ATIS-1000082

ATIS Standard on

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

Alliance for Telecommunications Industry Solutions

Approved May 2018

Abstract

This document provides a Technical Report on a SHAKEN APIs used to support a Centralized Signing and Signature Validation Server. These APIs provide a means for multiple and/or disparate network elements to use an HTTP-based RESTful interface to access SHAKEN Signing and Signature Validation servers.

Foreword

The Alliance for Telecommunication Industry Solutions (ATIS) serves the public through improved understanding between providers, customers, and manufacturers. The Packet Technologies and Systems Committee (PTSC) develops and recommends standards and technical reports related to services, architectures, and signaling, in addition to related subjects under consideration in other North American and international standards bodies. PTSC coordinates and develops standards and technical reports relevant to telecommunications networks in the U.S., reviews and prepares contributions on such matters for submission to U.S. International Telecommunication Union Telecommunication Sector (ITU-T) and U.S. ITU Radiocommunication Sector (ITU-R) Study Groups or other standards organizations, and reviews for acceptability or per contra the positions of other countries in related standards development and takes or recommends appropriate actions.

The SIP Forum is an IP communications industry association that engages in numerous activities that promote and advance SIP-based technology, such as the development of industry recommendations, the SIPit, SIPconnect-IT, and RTCWeb-it interoperability testing events, special workshops, educational seminars, and general promotion of SIP in the industry. The SIP Forum is also the producer of the annual SIP Network Operators Conference (SIPNOC), focused on the technical requirements of the service provider community. One of the Forum's notable technical activities is the development of the SIPconnect Technical Recommendation – a standards-based SIP trunking recommendation for direct IP peering and interoperability between IP Private Branch Exchanges (PBXs) and SIP-based service provider networks. Other important Forum initiatives include work in Video Relay Service (VRS) interoperability, security, Network-to-Network Interoperability (NNI), and SIP and IPv6.

Suggestions for improvement of this document are welcome. They should be sent to the Alliance for Telecommunications Industry Solutions, PTSC, 1200 G Street NW, Suite 500, Washington, DC 20005, and/or to the SIP Forum, 733 Turnpike Street, Suite 192, North Andover, MA, 01845.

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 an optional capability that could augment the standard. The standard is fully functional without the incorporation of this optional capability.

The **ATIS/SIP Forum IP-NNI Task Force** under the **ATIS Packet Technologies and Systems Committee (PTSC)** and the **SIP Forum Technical Working Group (TWG)** was responsible for the development of this document.

Table of Contents

1 Introduction 1
2 Normative References 1
3 Definitions, Acronyms, & Abbreviations 1
3.1 Definitions 2
3.2 Acronyms & Abbreviations 2
4 Architecture 2
5 General API Requirements 4
5.1 Resource Structure 4
5.2 Special Request Header Requirements 5
5.3 Special Response Header Requirements 5
6 Data Types 5
6.1 Datatype: signingRequest 5
6.2 Datatype: origTelephoneNumber 6
6.3 Datatype: destTelephoneNumber 6
6.4 Datatype: signingResponse 6
6.5 Datatype: verificationRequest 7
6.6 Datatype: serviceException 7
6.7 Datatype: verificationResponse 7
6.8 Datatype: exception 8
6.9 Datatype: policyException 8
6.10 Datatype: requestError 8
7 Exceptions 8
7.1 RESTful WebServices Exceptions 8
7.2 Service Exceptions 9
7.3 Policy Exceptions 9
8 API Interface 10
8.1 Signing API 10
8.1.1 Functional Behavior 10
8.1.2 Call Flow 11
8.1.3 Request (POST) 11
8.1.4 Response 12
8.2 Verification API 13
8.2.1 Functional Behavior 13
8.2.2 Call Flow 14
8.2.3 Request (POST) 15
8.2.4 Response 15

Table of Figures

Figure 4.1 – SHAKEN Reference Architecture 3
Figure 4.2 – SHAKEN STI-AS/STI-VS with Centralized Signing & Signature Validation Server 4

ATIS Standard on –

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

1 Introduction

This technical report defines a Representational State Transfer (REST)ful interface that can be used in the Signature based Handling of Asserted information using toKENs (SHAKEN) framework to sign and verify telephony identity:

- Secure Telephone Identity Authentication Service (STI-AS) exposes an Applications Programming Interface (API) to sign the provided Personal Assertion Token (PASSporT) which includes the SHAKEN extension as defined in [draft-wendt-stir-passport-shaken].
- Secure Telephone Identity Verification Service (STI-VS) exposes an API to verify the signed Secure Telephone Identity (STI) according to procedures defined in IETF RFC 8225.

