Resilient Access Admin Console User's Guide

Version 3.4.0

Resilient Network Systems

Table Of Contents

Resilient Access

The Vision

Resilient Access virtualizes real-world relationships and conditions of trust, resolve identities in the network, and enforce each party's policies. This enables disparate organizations and users to share sensitive information and applications while maintaining control and privacy. Enterprises, government agencies, and online platforms leverage Resilient Access to collaborate and share information with their ecosystems of partners and customers.

Resilient Access Policy Evaluation

We have taken the first steps in realizing that vision by building a dynamic distributed policy evaluation engine. The core tenets of this policy evaluation engine is there is no one party that acts as the gate keeper making the final policy decision like in the federated model. Policy evaluation is "syndicated" across the different parties that are involved in making the policy decisions.

Syndication and dynamic Policy evaluation

Resilient Access policies are logical expressions that reference one or more Authorities. An authority performs an evaluation function that in the basic case generates an GRANT or DENY decision. A policy evaluates to a GRANT if the Authorities that are part of the policy expression evaluates to grant. For example the Policy:
EmailAuthentication AND PhoneAuthentication
will evaluate to a GRANT if both the EmailAuthentication and PhoneAuthentication authorities evaluate to a GRANT.
An Authority can also return a third state and that is another Policy. This is the mechanism through which dynamic policy evaluation takes place that can address syndication and cross-organizational policy evaluation use cases. An use case that illustrates this is of a doctor attempting to attain the medical record for a patient. Lets assume there is a syndicate consisting of medical institutions, pharmacies and insurance companies. Doctors are associated with medical institutions, patients are associated with a medical institution and an insurance company. The syndicate uses the Resilient Access for managing the policies that would grant access to sensitive medical records. The syndicate operator may create a high level policy such as IsDoctor(DoctorID) AND IsPatient(PatientID) to facilitate the access of a patient's medical record by a doctor. Lets assume the DoctorID identifies the medical institution the doctor is associated with and the PatientID identifies the insurance company the patient is a member of. The syndicate operator will then forward the policy evaluation to the medical institution for the doctor and the insurance company for the patient. Each medical institution and insurance company that is part of the syndicate can define their own policy for validating that the doctor or patient is a member of their organization.

Privacy enhancing features

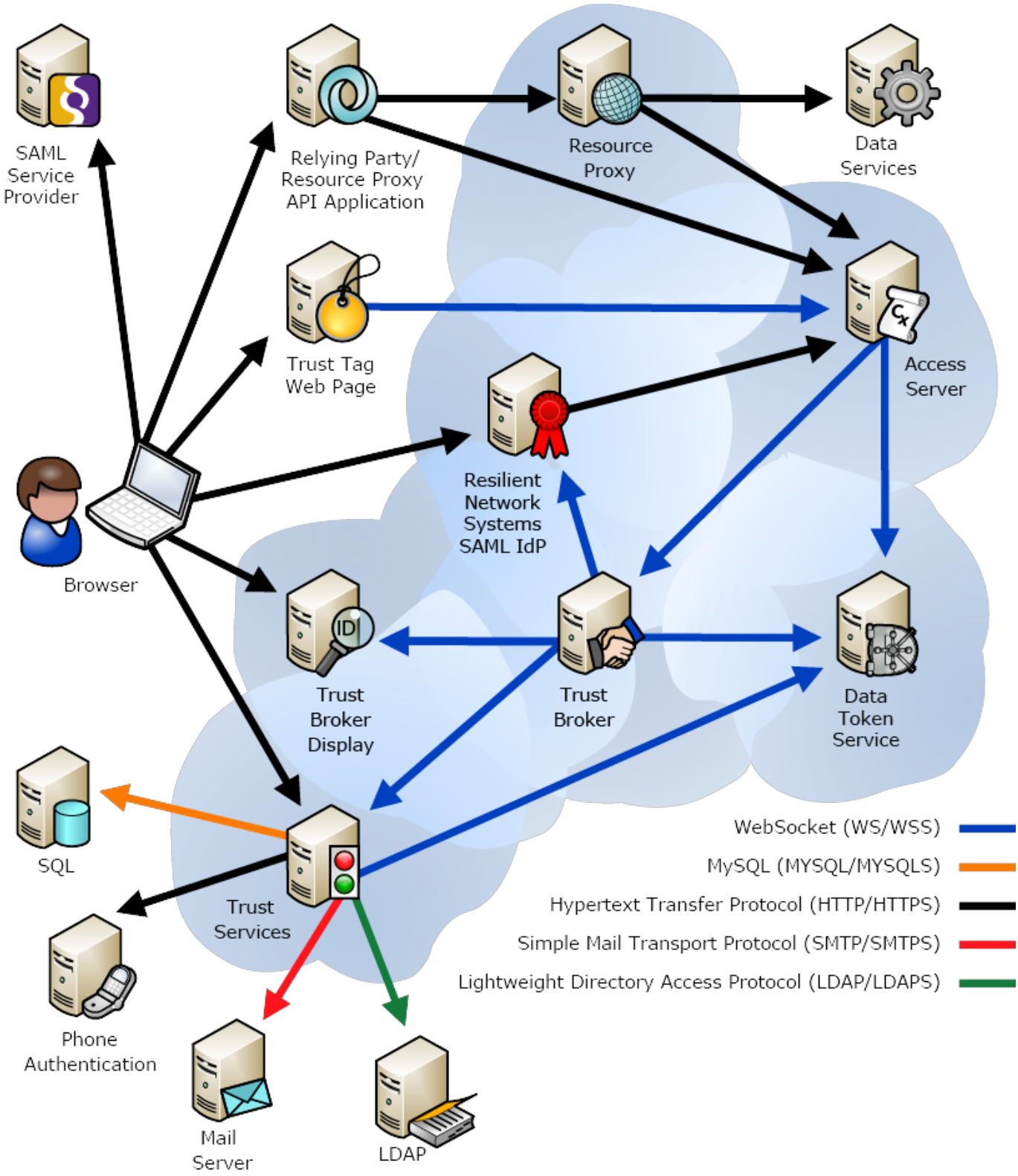
Policy evaluation in Resilient Access contains various privacy enhancing features. Policies can be crafted such that sensitive information does not flow through the network. The only party that would request for a sensitive data item would be the authority that requires the particular data item in order to perform its

policy evaluation tasks. An example of this is an LDAP Authentication Authority that need an UserID and a Password to perform its evaluation. The sensitive data item in this case is the password. The policy can be created such that this data item is not requested from the user when the policy evaluation commences. Only when the evaluation sequences arrives at the LDAP Authentication Authority, it will ask for the password through a Display Request that is posted to the user.

The other mechanism for privacy enhancement is through data obfuscation, or Zero Knowledge. In this case if sensitive personal identity information (PII) needs to flow through the network as part of a policy evaluation, policy can be crafted such that the PII data item is converted into an opaque token with the rules that the only party that can de-tokenize it is the Authority that is specified when the token is created. An example of this is say one Authority receives an email address, performs a database lookup and retrieves a SSN that need to be sent to another authority that can perform an evaluation based on the SSN. Instead of passing the sensitive info through Resilient Access un-encrypted the first authority can use a Data Token Service to obfuscate the SSN and identify the second authority as the one that can get the un-obfuscated value.

Resilient Access Components

The following diagram shows the components of Resilient Access and the relationship among them.



The Policy Workflow Engine

The Policy Workflow Engine is the orchestrator of the policy evaluation. It receives the policy expression from the Policy/Authority Store and routes the evaluation to the authorities. It uses the Trust Broker

Display to post the display request screens for additional credentials required by the policy. At the end of the evaluation it returns the result to the Access Server

Policy/Authority Store

The policies are configured in the Access Server. The Policy/Authority Store provides a Relying Party API for external services to connect to Resilient Access. Resilient provides several implementations of the Relying Party including Trust Tag, Data Proxy and SAML Identity Provider.

Authority Credentials

Authority Credentials act as the agent of the Policy Workflow Engine that interacts with the Browser. It manages the Trust Session that maintains the information of granted policies. Authority Credentials also redirects the browser to the Authorities Display Request screen as well as back to the Access Server at the end of policy evaluation.

Data Token Service

The Data Token Service provides an API for obfuscating data items into tokens. At the time the token is created the address of the service that can de-tokenize the data is also provided to the API. Only the service that has the permission to de-tokenize can retrieve the data value.

Authorities

Authorities perform an evaluation function and returns GRANT or DENY or another policy. If an authorities requires additional credentials then it sends a DISPLAY_REQUEST response to the Trust Broker for it to initiate a Display Request sequence through the Trust Broker. Resilient provides various types of authorities. These fall under the following broad categories:

- LDAP/AD Based Authorities - connects to an LDAP/AD and verifies identify such as password match (e.g. Password Authentication) or compares attributes retrieved from the LDAP/AD with expected values (e.g. Role Authorization)
- Database Authorities - connects to a Database and executes the configured SQL query for identity verification (e.g. Database Authentication) or compares data retrieved from the Database with expected values (e.g. Data lookups)
- User Created Authorities - these authorities integrate to services provided by third party identity or authentication service vendors. This category can also be used to build custom authorities that perform identity and authentication functions specific to an organization
- Policy Authorities - facilitates dynamic policy evaluation by retrieving attributes from LDAP/AD, Database or custom APIs and substituting them into the parameters of another policy. E.g. retrieve a phone number from an LDAP/AD and insert it into a Phone Authentication output policy

Registration and Sign In

About signing up

To sign up to use the Administrative Console, you must set a password and provide the following information.

- *Your corporate email address*—used to filter policies and authorities. Users with the same domain name in their email address can see the policies and authorities added by other users with the same domain name. Users with different domain names cannot see each others policies or authorities.
- *Your name*—used to personalize Administrative Console messages.
- *The name of your organization*—used to personalize the Administrative Console display.
- *Your phone number*—used to recover from a forgotten password.

Signing up

1. Visit https://tnaas.resilient-networks.com.
2. Click **Click Here to Sign Up**.
3. Type your corporate email address in the **Email** box.
4. Type a password in the **Password** and **Confirm Password** boxes.
5. Type your name in the **Name** box.
6. Type the name of your organization in the **Organization Name** box.
7. Type your phone number in the **Phone Number** box.
8. Click **Sign Up**.
9. Without closing the browser, check the email account of the address you provided in the **Email** box.
10. Open the email sent by rnsgateway@resilient-networks.com.
11. Memorize, copy, or write down the four-digit code in the email.
12. Return to the original browser tab and enter the four-digit code from the email into the **Authority Credentials** form.
13. Click **Submit**.

NOTE: If the sign up process fails to complete, see Recovering from an incomplete sign up.

Signing in

1. Visit https://tnaas.resilient-networks.com.
2. Type the email address of the account you provided during sign up in the **Email** box.
3. Type the password that you set during sign up in the **Password** box.
4. Click **Sign In**.

Recovering from a forgotten password

1. If you forgot your password, click **Trouble signing in**.

2. Then click **Forgot password**.
3. Type the email address of the account you provided during sign up in the **Email** box.
4. Click **Reset Password**.
5. Without closing the browser, check the email account of the address you provided in the **Email** box.
6. Open the email sent by rnsgateway@resilient-networks.com.
7. Copy the four-letter code in the email.
8. Return to the original browser tab and paste the four-letter code from the email into the **Authority Credentials** form.
9. Next you will see a **Phone Authentication Authority Credentials** form, select the radio button beside the phone number and click **Call**. (The phone number is the one you had entered during Sign Up)
10. Answer the phone and type 1 into the phone keypad when prompted.
11. Once the recorded voice has stopped talking, type the code displayed in the **Authority Credentials** form into the phone.
    NOTE: do not type the code until the recorded voice has finished speaking.
12. Type a new password into the **New Password** and **Confirm New Password** boxes.
13. Click **Reset Password**.

Recovering from an incomplete sign up

1. Click **Trouble signing in**.
2. Click **Did not finish signing up**.
3. Type the email you started signing up with in the **Email** box.
4. Type your name in the **Name** box.
   NOTE: this does not have to match the name you used during the original sign up.
5. Click **Complete sign up**.
6. Without closing the browser, check the email account of the address you provided in the **Email** box.
7. Open the email sent by rnsgateway@resilient-networks.com.
8. Copy the four-letter code in the email.
9. Return to the original browser tab and paste the four-letter code from the email into the **Authority Credentials** form.
10. Click **Submit**.

Create Policies

Resilient Access provides an intuitive interface for creating policies. Authorities are displayed in a panel on the left. Resilient provides a set of out of the Box authorities that are well suited for additional factors of authentication and authorization. This list currently consists of the following authorities

- **Email Authentication:** Ensures that the user can access the account of the email address that they provide. The Email Authentication Authority sends an email to the specified email address. Inside the email is a code. Upon receiving the email, the user must type the code into the web form.

- **SMS Authentication:** Provides out-of-band authentication. Ensures that the end user is in possession of a mobile phone and can receive an SMS message. After receiving the SMS from the SMS Authentication Authority, the end user is prompted to type the code in the Trust Broker Display screen.

- **Phone Authentication:** Provides out-of-band authentication. Ensures that the end user is in possession of a phone and can answer a specific phone number. After receiving a call from the Phone Authentication Authority, the end user types the code displayed on screen into the phone.**Voice Authentication:** Provides out-of-band authentication. Ensures that the end user is in possession of a phone and their voice matches the voice imprint stored in the Voice Authentication service . The first time, the user receives a phone call prompting to record their voice imprint by speaking out a series of authorization code values. The user then receives a second call to verify their voice imprint by speaking out the authorization code(s) displayed on the screen. With the voice imprint in file, Voice Authentication involves a single phone call and speaking out the authorization code(s) displayed on the screen. The system will prompt for up to 4 authorization codes to determine a voice imprint match.

- **Google Authenticator:** Provides an option to add a popular TOTP (Time-based One Time Password) authentication factor to a policy. This authentication displays a 6 digit number in a smart phone app that changes every 30 seconds. The user is prompted to enter the 6 digit number corresponding to the account they are trying to access in the *Authority Credentials* popup. A pre-requisite is to install the Google Authenticator smart phone app. There are 2 authorities, a provisioning authority *Google Authenticator Provisioning*, that should be part of a registration/provisioning policy. A QR code will be displayed in the *Authority Credentials* popup, the user should put their Google Authenticator app in 'Scan Barcode' mode and scan the QR code to create an account in the app. The 2nd authority is the *Google Authenticator* authentication authority that should be part of an authentication policy. This authority prompts the user to enter the 6 digit code corresponding to the provisioned account the user is attempting to authenticate into.

- **Acceptto Mobile Authenticator:** Provides an option to use a smart phone as an authentication device. There are 2 authorities, a provisioning authority *Acceptto Provisioning*, that should be part of a registration/provisioning policy. This authority provisions a user account in the Acceptto service. The user info required for provisioning is a user's name, verified email address, verified phone number, password and 2 security question and answers. Assuming the provisioning policy

executes on the submission of a user registration form, the provisioning policy should include *Email Authentication* and *Phone Authentication* or *SMS Authentication* as preceding factors for email and phone verification. The provisioning authority will ask the security question and answers if not provided in the user registration data and will prompt the user to download the *Acceptto It'sMe* iOS or Android apps. After installing the Acceptto app, the user should login with their provisioned email address. The 2nd authority is the *Acceptto Authentication* authority, that should be part of an authentication policy. When the authority executes, a push notification is sent to the Acceptto smart phone app requesting to accept the authentication request.

- **CAPTCHA Verification:** Protects against robots. Requires the end user to read a series of distorted characters that current computer programs cannot interpret.

- **LexisNexis Knowledge Based Authentication:** Given an individual's name, address and date of birth, the Lexis Nexis service will prompt for 3 identity questions with 4 options to select from. The sources for these questions are credit reporting agencies, mortgage records, DMV records, utility records and other public record systems. e.g. "Have you lived on any of the following streets" etc. If there are any incorrect answers, there is one additional follow up question. Rate limiting prevents bad actors to perform brute force attacks to break the system.

- **Infutor Identity Verification:** Resilient Access has integrations with the Infutor ID Max API that returns identity verification attributes and demographic data based on some basic input attributes such as name, email, phone number or address. The input attributes are full name and one or more of phone number, address and/or email. The service provides a score/rating of the match between the name and other identity attributes. For identities the verification succeeds, the authority returns various additional attributes that can be used as context for other policy decisions or can be returned as claims or assertions to relying parties. Similar to Neustar verification authorities there are three types of Infutor Identity Verification authorities
    - **Name - Phone Verification** - returns a score of the name to phone number match and other attribute
    - **Name - Email Verification** - returns a score of the name to email match and other attributes
    - **Name - Address Verification** - returns a score of the name to address match and other attributes.

- **FIDO U2F Authenticator:** The Fido Alliance has specified a second factor authentication standard that uses a USB hardware device that can be used for strong cryptographic authentication. An implementation of this standard by Google and Yubico is being used for enhanced security for Google products. The USB device acts as a certificate authority and issues an RSA key pair, storing the private key in the device and the public key is held by a U2F service implemented by an Identity provider. A cryptographic value is generated using the user ID, application ID and tenant ID attributes during the registration process. In the authentication phase the cryptographic value generated at authentication time is compared to the value generated during registration. Google Chrome and Firefox browsers incorporate the plugins to interact with the U2F device.

Resilient Access has a component that implements the U2F Server API and a client that incorporates U2F javascript libraries and client code to invoke the U2F Server API. This functionality is providesd by two authorities - U2F Registration and U2F Authentication. The U2F Registration authority is usually part of an account registration policy and the U2F Authentication authority can be used in an MFA authentication policy.

