|  |  |
| --- | --- |
| **Section Number & Heading** | **Requirement** |
| **2.1 Sequence Diagram** | - A VOP request from the Requesting PSP combines the Payment Account Number (IBAN) and the Name of the Payment Counterparty, or optionally includes an identification code.  - If multiple IBANs are to be verified, the Requesting PSP must send separate requests for each IBAN.  - The Requesting PSP submits the request, and the Responding PSP validates it, returning a response within a "Maximum Execution Time." |
| **2.2 VOP Scheme High-Level Flow** | - This section introduces the overall architecture, which includes the interactions with the EPC Directory Service (EDS) and how the VOP API integrates into the system.  - The document complements the VOP Scheme Rulebook, VOP API Specifications, and the API Security Framework. |
| **2.3 Use Cases (positive flows)** | - The VOP scheme supports three major use cases as positive flows:  1. **Name/IBAN check**: For both natural and legal persons.  2. **Identification code/IBAN check**: For legal persons only.  3. **Name/Identification code + additional attribute/IBAN check**: For either natural or legal persons with additional information.  - For each use case, the VOP Response includes codes such as Match (MTCH), No Match (NMTC), Close Match (CMTC), or Not Applicable (NOAP), along with potential error messages (HTTP 4xx/5xx). |
| **2.4 API Authentication and Authorisation** | - Describes the trust model for VOP API, which is part of the overall **API Security Framework**. The authentication process uses **TLS with Client Authentication** (via certificates issued by trusted authorities) to prove the identity of both the Requesting and Responding PSPs. |
| **2.4.1 Authentication** | - The Requesting PSP retrieves the Responding PSP’s API endpoint from the local copy of the EPC Directory Service (EDS), confirming the Responding PSP’s adherence to the VOP scheme.  - Mutual authentication between the Requesting PSP and Responding PSP is required via TLS with Client Authentication. |
| **2.4.2 Authorisation** | - The Responding PSP verifies the Requesting PSP’s adherence to the VOP EPC scheme by checking the National Authorization Number (NAN) and BIC combination via the **EDS**.  - Only authenticated clients are authorized to submit requests, with the Responding PSP verifying the validity of the BIC and NAN combination against the EDS data. |

|  |  |
| --- | --- |
| **Section Number & Heading** | **Requirement** |
| **3.1 Character Set and Data Types** | - The character set used for API parameters is **UTF-8**.  - Supported basic data types: **String**, **Boolean**, **ISODateTime**, **ISODate**, **UUID**, **BIC**, **LEI**, and **Integer** (32 bits).  - Specific string lengths: Max35Text, Max70Text, Max140Text, Max256Text, and Max500Text, which define the maximum number of characters allowed in a string.  - Character set for strings includes Latin characters, digits, and some punctuation (e.g., / - ? : ( ) . , ' + Space).  - A limited set of Latin characters is used in SEPA schemes, though additional characters may be exchanged based on prior agreements (bilateral/multilateral or through AOS).  - Refer to the “SEPA Requirements for an extended Character Set BEST PRACTICES” document for further details. |
| **3.2 Notations** | This section defines the notation used to describe attributes for both API requests and responses. |
| **3.2.1 Notation for Requests** | - **Attribute**: Describes the name of the attribute.  - **Type**: Specifies the data type of the attribute (either basic or complex, as defined in Section 3.1).  - **Condition**: Describes whether the attribute is **Mandatory**, **Optional**, or **Conditional**:  - **Mandatory**: Attribute must be provided by the client.  - **Optional**: Attribute is supported by the server but usage is optional for the client.  - **Conditional**: Attribute might be required under specific conditions. |
| **3.2.2 Notation for Responses** | - **Attribute**: Describes the name of the attribute.  - **Type**: Specifies the data type of the attribute (either basic or complex).  - **Condition**: Similar to requests, attributes in responses can be **Mandatory**, **Optional**, or **Conditional**:  - **Mandatory**: Attribute must always be included in the response.  - **Optional**: The server may include this attribute, but it is not required.  - **Conditional**: The attribute is included under certain conditions, as indicated in the description. |
| **3.2.3 Notations used for Requests and Responses** | - These notations apply to both requests from the client and responses from the server.  - **{Or, Or, Or}**: A sequence where exactly one of the elements must be included.  - **{Or – Optional, Or – Optional, Or – Optional}**: A sequence where at most one of the elements may be included, and all are optional.  - The YAML file used to define the API interface treats conditions like **"{Or"**, **"Or"**, and **"Or}"** as optional. Implementers are responsible for ensuring proper checks. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Section** | **Details** | **API Client Role** | **API Server Role** |