The only algorithm currently supported by this API is ES256.

The data set defined in this document could be expanded to accommodate other data types as needed (e.g., other PASSPort extensions that may need to be supported).

2 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.

[Ref 1] IETF RFC 8225, *PASSporT: Personal Assertion Token.*[1]

[Ref 2] PASSporT SHAKEN Extension.[2]

[Ref 3] IETF RFC 8224, *Authenticated Identity Management in the Session Initiation Protocol (SIP).*[3]

[Ref 4] ATIS-1000074, *Signature-based Handling of Asserted Information using toKENs (SHAKEN)*.[4]

[Ref 5] ATIS-1000080, *Signature-based Handling of Asserted information using toKENs (SHAKEN): Governance Model and Certificate Management.*[5]

3 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 >.

---

[1] This document is available from the Internet Engineering Task Force (IETF). < http://www.ietf.org >

[2] This document can be found at < https://datatracker.ietf.org/doc/draft-wendt-stir-passport-shaken/ >.

[3] This document is available from the Internet Engineering Task Force (IETF). < http://www.ietf.org >

[4] This document is available from the Alliance for Telecommunications Industry Solutions (ATIS) at < https://www.atis.org/docstore/product.aspx?id=28297 >.

[5] This document is available from the Alliance for Telecommunications Industry Solutions (ATIS) at < https://www.atis.org/docstore/product.aspx?id=28345 >

3.1 Definitions

Caller identity: The originating phone number included in call signaling used to identify the caller for call screening purposes. In some cases, this may be the Calling Line Identification or Public User Identity.

3.2 Acronyms & Abbreviations

API	Applications Programming Interface
CSCF	Call Session Control Function
PASSporT	Personal Assertion Token
HTTP	HyperText Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IMS	IP Multimedia Subsystem
IP-NNI	ATIS and SIP Forum IP Network-to-Network Joint Task Force
ISC	IMS Service Control
ITU	International Telecommunication Union
ITU-T	U.S. International Telecommunication Union Telecommunication Sector
ITU-R	U.S. ITU Radiocommunication Sector
JSON	JavaScript Object Notation
NNI	Network-to-Network Interoperability
PBXs	IP Private Branch Exchanges
PTSC	The Packet Technologies and Systems Committee
REST	Representational State Transfer
SHAKEN	Signature based Handling of Asserted information using toKENs
SIP	Session Initiation Protocol
SIPNOC	SIP Network Operators Conference
SKS	Secure Key Store
STI	Secure Telephone Identity
STI-AS	Secure Telephone Identity Authentication Service
STI-CR	Secure Telephone Identity Certificate Repository
STI-VS	Secure Telephone Identity Verification Service
STIR	Secure Telephone Identity Revisited
TWG	Technical Working Group
UTC	Coordinated Universal Time
UUID	Universally Unique Identifier
VRS	Video Relay Service

4 Architecture

Figure 4.1 depicts the SHAKEN reference architecture as described in Reference [4]. The reference architecture is based on the 3GPP IP Multimedia Subsystem (IMS) architecture, whereby the STI-AS and STI-VS are shown as IMS Application Servers, connecting to the IMS core Call Session Control Function (CSCF) via Session Initiation Protocol (SIP) IMS Service Control (ISC) interfaces.