- **FIDO2/WebAuthn Authenticator:** The FIDO Alliance and the W3C have produced the FIDO2/WebAuthn standard. It specifies the interaction between three parties:
    - *FIDO2 Authenticators* - protocols such as USB, BLE and NFC are used to connect authenticators to user agents; U2F authenticators are supported
    - *WebAuthn User Agents* - builtin browser and OS agents that bridge RP clients with FIDO2 authenticators
    - *RP clients* - JS browser clients and client side programs that act on behalf of RPs to register and authenticate users

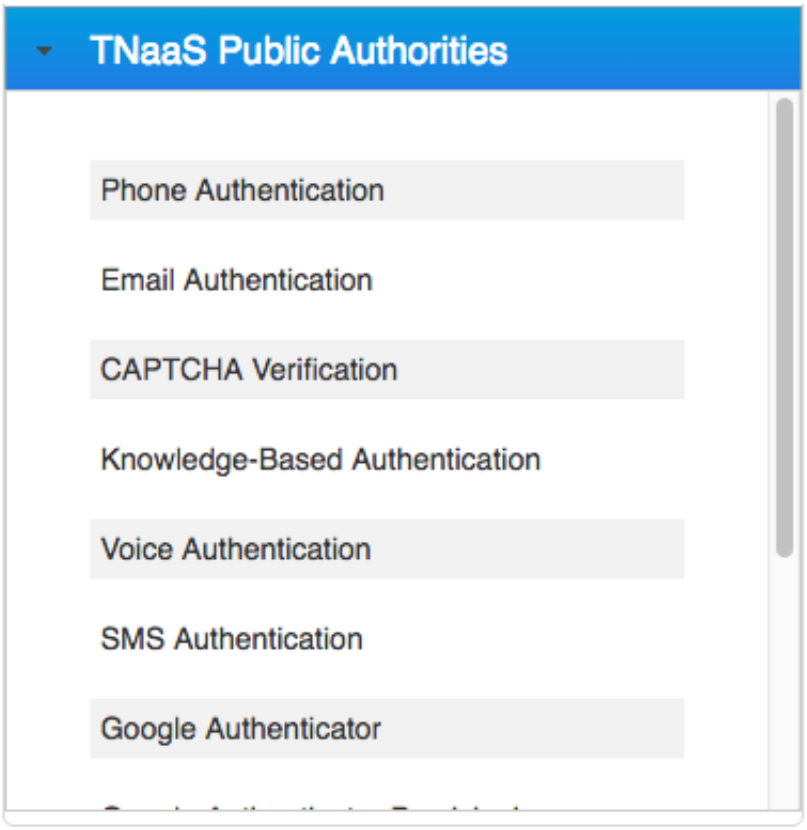
  The objective of this standard is to support user registration and authentication with strong public key credentials using a broad range of authenticators. This is important for several reasons:
    - Each user registration with an RP is done with a unique public key credential. The credential's private key never leaves the authenticator's secure enclave - it cannot be fished from the user and it cannot be stolen from an RP.
    - Google and Apple(rumored) are planning to add FIDO2 authenticators to Android and iOS. This will provide a ubiquitous, secure, easy-to-use authenticators on consumer phones.
    - Mozilla and Google have added WebAuthn user agents to Firefox and Chrome. Microsoft just announced the availability of one for Edge and Apple is rumored to be adding one to Safari.

  The result has the potential to practically eliminate the password as a consumer credential. In addition, mobile authenticators will also eliminate requiring the user to enter their user ID - the authenticators will automatically provide this to RPs.

  Resilient Access has a component that implements the WebAuthn RP function and is in the process of adding a WebAuthn RP client. Together these will greatly simplify the work required to add RP support for FIDO2/WebAuthn. This functionality is provided by two authorities - WebAuthn Registration and WebAuthn Authentication. An RP can include the WebAuthn Registration authority as an element of their account registration policy; and, can include the WebAuthn Authentication authority as an element of their authentication policy.

Administrator can also create their own authorities for identity and access verification functions that perform the verification function against their back-end resources as described in Create Authorities section. These authorities will appear in a section under the organization domain in the authorities selection UI.

The screenshot below shows the authority selection UI:

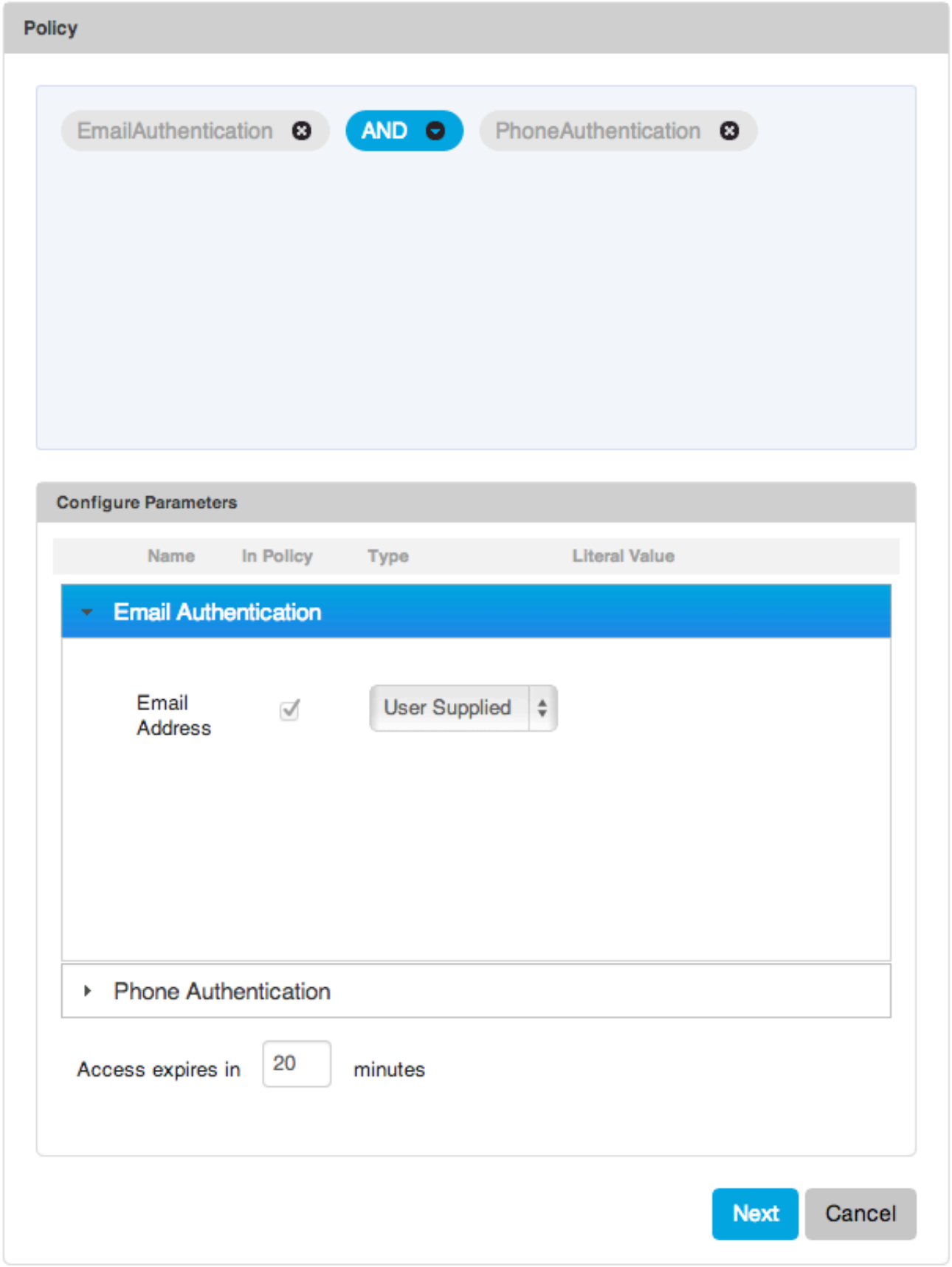


The right side consists of the policy expression panel and **Configure Parameters** section. The policy expression is created by selecting an authority from the **Authorities** control and dragging it to pale blue policy expression panel, the panel is highlighted to indicate it is a drop target. When there are more than one authorities in the policy expression panel, by default it uses AND as the join operator. This can be changed to one of the other three operators (OR, Ordered OR, Ordered AND). When there is a mix of different operators the precedence rules are:

1. Ordered AND
2. AND
3. Ordered OR
4. OR

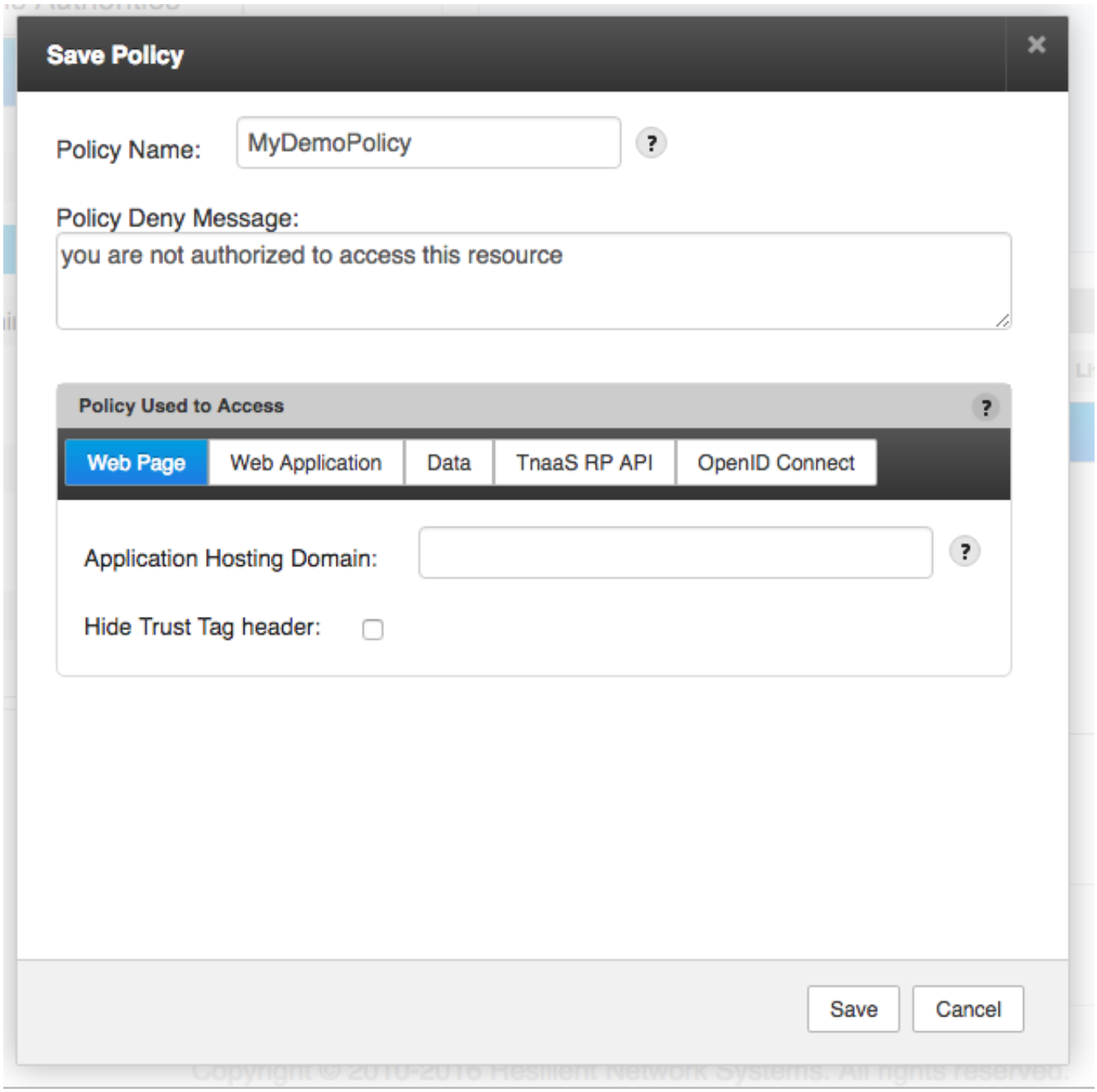
Since policy evaluation is a network operation, by default the AND and OR operators will evaluate the child operand with least complexity first (smallest sub-tree). Ordered OR and Ordered AND provides the semantics to override this and evaluate the child operands of the operator in strict left to right order.

The right hand side of the policy expression builder is show below.


Policy
EmailAuthentication
AND
PhoneAuthentication
Configure Parameters
Name
In Policy
Type
Literal Value
Email Authentication
Email Address
User Supplied
Phone Authentication
Access expires in
20
minutes
Next
Cancel

Authorities have parameters, the configuration of the authority can specify, certain parameters are required to be passed into the policy expression before the policy can be evaluated. For these authority parameter one should either select **User Supplied** or **Literal**. For **User Supplied** before policy evaluation begins the Trust Tag or Data Proxy policy evaluation UI will display prompts for these authority parameters. In the policy configuration one may optionally choose to pass any of the other authority parameters in the policy before evaluation. For the authority parameters that are not supplied into the policy, the policy evaluation run-time will dynamically request for these parameters through a Trust Broker Display screen. It is best practices for privacy enhancing policy evaluation to not pass parameters that contain sensitive information, like passwords before policy evaluation begins to minimize the transmission of sensitive data through the network.

Once the policy expression is built and on clicking the **Next** button, a popup is displayed for providing policy configuration parameters such as the Name and an optional custom deny message etc. The popup for policy configuration is shown below. Please refer to Web Page, Web Application, Data Proxy, TNaaS RP API and OpenID Connect for more details on the configurations of each of the *Policy Used to Access* options.


Save Policy
Policy Name:
MyDemoPolicy
Policy Deny Message:
you are not authorized to access this resource
Policy Used to Access
Web Page
Web Application
Data
TnaaS RP API
OpenID Connect
Application Hosting Domain:
Hide Trust Tag header:
Save
Cancel

Create Authorities

The Resilient Access Administrative Console provides graphical user interface (GUI) controls for adding and configuring custom authorities that perform an access control function. The authorities are grouped in the following categories:

- **Authentication authorities:**This type of authority performs username/password authentication. Intensity Analytics typing cadence can be combined with these authentication authorities for an additional factor besides the verification of the password entered.
- **Authorization authorities:**This type of authority performs authorization function such as membership in an LDAP/AD Group for role based authorization or the evaluation of an expression consisting of the attributes passed to the authority. Another category of authorization authority integrates with XACML Rules engine for customers to integrate their Policy Decision Point (PDP) with an overall access policy configured in RA.
- **Policy Authority**This type of authority extracts identity attributes from a user's record and passes them as inputs to authorities in another policy E.g. retrieving a user's phone number from an Active Directory and performing 2nd factor SMS or phone authentication. This is the mechanism through which Resilient policy can be configured to dynamically expand to more complex policies, given a small set of initial identity attributes (e.g. email or user ID), retrieve other identity attributes and perform additional authentication/verification functions.
- **Attribute Provider authorities**This type of authority extracts identity attributes out of an Identity Store and returns them as a part of the policy result. For SAML these attributes are part of the SAML assertion and for OpenID Connect are part of the user claims.
- **Policy Composition Authorities** These type of authorities are typically used to create adaptive access policies e.g. if a user is accessing an application within the organization's network, a simple password authentication policy is sufficient. If accessing the application from VPN then a 2 factor authentication policy must be applied. If accessing from a public wifi, then a more robust authentication policy must be imposed.

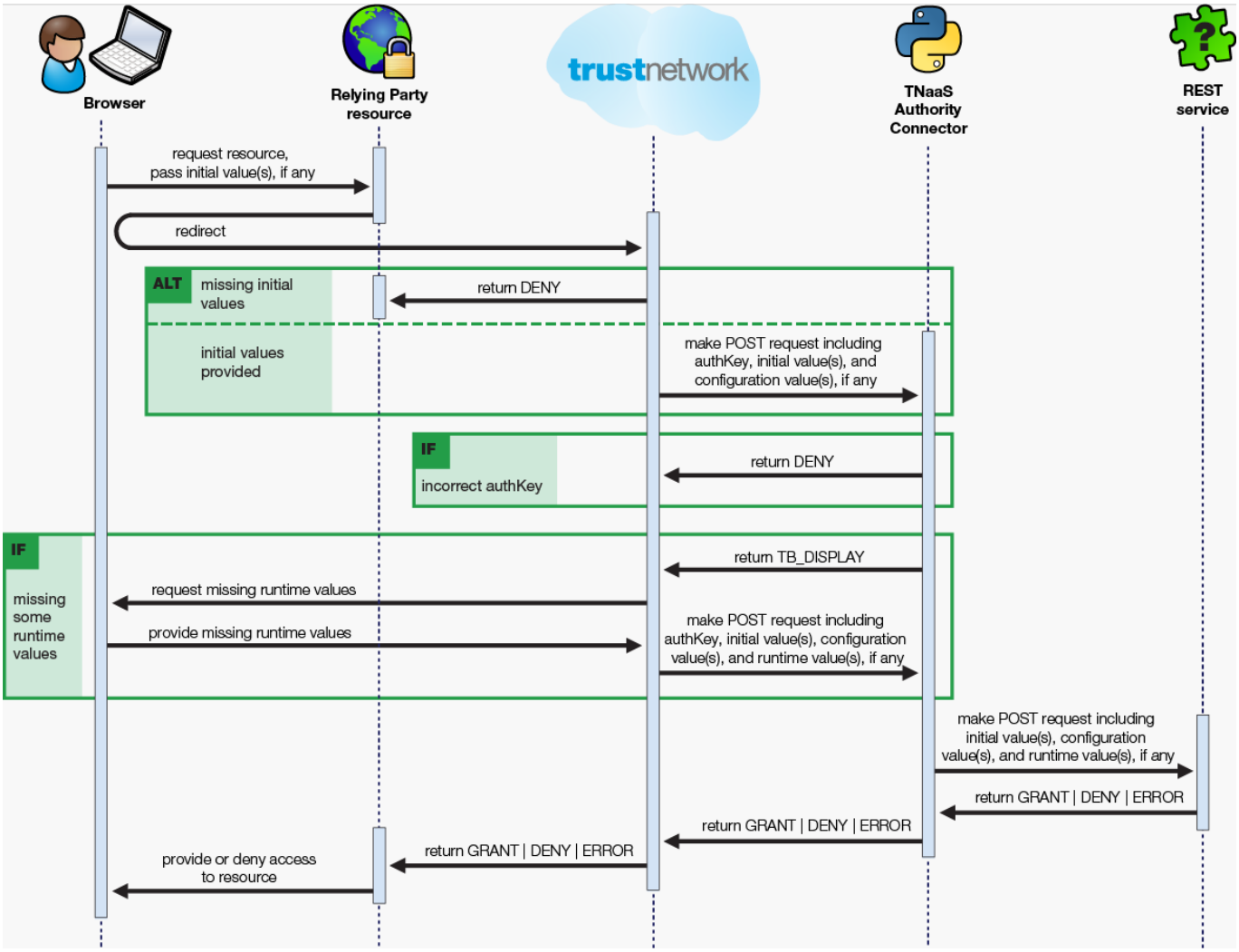
Authority Connector

Since an organization's identity store is usually never exposed outside of the organization's internal network, the LDAP/AD or Database authorities have two parts, a service that executes in the deployment of the Resilient Access Management system, usually hosted in the cloud and an Authority Connector service, a lightweight HTTP service that runs on a server within the organization's network and connects to the identity store. The hosted authority service connects with the Authority Connector over a secure protocol and only performs query functions for authentication and identity attribute retrieval.

