of the flow:
    1. In DIV there is an eBanking DIV landing page. As a result of this the code in the page can control the DIV flow between client eBanking DIV and SecureKey DIV; this is UI-controlled flow.
    2. In VME there is no page, therefore the flow must be controlled by eBanking VME module; this is BackEnd-controlled flow.
- Same vendor but different end points (APIs) – configuration change
- Different vendor license and security details – configuration change
- Different data packages
- No need to create a job in Verified.Me as is the case in DIV
- The main functional difference is that there is no processing done by the vendor and so data is immediately available as soon as the user ends their vendor flow. This means there is no polling required in DC to get the vendor output – different data retrieval implementation

The VME E2E service will have 4 steps

1. Client system makes a request to EBM-VME to use the service. This must be done from the back end.
2. Client's browser is redirected to the Verified.Me service. The client performs authorization and the actual login to another FI which makes the data available to the vendor. Vendor redirects client browser back to eBanking.
3. EBM-VME requests the data from the vendor and redirects the client browser back to its application.
4. Client system requests the data from EBM-VME.

EBM-VME REST API – Create Request in eBanking VME

/v1/json/vme/workflow

POST API endpoint to create new workflow for VME process. This is a public API, to be called by calling applications.

Request Data Model

```
{"applicantInfo": {
        "firstName": "Michael",
        "lastName": "McGee",
        "middleName": "George",
        "dateOfBirth": "string",
        "address": {
                "streetAddress": "20 Dundas St. West",
                "locality": "Toronto",
                "region": "ON",
                "postalCode": "L5M 6Z2",
                "country": "CA”
        },
        "phoneNumber": "+14556789000",
        "email": "michael@gmail.com"
},
"locale": "en-CA, fr-CA",
"callingAppParameter": {
        “callingAppReturnUrl”:”https://dc/vme/callback”
        “dataMatch”:
}
```

Response Data Model

```
{
        ”redirectUrl”:https://sk.vme/authorization_request_end_point,
        “worklowId”:”12345”
}
```

EBM-VME REST API – Get Request Status From eBanking VME

/v1/json/vme/workflows/{workflowId}/status

GET API endpoint to get workflow status for VME process. This is public API, that can be called by calling applications. This is a client helper or convenience API as it will not be used during the flow.

Request Data Model

{”workflowId":”123-456-789
}

Response Data Model

{“workflow”: {
”workflowId":”123-456-789”,
“status”:”SUCCESS,FAILURE,CANCEL,IN_PROGRESS”,
“matchStatus”:”PASS,FAIL”,
”startDate:””,
“endDate”:””
“durationInSec”:””
}
}

EBM-VME REST API – Get Request Result From eBanking VME

/v1/json/vme/workflows/{workflowId}/result

GET API endpoint to get workflow results: claims and decision. This is public API, to be called by Calling Applications.

Request Data Model

{”workflowId":”123-456-789”
}

Response Data Model

{”vmeData": {
”givenName": "Michael",
"familyName": "McGee",
"middleName": "George",
”title": ”prefix-to-name”,
”honorific”:”suffix-to-name”,
"dateOfBirth": ””,
"address": {
"streetAddress": "20 Yonge St.",
"locality": "Toronto",
"region": "ON",
"postalCode": "L5M 6Z2",
"country": "CA”
},
”phoneNumber": "+14556789000”,
”email": "michael@gmail.com” ,
”customerRefNum”:””,
”verificationDate”:””
},

“account”:{
“type”:”deposit,credit card,loan”,
“number”:”1234567890”,
“institution”:”01”,
“active”:”yes”,
}},
”matchResult”:{
”status”:”PASS,FAIL”,
“firstName”:”PASS,FAIL”,
”lastName”:”PASS,FAIL”,
“dateOfBirth”:”PASS,FAIL”,
”active”:”PASS,FAIL”
},
“workflow”:{
”workflowId":”123-456-789”,
“status”:”SUCCESS,FAILURE,CANCEL,IN_PROGRESS”,
“matchStatus”:”PASS,FAIL”,
”startDate:””,
“endDate”:””
“durationInSec”:””
}}

EBM-VME REST API – Allow Vendor To Send Auth Code to eBanking

/v1/json/vme/vendorCallback

GET API endpoint to allow the vendor to return the auth code to eBanking. This is public API to be called by the V.Me vendor.

Request Data Model

On Success

```
{
        ”code":”authorizationCode”,
        “state”:”workflowId”
}
```

On Failure

```
{
        “error”:”access_denied”
        “error_description”:””
}
```

Vendor uses standard OpenID defined error codes and error handling. In addition it has its own error code defined above for cases not covered by OpenID error handling

EBM-VME REST API – Allow Vendor To Get The Public Signing Key

/v1/json/vme/.well-known/jwks.json

GET API endpoint to allow the vendor to get the public key to verify the signature of the request.

The API will return the public key for signature. The key will be in the keystore file so that the key can be easily rotated.

File format:

```
{ "keys": [ {
        "alg": "RS256",
        "kty": "RSA",
        "use": "sig",
        "x5c": [ “public_key" ],
        "kid": "NjVBRjY5MDlCMUIwNzU4RTA2QzZFMDQ4QzQ2MDAyQjVDNjk1RTM2Qg"
} ]}
```

For field descriptions see https://auth0.com/docs/secure/tokens/json-web-tokens/json-web-key-set-properties

EBM-VME Verified.Me Integration - Authorization Request

Authorization request redirect link format

In order to authenticate the DAC, a signed request object is used. The authorization request is encoded as a compact serialized JWS object signed with the DAC's signing key. The payload of the JWS contains a JSON object with the fields being the OpenID Connect parameters below.