| **4.1 VOP API Technical Specifications** | Defines the overall structure of the API, including headers, request bodies, and response bodies, for successful VOP API execution. | Must structure requests according to defined standards. | Validate and process requests based on the defined structure and provide responses accordingly. |
| **4.1.1 API Access Methods** | - **Endpoint**: /vop/v1/payee-verifications  - **Method**: POST  - **Condition**: Mandatory  - **Description**: Submit a VOP Request. | Must use the correct endpoint and method to initiate VOP requests. | Must be prepared to process POST requests at the specified endpoint. |
| **4.1.2 Request Parameters** | Defines the headers and body elements required in API requests. | Must include all mandatory headers and body attributes in the specified format. | Validate and process the provided headers and request body attributes. |
| **4.1.2.1 Header** | - **X-Request-ID (UUID)**: Mandatory - **X-Request-Timestamp (ISODateTime)**: Mandatory | Generate and include these headers in the request with proper formats. | Validate headers for format and ensure correct timestamps. |
| **4.1.2.2 Request Body** | Contains elements such as **party**, **partyAccount**, **partyAgent**, **requestingAgent**, and optional data like unstructuredRemittanceInformation. | Provide mandatory data like Payee details, account information, and requesting/party agents. | Parse, validate, and process provided information. |
| **4.1.3 Response Parameters** | Defines the structure of responses, including headers, status codes, and body content. | Expect and handle defined headers and body responses appropriately. | Include appropriate headers and format in the response. |
| **4.1.3.1 Response Code** | HTTP 200 is used for successful requests. | Must handle 200 responses and analyze the response body for further processing. | Send HTTP 200 status for successful processing. |
| **4.1.3.2 Header** | - **X-Request-ID (UUID)**: Echoes client-provided ID. - **X-ResponseTimestamp (ISODateTime)**: Server response timestamp. | Use X-Request-ID for correlation. | Include these headers in response. |
| **4.1.3.3 Response Body** | Contains elements like partyNameMatch, partyIdMatch, and conditional attributes like matchedName. | Handle match codes to proceed with transactions (e.g., MTCH, NMTC, CMTC). | Provide appropriate match results in the response body based on processing rules. |
| **4.2 VOP API Data Model** | Provides detailed definitions of data types for elements such as Party Type, Account Type, Agent Type, and match codes. | Provide values matching the specifications for each data model element. | Validate and process these elements in requests. |
| **4.2.1 Party Type** | Specifies attributes like name, organisationId (LEI/BIC/others), and usage rules for combinations of IBAN + Name or Identifier Code. | Ensure the request adheres to specified combinations and usage rules. | Validate combinations and ensure they conform to Rulebook requirements. |
| **4.2.2 Account Type** | - **IBAN (Mandatory)**: Specifies the Payment Account Number. | Provide accurate IBAN for transactions. | Validate IBAN format and value. |
| **4.2.3 Agent Type** | - **BICFI (Mandatory)**: Provides the BIC for the Agent. | Include correct BIC for requesting/party agents. | Validate BIC codes and process requests accordingly. |
| **4.2.4 Generic Organisation Identification** | Attributes like identification, schemeNameCode, and issuer. One of schemeNameCode or schemeNameProprietary is mandatory when identification is used. | Follow ISO rules and include valid identification codes. | Validate provided codes and ensure compliance with Scheme Rulebook. |
| **4.2.5/6 Match Codes** | Defines codes for **Party Name Match** (MTCH, NMTC, CMTC, NOAP) and **Party Identification Match** (MTCH, NMTC, NOAP). | Interpret match codes to decide next actions in business flow. | Generate match codes based on processing results. |
| **4.3 Business Use Cases and Results** | Outlines matching scenarios for combinations like Name + IBAN and Identification Code + IBAN, including codes for various outcomes (e.g., MTCH, NMTC). | Use returned match codes to proceed with decision-making. | Provide matching results as per defined cases. |
| **4.4 Error Handling** | Provides structured error responses using RFC7807 standards with error codes and descriptions. | Handle errors as per provided codes and take corrective actions. | Return structured error responses for validation, timestamp, or client issues. |
| **4.5 HTTP Codes** | Supported HTTP status codes:  - **200 OK**: Successful - **400 Bad Request**: Validation errors - **401 Unauthorized** - **500 Internal Server Error** | Handle HTTP status codes to identify success or failure and take appropriate action. | Return correct HTTP status codes for client feedback and debugging. |

|  |  |  |
| --- | --- | --- |
| **Section** | **Details** | **Responder Role / API Server Requirements** |