The Resilient Access Authority Connector bundle consists of a few scripts, a lightweight python web server, and an encrypted file containing the configuration information specified in the GUI. With no need for a database and its own web server, the Resilient Access Authority features a very small footprint, making it easy for authority providers to extract and run the Resilient Access Authority Connector inside their own environments. Through this mechanism customer's backend resources such as LDAP/Active

Directory, Databases can remain behind the firewall with the Resilient Access Authority Connector taking responsibility for communications with Resilient Access through a secure interface that is based on key exchanges over TLS.

The following diagram illustrates the runtime sequence.



The steps to create a new authority and make it available for use in policies are:

1. Create and configure the Authority in Resilient Access Admin console as explained below for each of the different authority types.
2. Download and deploy the Resilient Access Authority Connector. if applicable
3. Execute Resilient Access Authority tests to make sure the authority is configured correctly.
4. Turn the authority online, it will now appear in the Authority selection panel in the Policy creation page under your organization's authorities section.

The following types of Authorities are supported in Resilient Access at this time:

Authentication Authorities

- Custom REST Authentication
- LDAP/AD Authentication
- Database Authentication

Authorization Authorities

- LDAP/AD Group Membership
- Attribute Authorization

Policy Authorities

- Custom REST Policy Authority
- LDAP/AD Policy Authority
- Database Policy Authority

Attribute Provider Authorities

- Custom REST Attribute Provider
- LDAP/AD Attribute Provider
- Database Attribute Provider

Policy Composition Authorities

- Simple Policy Authority
- Decision Authority

TNaaS Authority Connector

Prerequisites for TLS/SSL connections

If you specified the https protocol in the **TNaaS Authority Connector Host** box when configuring the authority, you will need the following files to complete the installation process:

- PEM encoded file containing the X.509 certificate to be used for TLS/SSL
- PEM encoded file containing the private key of the X.509 certificate

Copy these files to the local hard drive of the TNaaS Authority Connector host.

Ensure that the files are readable by root. As a best practice, Resilient recommends setting the minimum privileges on the PEM/CRT files.
sudo chmod 400 *file_name.extension*

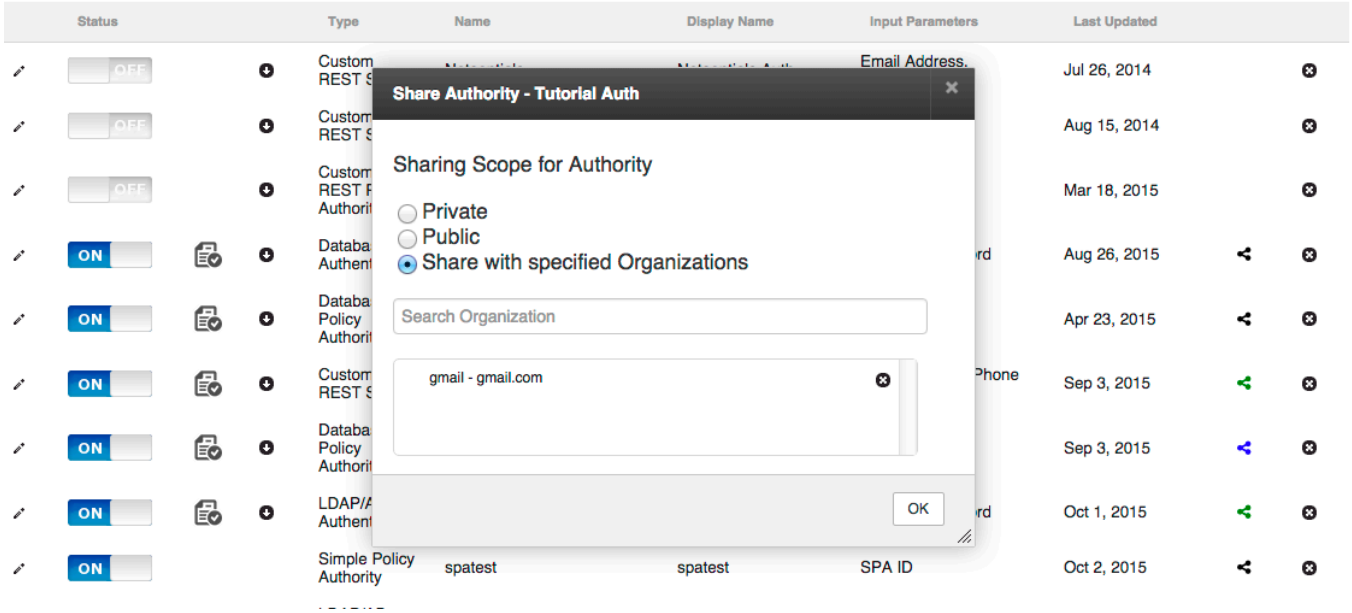
Installing the TNaaS Authority Connector

To install the TNaaS Authority Connector, complete the following steps.

1. Download the TNaaS Authority Connector from the Administrative Console.
2. Log onto the TNaaS Authority Connector host.
3. Copy the ZIP file containing the TNaaS Authority Connector scripts onto the TNaaS Authority Connector host.
4. Open a terminal prompt.
5. Extract the contents of the ZIP file.
   unzip *file_name*.zip
6. If you are not already logged in as a user with **sudo ALL** privileges, switch to one.
   su *sudo_ALL_user_name*
7. Execute the **tnaas-authority** shell script using **sudo**.
   sudo bash tnaas-authority.sh
   [-c *path*/*cert_file_name.extension* -k *path*/*prikey_file_name.extension*]
   where the **-c** and **-k** flags are required only if you specified the https protocol in the **Custom REST Service URL** box when configuring the authority.
8. If the web server starts successfully, you will see the following message:
   Bottle v0.13-dev server starting up (using
   CherryPyServer(keyfile='*prikey_file_name.extension*', certfile='*cert_file_name.extension*'))...
   Listening on http[s]://*host_name*:*port_number*/
   Hit Ctrl-C to quit.
9. After receiving the message that the web server is up and running, log into the Administrative Console, perform the TNaaS Authority Test to make sure the authority connector is functioning configured and functioning correctly and then switch your the authority from **OFF** to **ON** to use them in policies.

Sharing

Authorities can be shared across the Resilient Access tenancies through the sharing feature.



The sharing scope options are *Private*, *Public* or *Share with specified Organizations*. By default an authority's sharing scope is *Private*. This means the authority is only available within the tenancy the authority was created in. The sharing scope can be changed by clicking on the sharing icon towards the right of the authorities list table.

Setting the sharing scope to *Public*, will make the authority available to all other tenancies within Resilient Access. Setting the sharing scope to *Share with specified Organizations* will give the Resilient Access Admin user the option to search for organizations they wish to share their Authorities with. This would typically be partner organizations that the current organization want to make their authorities available for facilitating cross organizational access control.

In the authorities list the color of the sharing icon indicates the current sharing scope.

- Black - *Private*

- Green - *Public*
- Blue - *Share with specified Organization*

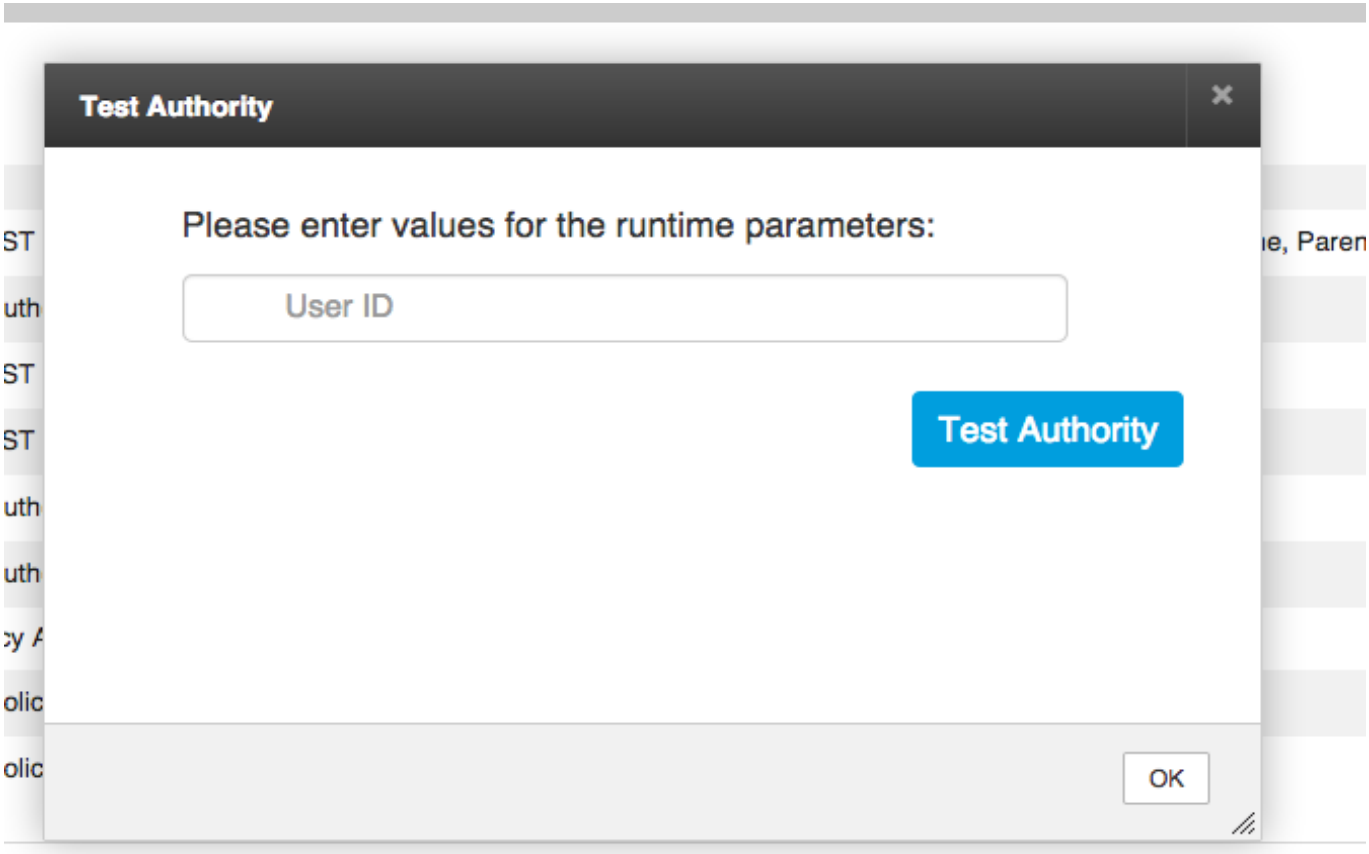
A shared authority will appear in the Authorities panel on the left hand side of the Create Policies page under a category named after the domain of the organization sharing the authority.

Testing

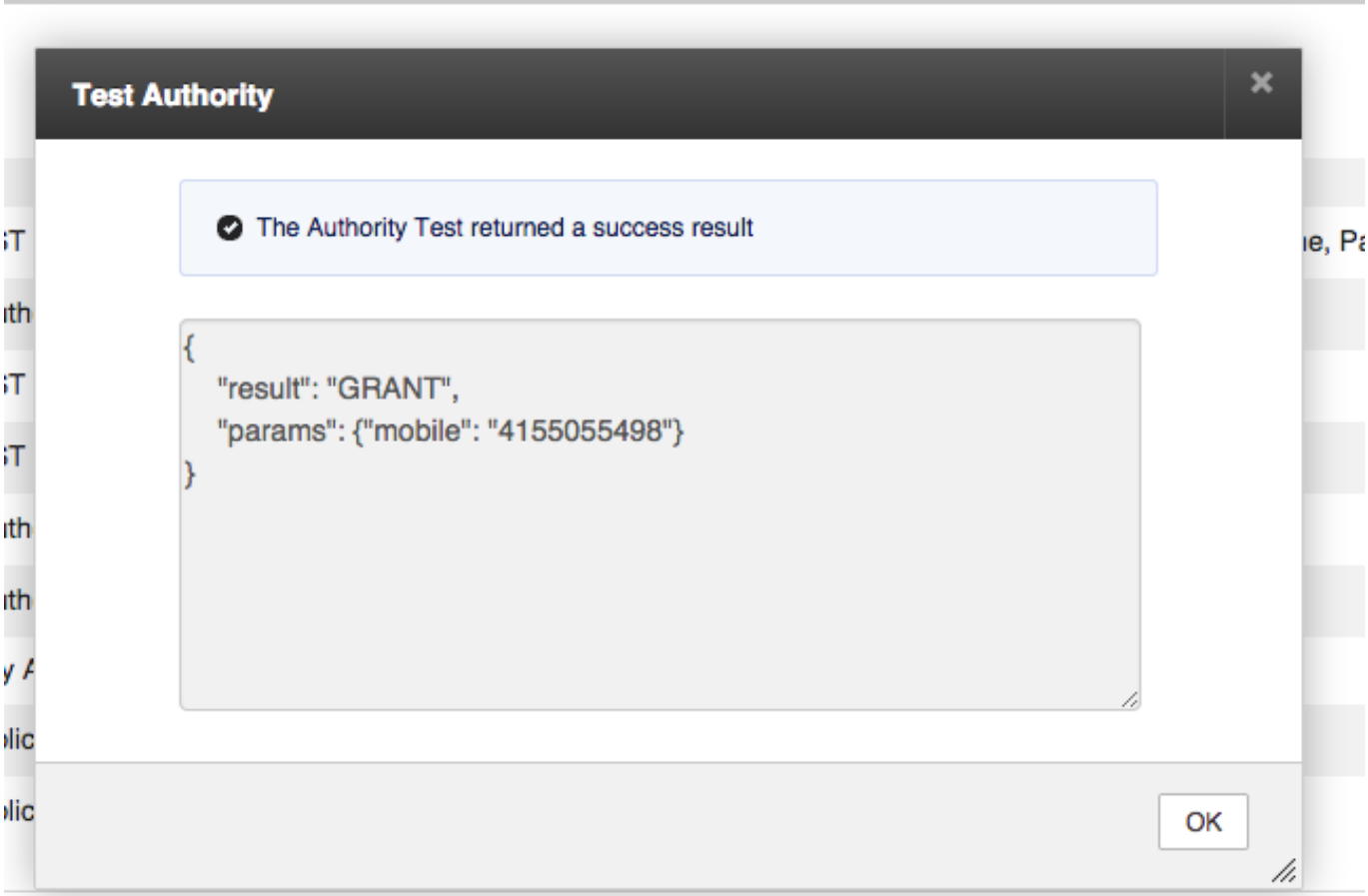
Authority Connector Test

Authorities that have a authority connector component has a testing feature that is available when the authority is made online. This allows the Resilient Admin user to perform tests to make sure the Authority has been configured correctly and returns the correct response when run-time parameters are passed in. The Authority can be tested by clicking on the test icon for the Authority in the Authorities list.

The Authority Test feature executes a HTTP request to the Authority connector in the exact same manner in which the Authority connector is invoked during policy evaluation. The following screenshots show the Authority Test feature:

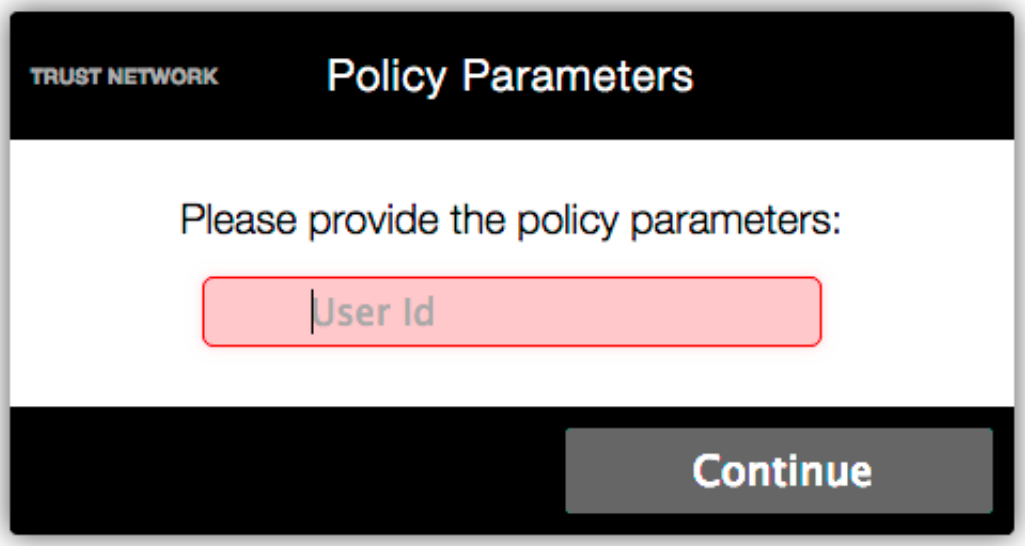


Enter values for runtime parameters and click the "Test Authority" button. A POST HTTP request will be sent to the Resilient Access Authority Connector with the values for the run-time parameters and the results will be displayed in the response page as shown below:

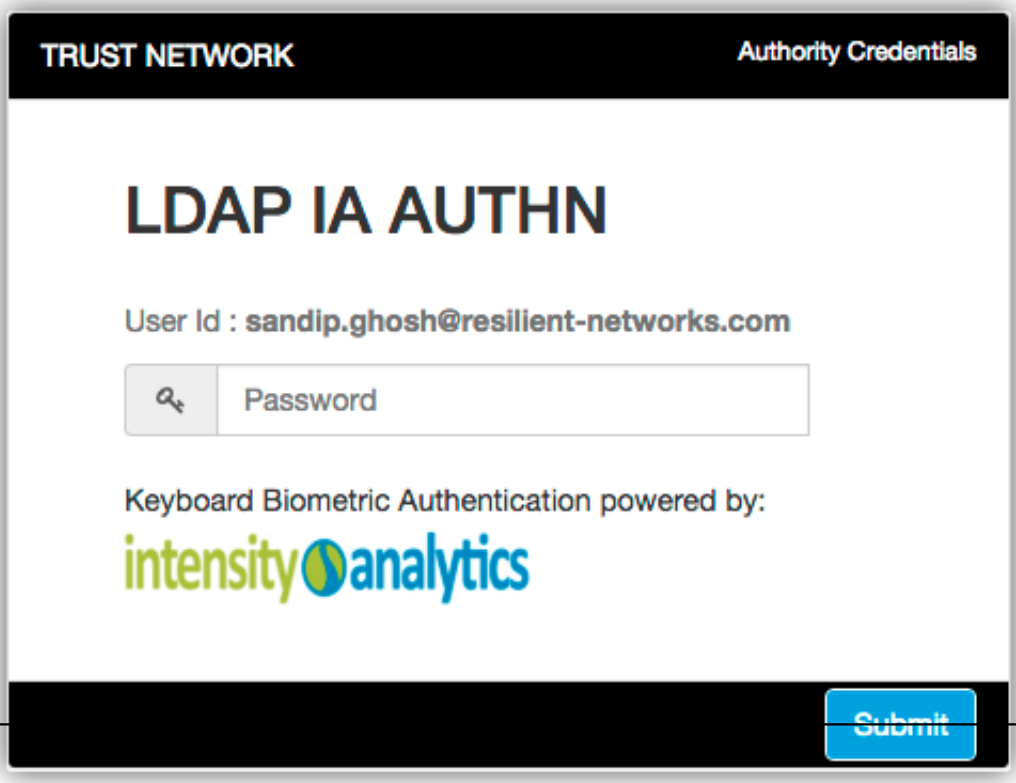


Policy Test

The policy can be tested by clicking on the test icon next to the policy in the policies list. This will open a new browser tab with a modal screen prompting the user to enter the runtime parameters for the policy. The policy will then evaluate using the Resilient Access policy evaluation UX, prompting for *Authority Credentials* as needed based on the configurations of the authorities that make up the policy. A status panel will display the result of the policy evaluation.