Request object structure

- ✓ **redirect_uri** - The URL that the authorization endpoint should redirect the response back to. This URL must be registered with Verified.Me to prevent returning authorization responses to unknown clients.
- ✓ **client_id** (required) : A client_id string provided to the DAC during provisioning.
- ✓ **scope** (required) : A space delimited list of scope values that identify the data that the DAC wants. The scope must include the OpenID scope as required by OpenID Connect as well as one or more of the Verified.Me defined scope values, such as Account Profile.
- ✓ **response_type** (required) : This must be set to the string value code.
- ✓ **state** (required) : **Use workflowId**
- ✓ **ui_locales** - A space delimited list of locale strings as described in RFC 5646. If provided, this will be used to select the language in the Verified.Me OIDC service.

The full set of parameters in the link to be passed to the Verified.Me OpenID Connect authorization endpoint

- ✓ **request** (required) : the JWS request object described above.
- ✓ **response_type** (required) : This must be set to the string value “code”.
- ✓ **client_id** (required) : A client_id string provided to the DAC during provisioning.
- ✓ **scope** (required) : A space delimited list of scope values that identify the data that the DAC wants. The scope must include the OpenID scope as required by OpenID Connect as well as one or more of the Verified.Me defined scope values, such as Account Profile.

Response

- ✓ **code** (required) : the *authorization code* to use in the token exchange for an *access token.*
- ✓ **state** (required) : the *state* provided in the authorization request.

EBM-VME Verified.Me Integration - Access Token Request

Access Token Request Format

The DAC is then able to exchange the *authorization code* for an *access token* by calling the Verified.Me OpenID Connect Endpoint. The DAC will authenticate using the private_key_jwt method to enable the Verified.Me OpenID Connect Token Endpoint to validate that it is the DAC making the call by verifying the signed token.

The token request is made by POSTing and the following parameters:

- ✓ **grant_type** (required) : The value MUST be set to "authorization_code".
- ✓ **code** (required) : The authorization code returned in the authorization grant response (above).
- ✓ **client_assertion_type** (required) : Must be set to the string urn:ietf:params:oauth:client-assertion-type:jwt-bearer
- ✓ **client_assertion** (required) : - A JWS encoded with compact serialization that is signed with the DAC's signing key. The payload of the token is a JSON object with the following claims:
  - ➢ **iss** (required) : The client_id of the DAC
  - ➢ **sub** (required) : The client_id of the DAC
  - ➢ **aud** (required) : The URL of the Verified.Me OpenID Connect Token Endpoint
  - ➢ **exp** (required) : The expiry time of the token. This time should be set to a small value (e.g. 5 minutes) into the future as the token will only be used to retrieve the access token.
  - ➢ **jti** (required) : A unique identifier for the JWT.
  - ➢ **client_id** (required) : The same client_id used in the authorization grant request.
  - ➢ **redirect_uri** (required) :The same redirect_uri used in the authorization grant request.

Response

- ✓ **access_token** (required) : the access token to be used in subsequent API calls for this session.
- ✓ **token_type** (required) : MUST be set to "bearer".
- ✓ **expires_in** (required) : number of seconds until this token expires. This will at minimum be set to the job.expiry_time .
- ✓ **id_token** (required) : a JWS as described in https://openid.net/specs/openid-connect-core-1_0.html#IDToken. The sub field will be a unique identifier for the user at the calling DAC. That is, different DACs will get a different value for the same user.

EBM-VME Verified.Me Integration - User Data Request

User Data Request

The DAC can now access the collected user data by sending a GET or POST request to the userinfo endpoint in accordance with OpenID Connect Core Section 5.3. The access token must be provided in the Authorization header as a bearer token.

Response - Personal Data

The data CIBC is consuming is called “Account Profile”. It has basic personal client data listed below

Field Name	Details
given_name	End-user's first name
family_name	End-user's "last name," including prefixes
middle_name	End-users "middle names." Includes all middle names, if more than one. Initials are acceptable.
title	End-user's name prefix title (i.e., Mr., Mrs., Dr., Sgt)
honorific	The suffix to the Client's name
birthdate	End user's date of birth (YYYY-MM-DD)
address	The end-user's primary postal address. This is an object with these fields: 1) streetAddress 2) locality 3) region 4) postalCode 5) country
phone_number	Person's primary number, mobile if available, in order of preference: 1) primary number if also marked mobile 2) most recent number marked mobile 3) home number if marked mobile 4) primary number landline or unknown 5) home number landline or unknown 6) user selection
email	Person's email address. In order of preference: 1) primary 2) user selection
customer_ref_num	A unique reference number that links back to the user's original CIF
verification_date	The date at which the FI server is attesting to the accuracy of this data

