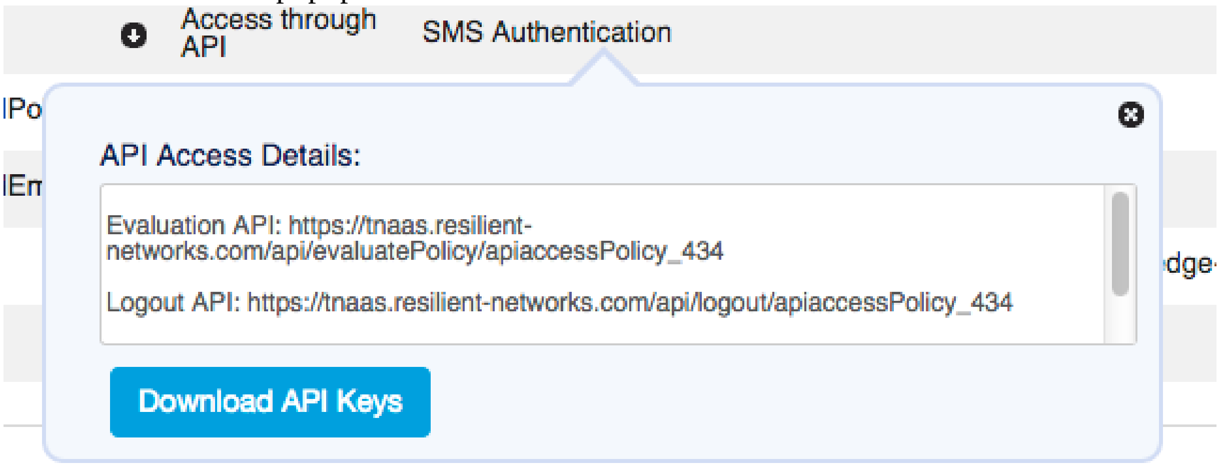
The Relying Party API is the programmatic way to integrate your applications with Resilient Access. Using the Relying Party API your application is in control of the interaction with the Resilient Access policy evaluation engine and your application provides the user interfaces for your users to interact with your applications.

## Create the API Access Policy

To build your Resilient Access Relying Party application, you should first create a policy in Resilient Access as described below:

1. Log into the Resilient Access admin console
2. Create a policy in Resilient Access and select **Access API** for **Policy Used For**
3. Click on the ⊕ button for the policy you created in the **Policy Used For** column of the policies list. You will see the popup for **API Access Details** as shown below:



Evaluation API: https://tnaas.resilient-networks.com/api/evaluatePolicy/apiaccessPolicy_434

Logout API: https://tnaas.resilient-networks.com/api/logout/apiaccessPolicy_434

4. Resilient Access generates a per policy API Key and a per organization (tenant) private-public key pairs (one for request and one for response) for securely using the Relying Party API. The section below describes the use of the API keys. From the **Policy Used For** popup the request private key and response public key can be downloaded.

## Use of API Keys

For secure interaction between the Relying Party and Resilient Access, the API makes use of API keys. There are three keys provided by Resilient Access when an API policy is created. These are:

- Policy API Key: This key is the API identifier of the Policy the Relying Party API is being invoked for. This key must be passed in the *X-API-KEY* request header parameter. If this key is not present or is not the correct value provided for the policy then the request will be rejected.
- Request Private Key: The private key of a RSA Key Pair. This key is used by the Relying Party to create a digital signature of the request body payload. It is optional, but if present it must be generated as follows:

1. Generate a SHA-256 hash of the request body
2. Base 64 encode the SHA-256 hash
3. Encrypt the value generated at step 2 using the request private key
4. This value should be set to the *X-SIGNATURE* request header.

The Relying Party API will verify the signature value passed matches the request body by generating a Base 64 encoded SHA-256 hash of the post body and comparing it with the *X-SIGNATURE* value decrypted with the Request public key. If they do not match then the request will be rejected. This ensures that the request from the Relying Party has not been tampered by a man in the middle.