TRUST NETWORK
Policy Parameters
Please provide the policy parameters:
User Id
Continue


Administrative Console
TNaaS Policy Test - WebApp


TRUST NETWORK
Authority Credentials
LDAP IA AUTHN
User Id : sandip.ghosh@resilient-networks.com
Password
Keyboard Biometric Authentication powered by:
intensity analytics
Submit

Custom REST Authentication

An Authority that calls a REST service that makes the authentication decision for the authority. The REST service must implement the Custom REST Authority API that provides a mechanism for developers to extend and enhance the access control capabilities of Resilient Access to meet their requirements.

Configuring the authority

To configure the authority, complete the following steps.

1. Sign into the Administrative Console.
2. Click **Create New Authority**.
3. Ensure that **Custom REST Authentication - V2** is selected in the **Authority Type** list box.
4. Type a name for the authority in the **Authority Name** box.
5. Type a name for the authority in the **Authority Display Name** box.
6. Type a description of the authority in the **Authority Description** box (optional).
7. Use the **Configuration Parameters** area to specify static name/value pairs to be sent with each evaluation request at runtime. Since the same value is sent every time, this type of parameter may be useful to send API credentials for web services the REST service authority is encapsulating. The config parameters can be sent either in the POST body or as HTTP header parameters
8. Use the **Runtime Parameters** area to add and configure the parameters that are used in the execution of the authority, e.g. email address or user ID. The values of these parameters will be supplied by the end user at runtime. For each runtime parameter, specify the following:
    - Type the name of the parameter in the **Name** box. The parameter name gets paired with the value provided at runtime and sent to the custom REST authority.
    - Type the label of the box displayed to the end user in the **Display Name** box.
    - If the parameter will contain a sensitive value, such as personally identifiable information, select the **Obfuscate** check box. This instructs Resilient Access to substitute an opaque token for the value as it transits the network, ensuring that the value is opaque to other authorities that may be part of the policy workflow.
    - If the user will provide the value in the initial request form, select the **Initial Request** check box. NOTE: Resilient recommends leaving the **Initial Request** check box blank if the value is sensitive or contains personally identifiable information.
    - Select the **Mask Input** checkbox to mask the values with bullet characters as the user types them in. This protects against shoulder surfing.
9. Type the fully qualified URL for the base address of the REST service in the **Custom REST Service URL** box including scheme and port (if applicable). The /token and /evaluate end-points must be implemented off this base address as per the Custom REST Authority API specifications.
10. Once you have finished configuring the custom authority, click **Create** or **Save**.

Retrieving API Credentials

From the Authorities list retrieve the API credentials for the JWT Bearer Token OAuth authentication

that is required for the Custom REST service API. This includes:

- Client ID
- Client Secret
- Public Key of an RSA Key Pair in PEM format (the Private Key is stored in Resilient Access)
- Key ID

LDAP/AD Authentication

An authority that authenticates a user in an LDAP or Active Directory. The Policy Workflow Engine privacy enhancing policy evaluation mechanism avoids the passing around of the password through the network by requesting for the password only when the LDAP/AD Authentication authority is invoked. The sensitive information is only exchanged between the user and the LDAP/AD Authentication authority over TLS.

Configuring the LDAP/AD Authentication Authority

To configure the authority, complete the following steps.

1. Sign into the Administrative Console
2. Click **Create New Authority.**
3. Select **LDAP/AD Authentication** from the **Authority Type** drop down.
4. Type a name for the authority in the **Authority Name** box.
5. Type a description of the authority in the **Authority Description** box (optional).
6. Type he fully qualified URI location of the Resilient Access Authority Connector in the **Resilient Access Authority Connector Host** box, including the number of the port on the Resilient Access Authority Connector host that will accept incoming connections. To encrypt the communications between the Trust Network and the Resilient Access Authority Connector, type **https**.
   http[s]://*fully_qualified_domain_name*:*port_number*
7. Configure the connection to the LDAP/AD server by entering:
    1. The LDAP Protocol connection URL using the ldap:// or ldaps:// scheme in the **Connection URL** box: **ldap[s]://*fully_qualified_domain_name*:*port_number***
    2. The LDAP/AD server connection user **Distinguished Name (DN)** in the **Connection Name** box: e.g. ***cn=admin,dc=acme,dc=com***
    3. The password for the user account to connect to the LDAP/AD server in the **Connection Password** box.
8. Configure how the user record for a member of the LDAP/AD will be found
    1. Enter the base path within the LDAP/AD where user records are stored in the **User Search Base** box. This field allows multiple base paths to be entered, by clicking the "+" icon to the right of the text box. If more than one base path is specified, all the specified base paths will be searched sequentially for the user record. e.g. ***(ou=employees,dc=acme,dc=com) OR (ou=contractors,dc=acme,dc=com)***
    2. If sub-paths below the base path should be searched for the user record, then click on the **Search Subtree** checkbox
    3. If entries within the LDAP/AD references other locations where user records are stored, then those locations will also be searched if the **Follow Referrals** checkbox is checked
    4. Enter the name of the user record attribute in the LDAP/AD that will be the user identifier when searching for the user record in the **User Identity Attribute** box. For example if the authentication is performed based on email address as the identity attribute and the "**mail**" attribute hold the email address in the user record then ***mail*** should be entered for **User Identity Attribute**
    5. Resilient Access has integrated with Intensity Analytics Behavioral Biometric

Authentication to seamlessly provide strong second factor authentication to policies created in Resilient Access. Intensity Analytics has a patented technology of calibrating the rhythm of a user's keystoke pattern and using that to create a unique user identity signature. This is then applied to detect if the person typing the password is the person being authenticated. As the user authenticates using their password the system calibrates and stores profiles of the keystroke rhythm until enough profiles have been created to accurately determine a user's keystroke rhythm. Subsequent authentication attempts will enforce the Intensity Analytics Behavioral Biometric Authentication as an additional authentication factor. To enable Intensity Analytics Behavioral Biometric Authentication click the **Include Intensity Analytics** checkbox.

9. Once you have finished configuring the LDAP/AD Authentication authority, click **Create** or **Save**.

Database Authentication

Authenticates a user by querying a database consisting of user records. Resilient privacy enhancing policy evaluation mechanism avoids the passing around of the password through the network by requesting for the password only when the Database Authentication authority is invoked. The sensitive information is only exchanged between the user and the Database Authentication authority over TLS.

Configuring the Database Authentication Authority

To configure the authority, complete the following steps.

1. Sign into the Administrative Console
2. Click **Create New Authority.**
3. Select **Database Authentication** from the **Authority Type** drop down.
4. Type a name for the authority in the **Authority Name** box.
5. Type a description of the authority in the **Authority Description** box (optional).
6. Type he fully qualified URI location of the Resilient Access Authority Connector in the **Resilient Access Authority Connector Host** box, including the number of the port on the Resilient Access Authority Connector host that will accept incoming connections. To encrypt the communications between Resilient and the Resilient Access Authority Connector, type **https**.
   http[s]://*fully_qualified_domain_name*:*port_number*
7. Configure the connection to the Database server by entering:
    1. The fully qualified hostname of the database server in the **Database Host** box:
    2. The name of the database containing the user records in the **Database Name** box
    3. The database user name in the **Database User Name** box
    4. The password for the user account to connect to the Database server in the **Database Password** box.
8. Configure how the database will be queried to find the user and verify their password.
    1. Enter the table in the database that holds the user records in the **Authentication Table** box
    2. Enter the column name that has the user identity attribute that will be passed during database authentication in the **UserId Column Name** box
    3. Enter the column name that stores the password in the **Password Column Name** box.
    4. If Passwords are encoded as MD5 values before storing them in the database then the **Password encoded as MD5** checkbox should be checked.
    5. Resilient Access has integrated with Intensity Analytics Behavioral Biometric Authentication to seamlessly provide strong second factor authentication to policies created in Resilient Access. Intensity Analytics has a patented technology of calibrating the rhythm of a user's keystoke pattern and using that to create a unique user identity signature. This is then applied to detect if the person typing the password is the person being authenticated. As the user authenticates using their password the system calibrates and stores profiles of the keystroke rhythm until enough profiles have been created to accurately determine a user's keystroke rhythm. Subsequent authentication attempts will enforce the Intensity Analytics Behavioral Biometric Authentication as an additional authentication factor. To enable Intensity Analytics Behavioral Biometric Authentication click the **Include Intensity Analytics** checkbox.

9. Once you have finished configuring the Database Authentication authority, click **Create** or **Save**.

LDAP/AD Group Membership

This type of authority authorizes access if the user is a member of a LDAP or AD group. This authority can be used for role based access control policies.

Configuring the LDAP/AD Group Membership Authority

To configure the authority, complete the following steps.

1. Sign into the Administrative Console
2. Click **Create New Authority.**
3. Select **LDAP/AD Group Membership** from the **Authority Type** drop down.
4. Type a name for the authority in the **Authority Name** box.
5. Type a description of the authority in the **Authority Description** box (optional).
6. Type he fully qualified URI location of the Resilient Access Authority Connector in the **Resilient Access Authority Connector Host** box, including the number of the port on the Resilient Access Authority Connector host that will accept incoming connections. To encrypt the communications between the Trust Network and the Resilient Access Authority Connector, type **https**.
   http[s]://*fully_qualified_domain_name*:*port_number*
7. Configure the connection to the LDAP/AD server by entering:
    1. The LDAP Protocol connection URL using the ldap:// or ldaps:// scheme in the **Connection URL** box: **ldap[s]://*fully_qualified_domain_name*:*port_number***
    2. The LDAP/AD server bind account **Distinguished Name (DN)** in the **Connection Name** box: e.g. ***cn=admin,dc=acme,dc=com***
    3. The password for the bind account to connect to the LDAP/AD server in the **Connection Password** box.
8. Configure how the user record for a member of the LDAP/AD will be found
    1. Enter the base path within the LDAP/AD where user records are stored in the **User Search Base** box. This field allows multiple base paths to be entered, by clicking the "+" icon to the right of the text box. If more than one base path is specified, all the specified base paths will be searched sequentially for the user record. e.g. ***(ou=employees,dc=acme,dc=com) OR (ou=contractors,dc=acme,dc=com)***
    2. If sub-paths below the base path should be searched for the user record, then click on the **Search Subtree** checkbox
    3. If entries within the LDAP/AD references other locations where user records are stored, then those locations will also be searched if the **Follow Referrals** checkbox is checked
    4. Enter the name of the user record attribute in the LDAP/AD that will be the user identifier when searching for the user record in the **User Identity Attribute** box. For example if the authentication is performed based on email address as the identity attribute and the "**mail**" attribute hold the email address in the user record then ***mail*** should be entered for **User Identity Attribute**
9. Configure Group membership attributes as follows:
    1. Specify the Base DN for the LDAP/AD group the authority is verifying membership of.
    2. Specify the attribute that lists the members of the group within the Group DN. For AD this attribute is typically member while for LDAPs this attribute is typically uniqueMember

10. Once you have finished configuring the LDAP/AD Authentication authority, click **Create** or **Save**.

Attribute Authorization

This type of authority receives a number of attributes as run-time parameters. Typically these attributes are retrieved from identity stores like LDAP/AD or Databases and the attribute authorization authority is configured as an authority in the output policy of a LDAP/AD or Database policy authority. The authority provides an intutive user interface for building a logical expression of any complexity using the run-time parameters and literals as shown in the example below.

Configuring the authority

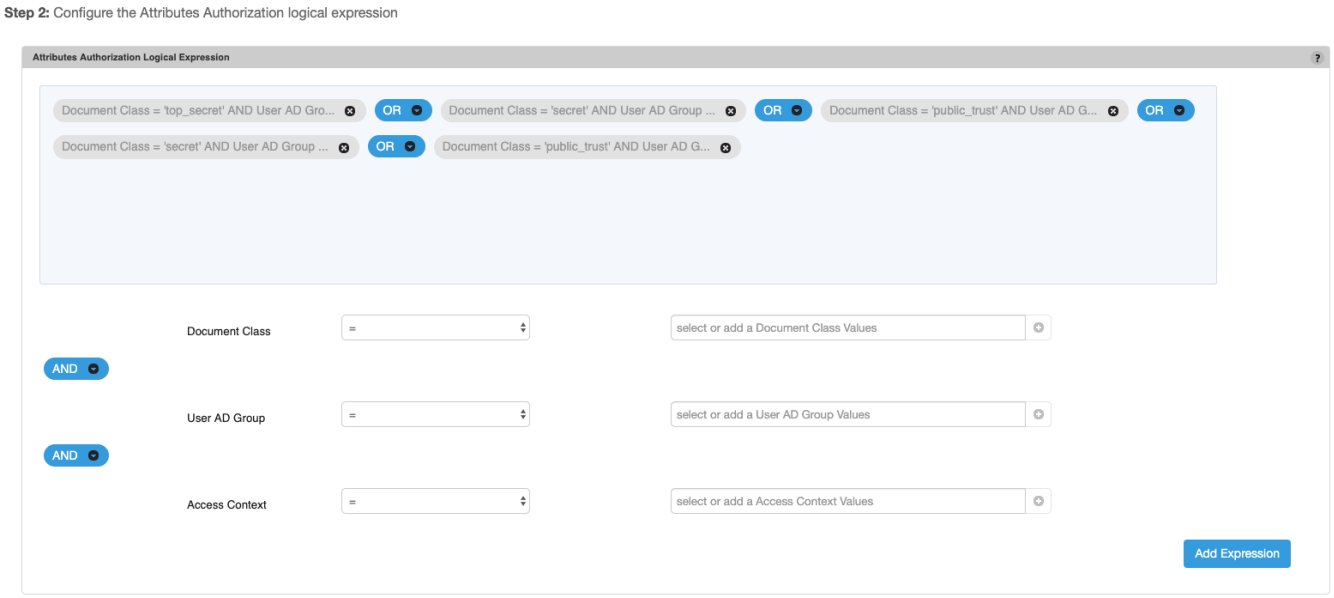
To configure the authority, complete the following steps.

1. Sign into the Administrative Console.
2. Click **Create New Authority**.
3. Ensure that **Attribute Authorization** is selected in the **Authority Type** list box.
4. Type a name for the authority in the **Authority Name** box.
5. Type a name for the authority in the **Authority Display Name** box.
6. Type a description of the authority in the **Authority Description** box (optional).
7. Use the **Runtime Parameters** area to add and configure the parameters that are used in the execution of the authority, e.g. email address or user ID. The values of these parameters will be supplied by the end user at runtime. For each runtime parameter, specify the following:
    - Type the name of the parameter in the **Name** box. The parameter name gets paired with the value provided at runtime and sent to the custom REST authority.
    - Type the label of the box displayed to the end user in the **Display Name** box.
    - If the parameter will contain a sensitive value, such as personally identifiable information, select the **Obfuscate** check box. This instructs Resilient Access to substitute an opaque token for the value as it transits the network, ensuring that the value is opaque to other authorities that may be part of the policy workflow.
    - If the user will provide the value in the initial request form, select the **Initial Request** check box. NOTE: Resilient recommends leaving the **Initial Request** check box blank if the value is sensitive or contains personally identifiable information.
    - Select the **Mask Input** checkbox to mask the values with bullet characters as the user types them in. This protects against shoulder surfing.
8. The logical expression is built in the **Attributes Authorization Logical Expression** panel by creating sub-expressions that combine the run-time parameters. The sub-expressions are then joined through the AND and OR join operators. To create Attribute Authorization expression follow the steps below
    1. For each run-time parameter specify at-least one value, multiple values can be provided by either clicking the + button next to the parameter value box or by clicking the ENTER key. Values can be removed by clicking the X for an entry in the drop-down of the parameter value. When multiple values are specified for a run-time parameter the operator should be IN or NOT IN
    2. Specify the join operator for each run-time parameter expression. Typically the run-time parameters are joined using the same join operator.
    3. Click **Add Expression** to add the sub-expression to the overall Attribute Authorization

expression

4. Specify the join operator that joins the sub-expression when there are multiple sub-expressions. Typically the same join operator is used to join all the sub-expressions.
5. When mouse hovers over an sub-expression the entire expression is displayed in a popup tooltip.

9. Once you have finished configuring the Attribute Authorization authority, click **Create** or **Save**.



Custom REST Policy Authority

An Authority that calls a REST service that returns attributes if access is granted. These attributes are mapped to the input parameters of the output policy configured. The REST service must implement the Custom REST Authority API that provides a mechanism for developers to extend and enhance the access control capabilities of Resilient Access to meet their requirements.

Configuring the authority

To configure the authority, complete the following steps.

