**ATIS-0x0000x**

ATIS Standard on

**Technical Report on SHAKEN API for a Centralized Signing and Signature Validation Server**

**Alliance for Telecommunications Industry Solutions**

Approved Month DD, YYYY

**Abstract**

This document provides a Technical Report on Originating Party Spoofing in IP Communication Networks. It describes problems associated with originating party spoofing in IP communication networks, identifies potential mitigation options, analyze pros and cons of mitigation options.

**Foreword**

The Alliance for Telecommunications Industry Solutions (ATIS) serves the public through improved understanding between carriers, customers, and manufacturers. The [**COMMITTEE NAME**] Committee [**INSERT MISSION**]. [**INSERT SCOPE**].

The mandatory requirements are designated by the word *shall* and recommendations by the word *should*. Where both a mandatory requirement and a recommendation are specified for the same criterion, the recommendation represents a goal currently identifiable as having distinct compatibility or performance advantages. The word *may* denotes a optional capability that could augment the standard. The standard is fully functional without the incorporation of this optional capability.

Suggestions for improvement of this document are welcome. They should be sent to the Alliance for Telecommunications Industry Solutions, [**COMMITTEE NAME**], 1200 G Street NW, Suite 500, Washington, DC 20005.

At the time of consensus on this document, [**COMMITTEE NAME**], which was responsible for its development, had the following leadership:

[**LEADERSHIP LIST**]

The **[SUBCOMMITTEE NAME]** Subcommittee was responsible for the development of this document.

**Revision History**

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
|  |  |  |  |

**Table of Contents**

[INSERT]

**Table of Figures**

[INSERT]

**Table of Tables**

[INSERT]

# Introduction

This technical report defines the RESTful interface to be used in the SHAKEN framework to sign and verify telephony identity:

* STI-AS (Secure Telephone Identity Authentication Service) shall expose an API to sign the provided PASSporT token which includes the SHAKEN extension as defined in [draft-wendt-stir-passport-shaken]
* STI-VS (Secure Telephone Identity Verification Service) shall expose an API to verify the signed STI according to procedures defined in draft-ietf-stir-passport

The only algorithm currently supported by this API is ES256.

Editor’s Note: add disclaimer that this is a reference implementation.

# Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

1. “RESTful Web Services Standards” - <http://tss.att.com/document/R113140.pdf>.
2. STIR-Passport: <https://tools.ietf.org/html/draft-ietf-stir-passport-10>
3. SHAKEN extensions for PASSporT : https://datatracker.ietf.org/doc/draft-wendt-stir-passport-shaken/
4. SIP based framework is defined in RFC 4474bis:  <https://tools.ietf.org/html/draft-ietf-stir-rfc4474bis-15>
5. SHAKEN framework spec [ATIS-1000080]

# Definitions, Acronyms, & Abbreviations

For a list of common communications terms and definitions, please visit the *ATIS Telecom Glossary*, which is located at < <http://www.atis.org/glossary> >.

## Definitions

**Caller identity:** The originating phone number included in call signalling used to identify the caller for call screening purposes.In some cases this may be the Calling Line Identification or Public User Identity. For the purposes of this study, the caller identity may be set to an identity other than the caller’s Calling Line Identification or Public User Identity.

## Acronyms & Abbreviations

|  |  |
| --- | --- |
| Acronym | Term |
| STI  | Secure Telephone Identity |
| STI-AS | STI Authentication Service |
| STI-VS | STI Verification Service |
| SHAKEN  | Signature based HAndling of Asserted Information using toKENs  |
| STIR | Secure Telephone Identity Revisited |
| UUID  | Universally Unique Identifier |
| PASSporT | Persona Assertion Token |

# Architecture



Figure 4.1 – SHAKEN Reference Architecture

Xxxxx



# General API Requirements

1. STI-AS and STI-VS have to expose a RESTful web services implemented using HTTP and aligned with the principles of RESTful API.
2. Only JSON based data format is supported. APIs use “application/json” content type
3. All validations will be described below in the error handling sections for each API explicitly .
4. POST HTTP request is used for the both APIs.
5. HTTP 1.1 protocol version has to be supported by server side.