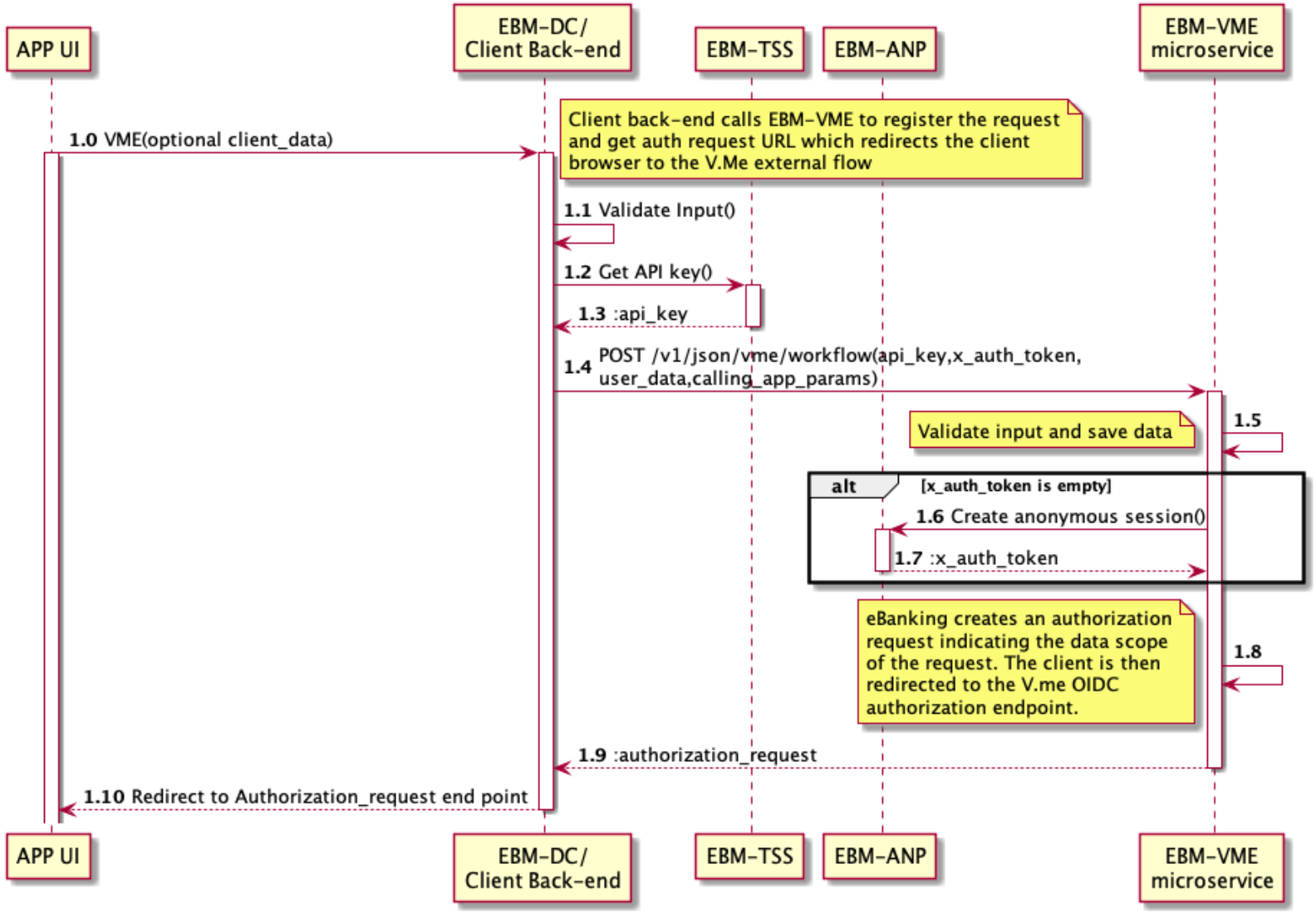
EBM-VME Verified.Me Integration - User Data Request cont.