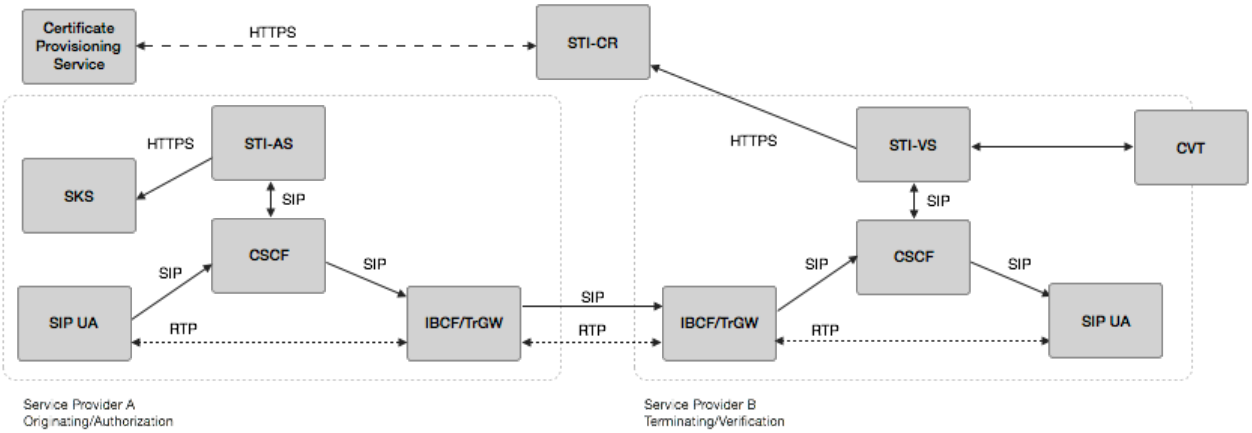


Figure 4.1 – SHAKEN Reference Architecture

As service providers incorporate SHAKEN into their infrastructure, they may need to deploy SHAKEN capabilities into multiple networks; some networks may be IMS-based, and some may not. Furthermore, service providers may determine that the STI-AS and/or STI-VS functions are better suited to be invoked at points other than the network core, such as at the network edge. The use of SIP as the STI-AS/STI-VS access protocol may not be suitable when initiating authentication and/or verification requests from locations other than an IMS core.

Because of the potential need for a service provider to initiate authentication and verification in multiple networks and/or from different network elements within their infrastructure, it would be beneficial to share a centralized authentication and verification service, calling upon these services from various points within a service provider's infrastructure.

This technical report describes a means of decomposing the STI-AS and STI-VS functions and exposing a HyperText Transfer Protocol (HTTP)-based RESTful API that can be used to request SHAKEN authentication and verification services. The API can be used by diverse network elements within a service provider's network to make SHAKEN authentication and verification requests of shared, centralized Signing and Signature Validation servers.

As shown in Figure 4.2, the overall STI-AS functionality is decomposed into two parts: a Signing server function and an authenticator function. Likewise, the STI-VS is decomposed into a Signature Validation server function and a verifier function. The HTTP-based API is used between the authenticator and Signing server functions and between the verifier and Signature Validation server functions. Figure 4.2 depicts a combined Signing and Signature Validation Server function, but this is optional.

The authenticator and verifier functions initiate the signing and validation requests via the API described in this document. The authenticator and verifier functions may be integrated into other network elements or may be developed as stand-alone functions. For example, a stand-alone authenticator function in an implementation of the SHAKEN reference architecture would receive SIP INVITE messages from the CSCF and formulate and send an HTTP signing request to the Signing server per the API described in this document. The Signing server performs the signing/authentication functions and formulates an API response containing an Identity header that the authenticator adds to the SIP INVITE message returned to the CSCF. Alternatively, the authenticator function could be integrated into the CSCF, whereby the CSCF would directly support the new API.

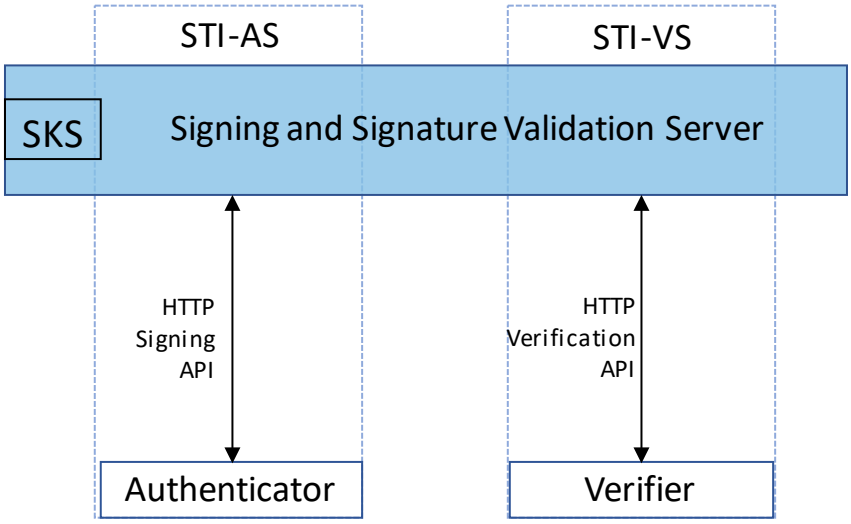


Figure 4.2 – SHAKEN STI-AS/STI-VS with Centralized Signing & Signature Validation Server