## Resource Structure

REST resources are defined with respect to a “server Root” :

 “serverRoot” = http://{hostname}:{port}/{optionalRoutingPath}

The resource structure is provided below:

 

‘apiVersion’ should be set to “1”.

## Special Request Header Requirements

The following headers are expected to be sent in all HTTP requests:

| Header Name | Mandatory? | Description |
| --- | --- | --- |
| X-RequestID | N | According to the general agreement the transaction UUID should be published by component calling an exposed by other component API in order to make possible the transaction traceability in case of troubleshooting and fault analysis. Generated UUIID should be compliant with RFC 4122. If received will not be validated explicitly by server. If not received it will be automatically generated by STI-AS/VS service on request receipt.Received/Generated transaction UUID will be returned back in the corresponding HTTP response in “X-RequestID” header. |
| X-InstanceID | N | For  auditing  purpose  each  component  calling  the  API should identify  itself  by sending its  identity  (  e.g. Instar  name , VNFC name/UUID , VM name/UUID ...) in  "**X-InstanceID**" header .  |
| Content-Type | Y | Determines the format of the request body. Valid value is: “**application/json**”.Requests with other types will be rejected with “415 Unsupported Media type” HTTP status code. |
| Accept | N  | If specified has to contain “**application/json**” content type, otherwise HTTP request will be rejected with “406 Not Acceptable“ HTTP Status Code.If not specified will be y default handled as “**application/json**”. |

## Special Response Header Requirements

The following headers are expected to be sent in all HTTP responses:

| Header Name | Mandatory? | Description |
| --- | --- | --- |
| X-RequestID | Y | Received/Generated transaction UUID will be returned back in the corresponding HTTP response. |
| Content-Type | Y | Determines the format of the response body. Valid value is : **“application/json”**  |

# Data Types

## Datatype: signingRequest

|  |  |  |  |
| --- | --- | --- | --- |
| Key Name | Key Value Type | Required? | Description |
| attest | String Allowed values :  [“A” , “B” , “C”] | Y | SHAKEN extension to PASSporT.Indicator identifying the service provider that is vouching for the call as well as a clearly indicating what information the service provider is attesting to.SHAKEN spec requires “attest” key value be set to uppercase characters “A”, “B”, or “C”.  |
| dest | destTelephoneNumber | Y | Represents the called party. Array containing **one or more** identities of telepnoneNumber type. |
| iat | Integer | Y | “Issued At Claim”: Should be set to the date and time of issuance of the PASSporT Token. The time value should be in the Numeric Date format defined in RFC 7519 : number of seconds elapsed since 00:00:00 UTV , Thursday , 1 January 1970 not including leap seconds . |
| orig | origTelephoneNumber | Y | Represents the asserted identity of the originator of the personal communications signaling. |
| origid | String | Y | The unique origination identifier (“origid”) is defined as part of SHAKEN extension to PASSporT. This unique origination identifier should be a globally unique string corresponding to a UUID (RFC 4122).Note: VM UUID can be used as a unique originator identifier. |

## Datatype: origTelephoneNumber

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| tn | String Allowed Characters :  [0-9] ,\*,#,+, and visual separators defined in RFC 3966 : “.”, “-“, “(“, “)”. | Y | Telephone Number of Originating/Destination identity.Server will remove all non-numeric characters if received except start (\*) and pound (#) characters.Ex. : (+1)235-555-1212 🡪 12355551212 |

## Datatype: destTelephoneNumber

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| tn | List of Strings  [1 … unbounded] Allowed Characters :  [0-9] ,\*,#,+, and visual separators defined in RFC 3966: “.”, “-“, “(“, “)”. | Y | List containing **one or more** identities of String type.Server will remove all non-numeric characters if received except start (\*) and pound (#) characters.Ex. : (+1)235-555-1212 🡪 12355551212 |