Response - Account Data

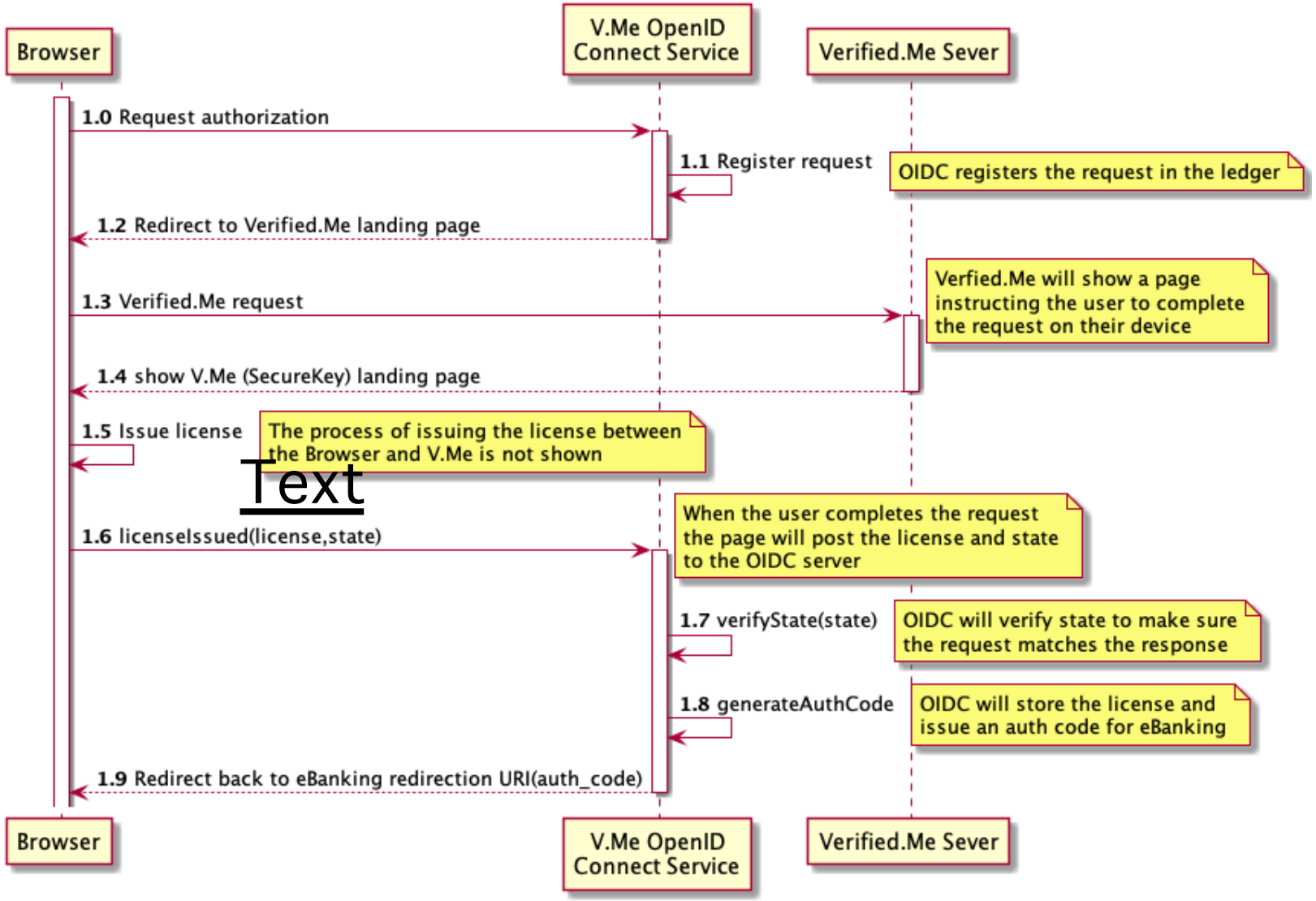
In addition, “Account Profile” offers basic account verification data

Top Field Name	Sub-field Names	Details
Account		An open financial account belonging to the user. The older account is preferred. Order of preference is: • Deposit Account • Credit Card account • Loan Account
	type	the type of account (i.e. deposit, loan, credit card)
	number	the full transit plus account numbers of the selected account. There is a proposal to also pass the transit number if the account field of this bundle. The first five numbers are the transit number, followed by the account number (i.e. "9345334011111222233334444" Where the 93453 is the Transit number and the 34011111222233334444 is the account number.)
	institution	The institution number associated to the FI (i.e. 01)
	active	The status of the account. "True" means the account is active, "False" means it is not, but detailed account status (i.e. reason) is not provided

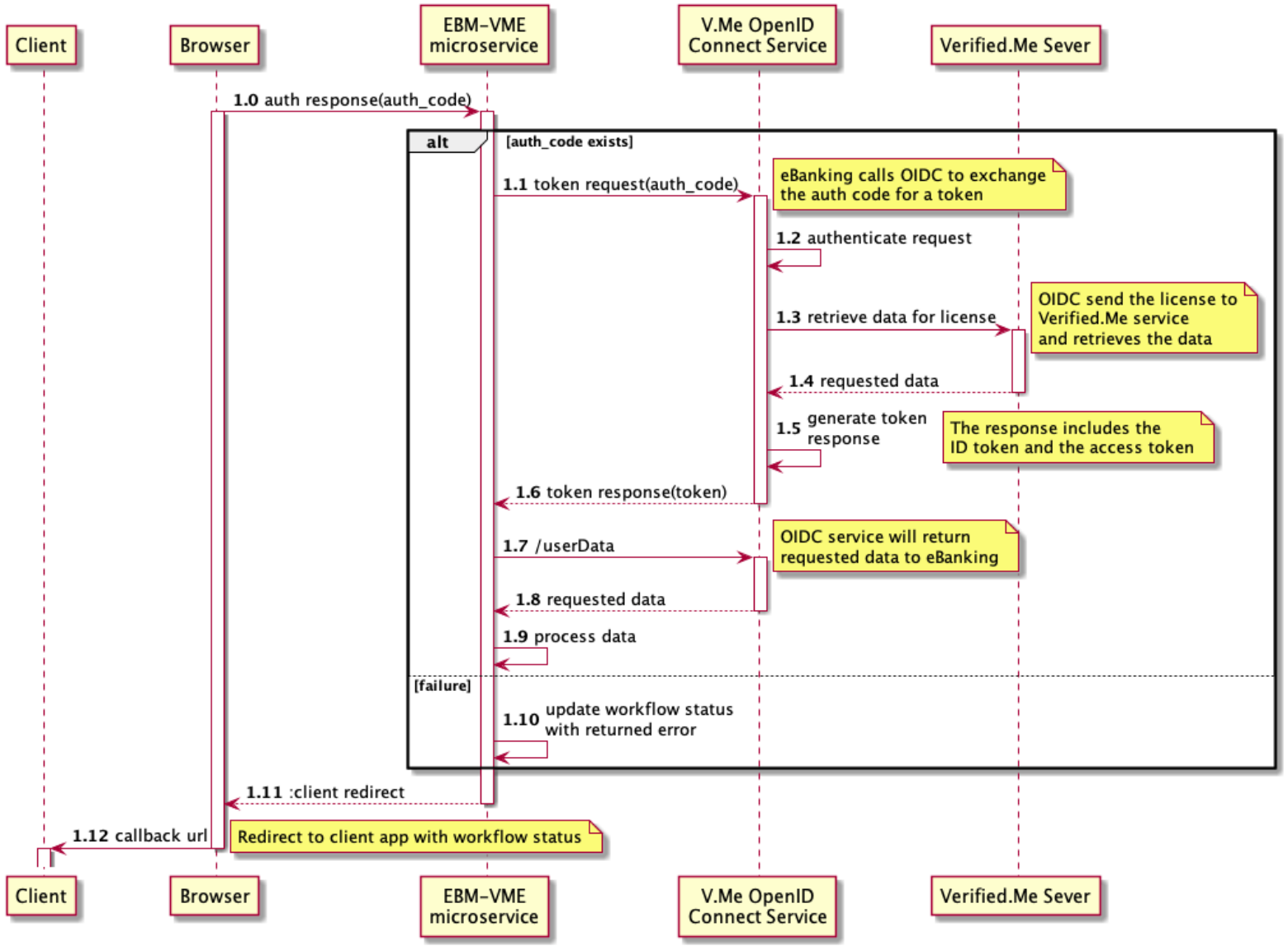
Client Service Request Initialization Using eBanking VME Service