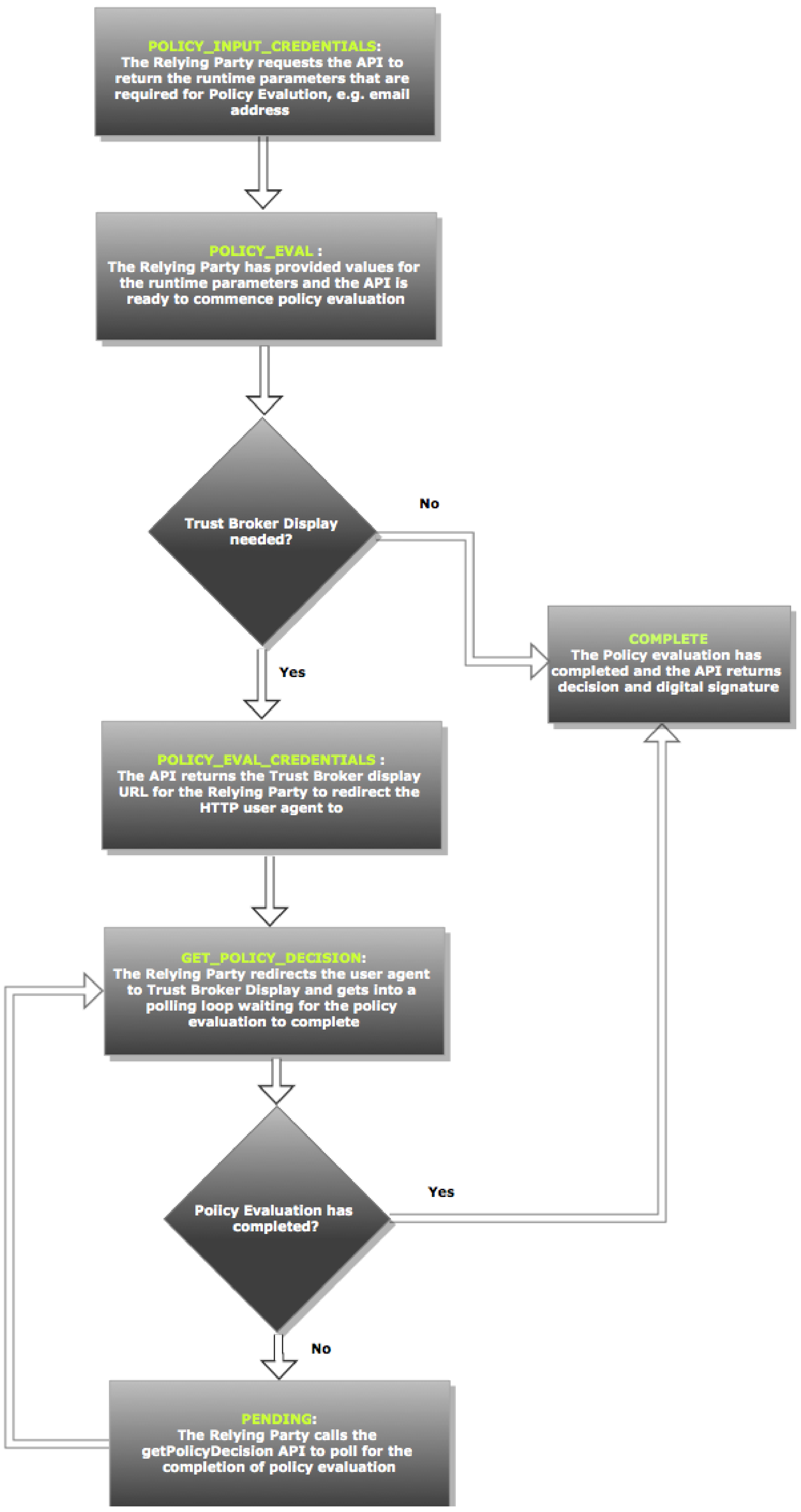
- Response Public Key: The public key of a RSA Key Pair. Resilient Access will always generate an *X-SIGNATURE* response header using a similar process as described above. The Base 64 encoded SHA-256 value of the response body is encrypted with the Response Private Key and this value is set to the *X-SIGNATURE* response header. The Relying Party can then generate a Base 64 encoded SHA-256 hash of the response body and compare it with the *X-SIGNATURE* value decrypted with the Response Public Key to ensure there is no tampering of the response from TNaaS Relying Party API