## Datatype: siginingResponse

|  |  |  |  |
| --- | --- | --- | --- |
| Key Name | Key Value Type | Required? | Description |
| identity | String Cannot be NULL | Y | Identity header value as defined in RFC4474bis with “identityDigest” in full format and mandatory “info” and “alg” header field parameters. The “info” header field parameter contains the public key URL of the certificate used during STI signing. The “alg” header field parameter contains the algorithm used during STI signing.  |

## Datatype: verificationRequest

|  |  |  |  |
| --- | --- | --- | --- |
| Key Name | Key Value Type | Required? | Description |
| identity | String  | Y | Identity header value as defined in RFC4474bis with “identityDigest” in full format and mandatory “info” and “alg” header field parameters. |
| to | destTelephoneNumber | Y | Represents the called party. Array containing **one or more** identities of telepnoneNumber type. This shall be set to the value of the “To:” header field parameter in the incoming SIP Invite |
| time | Integer | Y | This shall be set based on the value of the Date header field parameter in the incoming Invite.The time value should be in the Numeric Date format defined in RFC 7519 : number of seconds elapsed since 00:00:00 UTV , Thursday , 1 January 1970 not including leap seconds . |
| from | origTelephoneNumber | Y | Represents the asserted identity of the originator of the personal communications signaling.This shall be set to the value of the “P-Asserted-Identity”, if available, or “From” header field parameter in the incoming Invite.  |

## Datatype: serviceException

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| serviceException | exception | Yes | Service Exception |

## Datatype: verificationResponse

|  |  |  |  |
| --- | --- | --- | --- |
| Key Name | Key Value Type | Required? | Description |
| reasoncode | Integer  | N | Reason Code to be used in case of failed verification by STI-VS to build SIP Reason header if required.Currently possible values are defined as follows (please pay attention they can be extended/changed in the future) :403,428 ( will not be returned in the initial release) ,436,437,438403 – “Stale Date header received” 436 – Bad Public Key Certificate URI |
| reasontext | String | N | Reason Text to be used in case of failed verification by STI-VS to build SIP Reason header if required.Currently possible values are defined as follows (please pay attention they can be extended/changed in the future) :403 - “Stale Date”428 - “Use Identity Header” (will not be returned in the initial release) 436 – “Bad Identity Info” 437 – “Unsupported Credential”438 – “Invalid Identity Header” |
| reasondesc | String | N | Reason details description . Can be used for logging and troubleshooting. |
| verstat | String{“TN-Validation-Passed”,“TN-Validation-Failed”,“No-TN-Validation”} | Y | Verification Status :**TN-Validation-Passed** - The calling number passed the validation**TN-Validation-Faile**d - The calling number failed the validation**No-TN-Validation** - No validation number was performed |

## Datatype: exception

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| messageId | string | Yes | Unique message identifier of the format ‘ABCnnnn’ where ‘ABC’ is either ‘SVC’ for Service Exceptions or ‘POL’ for Policy Exception. Exception numbers may be in the range of 0001 to 9999 where 0001 to 2999 are defined by OMA and 3000-9999 are available and undefined. |
| text | string | Yes | Message text, with replacement variables marked with %n, where n is an index into the list of <variables> elements, starting at 1 |
| variables | string | No | List of zero or more strings that represent the contents of the variables used by the message text |
| url | string | No | Hyperlink to a detailed error resource e.g., an HTML page for browser user agents. Currently will not be used. |

## Datatype: policyException

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Required? | Description |
| policyException | exception | Yes | Policy Exception |

## Datatype: requestError

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field |  | Type | Required? | Description |
| requestError | policyException or serviceException | exception | Yes | Request Error Message |

# Exceptions

## RESTful WebServices exceptions

RESTful services generate and send exceptions to clients in response to invocation errors. Exceptions send HTTP status codes (specified later in this document for each operation). HTTP status codes may be followed by an optional JSON exception structure [(“requestError” datatype](#_Datatype:_requestError)). Two types of exceptions may be defined: service exceptions and policy exceptions.

## Service exceptions

When a service is not able to process a request, and retrying the request with the same information will also result in a failure, and the issue is not related to a service policy issue, then the service will issue a fault using the service exception fault message. Examples of service exceptions include invalid input, lack of availability of a required resource or a processing error.