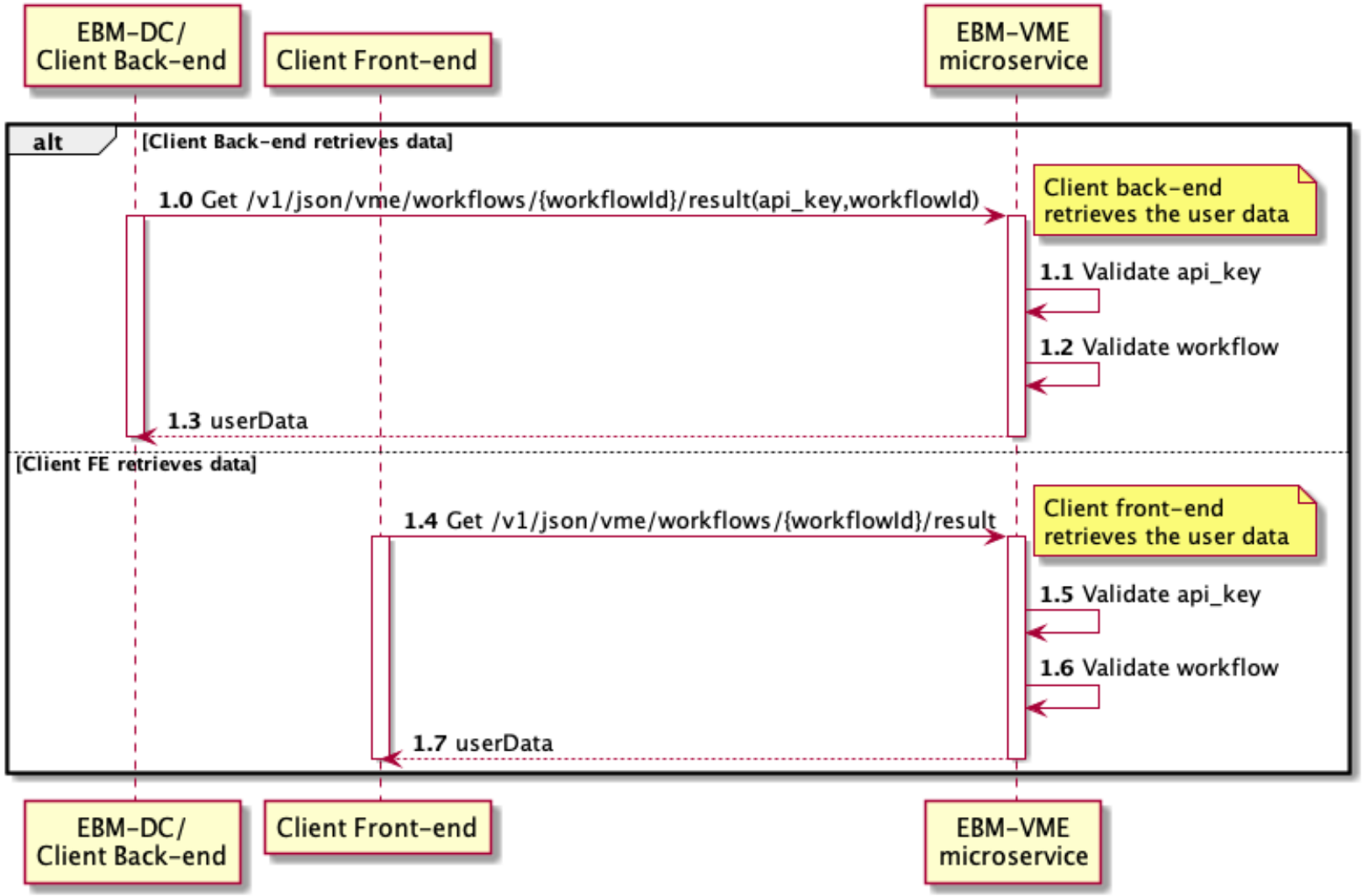
User Authorization & FI Login Through Verified.Me