1. Sign into the Administrative Console.
2. Click **Create New Authority**.
3. Ensure that **Custom REST Policy Authority - V2** is selected in the **Authority Type** list box.
4. Type a name for the authority in the **Authority Name** box.
5. Type a name for the authority in the **Authority Display Name** box.
6. Type a description of the authority in the **Authority Description** box (optional).
7. Use the **Configuration Parameters** area to specify static name/value pairs to be sent with each evaluation request at runtime. Since the same value is sent every time, this type of parameter may be useful to send API credentials for web services the REST service authority is encapsulating. The config parameters can be sent either in the POST body or as HTTP header parameters
8. Use the **Runtime Parameters** area to add and configure the parameters that are used in the execution of the authority, e.g. email address or user ID. The values of these parameters will be supplied by the end user at runtime. For each runtime parameter, specify the following:
    - Type the name of the parameter in the **Name** box. The parameter name gets paired with the value provided at runtime and sent to the custom REST authority.
    - Type the label of the box displayed to the end user in the **Display Name** box.
    - If the parameter will contain a sensitive value, such as personally identifiable information, select the **Obfuscate** check box. This instructs Resilient Access to substitute an opaque token for the value as it transits the network, ensuring that the value is opaque to other authorities that may be part of the policy workflow.
    - If the user will provide the value in the initial request form, select the **Initial Request** check box. NOTE: Resilient recommends leaving the **Initial Request** check box blank if the value is sensitive or contains personally identifiable information.
    - Select the **Mask Input** checkbox to mask the values with bullet characters as the user types them in. This protects against shoulder surfing.
9. Type the fully qualified URL for the base address of the REST service in the **Custom REST Service URL** box including scheme and port (if applicable). The /token and /evaluate end-points must be implemented off this base address as per the Custom REST Authority API specifications.
10. Specify the names of the attributes that will be returned in the assertions object in the response to the /evaluate API if the "result"="GRANT".
11. Define the output policy and configure the policy parameters using the steps below:
    11. Click the **Create Output Policy** button to define the output policy. The output policy is created in a popup window with a similar interface as the Create Policies page. Drag and drop authorities and define the output policy

12. The **Configure Policy Parameters** table will list the parameters of the output policy. These can either be mapped to ***Runtime Parameter*** defined above or a ***Literal Value*** or a ***Query Result.***
    - If **Mapped Type** is ***Runtime Parameter*** then **Mapped Value** will be populated with the runtime parameters defined. Select the one to use from the drop-down
    - If **Mapped Type** is ***Literal*** enter the value in the **Mapped Value** box
    - If **Mapped Type** is ***Query Result*** then **Mapped Value** will be populated with the attributes defined in the **Custom REST Service Attributes** section. Select the one to use from the drop-down.

12. Once you have finished configuring the custom authority, click **Create** or **Save**.

Retrieving API Credentials

From the Authorities list retrieve the API credentials for the JWT Bearer Token OAuth authentication that is required for the Custom REST service API. This includes:

- Client ID
- Client Secret
- Public Key of an RSA Key Pair in PEM format (the Private Key is stored in Resilient Access)
- Key ID

LDAP/AD Policy Authority

LDAP/AD Policy Authority performs a lookup in an LDAP/AD and returns attributes from the AD. These attributes can then be passed to the parameters of the output policy configured.

Configuring the LDAP/AD Policy Authority

To configure the authority, complete the following steps.

1. Sign into the Administrative Console.
2. Click **Create New Authority**.
3. Ensure that **LDAP/AD Policy Authority** is selected in the **Authority Type** list box.
4. Type a name for the authority in the **Authority Name** box.
5. Type a name for the authority in the **Authority Display Name** box.
6. Type a description of the authority in the **Authority Description** box (optional).
7. Type the fully qualified URI location of the Resilient Access Authority Connector in the **Resilient Access Authority Connector Host** box, including the number of the port on the Resilient Access Authority Connector host that will accept incoming connections. To encrypt the communications between Resilient and the Resilient Access Authority Connector, type **https**.
   http[s]://*fully_qualified_domain_name*:*port_number*
8. Use the **Runtime Parameters** area to add and configure the parameters without literal values. The values of these parameters will be supplied by the end user at runtime. For each runtime parameter, specify the following:
   - Type the name of the parameter in the **Name** box. The parameter name gets paired with the value provided at runtime and sent to the custom REST authority.
   - Type the label of the box displayed to the end user in the **Display Name** box.
   - If the parameter will contain a sensitive value, such as personally identifiable information, select the **Obfuscate** check box. This instructs Resilient Access to substitute an opaque token for the value as it transits the network, ensuring that the value never passes through the central Policy Workflow Engine component and does not get stored in the Trust History.
   - If the user will provide the value in the initial request form, select the **Initial Request** check box. NOTE: Resilient recommends leaving the **Initial Request** check box blank if the value is sensitive or contains personally identifiable information.
   - Select the **Mask Input** check box to mask the values with bullet characters as the user types them in. This protects against shoulder surfing.
9. Configure the connection to the LDAP/AD server by entering:
   1. The LDAP Protocol connection URL using the ldap:// or ldaps:// scheme in the **Connection URL** box: **ldap[s]://*fully_qualified_domain_name*:*port_number***
   2. The LDAP/AD server connection user **Distinguished Name (DN)** in the **Connection Name** box: e.g. ***cn=admin,dc=acme,dc=com***
   3. The password for the user account to connect to the LDAP/AD server in the **Connection Password** box.
10. Configure how the user record for a member of the LDAP/AD will be found
    1. Enter the base path within the LDAP/AD where user records are stored in the **User Search**

Base box. This field allows multiple base paths to be entered, by clicking the "+" icon to the right of the text box. If more than one base path is specified, all the specified base paths will be searched sequentially for the user record. e.g. ***(ou=employees,dc=acme,dc=com) OR (ou=contractors,dc=acme,dc=com)***

2. If sub-paths below the base path should be searched for the user record, then click on the **Search Subtree** checkbox
3. If entries within the LDAP/AD references other locations where user records are stored, then those locations will also be searched if the **Follow Referrals** checkbox is checked
4. Enter the name of the user record attribute in the LDAP/AD that will be the user identifier when searching for the user record in the **User Identity Attribute** box. For example if the authentication is performed based on email address as the identity attribute and the "**mail**" attribute hold the email address in the user record then ***mail*** should be entered for **User Identity Attribute**

5. Select the runtime parameter that will match the **User Identity Attribute** for the LDAP/AD record lookup

11. Specify the attributes to retrieve from the LDAP/AD by entering the attribute names in the **LDAP/AD Attributes to retrieve** section
12. Define the output policy and configure the policy parameters using the steps below
    1. Click the **Create Output Policy** button to define the output policy. The output policy is created in a popup window with a similar interface as the Create Policies page. Drag and drop authorities and define the output policy
    2. The **Configure Policy Parameters** table will list the parameters of the output policy. These can either be mapped to ***Runtime Parameter*** defined above or a ***Literal*** or a ***Query Result.***
        - If **Mapped Type** is ***Runtime Parameter*** then **Mapped Value** will be populated with the runtime parameters defined. Select the one to use from the drop-down
        - If **Mapped Type** is ***Literal*** enter the value in the **Mapped Value** box
        - If **Mapped Type** is ***Query Result*** then **Mapped Value** will be populated with the attributes defined in the **LDAP/AD Attributes to retrieve** section. Select the one to use from the drop-down.
13. Once you have finished configuring the custom authority, click **Create** or **Save**.

Database Policy Authority

Database Policy Authority performs a SQL Query in a Database and maps the results of the query to parameters of the output policy configured. The query should only return one result row for the authority to correctly forward the evaluation to the output policy.

Configuring the Database Policy Authority

To configure the authority, complete the following steps.

1. Sign into the Administrative Console.
2. Click **Create New Authority**.
3. Ensure that **Database Policy Authority** is selected in the **Authority Type** list box.
4. Type a name for the authority in the **Authority Name** box.
5. Type a name for the authority in the **Authority Display Name** box.
6. Type a description of the authority in the **Authority Description** box (optional).
7. Type the fully qualified URI location of the Resilient Access Authority Connector in the **Resilient Access Authority Connector Host** box, including the number of the port on the Resilient Access Authority Connector host that will accept incoming connections. To encrypt the communications between the Resilient Access and the Resilient Access Authority Connector, type **https**.
   http[s]://*fully_qualified_domain_name*:*port_number*
8. Use the **Runtime Parameters** area to add and configure the parameters without literal values. The values of these parameters will be supplied by the end user at runtime. For each runtime parameter, specify the following:
   - Type the name of the parameter in the **Name** box. The parameter name gets paired with the value provided at runtime and sent to the custom REST authority.
   - Type the label of the box displayed to the end user in the **Display Name** box.
   - If the parameter will contain a sensitive value, such as personally identifiable information, select the **Obfuscate** check box. This instructs Resilient to substitute an opaque token for the value as it transits the network, ensuring that the value never passes through the central Policy Workflow Engine component and does not get stored in the Trust History.
   - If the user will provide the value in the initial request form, select the **Initial Request** check box. NOTE: Resilient recommends leaving the **Initial Request** check box blank if the value is sensitive or contains personally identifiable information.
   - Select the **Mask Input** check box to mask the values with bullet characters as the user types them in. This protects against shoulder surfing.
9. Configure the connection to the Database server by entering:
   1. The fully qualified hostname of the database server in the **Database Host** box:
   2. The name of the database containing the user records in the **Database Name** box
   3. The database user name in the **Database User Name** box
   4. The password for the user account to connect to the Database server in the **Database Password** box.
10. Build the SQL Expression to extract attributes through the SQL query builder.
    1. First select the table name and click on the + button to add the table to the expression
    2. After the first table is added the controls for adding columns to the SQL query will appear.

Add each column by specifying the **Column Name** the type whether *String* or *Numeric* and the table the column belongs to and then the + button to add the column to the SQL query

3. After adding a column the controls for adding WHERE clauses will appear. The left-hand side of the WHERE clause expression will be populated with the columns in the SELECT section, the right-hand side can either be ***Runtime Parameter*** a ***SQL Column*** or a ***Literal***. The widget next to it will be appropriately populated to specify the right-hand side value. Click the + button to add the WHERE clause.
4. For either the SELECT column or the FROM table or the WHERE clause can be delete by clicking on the delete icon. The SQL expression will be updated as required to ensure it always remains a valid SQL query.

11. Define the output policy and configure the policy parameters using the steps below
    1. Click the **Create Output Policy** button to define the output policy. The output policy is created in a popup window with a similar interface as the Create Policies page. Drag and drop authorities and define the output policy
    2. The **Configure Policy Parameters** table will list the parameters of the output policy. These can either be mapped to ***Runtime Parameter*** defined above or a ***Literal*** or a ***Query Result.***
        - If **Mapped Type** is ***Runtime Parameter*** then **Mapped Value** will be populated with the runtime parameters defined. Select the one to use from the drop-down
        - If **Mapped Type** is ***Literal*** enter the value in the **Mapped Value** box
        - If **Mapped Type** is ***Query Result*** then **Mapped Value** will be populated with columns in the SQL query defined in the **Build the SQL Expression to extract attributes** section. Select the one to use from the drop-down.
12. Once you have finished configuring the custom authority, click **Create** or **Save**.

Simple Policy Authority

A Simple Policy Authority returns the output policy configured, mapping the runtime parameters passed to it to the parameters required by the output policy. This type of authority is well suited to define the top level policy when the policy consists of several lower level policies where each lower level policy performs a similar function. E.g. an isEmployee policy that checks against different identity sources to verify if an employee is a member of an organization.

Configuring the Simple Policy Authority

To configure the authority, complete the following steps.

1. Sign into the Administrative Console.
2. Click **Create New Authority**.
3. Ensure that **Simple Policy Authority** is selected in the **Authority Type** list box.
4. Type a name for the authority in the **Authority Name** box.
5. Type a name for the authority in the **Authority Display Name** box.
6. Type a description of the authority in the **Authority Description** box (optional).
7. Use the **Runtime Parameters** area to add and configure the parameters without literal values. The values of these parameters will be supplied by the end user at runtime. For each runtime parameter, specify the following:
    - Type the name of the parameter in the **Name** box. The parameter name gets paired with the value provided at runtime and sent to the custom REST authority.
    - Type the label of the box displayed to the end user in the **Display Name** box.
    - If the parameter will contain a sensitive value, such as personally identifiable information, select the **Obfuscate** check box. This instructs Resilient to substitute an opaque token for the value as it transits the network, ensuring that the value never passes through the central Policy Workflow Engine component and does not get stored in the Trust History.
    - If the user will provide the value in the initial request form, select the **Initial Request** check box. NOTE: Resilient recommends leaving the **Initial Request** check box blank if the value is sensitive or contains personally identifiable information.
    - Select the **Mask Input** check box to mask the values with bullet characters as the user types them in. This protects against shoulder surfing.
8. Define the output policy and configure the policy parameters using the steps below:
    1. Click the **Create Output Policy** button to define the output policy. The output policy is created in a popup window with a similar interface as the Create Policies page. Drag and drop authorities and define the output policy
    2. The **Configure Policy Parameters** table will list the parameters of the output policy. These can either be mapped to ***Runtime Parameter*** defined above or a ***Literal Value***. If selecting a ***Runtime Parameter*** then select the one to use from the **Mapped Value** drop-down, if **Mapped Type** is ***Literal Value*** then the value should be entered in the **Mapped Value** box.
9. Once you have finished configured the Simple Policy Authority, click **Create** or **Save**

Decision Authority

This authority type is used for routing the policy evaluation to a particular branch based on a criteria on one of the runtime parameters that are among its inputs. We can think of this authority type functioning similar to a *switch* statement in a programming language or rules engine. A few pre-defined decision criterias based on common use cases such as domain in an email address or IP address/CIDR range are provided. A custom decision criteria can be created by defining a regular expression extraction rule from the value of the parameter.

The cases within the decision switch consists of key-value pairs where the *key* is the value extracted off the decision criteria runtime parameter and the *value* is an existing Authority that is online and in-scope for routing the policy evaluation to. The Authority needs to be online and have at-least one parameter in common with the runtime parameters configured for the Decision Authority in order for it be eligible for selection as the authority to forward the evaluation to. An e.g. is a Decision Authority that takes 2 parameters - a email address (parameter name *email*) as a user identifier parameter and an IP address (parameter name *ipAddress*) that is used as the decision criteria parameter. An eligible Authority that can be selected for routing the policy evaluation must have at-least one parameter in common (e.g the *email*). The parameters received by the Decision Authority is passed to the matching parameter inputs of the authority the policy evaluation routes to.

Configuring the Decision Authority

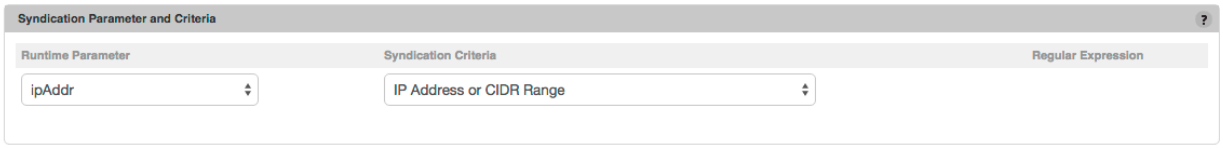
To configure the authority, complete the following steps.

1. Sign into the Administrative Console
2. Click **Create New Authority.**
3. Select **Decision Authority** from the **Authority Type** drop down.
4. Type a name for the authority in the **Authority Name** box.
5. Type a description of the authority in the **Authority Description** box (optional)
6. Use the **Runtime Parameters** area to add and configure the parameters without literal values. The values of these parameters will be supplied by the end user at runtime. For each runtime parameter, specify the following:
    - Type the name of the parameter in the **Name** box. The parameter name gets paired with the value provided at runtime and sent to the custom REST authority.
    - Type the label of the box displayed to the end user in the **Display Name** box.
    - If the parameter will contain a sensitive value, such as personally identifiable information, select the **Obfuscate** check box. This instructs Resilient Access to substitute an opaque token for the value as it transits the network, ensuring that the value never passes through the central Policy Workflow Engine component and does not get stored in the Trust History.
    - If the user will provide the value in the initial request form, select the **Initial Request** check box. NOTE: Resilient recommends leaving the **Initial Request** check box blank if the value is sensitive or contains personally identifiable information.
    - Select the **Mask Input** check box to mask the values with bullet characters as the user types them in. This protects against shoulder surfing.

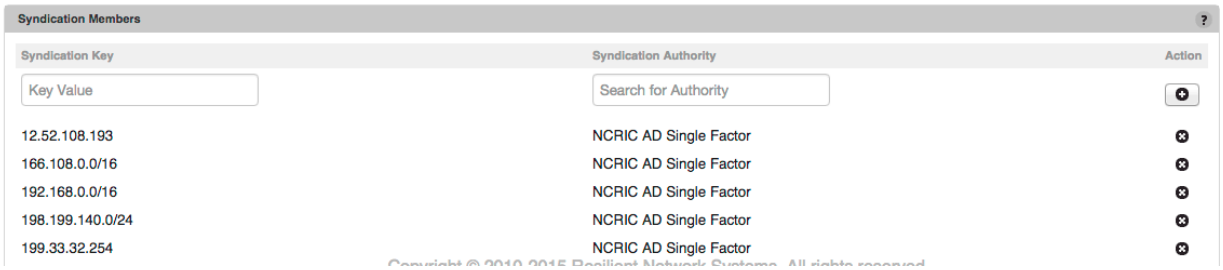
7. Configure the decision runtime parameter and the decision criteria:
    1. Select the runtime parameter the decision will be based on in the **Runtime Parameter** drop-down.
    2. Select the decision criteria to use in the **Decision Criteria** drop- down. Select between the following options:
        1. Select *Parameter Value* if the entire parameter value will be used as the decision criteria
        2. Select *Extract Domain from Email* if the decision criteria is based on the domain in an email address
        3. Select *Extract sub-domain from Hostname* if the decision criteria is based on the sub-domain in a hostname of a URL
        4. Select *IP Address or CIDR Range* if the decision criteria is based on a IPV4 IP address value e.g. 192.168.1.1 or within a CIDR Range such as 192.168.1.0/24 which includes all IP addresses between 192.168.1.0 - 192.168.1.255
        5. Select *Custom Regular expression to extract from Parameter Value* if the decision criteria is extracted from a runtime parameter based on the specified regular expression. In this case the regular expression must be specified in the **Regular Expression** box.
8. Configure the Key-Value pairs that make up the routing options for the Decision Authority
    1. For each routing option enter the key in the **Key Value** box and start typing the authority name in the **Search Authority** box and the Authorities that are eligible and match the partial name entered will appear. One of the eligible authorities must be selected and clicking the + button will add the authority.
    2. If you wish to route the evaluation to an authority if none of the key-value pairs match then check the **Has a no match case** box and select the default authority to route to by entering the partial name for the authority in the **Authority to invoke if no match** box
9. Once you have finished configuring the Decision authority, click **Create** or **Save**.