5 General API Requirements

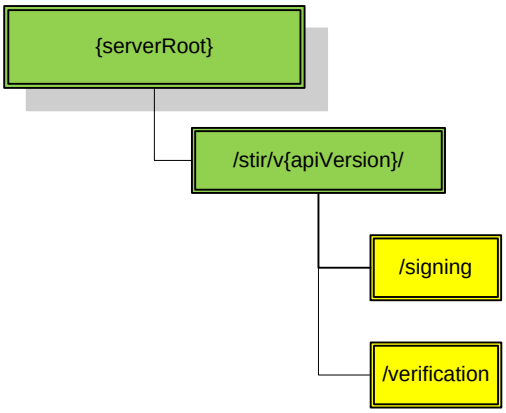
1. STI-AS and STI-VS have to expose RESTful web services implemented using HTTP and aligned with the principles of RESTful API.
2. Only JavaScript Object Notation (JSON)-based data format is supported. APIs use “application/json” content type.
3. All validations will be described below in the error handling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

5.1 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”.

5.2 Special Request Header Requirements

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

Header Name	Mandatory?	Description
X-RequestID	N	The **X-RequestID** transaction ID should be included in order to make possible the transaction traceability in case of troubleshooting and fault analysis. If received, it 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 ID will be returned back in the corresponding HTTP response in "X-RequestID" header.
X-InstanceID	N	For auditing purposes, each component calling the API should identify itself by sending its identity (e.g., 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 default handled as "**application/json**".

5.3 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 **X-RequestID** transaction ID will be returned back in the corresponding HTTP response.
Content-Type	Y	Determines the format of the response body. Valid value is: **"application/json"**.

6 Data Types

6.1 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 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 TNs.

Key Name	Key Value Type	Required?	Description
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 Coordinated Universal Time (UTC), 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).

6.2 Datatype: origTelephoneNumber

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

6.3 Datatype: destTelephoneNumber

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

6.4 Datatype: signingResponse

Key Name	Key Value Type	Required?	Description
identity	String Cannot be NULL	Y	Identity header value as defined in RFC 8224 with "identityDigest" in full format and mandatory "info" parameter. The "info" header field parameter contains the public key URL of the certificate used during STI signing.

6.5 Datatype: verificationRequest

Key Name	Key Value Type	Required?	Description
identity	String	Y	Identity header value as defined in RFC 8224 with "identityDigest" in full format and mandatory "info" parameter.
to	destTelephoneNumber	Y	Represents the called party. Array containing **one or more** identities of destination TNs. This is set to the value of the "To:" header field parameter in the incoming SIP Invite.
time	Integer	Y	This is 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 UTC, Thursday, 1 January 1970 not including leap seconds.
from	origTelephoneNumber	Y	Represents the asserted identity of the originator of the personal communications signaling. This is set to the value of the "P-Asserted-Identity", if available, or "From" header field parameter in the incoming Invite.

6.6 Datatype: serviceException

Field	Type	Required?	Description
serviceException	exception	Y	Service Exception.

6.7 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: 403, 428 (recommendation is to not use this Reason Code until a point where all calls on the VoIP network are mandated to be signed), 436, 437, 438.
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: 403 - "Stale Date" 428 - "Use Identity Header" (recommendation is to not use this Reason Text until a point where all calls on the VoIP network are mandated to be signed) 436 – "Bad Identity Info" 437 – "Unsupported Credential" 438 – "Invalid Identity Header"

Key Name	Key Value Type	Required?	Description
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 number passed the validation. **TN-Validation-Faile**d - The number failed the validation. **No-TN-Validation** - No number validation was performed.

6.8 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 the Open Mobile Alliance 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.

6.9 Datatype: policyException

Field	Type	Required?	Description
policyException	exception	Yes	Policy Exception.

6.10 Datatype: requestError

Field	Type	Required?	Description
requestError	policyException or serviceException	Yes	Request Error Message.

7 Exceptions

7.1 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). Two types of exceptions may be defined: service exceptions and policy exceptions.

7.2 Service Exceptions

When a service is not able to process a request, 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.

7.3 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 due to internal error.