A service exception uses the letters 'SVC' at the beginning of the message identifier. ‘SVC’ service exceptions used by SHAKEN API are defined below:

| Exception ID  | Exception text | HTTP Status Code | Exception  Variables | Error Description |
| --- | --- | --- | --- | --- |
| SVC4000 | Error: Missing request body. | 400 | - | **MISSING\_BODY**The API failed due to missing body. |
| SVC4001 | Error: Missing mandatory parameter ‘%1’. | 400 | %1 – parameter name | **MISSING\_INFORMATION**The API failed due to missing mandatory parameter |
| SVC4002 | Error: Requested response body type ‘%1’ is not supported. | 406 | %1 – not supported response body type | **NOT\_ACCEPTABLE\_RESPONSE\_BODY\_TYPE**A request was made of a resource for a non-supported message body format  |
| SVC4003 | Error: Requested resource was not found. | 404 | - | **RESOURCE\_NOT\_FOUND**The server has not found anything matching the Request-URI |
| SVC4004 | Error: Unsupported request body type, expected ‘%1’. | 415 | %1 – content type (’application/json’) | **UNSUPPORTED\_REQUEST\_BODY\_TYPE**Received unsupported message body type |
| SVC4005 |  Error: Invalid ‘%1’ parameter value: %2. | 400 | %1 – parameter name%2– short error description  | **INVALID\_PARAMETER\_VALUE**Parameter’s value is invalid  |
| SVC4006 | Error: Failed to parse received message body: %1.  | 400 | %1-“invalid message body length specified”/”invalid JSON body” | **FAILED\_TO\_PARSE\_MSG\_BODY** |
| SVC4007 | Error : Missing mandatory Content-Length header | 411 | - | **MISSING\_BODY\_LENGTH**The Content-Length header was not specified. |

## Policy exceptions

When a service is not able to complete because the request fails to meet a policy criteria, then the service will issue a fault using the policy exception fault message. To clarify how a policy exception differs from a service exception, consider that all the input to an operation may be valid as meeting the required input for the operation (thus no service exception), but using that input in the execution of the service may result in conditions that require the service not to complete. Examples of policy exceptions include API violations, requests not permitted under a governing service agreement or input content not acceptable to the service provider.

A Policy Exception uses the letters 'POL' at the beginning of the message identifier. ‘POL’ policy exceptions used by SHAKEN API are defined below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Exception ID  | Exception text | HTTP Status Code | Exception  Variables | Error Description |
| POL4050 | Error: Method not allowed | 405 | - | The resource was invoked with unsupported operation. |
| POL5000 | Error: Internal Server Error. Please try again later | 500 | - | The request failed either due to internal vIRC problem. |

# API Interface

## Signing API

###  Functional Behavior

Used to create the PASSporT signature with private key certificate.

The Authenticator sends a signingRequest including the following to the SHAKEN Signing Service:

1. The “orig” parameter is populated using the PAI field if present, otherwise using the From header field in the SIP Invite.
2. The “dest” parameter is populated using the To header field in the SIP Invite.
3. The “iat” parameter is populated using the “Date” header field in the SIP Invite. If there is no “Date” header field in the SIP Invite, a Date header field is added to the SIP INVITE.
4. The “origid” parameter is determined as described in ATIS-1000074 for the “origid” field in the PASSporT.
5. The “attest” parameter is determined as described in ATIS-1000074 for the “attest” field in the PASSporT.
6. The signingRequest is then sent to the SHAKEN Signing Service.

The SHAKEN Signing Service performs the following steps:

1. Validate the incoming signing request parameters in terms of parameter’s type and format.

2. Validate the “iat” parameter value in terms of “freshness”: the request with “iat” value with time different by more than one minute from the current time will be rejected.