## evaluatePolicy API

### Policy Evaluation State Diagram

The following diagram shows the Policy Evaluation State Diagram

### Policy Evaluation State Machine


POLICY_INPUT_CREDENTIALS: The Relying Party requests the API to return the runtime parameters that are required for Policy Evalution, e.g. email address
POLICY_EVAL : The Relying Party has provided values for the runtime parameters and the API is ready to commence policy evaluation
Trust Broker Display needed?
No
Yes
COMPLETE The Policy evaluation has completed and the API returns decision and digital signature
POLICY_EVAL_CREDENTIALS : The API returns the Trust Broker display URL for the Relying Party to redirect the HTTP user agent to
GET_POLICY_DECISION: The Relying Party redirects the user agent to Trust Broker Display and gets into a polling loop waiting for the policy evaluation to complete
Policy Evaluation has completed?
Yes
No
PENDING: The Relying Party calls the getPolicyDecision API to poll for the completion of policy evaluation

## Relying Party API Request/Response

#### Policy Input Credentials

This is the initial request from the Relying Party to get the policy input parameters.

```
API: https://tnaas.resilient-networks.com/api/evaluatePolicy/
HTTP Method: POST
Request Headers:
    Accept: application/json

    Content-Type: application/json
    X-API-KEY: <The API key returned by TNaaS>
    X-SIGNATURE:  <MD5 digest of request body encrypted with REQUEST_P
RIVATE_KEY> (optional)
Request Body:
{
    state: POLICY_INPUT_CREDENTIALS
}

Response Headers:
    Accept: application/json
    Content-Type: application/json
    X-SIGNATURE: <MD5 digest of response body encrypted with RESPONSE_
PRIVATE_KEY>

Response Body:
{
    'state': 'POLICY_INPUT_CREDENTIALS',
    'contextID': '<context GUID>',
    'policyParameters': [
      {
         'name': '<input param name>',
         'displayName': '<input param display name>',
         'type': 'text' | 'password'
      },
      { ... }
   ]
}
```

#### Evaluate Policy

Relying Party should build an HTML form requesting user to enter policy input parameters and collects the user's input and then calls this API to start the policy evaluation.

```
API: https://tnaas.resilient-
networks.com/api/evaluatePolicy/<policyName>
HTTP Method: POST
Request Headers:
    Accept: application/json
    Content-Type: application/json
    X-API-KEY: <The API key returned by TNaaS>
    X-SIGNATURE: <MD5 digest of request body encrypted with REQUEST_PR
IVATE_KEY>
Request Body:
{
    'sessionID': '<session ID GUID>', (If the relying party has it fro
m a previous session with the API)
    'contextID': '<context GUID>',
    'state': 'POLICY_EVAL',
    'parameters' {
       '<param_name>': '<param_value',
       ...
    }
}

Response Headers:
    Accept: application/json
    Content-Type: application/json
    X-SIGNATURE: <MD5 digest of response body encrypted with RESPONSE_
PRIVATE_KEY>

Response Body:
{
    'contextID': <context GUID>
    'state': 'POLICY_EVAL_CREDENTIALS' | 'COMPLETE',
    'redirectURL': '<The Authority Credentials URL if state is POLICY_
EVAL_CREDENTIALS>',
    'timeout': '<DateTime in milliseconds from epoch when Relying Part
y should stop polling and generate an error message>', (if state=POLIC
Y_EVAL_CREDENTIAL)
    'decision': 'GRANT' | 'DENY' | 'ERROR', (if state=COMPLETE)
    'message': <Deny message configured in TNaaS for DENY, error messa
ge returned by policy evaluation if ERROR> (if state=COMPLETE)
    'sessionID': '<session ID GUID>', (if state=COMPLETE)
    'expiration': <DateTime in milliseconds from epoch> (if state=COMP
LETE)
}
```