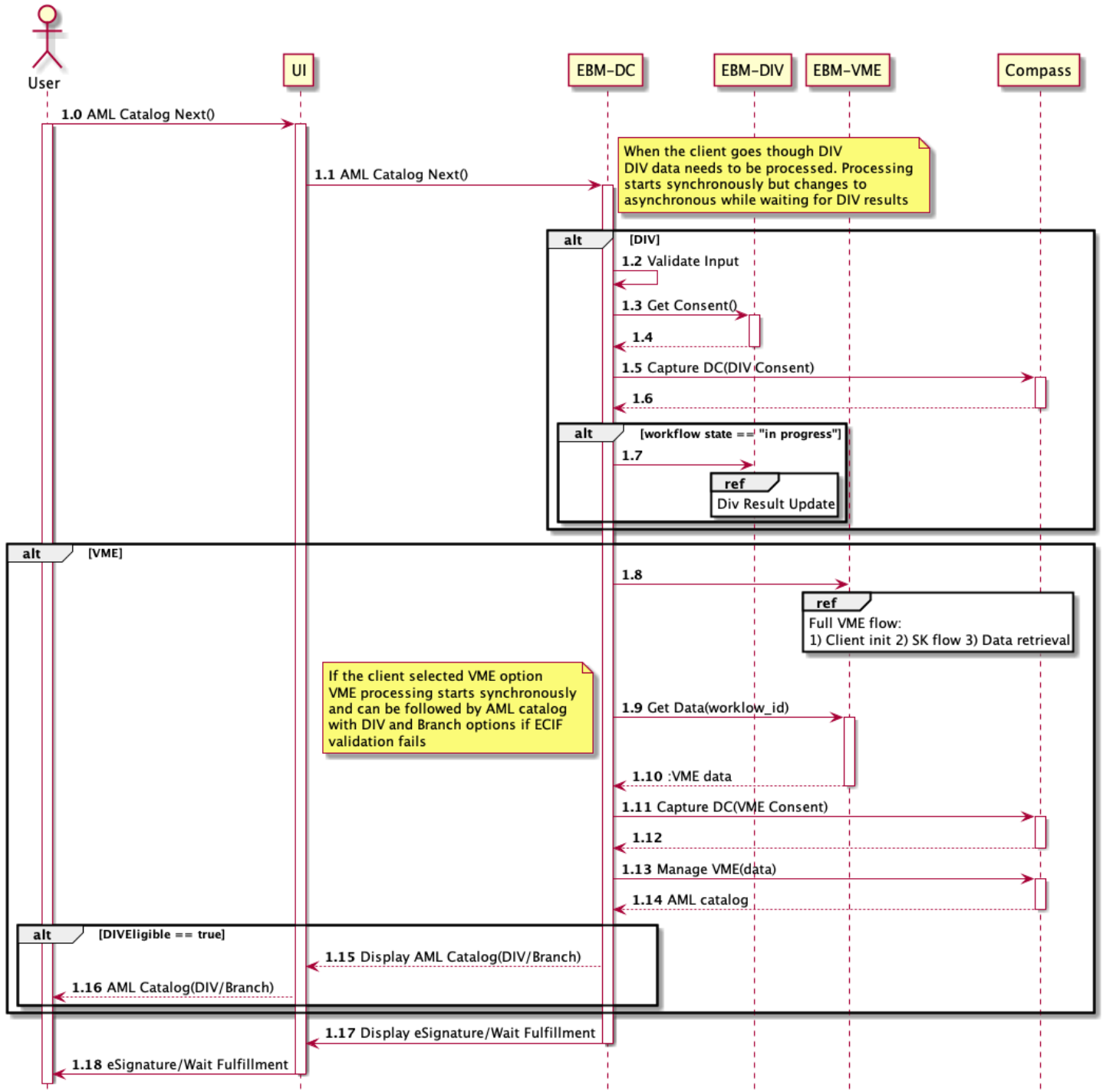
eBanking Data Retrieval From Verified.Me