3. Normalize to the canonical form the received telephony numbers if needed (remove visual separators and leading “+”).

3. Build SHAKEN PASSport protected JWT header (with “ppt” SHAKEN extension).

4. Build SHAKEN PASSporT JWT payload by keeping lexicographic order and removing space and line breaking characters.

6. Generate PASSporT signature with appropriate certificate private key.

7. Build Full Form of PASSporT.

8. Build SIP “Identity” header value by using identity digest from the previous step and add “info” parameter with angle bracketed URI to acquire the public key of certificate used during PASSporT signing

9. In case of successfully signing build and send “siginingResponse” to the Authenticator, otherwise send error.

Upon receipt of the signingResponse, the Authenticator uses the “identity” parameter in the response to populate the SIP Identity header field and forwards the request. If no identity parameter is received in a response, the Authenticator forwards the request without adding a SIP Identity header field.

###  Call Flow



* + 1. **Request (POST)**

The used resource is: https://{serverRoot}/stir/v1/signing

|  |  |
| --- | --- |
| Name  | Description |
| serverRoot | Server base URL : hostname+port+base pathHostname shall contain the Global FQDN of Signing Service |

* + - 1. **Request Body**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Data Type | Required? | Brief description |
| Signing Request | signingRequest | Yes | Contains the JSON structure of the signing request (PASSporT payload claims)  |

* + - 1. **Request Sample**
1. POST /stir/v1/signing HTTP/1.1
2. Host : stir.att.com
3. Accept : application/json
4. X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
5. Content-Type: application/json
6. Content-Length : …

{

 "signingRequest”: {

 "attest": “A”,

 "orig”: {

 “tn”: “12155551212”

 },

 “dest”: {

 “tn” : [

 “12355551212”

 ]

 },

 "iat”: 1443208345,

 “origid”: “de305d54-75b4-431b-adb2-eb6b9e546014”

 }

}

Editor’s Note: reflect name changes in example

* + 1. **Response**
			1. **Response Body**

Response body is returned as JSON object (Content-Type: application/son).

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Data Type | Required? | Brief description |
| Signing Response | signingResponse | Yes | Contains the JSON structure of the signing response (SIP Identity header field value). |

* + - 1. **Response Sample (Success)**
1. HTTP/1.1 200 Ok
2. X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
3. Content-Type : application/json
4. Content-Length : …

{

 "signingResponse": {

 "identity" : “eyJhbGciOiJFUzI1NiIsInR5cCI6InBhc3Nwb3J0IiwicHB0Ijoic2hha2VuIiwieDV1IjoiaHR0cDov L2NlcnQtYXV0aC5wb2Muc3lzLmNvbWNhc3QubmV0L2V4YW1wbGUuY2VydCJ9eyJhdHRlc3QiOiJBIiwiZGVzdCI6eyJ0biI6IisxMjE1NTU1MTIxMyJ9LCJpYXQiOiIxNDcxMzc1NDE4Iiwib3JpZyI6eyJ0biI64oCdKzEyMTU1NTUxMjEyIn0sIm9yaWdpZCI6IjEyM2U0NTY3LWU4OWItMTJkMy1hNDU2LTQyNjY1NTQ0MDAwMCJ9.\_28kAwRWnheXyA6nY4MvmK5JKHZH9hSYkWI4g75mnq9Tj2lW4WPm0PlvudoGaj7wM5XujZUTb\_3MA4modoDtCA;info=<<https://cert.example2.net/example.cert>>;alg=ES256”

 }

}

* + - 1. **Response Sample (Failure)**
1. HTTP/1.1 400 Bad RequestProp
2. X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
3. Content-Type : application/json
4. Content-Length : …

{

 “requestError”: {

 “serviceException”: {

 “messageId”: “SVC4501”

 “text”: “Error: Invalid Content. Missing mandatory parameter ‘%1’”,

 “variables”: [“iat”]