8 API Interface

8.1 Signing API

8.1.1 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:

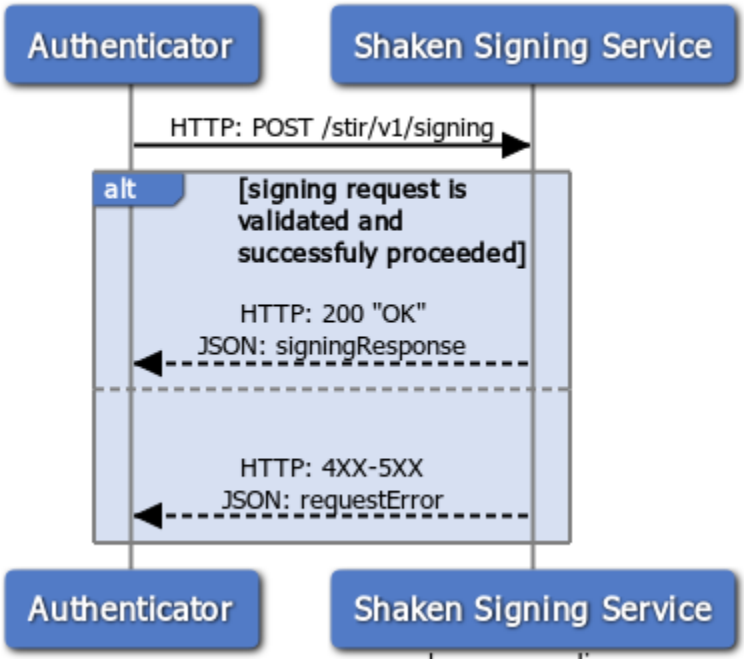
1. The “orig” parameter is populated using the PAI field if present, otherwise using the From header field in the SIP Invite.
2. The “dest” parameter is populated using the To header field in the SIP Invite.
3. The “iat” parameter is populated using the “Date” header field in the SIP Invite. If there 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 “+”).
4. Build SHAKEN PASSport protected JWT header (with “ppt” SHAKEN extension).
5. 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 used to acquire the public key of certificate used during PASSporT signing.
9. If successfully signed, build and send “signingResponse” 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.

8.1.2 Call Flow



8.1.3 Request (POST)

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

Name	Description
serverRoot	Server base URL: hostname+port+base path Hostname contains the Global FQDN of Signing Service.

8.1.3.1 Request Body

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

8.1.3.2 Request Sample

```
POST  /stir/v1/signing  HTTP/1.1
Host: stir.example.com
Accept: application/json
X-InstanceID : de305d54-75b4-431b-adb2-eb6b9e546014
X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
Content-Type: application/json
Content-Length: …
{
   "signingRequest": {
```

```
                "attest": “A”,
                "orig”: {
                        “tn”: “12155551212”
                    },
                 “dest”: {
                        “tn”: [
                                “12355551212”
                            ]
                        },
                 "iat”:  1443208345,
                 “origid”: “de305d54-75b4-431b-adb2-eb6b9e546014”
    }
}
```


8.1.4 Response

8.1.4.1 Response Body

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

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

8.1.4.2 Response Sample (Success)


```
HTTP/1.1 200 OK
X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
Content-Type: application/json
Content-Length: …

{
   "signingResponse": {
    "identity": “eyJhbGciOiJFUzI1NiIsInR5cCI6InBhc3Nwb3J0IiwicHB0Ijoic2hha2VuIiwieDV1IjoiaHR0cDov
L2NlcnQtYXV0aC5wb2Muc3lzLmNvbWNhc3QubmV0L2V4YW1wbGUuY2VydCJ9eyJhdHRlc3QiOiJBIiwiZGVzdC
I6eyJ0biI6IisxMjE1NTU1MTIxMyJ9LCJpYXQiOiIxNDcxMzc1NDE4Iiwib3JpZyI6eyJ0biI64oCdKzEyMTU1NTUxMj
EyIn0sIm9yaWdpZCI6IjEyM2U0NTY3LWU4OWItMTJkMy1hNDU2LTQyNjY1NTQ0MDAwMCJ9._28kAwRWnheX
yA6nY4MvmK5JKHZH9hSYkWI4g75mnq9Tj2lW4WPm0PlvudoGaj7wM5XujZUTb_3MA4modoDtCA;info=<https:/
/cert.example2.net/example.cert>”
  }
}
```


8.1.4.3 Response Sample (Failure)


```
HTTP/1.1 400 Bad Request
X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
Content-Type: application/json
Content-Length: …
{
  “requestError”: {
            “serviceException”: {
```

```
                                    "messageId": "SVC4001"
                                    "text": "Error: Missing mandatory parameter '%1'",
                                    "variables": ["iat"]
                                   }
                        }
}
```


8.1.4.4 HTTP Response Codes

Response code	Service/Policy Exception	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 due to internal signing server problem.