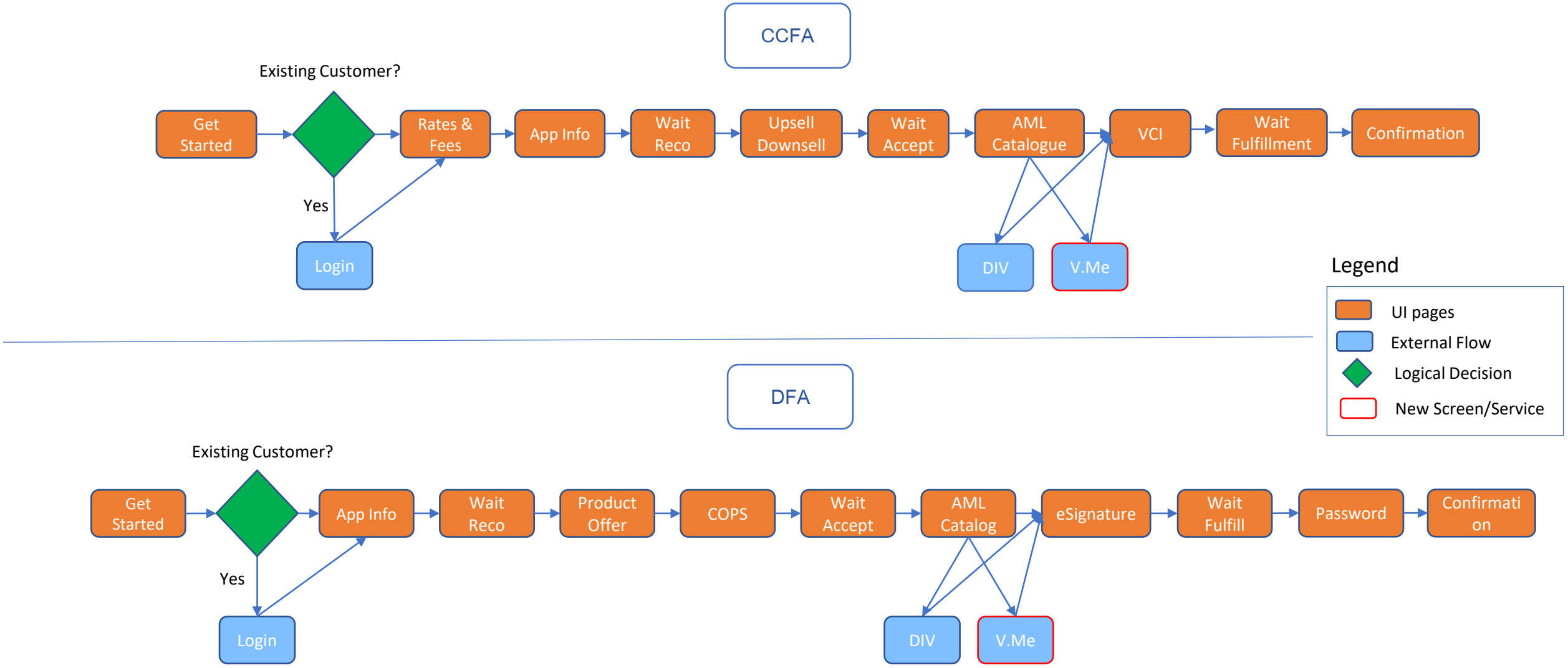
Client Data Retrieval From eBanking VME service