 }

 }

}

* + - 1. **HTTP Response Codes**

|  |  |  |
| --- | --- | --- |
| Response code | Service/PolicyException | Reason /Description |
| 200 | N/A | Successful signing |
| 400 | SVC4000 | Missing JSON body in the request |
| 400 | SVC4001 | Missing mandatory parameter |
| 406 | SVC4002 | Not supported body type is specified in Accept HTTP header |
| 415 | SVC4004 | Received unsupported message body type in Content-Type HTTP header |
| 400 | SVC4005 | Invalid parameter value |
| 400 | SVC4006 | Failed to parse JSON body |
| 411 | SVC4007 | Missing mandatory Content-Length header |
| 405 | POL4050 | Method Not Allowed : Invalid HTTP method used ( all methods except POST will be rejected for the specific resource URL)  |
| 500  | POL5000 | The POST request failed either due to internal signing server problem. |

* 1. **Verification API**
		1. **Functional Behavior**

 The Verification API is used to verify the signature provided in the Identity header field and to determine that the signing service credentials demonstrate authority over the call originating identity.

Upon receipt of a SIP INVITE containing a SIP Identity header field parameter, the Verifier builds a verficiationRequest as follows:

1. The “from” parameter is populated using the PAI field if present, otherwise using the From header field in the SIP Invite.
2. The “to” parameter is populated with the To header field from the SIP Invite.
3. The “time” parameter value is populated with the “iat” parameter in the PASSporT .
4. The “identity” parameter value is populated using the Identity header field in the SIP Invite.
5. The Verifier then sends the HTTPS Post to request verification.

Upon receipt of the verificationRequest, the SHAKEN Verification Service performs the following steps. Each step is associated with the appropriate error case(s) specified in the section “Mapping of verification failure cases to the returned SIP Reason header parameters” The error case numbers **En** per each step is specified in parentheses.

1. Validate the incoming verification request parameters in terms of parameter’s type and format (E1 and E2).

2. Validate the “time” parameter value in terms of “freshness”: a request with a “time” value which is different by more than one minute from the current time will be rejected (E3)

3. Parse the “identity” parameter value:

 - full form of PASSporT is required by SHAKEN : “identity-digest” parameter of Identity header has to be parsed to validate the full form format ( 3 data portions delimited with dot (“.”) ) .If the expected format is not matched 🡪 reject request on the Invalid PASSporT form (E4)

 - If “ppt” parameter is specified and its value is not “shaken” 🡪 reject request (E5)

 - If “info” parameter is not specified 🡪 reject request (E6)

 - If the URI specified in “info” parameter is not syntactically valid 🡪 reject request (E7)

4. Decode “identity-digest” parameter value to extract from the first portion (**PASSporT header** ) “ppt” , “typ”,”alg” and “x5u” claims :

 - If one of the mentioned claims is missing -> reject request ( E9)

 - if extracted “typ” value is not equal to “passport” 🡪 reject request (E11)

 - if extracted “alg” value is not equal to “ES256” 🡪 reject request ( E12)

 - if extracted “x5u” value is not equal to the URI specified in the “info” parameter of Identity header 🡪 reject request (E10)

 - If extracted “ppt” is not equal to “shaken” 🡪 reject request (E13)

5. Decode “identity-digest” parameter value to extract from the second portion (**PASSporT payload**) “dest” , “orig” , “attest”, “origid” and “iat” claims :

 - on missing mandatory claims reject request ( E14)

 - validate the extracted from payload “iat” claim value in terms of “freshness” relative to “time” value: request with “expired” “iat” will be rejected🡪 reject request (E15)

 - Normalize to the canonical form the received in the “verificationRequest” “from” and “to” telephone numbers (remove visual separators and leading “+”) and compare them with ones extracted from the

“orig” and “dest” claims of PASSporT payload. If they are not identical 🡪 reject request (E16)

 6. Dereference “info” parameter URI to a resource that contains the public key of the certificate used by signing service to sign a request . If there is a failure to dereference the URI due to timeout or a non-existent resource the request shall be rejected ( E8).

7. Validate the issuing CA . On the failure to authenticate the CA ( for example not valid, no root CA) request will be rejected (E17))

8. Validate the signature of “identity” digest parameter. On failure reject the request (E18).

9. If there is no “origid” and/or “attest” claims in the decrypted PASSporT payload 🡪 reject (E19 , E20)).

10. Compare all PASSporT claims from decrypted identity digest ( except “origid” and “attest”) and claims from PASSporT header and payload validated at step 5 and 6 above. If one of the claim’s value does not match the request should be be rejected ( E21 , E22) .