Step 1: Configure the runtime parameters for the Authority

Step 2: Configure the parameter and criteria of the syndication



Step 3: Configure the members of the syndication



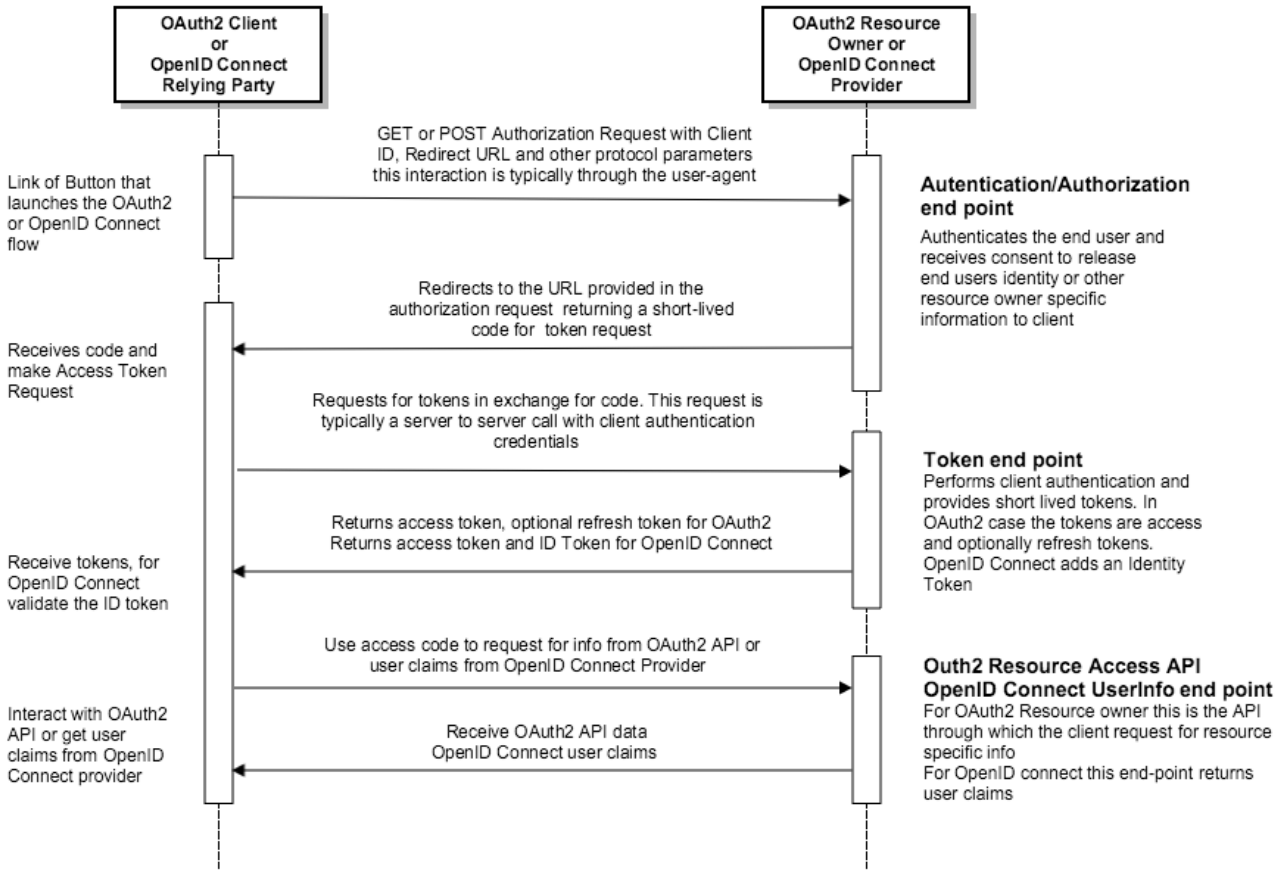
OpenID Connect

OpenID Connect 1.0 is a REST style authentication protocol that provides an identity layer over the OAuth 2.0 (OAuth2) protocol. OAuth2 is a ubiquitous authorization protocol that is used for various integration functions both at the API layer and the application layer.

OAuth2 and OpenID Connect 1.0 Primer

This section briefly describes the OAuth2 and OpenID Connect workflows. If you are an OAuth2 expert, you can skip to the next section.

The diagram below describes a typical Oauth2 and OpenID Connect interaction:



OpenID Connect extensions

While OAuth2 is a framework for performing end-user or API authorization to gain access to a secure API provided by a Resource owner, OpenID Connect is an authentication protocol that interacts with an end-user to authenticate the user and then provide a set of user identity claims to the OpenID Connect Relying Party (RP). In addition to the access token, the OpenID Connect protocol requires the OpenID Connect Provider (OP) to return a ID token. This ID token must be a JSON Web Token (JWT) and have a specific set of user identity claims that are specified here. The Relying Party is expected to validate the ID token before it requests for user identity information.

The user info end-point of the OP should provide OpenID Connect standard user identity claims in the specified attribute names and OP specific additional claims that it wishes to provide for a valid access token.

Resilient Access OpenID Connect Provider

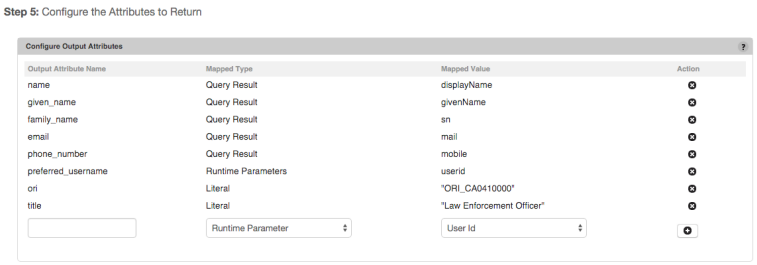
The Resilient Access OpenID Connect Provider is fully compliant with the OpenID Connect 1.0 and OAuth 2.0 specs. It provides an intuitive and customizable interface for building OpenID Connect Relying Party applications that can make full use of the network based access management workflow capabilities in TNaaS. It provides feature to configure the authentication user experience to match your organizations branding and themes. It also provides additional functionality beyond the OpenID Connect spec to make it easier to integrate with your Relying Party applications such as configuration options for returning application specific claims and logout functionality. The following sections provide step by step guide to build your OpenID Connect Relying Party application

Configuration

The first step is to create the Resilient Access policy that will be used for OP authentication. The policy can use all the Resilient Access capabilities for building an access management workflow. In this example we will create a two factor authentication policy that authenticates a user against an Active Directory, retrieves the phone number from the active directory and performs 2nd factor phone authentication and retrieves the user info claims through an LDAP/AD Attribute Provider authority.

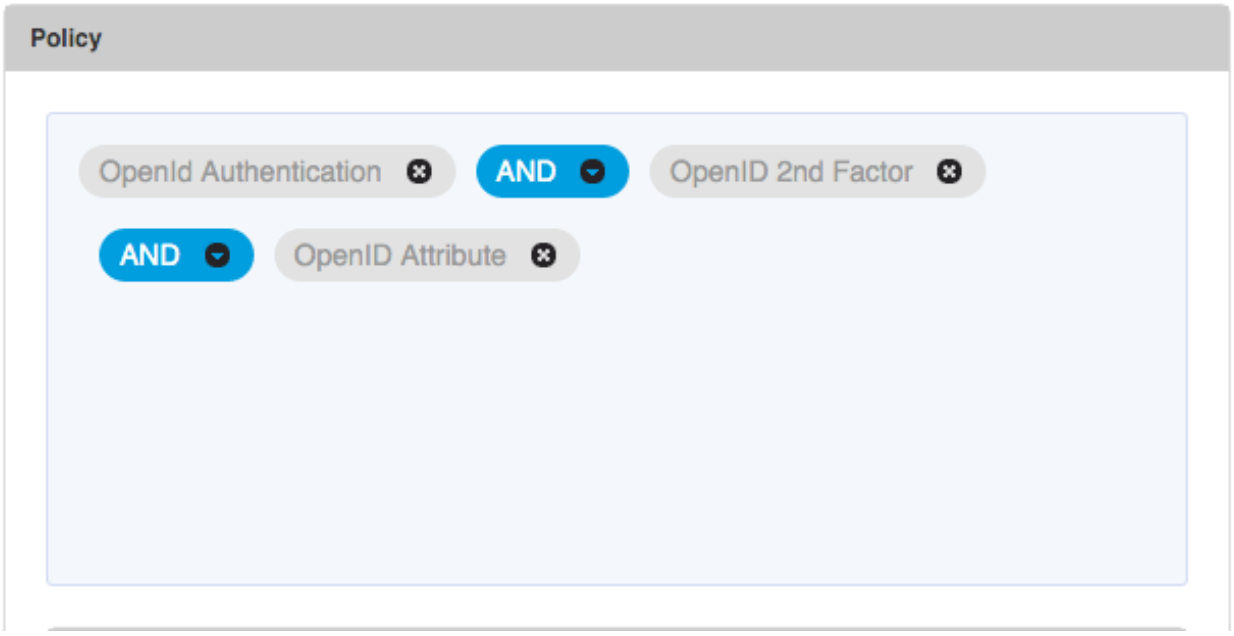
Create the authorities for your authentication policy

1. Create the LDAP/AD Authentication authority to authenticate users against your organization's AD. Deploy the authority connector and make authority online.
2. Create the LDAP/AD Policy Authority to retrieve the phone numbers from your organization's AD and perform phone authentication. Deploy the authority connector and make authority online.
3. Create the LDAP/AD Attribute Provider Authority to retrieve the user claim attributes from the AD and any other attributes that you wish to return to your RP application as additional claims. Map the AD attributes to the corresponding OpenID Connect standard claims in the *Configure Output Attributes* section of the LDAP/AD Attribute Provider authority. Deploy the authority

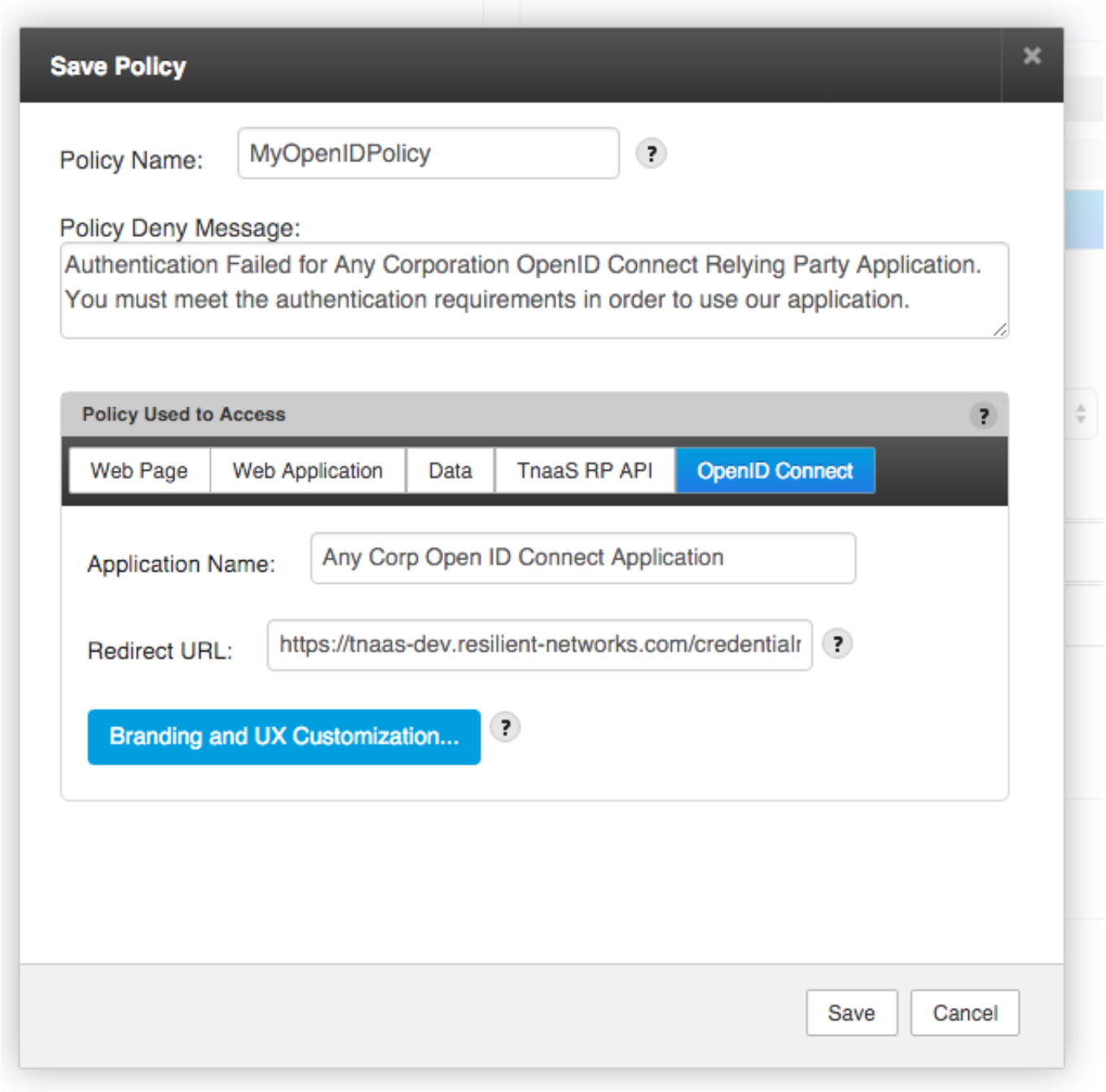


Create the authentication policy

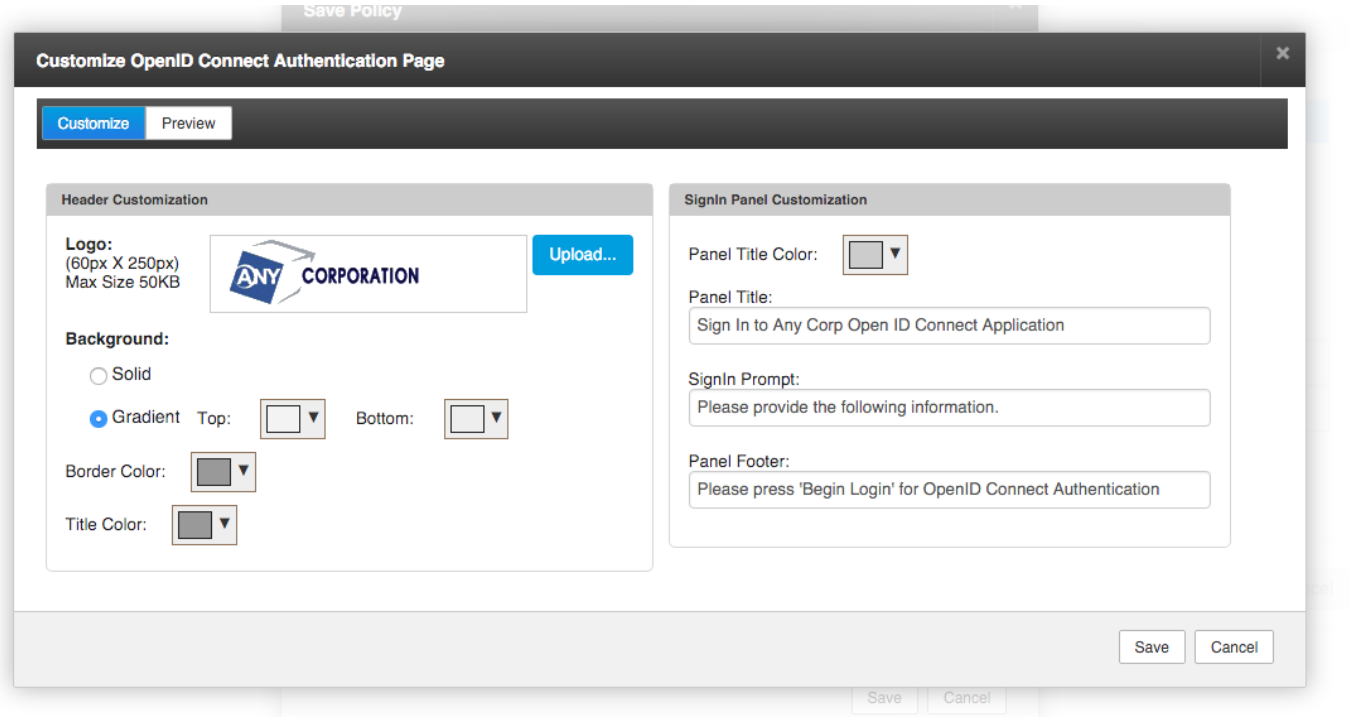
Now that the authorities required for the policy have been created and deployed on your authority connector server, we can create the authentication policy by simply dragging and dropping the authorities into the *Policy* panel. For our example 2 factor authentication policy the policy expression looks as follows:



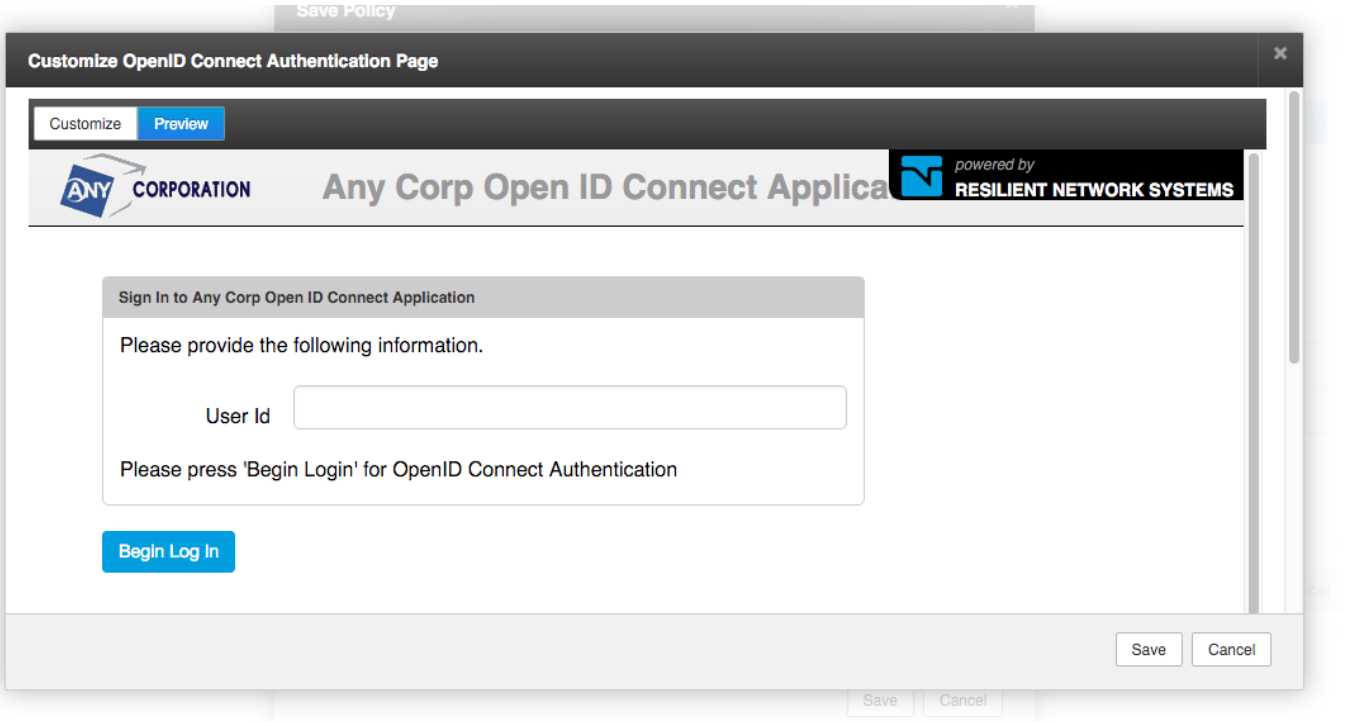
Click next to complete the creation of the Policy. Specify a name and your customized deny message and select *OpenID Connect* from the *Policy Used to Access* panel. For your OpenID Connect application specify the *Application Name* and *Redirect URL*. As per the OpenID Connect spec the Redirect URL specified in the configuration must match the `redirect_uri` parameter that is passed to the **/openId/authenticate** end-point.