| **4.1 VOP API Technical Specifications** | Defines the overall structure of the API, including headers, request bodies, and response bodies, for successful VOP API execution. | Validate the structure of incoming requests against API specifications and generate appropriate responses adhering to standards. |
| **4.1.1 API Access Methods** | - **Endpoint**: /vop/v1/payee-verifications  - **Method**: POST  - **Condition**: Mandatory  - **Description**: Submit a VOP Request. | Ensure the /vop/v1/payee-verifications endpoint is active and correctly configured to accept POST requests. |
| **4.1.2 Request Parameters** | Defines the headers and body elements required in API requests. | Parse incoming requests, validate mandatory headers and body attributes, and process them according to API specifications. |
| **4.1.2.1 Header** | - **X-Request-ID (UUID)**: Unique request ID from client. - **X-Request-Timestamp (ISODateTime)**: Request timestamp. | Validate X-Request-ID as a valid UUID. Ensure X-Request-Timestamp is in ISODateTime format and within the allowed time window. |
| **4.1.2.2 Request Body** | Contains elements such as **party**, **partyAccount**, **partyAgent**, **requestingAgent**, and optional data like unstructuredRemittanceInformation. | Validate and process all mandatory elements in the request body.  Optional data should be handled gracefully if present. |
| **4.1.3 Response Parameters** | Defines the structure of responses, including headers, status codes, and body content. | Ensure responses follow the specified structure with appropriate headers and body content. |
| **4.1.3.1 Response Code** | HTTP 200 is used for successful requests. | Return HTTP 200 status for successfully processed requests. |
| **4.1.3.2 Header** | - **X-Request-ID (UUID)**: Echoes client-provided ID. - **X-ResponseTimestamp (ISODateTime)**: Server response timestamp. | Include X-Request-ID in the response header for request correlation. Add X-ResponseTimestamp with the current timestamp in ISODateTime. |
| **4.1.3.3 Response Body** | Contains elements like partyNameMatch, partyIdMatch, and conditional attributes like matchedName. | Populate response body with matching results (MTCH, NMTC, etc.) as per validation.  Include matchedName if applicable. |
| **4.2 VOP API Data Model** | Provides detailed definitions of data types for elements such as Party Type, Account Type, Agent Type, and match codes. | Validate and process incoming data against the defined data models, ensuring compliance with format and conditions. |
| **4.2.1 Party Type** | Specifies attributes like name, organisationId (LEI/BIC/others), and usage rules for combinations of IBAN + Name or Identifier Code. | Verify and handle combinations such as **Name + IBAN** or **Identifier Code + IBAN**, ensuring they adhere to Scheme Rulebook standards. |
| **4.2.2 Account Type** | - **IBAN (Mandatory)**: Specifies the Payment Account Number. | Validate the provided IBAN for format and correctness using ISO 13616 standards. |
| **4.2.3 Agent Type** | - **BICFI (Mandatory)**: Provides the BIC for the Agent. | Validate BICFI codes for correctness and ensure they match directory standards. |
| **4.2.4 Generic Organisation Identification** | Attributes like identification, schemeNameCode, and issuer. One of schemeNameCode or schemeNameProprietary is mandatory when identification is used. | Validate organisation identification details against ISO standards and rulebook exceptions. |
| **4.2.5/6 Match Codes** | Defines codes for **Party Name Match** (MTCH, NMTC, CMTC, NOAP) and **Party Identification Match** (MTCH, NMTC, NOAP). | Determine and return appropriate match codes based on processing results. Ensure logic adheres to defined rules. |
| **4.3 Business Use Cases and Results** | Outlines matching scenarios for combinations like Name + IBAN and Identification Code + IBAN, including codes for various outcomes (e.g., MTCH, NMTC). | Process matching scenarios and return corresponding results, ensuring they align with rulebook-defined combinations and codes. |
| **4.4 Error Handling** | Provides structured error responses using RFC7807 standards with error codes and descriptions. | Generate structured error responses with relevant HTTP status codes and descriptive attributes for easier debugging by clients. |
| **4.4.1 Error Response Parameters** | Attributes like type, code, title, and status in structured JSON for detailed error descriptions. | Include detailed error information in JSON format for all error scenarios. |
| **4.4.2 Error Cases** | Lists error scenarios and corresponding codes (e.g., FORMAT\_ERROR, TIMESTAMP\_INVALID, CLIENT\_INVALID). | Handle identified error cases appropriately, returning the correct HTTP status code and detailed error message. |
| **4.5 HTTP Codes** | Supported HTTP status codes:  - **200 OK**: Successful - **400 Bad Request**: Validation errors - **401 Unauthorized** - **500 Internal Server Error** | Return appropriate HTTP status codes to indicate the result of the processing and provide additional details in response bodies where applicable. |