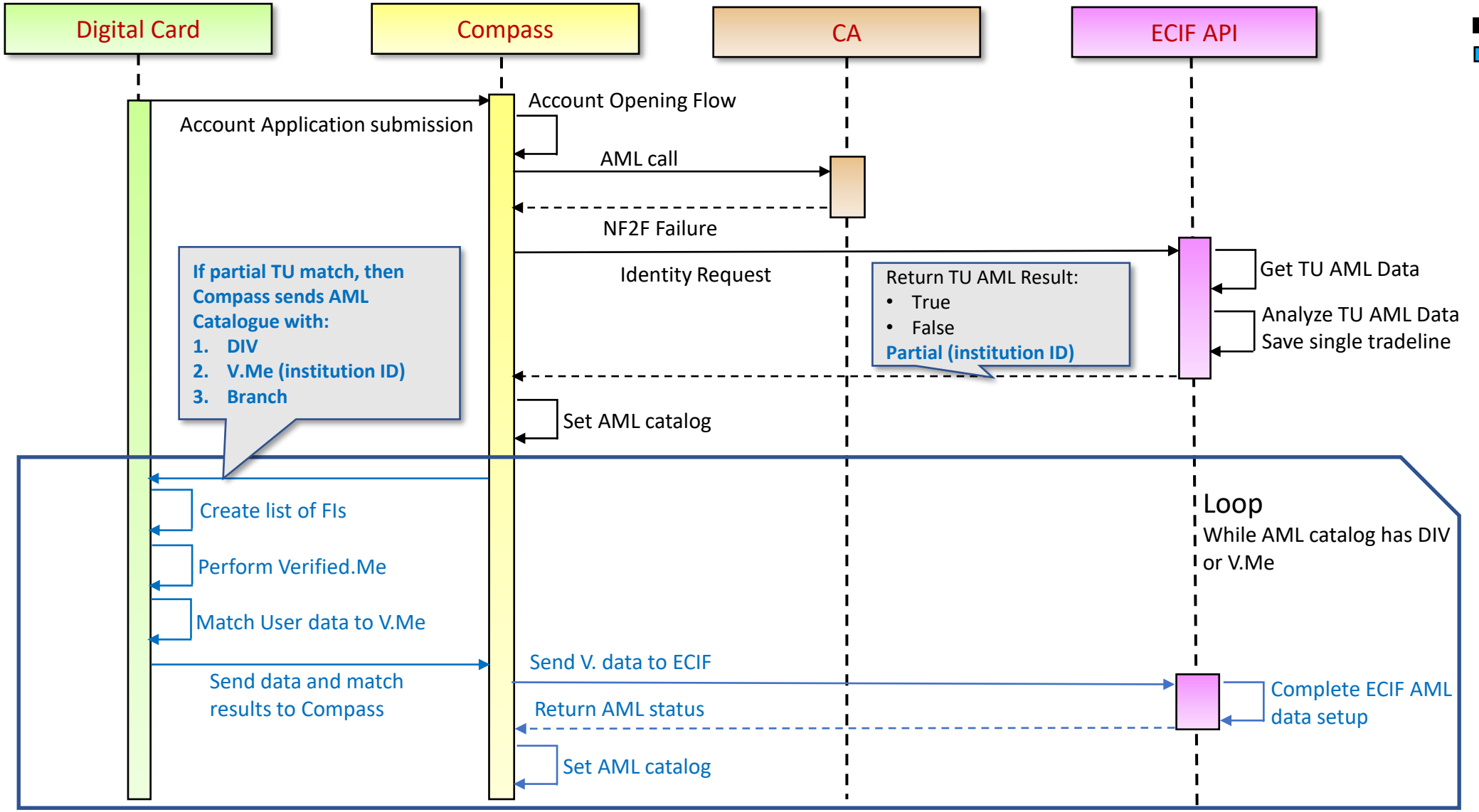
Digital Cart & VME Service Integration



Screen Flows with V.Me



EBM-VME Verified.Me as part of E2E AML Processing

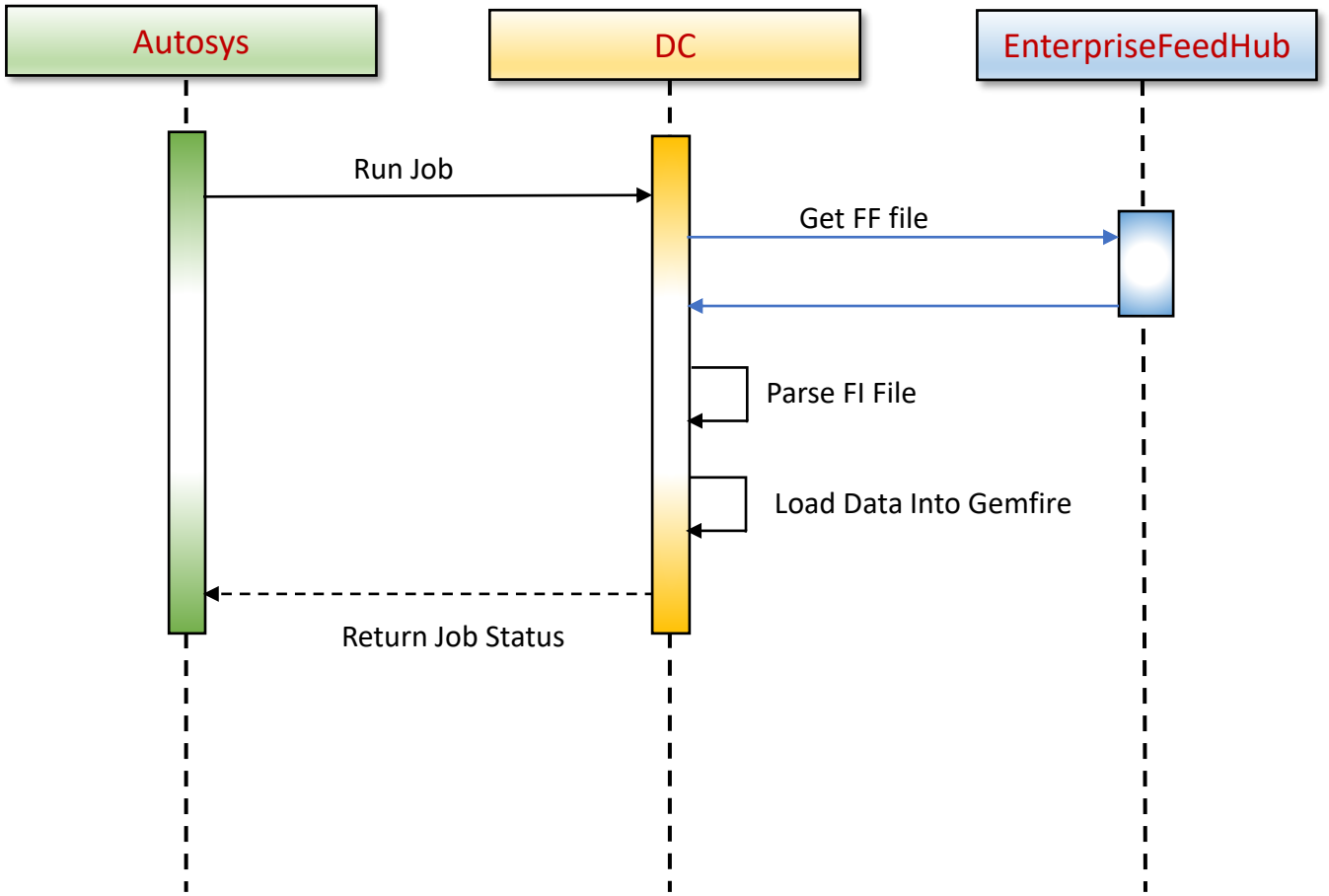


Current functionality

New functionality

1. Compliance Authority is called to perform AML validation
2. If there is AML compliance, the case proceeds to post compliance processing
3. If compliance fails due to ID validation, the Identity Validation service is called
4. Compass calls ECIF Identity API
5. Identity API calls TU
6. TU performs matching logic and provides a response
7. Identity API inspects TU response, updates ECIF and returns result to Compass
8. Compass prepares AML catalog based on ECIF response
9. eBanking presents AML catalog
10. eBanking performs V.Me if selected
11. If successful eBanking matches data and sends it to Compass
12. Compass sends the V.Me data to ECIF and gets AML status
13. Compass updates AML catalog and sends it to eBanking

Weekly Batch Job to Process FI File



- The FI file carries the list of all financial institutions in Canada
- It's posted on the FeedHub once a week
- eBanking needs to process the file by extracting financial institution name, number and parent number
- This data needs to be stored either in Gemfire as this is the only storage mechanism available in eBanking or possibly as a text file for DC module consumption
- The data will be used by DC module when presenting the AML catalogue to the client
- eBanking will compare the FI number received from TU AML against the list of FI institutions and exclude the matching institution or parent institution from the suggested list of V.Me FIs because they are not eligible if they are the source of TU AML data