Click the *Branding and UX Customization...* button to brand and theme the authentication UX to match your application.



You can instantly preview how your authentication start screen will appear through the Preview tab.



Save the *Customize OpenID Connect Authentication Page* and the *Save Policy* popups and you can now view the OP integration details by clicking the icon next to OpenID Connect authentication policy you just created. Resilient Access generates a Client ID and Client Secret and provides the OP end-points for your RP application.

Test the OpenID Connect Integration

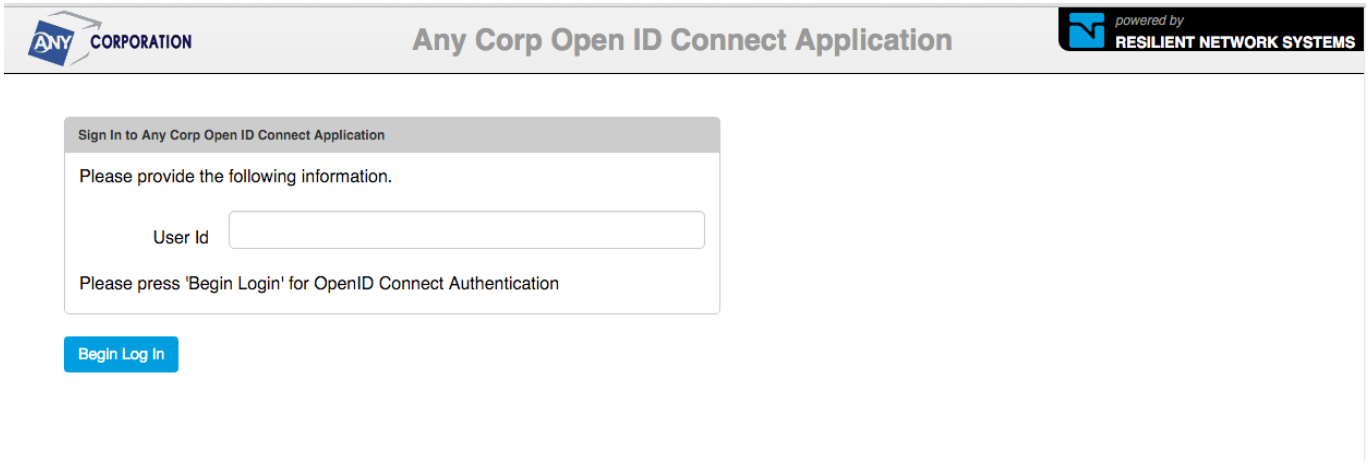
You can quickly test the OpenID Connect integration using our online OpenID Connect sample client at the URL:

https://oidc.resilient-networks.com/login?clientid=<YOUR_CLIENT_ID>&secret=<YOUR_CLIENT_SECRET>

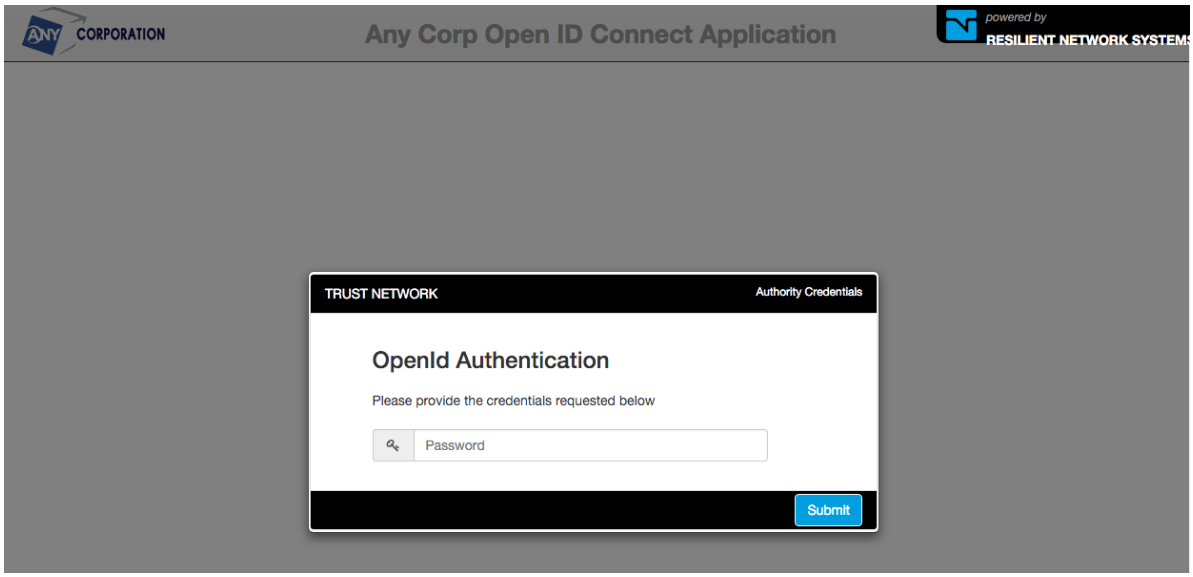
Replace YOUR_CLIENT_ID and YOUR_CLIENT_SECRET with the values from your OpenID Connect Policy details popup. The exposure of the Client Secret in the RP Client app is just for demo purposes, in a real world application the client secret is never sent as a part of web browser HTTP request.

The following screenshots shows the authentication flow for our customized OpenID Connect RP application

From a link or button in the OpenID Connect RP app, the **/openId/authenticate** end-point is called which lands the user into the customized authentication page.

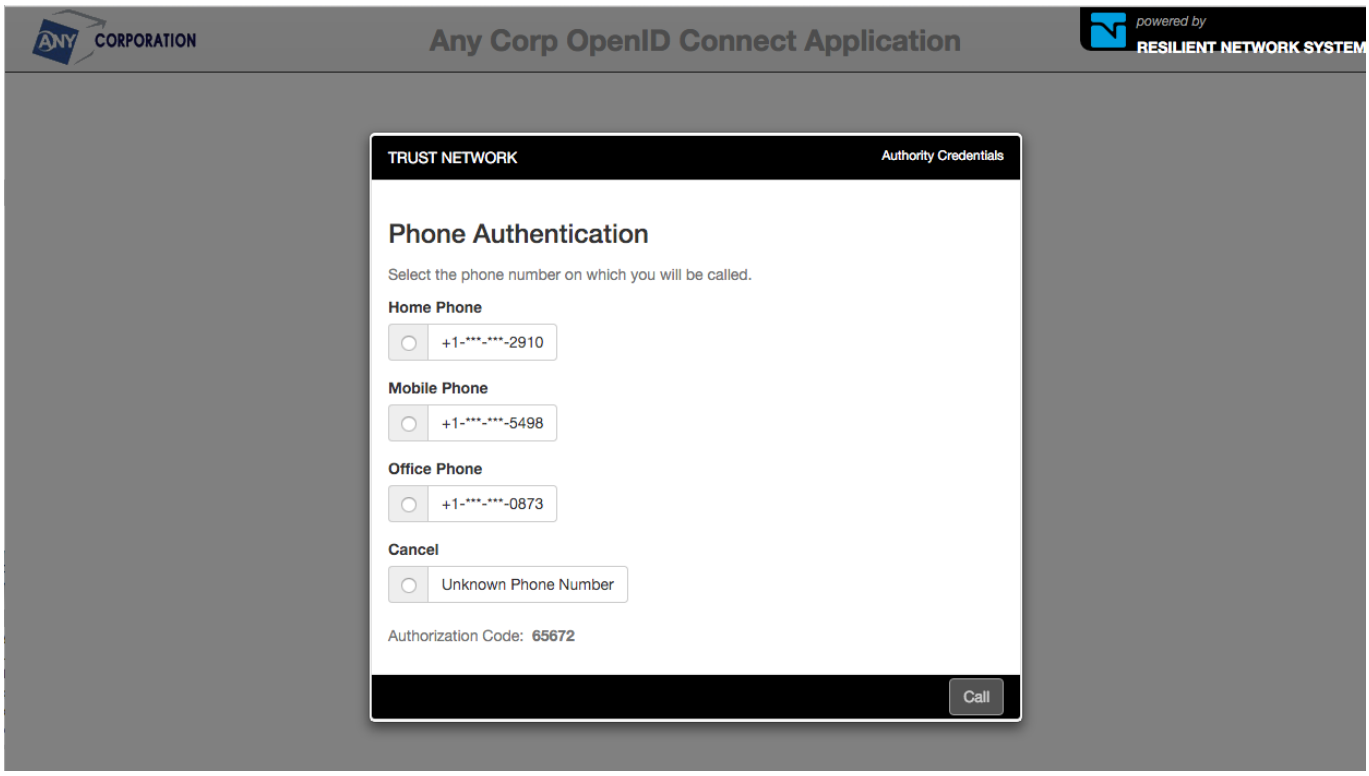


Enter the userid/email and click *Begin Log In* to start the authentication process. Will be prompted for

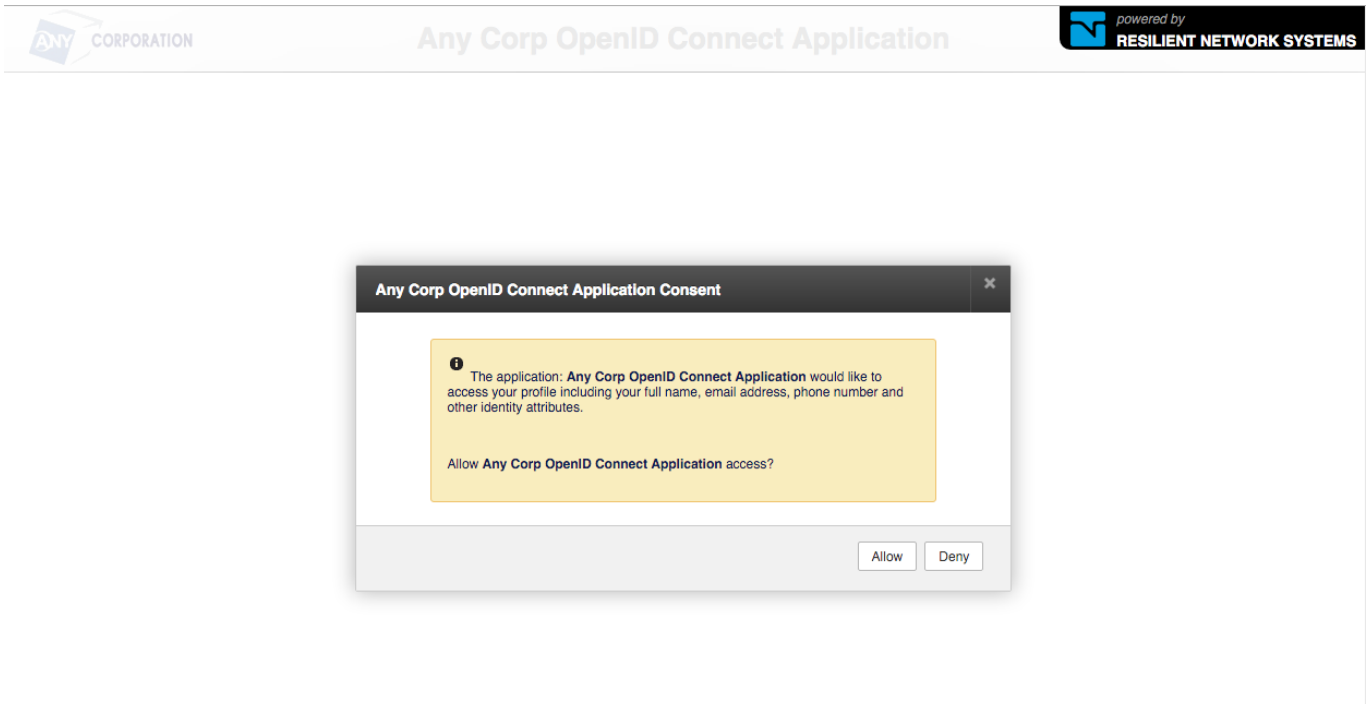


the password.

After passing through the AD Authentication, the 2nd factor phone authentication will be prompted.



Selecting a phone number, will call the phone and the voice prompts will prompt for the authorization code. After successfully completing the second factor authentication, the OpenID Connect consent dialog will be displayed.



On clicking *Allow* the authentication flow will complete and control will be passed to the *Redirect URL* for exchanging the code for the access token and retrieving the end-user claims.

Implementing the OpenID Connect Client

Now that you have configured your OpenID Connect RP application's authentication policy you are ready to implement the client code in your preferred development environment. Resilient Access implements the Authentication using Authorization Code Flow of the OpenID Connect spec. The sections below has the details of the OpenID Connect end-points Resilient Access provides. Resilient Access provides a minimal reference OpenID Connect client implementation in GoLang on GitHub. You can use this as the starting point for your OpenID Client implementation.

Authenticate end-point

```
Endpoint URL: https://tnaas.resilient-networks.com/openId/authenticate
Method: GET or POST
Parameters:
    scope: openid - REQUIRED
    response_type: code - REQUIRED
    client_id: <TNAAS Generated Client ID> - REQUIRED
    redirect_uri: <Relying Party Redirect URI after authentication> -
REQUIRED
    state: <Random value from RP to prevent CSRF, XSRF> - OPTIONAL
    nonce: <Prevent replay attacks> - OPTIONAL

Example:
GET https://tnaas.resilient-networks.com/openId/authenticate?
    response_type=code
    &scope=openid%20profile%20email
    &client_id=046024858031286
    &state=af0ifjsldkj
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb HTTP/1.1
```

- The `scope` parameter must contain the value **openid**. It can contain other scopes defined in OAuth2 and OpenID Connect specs
- The `response_type` parameter must be **code**.
- The `redirect_uri` parameter must be exactly the same as specified in the *Redirect URL* field in the TNaaS policy configurations not including query and fragment portions. You may pass instance specific query parameters in the redirect URL. The `redirect_uri` may be URL encoded.
- If this end-point is called in a HTTP POST the parameters must be passed in `application/x-www-form-urlencoded` format.

Token end-point

The token end-point performs client authentication before it will process the request from the RP. TNaaS OP uses the `client_secret_jwt` type of OpenID Connect Client Authentication. This means the RP application must create a JSON Payload as specified below. The signature of the JWT must use HMAC SHA256 (`HS256`) Algorithm as per the JWT spec. The secret for the HMAC is the client secret provided when the policy is created in Resilient Access. The ID token is also a HMAC SHA256 JWT with the required fields as per the ID Token specification.

```
Endpoint URL: https://tnaas.resilient-networks.com/openId/token
Method: POST
Headers:
    Content-Type: application/x-www-form-urlencoded
    Accept: application/json
Post Parameters:
```

```
    grant_type=authorization_code
    code=<The code received in the redirect from /authenticate>
    redirect_uri=<The redirect URL specified in the policy config>
    client_assertion_type=urn:ietf:params:oauth:client-assertion-
type:jwt-bearer
    client_assertion=<The client assertion JWT>

Client Assertions JWT Payload:
{
   "iss": <The Client ID from TNaaS>,
   "sub": <The Client ID from TNaaS>,
   "aud": "https://tnaas.resilient-networks.com/openId/token",
   "jti": <The unique identifier from RP, e.g. a UUID>,
   "exp": <The time in milliseconds when the client assertion will exp
ire>,
}

Response:
   Content-Type: application/json
   Response Body:
   {
       "access_token": "<The access token for getting user info>",
       "token_type": "Bearer",
       "expires_in": "<The time in seconds when the authenticated cred
entials will expire>",
       "id_token": "<Authentication JWT>"
    }

Example:
POST /token HTTP/1.1
  Host: tnaas.resilient-networks.com
  Content-Type: application/x-www-form-urlencoded
  Body:
    grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
    &client_assertion_type=urn:ietf:params:oauth:client-assertion-
type:jwt-bearer
    &client_assertion=eyJhbGciOiJSUzI1N....

Response:
{
    "access_token":"P1tWrH8whlm1Gh7”,
    "token_type":"Bearer”,
    "id_token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJodHRwc
zpcL1wvdG5hYXMtZGV2LnJlc2lsaWVudC1uZXR3b3Jrcy5jb21cL29wZW5JZCIsInN1YiI
6ImQ2MjAyY2MyLTdhYmEtNGQ2OC04ZmNiLTExMzZkMmMyZTc0ZSIsImF1ZCI6IjgxMDU1N
```

```
jcwNzgwNTA0NCIsImV4cCI6MTQ1NTA1NzE4NDgzOSwiaWF0IjoxNDU1MDU2ODg0ODM5LCJ
hdXRoX3RpbWUiOjE0NTUwNTY4ODIwMjQsIm5vbmNlIjoiNjA0ZjFhMjMtOTY3ZC00ZTcxL
WIwMTYtY2Y5ODRiMmYwODY0In0.nqezBftUtnCtjConcLKn_4_QYwo2xB3_YyHV8Y522s0
”,
    "expires_in":1201,
    "state":"7150dab6-1267-4413-8244-28c99c400d66”
}

The JSON Payload of the ID Token:
{
    "iss":"https:\/\/tnaas-dev.resilient-networks.com\/openId”,
    "sub":"d6202cc2-7aba-4d68-8fcb-1136d2c2e74e”,
    "aud":"810556707805044”,
    "exp":1455057184839,
    "iat":1455056884839,
    "auth_time":1455056882024,
    "nonce":"604f1a23-967d-4e71-b016-cf984b2f0864”
}
```