**Get Policy Decision**

If the state=*POLICY_EVAL_CREDENTIALS* then Relying Party should call this API in a loop polling for the end of policy evaluation.

```
API: https://tnaas.resilient-
networks.com/api/evaluatePolicy/<policyName>
HTTP Method: POST
Request Headers:
    Accept: application/json
    Content-Type: application/json
    X-API-KEY: <The API key returned by TNaaS>
    X-SIGNATURE: <MD5 digest of request body encrypted with REQUEST_PR
IVATE_KEY>
Request Body:
{
    'contextID': '<context GUID>',
    'state': 'GET_POLICY_DECISION
}

Response Headers:
    Accept: application/json
    Content-Type: application/json
    X-SIGNATURE: <MD5 digest of response body encrypted with RESPONSE_
PRIVATE_KEY>
Response Body:
{
  'state': 'PENDING' | 'COMPLETE',
  'contextID': '<context GUID>',
  'decision': 'GRANT' | 'DENY' | 'ERROR' (if state=COMPLETE)
  'message': <Deny message configured in TNaaS for DENY, error message
 returned by policy evaluation if ERROR> (if state=COMPLETE)
  'sessionID': <session ID GUID> (if state=COMPLETE)
  'expiration': <DateTime in milliseconds from epoch> (if state=COMPLE
TE)
}
```