8.2 Verification API

8.2.1 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 verificationRequest 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 RFC7519 encoded Date header field from the SIP Invite.
4. The "identity" parameter value is populated using the Identity header field in the SIP Invite.
5. The Verifier then sends the HTTP Post to request verification.

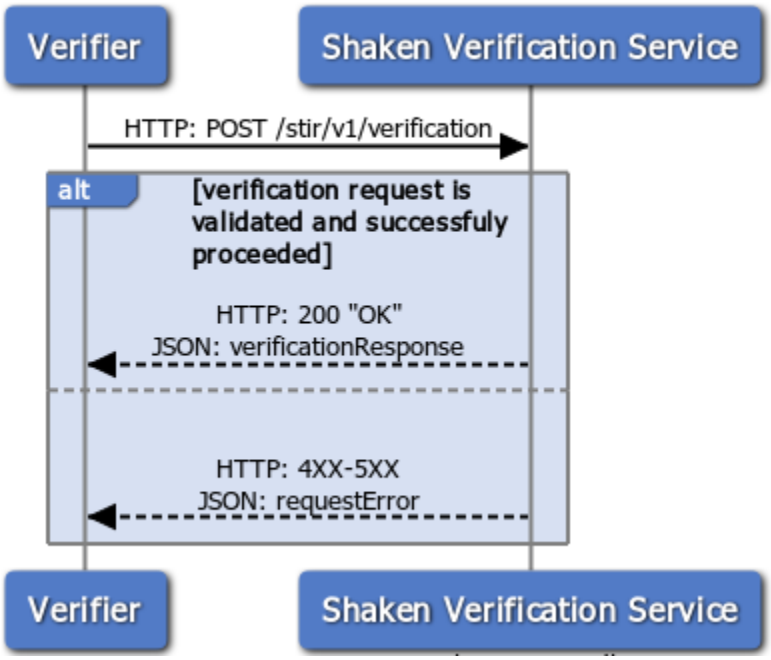
Upon receipt of the verificationRequest, the SHAKEN Verification Service performs the following steps. Each step is associated with the appropriate error case(s) specified in the section "Mapping of verification failure cases to the returned SIP header parameters". The error case numbers **En** per each 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:
    a. full form of PASSporT is required by SHAKEN: "identity-digest" parameter of Identity header has to be parsed to validate the full form format [three data portions delimited with dot (".")]. If the expected format is not matched → reject request on the Invalid PASSporT form (E4).

b. If “ppt” parameter is specified and its value is not “shaken” → reject request (E5).
c. If “info” parameter is not specified → reject request (E6).
d. 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:
   a. If one of the mentioned claims is missing -> reject request (E9).
   b. If extracted “typ” value is not equal to “passport” → reject request (E11).
   c. If extracted “alg” value is not equal to “ES256” → reject request (E12).
   d. If extracted “x5u” value is not equal to the URI specified in the “info” parameter of Identity header → reject request (E10).
   e. 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:
   a. On missing mandatory claims reject request (E14).
   b. 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).
   c. On invalid “attest” claim reject request (E19).
   d. 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 is 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).

8.2.2 Call Flow



8.2.3 Request (POST)

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

Name	Description
serverRoot	Server base URL: hostname+port+base path. Hostname contains the Global FQDN of Verification Service.

8.2.3.1 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).