* + 1. **Call Flow**



* + 1. **Request (POST)**

The used resource is: https://{serverRoot}/stir/v1/verification

|  |  |
| --- | --- |
| Name  | Description |
| serverRoot | Server base URL : hostname+port+base pathHostname shall contain the Global FQDN of Verification Service |

* + - 1. **Request Body**

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Data Type | Required? | Brief description |
| Verification Request | verificationRequest | Yes | Contains the JSON structure of the verification request (PASSporT payload claims + identity header)  |

* + - 1. **Request Sample**
1. POST /stir/v1/verification HTTP/1.1
2. Host : stir.att.com
3. Accept : application/json
4. X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
5. Content-Type: application/json
6. Content-Length : …

{

 “verificationRequest”: {

 “from”: {

 “tn”: “12155551212”

 },

 “to”: {

 “tn” : [

 “12355551212”

 ]

 },

 “time”: 1443208345,

 “identity” : “eyJhbGciOiJFUzI1NiIsInR5cCI6InBhc3Nwb3J0IiwicHB0Ijoic2hha2VuIiwieDV1IjoiaHR0cDov L2NlcnQtYXV0aC5wb2Muc3lzLmNvbWNhc3QubmV0L2V4YW1wbGUuY2VydCJ9eyJhdHRlc3QiOiJBIiwiZGVzdCI6eyJ0biI6IisxMjE1NTU1MTIxMyJ9LCJpYXQiOiIxNDcxMzc1NDE4Iiwib3JpZyI6eyJ0biI64oCdKzEyMTU1NTUxMjEyIn0sIm9yaWdpZCI6IjEyM2U0NTY3LWU4OWItMTJkMy1hNDU2LTQyNjY1NTQ0MDAwMCJ9.\_28kAwRWnheXyA6nY4MvmK5JKHZH9hSYkWI4g75mnq9Tj2lW4WPm0PlvudoGaj7wM5XujZUTb\_3MA4modoDtCA;info=<<https://cert.example2.net/example.cert>>;alg=es256”

 }

}

* + 1. **Response**
			1. **Response Body**

Response body is returned as JSON object (Content-Type: application/son).

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Data Type | Required? | Brief description |
| Verification Response | verificationResponse | Yes | Contains the JSON structure of the verification response. |

* + - 1. **Mapping of verification failure cases to the returned SIP Reason header field parameters**