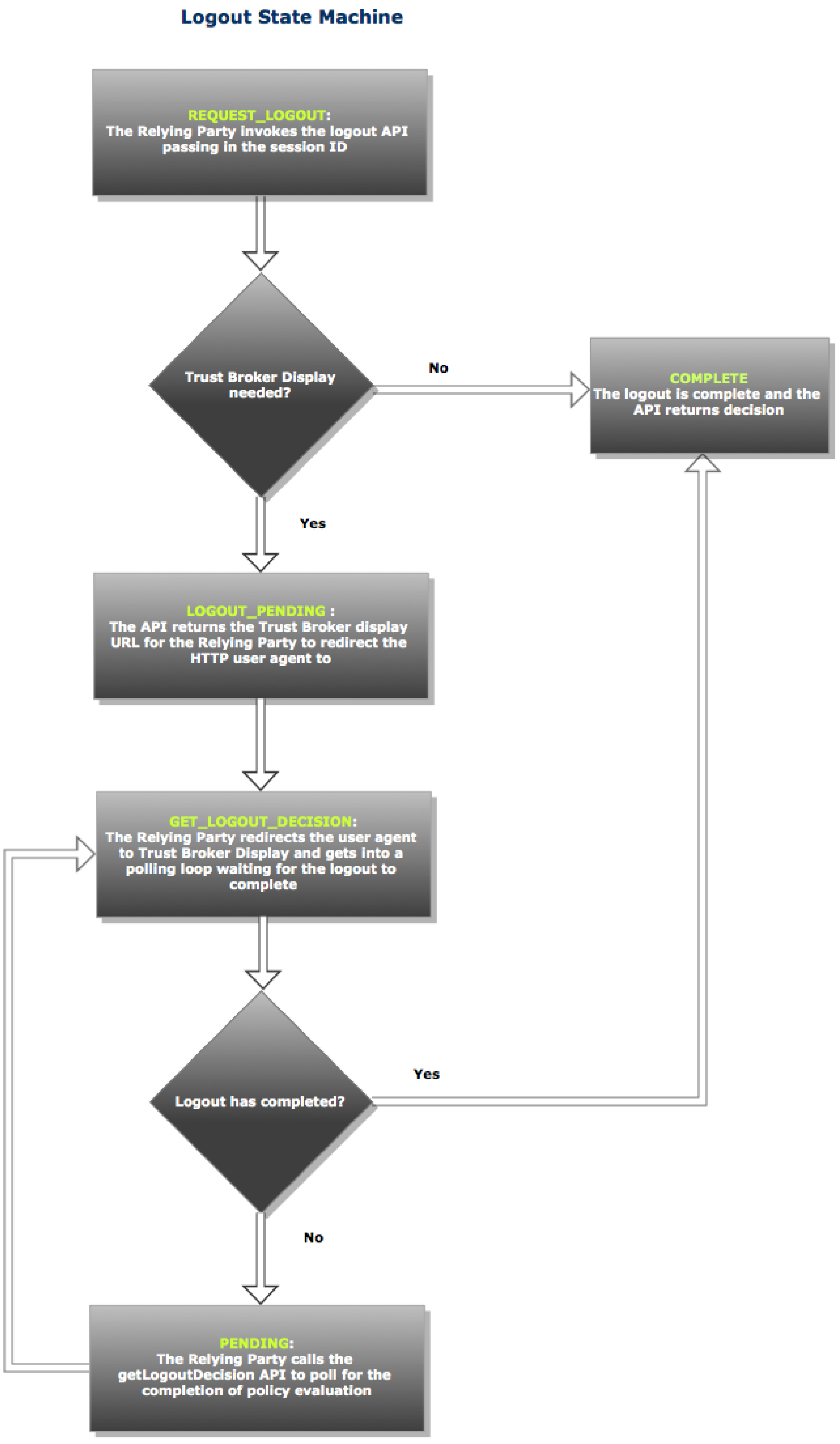
**Protocol Errors**

- Bad Credentials
  - The Relying Party has not sent or sent incorrect X-API-KEY or X-SIGNATURE headers
  - HTTP Status code: 401, response body will have decision=ERROR and error message
- Bad Request Body
  - The Relying Party has passed an incorrect 'state' attribute or missing data for the state

    - attribute, e.g. no "parameters" for state=POLICY_EVAL
    - The contextID is missing or not valid
    - HTTP Status code: 400, response body will have decision=ERROR and error message
- Deny Evaluation Result
    - HTTP Status code: 401, response body will have decision=DENY and message set to the deny message configured in TNaaS
- Trust Network evaluation error
    - Internal error occurred during policy evaluation
    - HTTP Status code: 500, response body will have decision=ERROR and exception message generated by trust network component

## logout API

### logout state diagram


Logout State Machine
REQUEST_LOGOUT: The Relying Party invokes the logout API passing in the session ID
Trust Broker Display needed?
No
COMPLETE The logout is complete and the API returns decision
Yes
LOGOUT_PENDING : The API returns the Trust Broker display URL for the Relying Party to redirect the HTTP user agent to
GET_LOGOUT_DECISION: The Relying Party redirects the user agent to Trust Broker Display and gets into a polling loop waiting for the logout to complete
Logout has completed?
Yes
No
PENDING: The Relying Party calls the getLogoutDecision API to poll for the completion of policy evaluation

## Logout API Request/Response

### Request Logout

```
API: https://tnaas.resilient-networks.com/api/logout/<policyName>
HTTP Method: POST
Request Headers:
    Accept: application/json
    Content-Type: application/json
    X-API-KEY: <The API key returned by TNaaS>
    X-SIGNATURE: <MD5 digest of request body encrypted with REQUEST_PR
IVATE_KEY>
Request Body:
{
    'sessionID': <session ID GUID from successful policy evaluation>,
    'state': 'REQUEST_LOGOUT'
}

Response Headers:
    Accept: application/json
    Content-Type: application/json
    X-SIGNATURE: <MD5 digest of response body encrypted with RESPONSE_
PRIVATE_KEY>
Response Body:
{
  'state': 'LOGOUT_PENDING' | 'COMPLETE',
  'contextID': '<context GUID>',
  'redirectURL': '<Trust Broker display URL to redirect browser to>' (
if state=LOGOUT_PENDING),
  'timeout': <DateTime in milliseconds from epoch when Relying Party s
hould stop polling and generate an error message> (if state=LOGOUT_PEN
DING)
  'decision': 'SUCCESS' | 'ERROR' (if state=COMPLETE)
  'message': <error message returned by policy evaluation if ERROR> (i
f state=COMPLETE)
}
```

### Get Logout Decision

```
API: https://tnaas.resilient-networks.com/api/logout/<policyName>
HTTP Method: POST
Request Headers:
    Accept: application/json
    Content-Type: application/json
    X_API_KEY: <The API key returned by TNaaS>
    X-SIGNATURE: <MD5 digest of request body encrypted with REQUEST_PR
IVATE_KEY>
Request Body:
{
    'state': 'GET_LOGOUT_DECISION',
    'contextID': '<context GUID>'
}

Response Headers:
    Accept: application/json
    Content-Type: application/json
    X-SIGNATURE: <MD5 digest of response body encrypted with RESPONSE_
PRIVATE_KEY>
Response Body:
{
  'state': 'PENDING' | 'COMPLETE',
  'decision': 'SUCCESS' | 'ERROR' (if state=COMPLETE)
  'message': <error message returned by policy evaluation if ERROR> (i
f state=COMPLETE)
}
```