8.2.3.2 Request Sample

```
POST  /stir/v1/verification  HTTP/1.1
Host: stir.example.com
Accept: application/json
X-InstanceID : de305d54-75b4-431b-adb2-eb6b9e546014
X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
Content-Type: application/json
Content-Length: …
{
  “verificationRequest”: {
                “from”: {
                        “tn”: “12155551212”
                    },
                 “to”: {
                        “tn”: [
                                “12355551212”
                            ]
                        },
                   “time”:  1443208345,
                   “identity”:
“eyJhbGciOiJFUzI1NiIsInR5cCI6InBhc3Nwb3J0IiwicHB0Ijoic2hha2VuIiwieDV1IjoiaHR0cDov
L2NlcnQtYXV0aC5wb2Muc3lzLmNvbWNhc3QubmV0L2V4YW1wbGUuY2VydCJ9eyJhdHRlc3QiOiJBIiwiZGVzdC
I6eyJ0biI6IisxMjE1NTU1MTIxMyJ9LCJpYXQiOiIxNDcxMzc1NDE4Iiwib3JpZyI6eyJ0biI64oCdKzEyMTU1NTUxMj
EyIn0sIm9yaWdpZCI6IjEyM2U0NTY3LWU4OWItMTJkMy1hNDU2LTQyNjY1NTQ0MDAwMCJ9._28kAwRWnheX
yA6nY4MvmK5JKHZH9hSYkWI4g75mnq9Tj2lW4WPm0PlvudoGaj7wM5XujZUTb_3MA4modoDtCA;info=<https:/
/cert.example2.net/example.cert>”
   }
}
```

8.2.4 Response

8.2.4.1 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.

8.2.4.2 Mapping of Verification Failure Cases to the Returned SIP Reason Header Field Parameters

Error Case Number	Error Case ("reasondesc")	HTTP Status Code	"reasoncode"	"reasontext"	"verstat"
E1	Missing mandatory parameters in the verification request ("from", "to", "time", "identity").	400 with service exception	-	-	No-TN-Validation
E2	Received invalid parameters (invalid "from"/"to" tn format, "time" 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 in compact form instead of required by SHAKEN spec full form.	200	438	Invalid Identity Header	No-TN-Validation
E5	Identity header is received with 'ppt' parameter value that is not 'shaken'.	200	438	Invalid Identity Header	No-TN-Validation
E6	Missing 'info' parameter in the 'identity'.	200	436	Bad identity Info	No-TN-Validation
E7	Invalid 'info' URI.	200	436	Bad identity Info	No-TN-Validation
E8	Failed to dereference 'info' URI.	200	436	Bad identity Info	No-TN-Validation
E9	Missing '%1' claim in the PASSporT header. %1 - "ppt", "typ", "alg", "x5u"	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	Invalid Identity Header	No-TN-Validation
E14	Missing '%1' mandatory claim in PASSporT payload %1 - "dest", "orig", "attest", "origid", "iat"	200	438	Invalid Identity Header	No-TN-Validation

Error Case Number	Error Case (“reasondesc”)	HTTP Status Code	“reasoncode”	“reasontext”	“verstat”
E15	‘iat’ from PASSporT payload is not fresh.	200	403	Stale Date	No-TN-Validation
E16	‘%1’ claim from PASSporT payload doesn’t match the received in the verification request claim. %1 - “orig”, “dest”	200	438	Invalid Identity Header	No-TN-Validation
E17	Failed to authenticate CA.	200	437	Unsupported credential	TN-Validation-Failed
E18	Signature validation failed.	200	438	Invalid Identity Header	TN-Validation-Failed
E19	‘attest’ claim in PASSporT payload is not valid.	200	438	Invalid Identity Header	No-TN-Validation

8.2.4.3 Response Sample (Success + Successful Validation)

```
HTTP/1.1 200 OK
X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
Content-Type: application/json
Content-Length: …
{
  "verificationResponse": {
            “verstat”: “TN-Validation-Passed”
    }
}
```

8.2.4.4 Response Sample (Success + Failed Validation)

```
HTTP/1.1 200 OK
X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
Content-Type: application/json
Content-Length: …

{
  "verificationResponse": {
           “reasoncode”: 436,
            “reasontext”: “Bad Identity Info”,
            “reasondesc”: “Invalid ‘info’ URI”,
            “verstat”: “No-TN-Validation”
    }
}
```

8.2.4.5 Response Sample (Failure)

```
HTTP/1.1 400 Bad Request
X-RequestID: AA97B177-9383-4934-8543-0F91A7A02836
Content-Type: application/json
Content-Length: …
```

```
{
  “requestError”: {
            “serviceException”: {
                                    “messageId”: “SVC4001”
                                    “text”: “Error: Missing mandatory parameter ‘%1’”,
                                    “variables”: [“iat”]
                                   }
                        }
}
```

8.2.4.6 HTTP Response Codes

Response code	Service/Policy Exception	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 due to internal signing server problem.