| Error Case Number | Error Case  | HTTP Status Code | SIP Reason Code | SIP Reason Text | “verstat”  |
| --- | --- | --- | --- | --- | --- |
| E1 | Missing mandatory parameters in the verification request ( “orig’, “dest” ,”iat” , “identity”) | 400 with service exception | - | - | No-TN-Validation |
| E2 | Received invalid parameters( invalid “tn” , “iat” value …) | 400 with service exception  | - | - | No-TN-Validation |
| E3 | Received “iat” value is not “fresh”  | 200  | 403 | Stale Date | No-TN-Validation |
| E4 | Identity header with “identity-digest” in compact form instead of required by SHAKEN spec full form. | 200 | 438 | InvalidIdentity Header | No-TN-Validation |
| E5 | Identity header is received with “ppt” parameter value that is not “shaken”  | 200 | 438 | InvalidIdentity Header | No-TN-Validation |
| E6 | Missing “info” parameter in the “identity”  | 200 | 436 | Bad identity Info | No-TN-Validation |
| E7 | “info” parameter from “identity” is invalid (syntactically invalid URI) | 200  | 436 | Bad identity Info | No-TN-Validation |
| E8 | Failed to dereference “info” URI  | 200  | 436 | Bad identity Info | No-TN-Validation |
| E9 | “ppt”/”typ”/”alg”/”x5u” claims missing in the PASSporT header  | 200 | 436 | Bad identity Info | No-TN-Validation |
| E10 | “x5u” from PASSporT header doesn’t match the “info” parameter of identity header value | 200  | 436 | Bad identity Info | No-TN-Validation |
| E11 | “typ” from PASSporT header is not “passport” | 200  | 437  | Unsupported credential | No-TN-Validation |
| E12 | “alg” from PASSporT header is not “ES256” | 200 | 437  | Unsupported credential | No-TN-Validation |
| E13 | “ppt” from PASSporT header is not “shaken” | 200 | 438 | InvalidIdentity Header | No-TN-Validation |
| E14 | Missing mandatory claims in PASSporT payload ( “dest” , “orig” , “attest” , “origid” ) | 200 | 438 | InvalidIdentity Header | No-TN-Validation |
| E15 | “iat” from in PASSporT payload is not “fresh” | 200  | 403 | Stale Date | No-TN-Validation |
| E16 | “orig” and “dest” claims from PASSporT payload don’t match the received in the verification request corresponding claims  | 200 | 438 | InvalidIdentity Header | No-TN-Validation |
| E17 | Failed to authenticate CA | 400 | 437 | Unsupported credential | TN-Validation-Failed |
| E18 | Signature validation failed  | 200  | 438 | InvalidIdentity Header | TN-Validation-Failed |
| E19 | Missing SHAKEN extension “attest” claim in the decrypted PASSporT | 200 | 438 | InvalidIdentity Header | TN-Validation-Failed |
| E20 | Missing SHAKEN extension “origid” claim in the decrypted PASSporT | 200 | 438 | InvalidIdentity Header | TN-Validation-Failed |
| E21 | “orig” /”dest” claims from decrypted payload don’t match the ones received in the INVITE  | 200 | 438  | Invalid Identity Header  | TN-Validation-Failed |
| E22 | “iat” claim from decrypted payload doesn’t match the “iat” from PASSporT payload. | 200 | 438  | Invalid Identity Header  | TN-Validation-Failed |
| E23 | Successful verification | 200 | - | - | TN-Validation-Passed |

* + - 1. **Response Sample (Success + Successful Validation)**
1. HTTP/1.1 200 Ok
2. X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
3. Content-Type : application/json
4. Content-Length : …

{

 "verificationResponse": {

 “verstat”: “TN-Validation-Passed”

 }

}

* + - 1. **Response Sample (Success + Failed Validation)**
1. HTTP/1.1 200 Ok
2. X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
3. Content-Type : application/json
4. Content-Length : …

{

 "verificationResponse": {

 “reasoncode”: 436200,

 “reasontext”: “Bad Identity Info”,

 “reasondesc”: “Info URI dereferencing failure”,

 “verstat”: “TN-Validation-Failed”

 }

}

* + - 1. **Response Sample (Failure)**
1. HTTP/1.1 400 Bad Request
2. X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
3. Content-Type : application/json
4. Content-Length : …

{

 “requestError”: {

 “serviceException”: {

 “messageId”: “SVC4501”

 “text”: “Error: Invalid Content. Missing mandatory parameter ‘%1’”,

 “variables”: [“iat”]

 }

 }

}

* + - 1. **HTTP Response Codes**

|  |  |  |
| --- | --- | --- |
| Response code | Service/PolicyException | Reason /Description |
| 200 | N/A | Successful signing |
| 400 | SVC4000 | Missing JSON body in the request |
| 400 | SVC4001 | Missing mandatory parameter |
| 406 | SVC4002 | Not supported body type is specified in Accept HTTP header |
| 415 | SVC4004 | Received unsupported message body type in Content-Type HTTP header |
| 400 | SVC4005 | Invalid parameter value |
| 400 | SVC4006 | Failed to parse JSON body |
| 411 | SVC4007 | Missing mandatory Content-Length header |
| 405 | POL4050 | Method Not Allowed : Invalid HTTP method used ( all methods except POST will be rejected for the specific resource URL)  |
| 500  | POL5000 | The POST request failed either due to internal signing server problem. |

# Conclusions

**Annex A**

(normative/informative)

# A XXXX

This annex will document supportive material