A REST service that provides an authentication or authorization function can integrate with Resilient Access (RA) by implementing the following API to be part of a RA policy workflow. This API will allow the REST service to authenticate itself with RA and then execute an access control operation that might include interactions with the users.

This API uses the OAuth 2.0 JWT Bearer Token specification, an emerging standard for secure API access authentication that is gaining widespread adoption. This link from Salesforce documentation site has good explanation and sample code in Java for implementing this authentication flow.

/token

The REST service must implement the /token API as per the OAuth 2.0 JWT Bearer Token Flow and issue an access token. RA will send this access token in all subsequent interactions with the REST service. The REST service must return a forbidden response if the access token is not passed in a Authorization HTTP header parameter.

```
URL: /token
Method: POST
Consumes: application/x-www-form-urlencoded
Produces: application/json
Parameters:
    type: body
    name: client_id
    value: the OAuth 2.0 client ID issued by the Custom REST Authority
 configuration

    type: body
    name: client_secret
    value: the OAuth 2.0 client secret issued by the Custom REST Autho
rity configuration

    type: body
    name: grant_type
    value: urn:ietf:params:oauth:grant-type:jwt-bearer

    type: body:
    name: assertion
    value: assertion JWT, see below for the format of this assertion:

Responses:
    200: { "access_token":""}
    403,400: { "access_token": "ERROR_<error_code>", "message": "" }

Assertion JWT:
Payload:
```

```
{
    "iss": ,
    "sub":  - A type-4 UUID generated by RA that will identify the int
eraction between RA and the REST service
    "aud": "/token",
    "jti": A type-4 UUID
    "exp": The time in seconds when the token expires, should be limit
ed to 1 minute from current time.
}

Header:
{
    "alg": "RS256",
    "typ": "JWT",
    "kid": ""
}

Signature: JWT signed with the public key of the RSA key-
pair issued to the REST service.
```

/evaluate

The evaluate API performs the service operation. It can interact with the user by returning the "result": "DISPLAY_REQUEST" JSON attribute.

```
URL: /evaluate
Method: POST
Consumes: application/json
Produces: application/json
Parameters:
    type: header
    name: Authorization
    value: "Bearer "

Request Body:
    {
        "requestId":
        "context": {
            "param1": "value1",
            "param2": "value2"
        }
        "config": {
            "param1": "value1",
            "param2": "value2"
```

```
        }
    }

Response:
200:
{
    "requestId": type-4 UUID that was passed in the request body
    "result": "GRANT" || "DENY" || "DISPLAY_REQUEST" || "ERROR"
    // If result is "ERROR", an optional error message
    "error":
    // If "result" is "GRANT", the REST service can optionally return
claims in an "assertion" object
    "assertions": {
        "claim1": "value1",
        "claim2": "value2"
    }
    // If "result" is "DISPLAY_REQUEST" then REST service should retur
n a "display" object. Please see below for the appearance of the user
dialog.
    "display": {
        "title":
        "instructionText":
        "errorText":
        "footerText":
        // The "display" object must contain one or more "item" object
s in the "items" array. Each "item" object typically translates to a H
TML form element for a browser user-
agent. An example of each type is provided.
        "items": [
            // text input field
            {
                "type": "text"
                "name": <form element name>
                "label": <label>
            },
            // HTML5 "number" input type, mobile browsers will display
 the numeric keypad
            {
                "type": "number"
                "name": <form element name>
                "label": <label>
             },
             // HTML5 "tel" input type, mobile browsers will display t
he phone number input keypad
             {
                 "type": "tel"
```

```
                "name": <form element name>
                "label": <label>
            },
            // HTML5 "email" input type, mobile browsers will display
 the email input keypad
            {
                "type": "email"
                "name": <form element name>
                "label": <label>
            },
            // password input field, hide the text entered in the inp
ut field.
            {
                "type": "password"
                "name": <form element name>
                "label": <label>
            },
            // displays a static field with the label and value
            {
                "type": "static"
                "name": <form element name>
                "label": <label>
                "value": <value>
             },
             // displays a text area field that can have a consent, a
greement, etc
             {
                 "type": "textarea"
                 "name": <form element name>
                 "label": <label>
                 "value": <value> // can embed HTML
             },
             // displays an HTML select or equivalent input field.
             {
                 "type": "dropdown"
                 "name": <form element name>
                 "label": <label>
                 "options": [
                     {
                         "value": <option value>
                         "label": <option label>
                     }
                 ]
             },
             // displays a set of radio button options
             {
```

```
                    "type": "radio"
                    "name": <form element name>
                    "label": <label>
                    "options": [
                        {
                            "value": <option value>
                            "label": <option label>
                         }
                     ]
                },
                // a group of check boxes
                {
                    "type": "checkbox"
                    "name": <form element name>
                    "label": <label>
                    "options": [
                        {
                            "name": <checkbox name>
                            "value": <checkbox value>
                            "label": <checkbox label>
                        }
                    ]
                },
                // displays a hidden field for passing state etc
                {
                    "type": "hidden"
                    "name": <form element name>
                    "value": <value>
                },
            ]
        }
}

400, 403: {
    "error_code": error code
    "error": error message
}
```

The user interaction dialogs are modelled around HTML forms. A REST service can have a series of these interactions as part of an operation. In a multi-step interaction each step represents a form submission and the data gathered in a form submission is passed back to the REST service for the next call to the /evaluate API in the "context" object. The "name" of the "item" object specifies the context

parameter that will hold the value of the field submitted. A "hidden" item can be useful for maintaining state across a multi-step interaction.

The following diagram shows the layout of the data sent by the REST service in the "display" JSON object in the user interaction dialog.