- The `grant_type` parameter should be **authorization_code**
- The `code` parameter should be the code received in the redirect from the */authenticate* end-point
- The `redirect_uri` parameter should be exactly the same as the *Redirect URL* field configured in the policy in Resilient Access
- The `client_assertion_type` parameter should be **urn:ietf:params:oauth:client-assertion-type:jwt-bearer**
- The `client_assertion` JWT should use the HMAC SHA256 algorithm and payload should have the attributes shown above.
- The RP Client should successfully decode and validate the ID Token JWT before proceeding to the next step

User Info end-point

The last step of the OpenID Connect protocol is for the RP application to call the /token end-point to get the identity claims for the authenticated user.

```
Endpoint URL: https://tnaas.resilient-networks.com/openId/userinfo
Method: GET
Header:
    Authorization: Bearer <Access Token>

Response:
Header:
    Content-Type: application/json
Body:
{
    "sub":"<The user id in ID provider>",
```

```
    <user info attributes specified in config>
}

Example:
GET /userinfo HTTP/1.1
  Host: tnaas.resilient-networks.com
  Authorization: Bearer SlAV32hkKG

HTTP/1.1 200 OK
  Content-Type: application/json
  {
   "sub": "248289761001",
   "name": "Jane Doe",
   "given_name": "Jane",
   "family_name": "Doe",
   "preferred_username": "j.doe",
   "email": "janedoe@example.com",
   "picture": "http://example.com/janedoe/me.jpg"
  }
```

- The Authorization header must be provided with value of **Bearer <Access Token>**

logout end-point

The Resilient Access OP provides an extension to the OpenID Connect spec that implements application logout functionality that can be useful for RP applications.

```
Endpoint URL: https://tnaas.resilient-networks.com/openId/logout
Method: GET
Parameters:
    client_id: <TNAAS Generated Client ID> - REQUIRED
    redirect_uri: <Relying Party Redirect URI after logout> - REQUIRED
    sub: <The sub attribute in the ID Token and /userinfo response> -
REQUIRED
```

- The `sub` parameter should be set to value returned in the `sub` attribute of the ID Token and */userinfo* responses

SAML

SAML is the most commonly used enterprise standard for access control and is used by most enterprise IdAM vendor integrations. It is an XML protocol that involves a SAML service provider (SP), the RP application and the SAML identity provider (IdP), Resilient Access in this case, publishes their respective SAML metadata XML files and provides it to the other party. The metadata XML encrypts the public key of the RSA key pair that is used to encrypt the data that is exchanged between the two sides. The metadata also specifies the identity attributes the IdP will return to the SP application on successful authentication. Briefly, the interaction begins with the SP application sending an AuthN request to the IdP which involves a redirection to the IdP user interface for performing the authentication that may involve verification of user credentials and on successful authentication results in the IdP sending a signed and encrypted SAML assertion with the identity attributes of the user being authenticated. More details on the protocol on Wikipedia

Resilient Access fully supports the SAML 2.0 standard and has pre-built integrations with some of the most popular cloud platforms and service including Microsoft Office 365, Google G Suite, Salesforce, Box among others. Cloud service or internal applications that are currently not integrated can be added by our professional services team. Resilient also provides customer branded Single Sign On (SSO) portal that combines all the cloud services and internal applications that are used by the customer's workforce.

Any policy created using the rich policy workflow infrastructure in RA can be used as the access control policy for SAML SSO. Due to the complexities of the SAML protocol, RA Admin Console currently does not have the UI to configure a SAML SP - RA IdP integration. Developers/IT staff can still build their custom access policies and contact Resilient sales and professional services team to deploy their apps within their custom branded SSO portal.

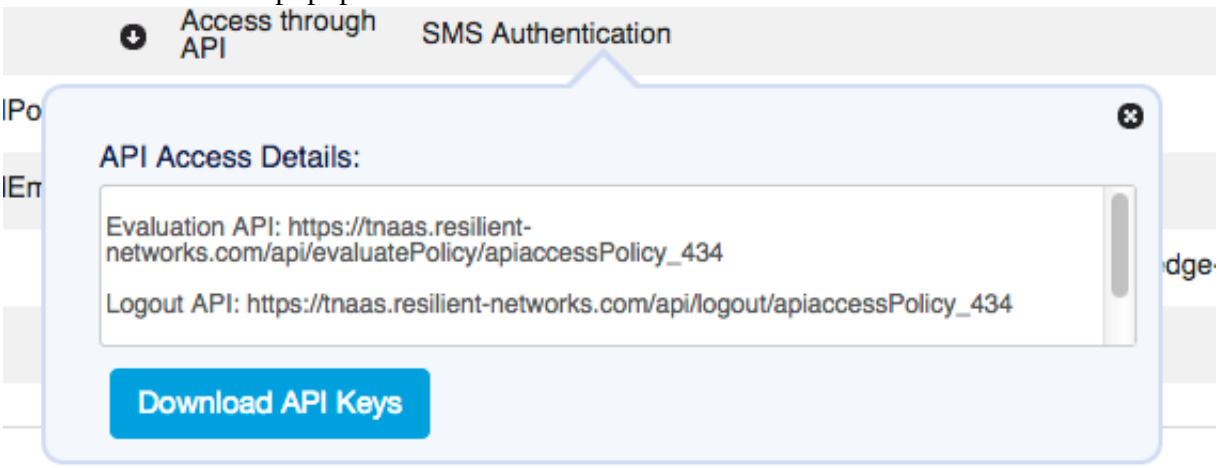
Relying Party API

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:



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.

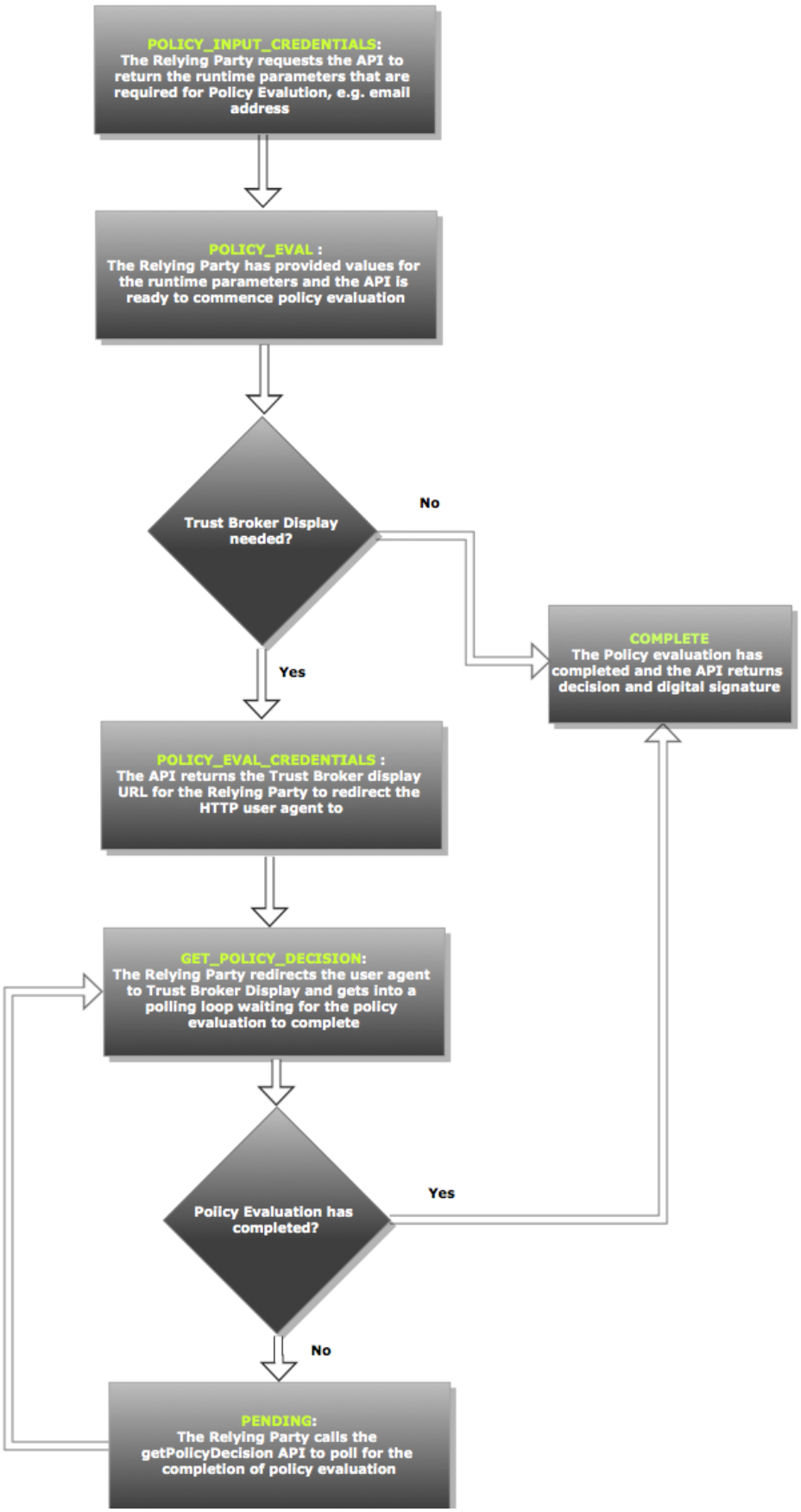
- Response Public Key: The public key of a RSA Key Pair. Resilient Access will always generate an *X-SIGNATURE* response header using a similar process as described above. The Base 64 encoded SHA-256 value of the response body is encrypted with the Response Private Key and this value is set to the *X-SIGNATURE* 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



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)
}
```

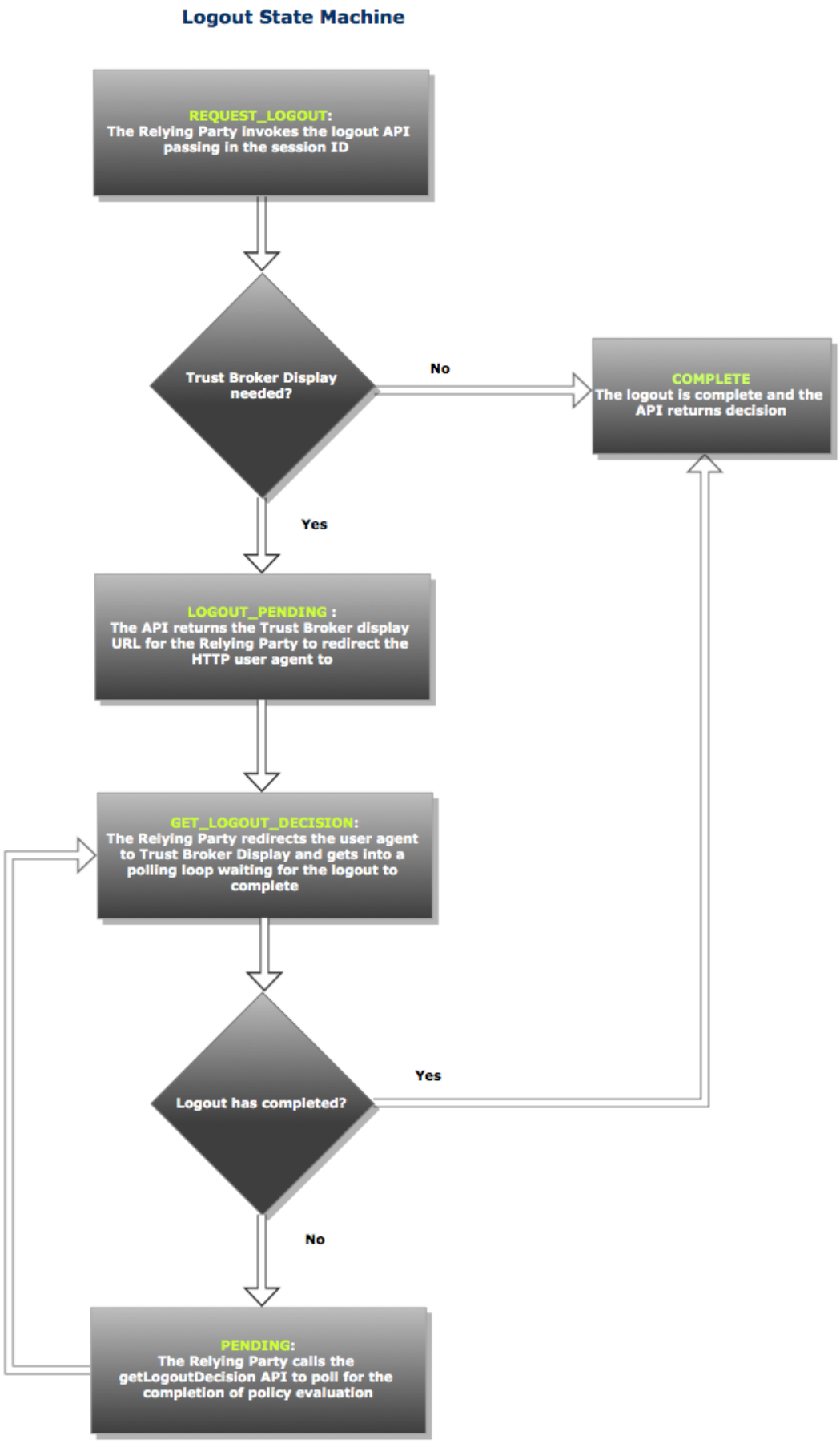
Protocol Errors

- 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)
}
```

Trust Tag

Trust Tag is a java script file hosted by Resilient Access that executes when the page that contains the SCRIPT tag is loaded in the browser. Trust Tag is a Resilient Access relying party implementation. The Trust Tag script tag should be the first script tag defined in the head tag of the HTML source.

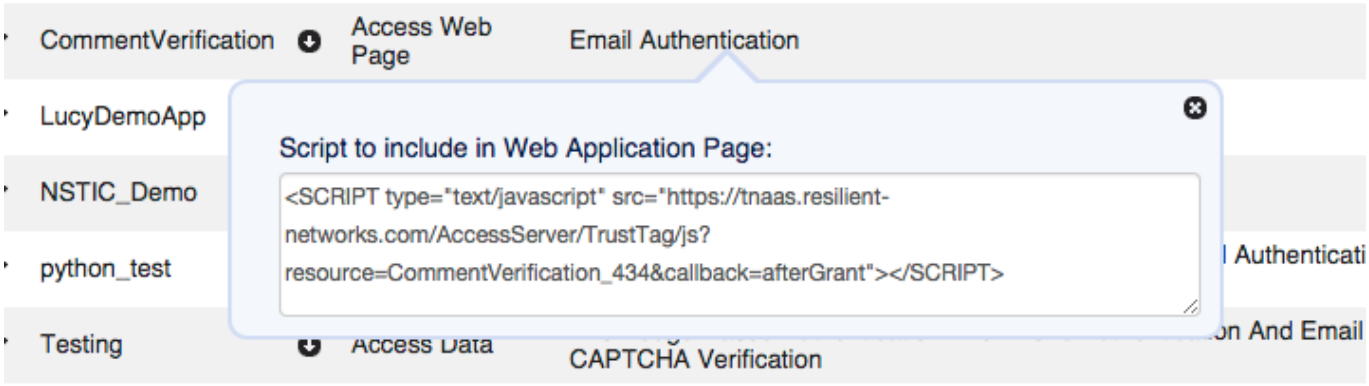
There are two variations of Trust Tag:

- Web Page - Useful for enforcing a policy on individual web pages or parts of an application
- Web Application - Useful for enforcing a policy on web applications to securely control access to a web application

Access Web Page

This variant of Trust Tag is well suited for enforcing a policy in a web page or specific sections of a web page. In this interactive guide we have used the Email Authentication policy to verify the user is in possession of the email address they entered before they can post a comment.

This variant of Trust Tag enforces the policy purely through Javascript. After creating a policy of type **Access Web Page**, from the Policy list click on the icon next to the policy to use. It will display the HTML SCRIPT tag code to insert into your HTML or server script page as shown below:



Simply copy the SCRIPT tag and insert it into the <head> tag of each web page that you wish to enforce the policy on. This script tag should be the first SCRIPT tag in the HTML source. This will instantly integrate the Trust Network policy evaluation engine into your web page through javascript injection.

Since this variant of Trust Tag enforces the policy through javascript after the server has returned the response for the web page, it is recommended to dynamically build the web page using AJAX techniques so that sensitive information is only visible to the end user if the policy evaluation succeeds. Trust Tag facilitates this by invoking a Javascript function called "afterGrant" if it is defined in the page. It is recommended to use AJAX to retrieve and render contents of the page in this callback function.

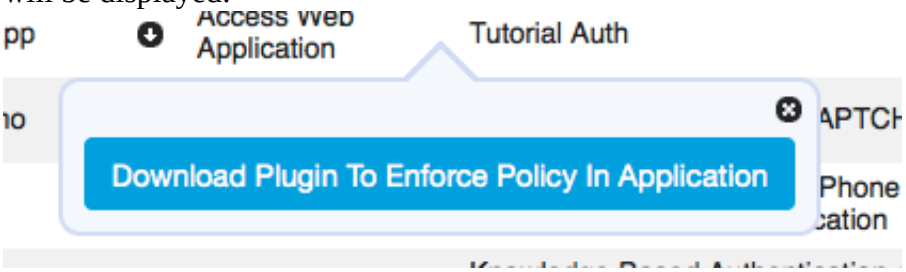
Access Web Page Trust Tag uses Cross Origin Resource Sharing (CORS) to facilitate the secure execution of cross-domain Javascript code. This feature works in all modern browsers as shown in the Browser Support section of the Wikipedia link. In order to successfully execute Access Web Page Trust Tag, the web page should be served from the domain value specified in the **Application Hosting Domain** field including the HTTP scheme, e.g. ***https://www.example.com***

Access Web Application

The Web Application variant of Trust Tag is well suited for access control of web applications. In this interactive guide we have used a combination of Database Policy Authority and Phone Authentication to demonstrate web application access control through Trust Tag. This interactive guide is available to users who have created an account on Resilient Access. The Database Policy Authority extracts the name and phone number entered during user registration and passes the name as an identity attribute to this Wordpress application and performs a Phone Authentication using the phone number.

Access Web Application Trust Tag consists of a few server side scripts that are available as a plugin for a web application. By adding a single line of code in the main template/layout file of the web application, access to the application controlled through the policy defined in Resilient Access. The steps for integration with your web application are as follows:

1. Create the policy you wish to enforce in Resilient Access, you must specify a valid hostname where the web application will be hosted, e.g. ***https://example.com***. Select the **Access Web Application** tab and then select the W**eb Application Technology** your web app is built using and the name of the web application (context path). Enter "/" if the web application will be installed in the root context.
2. In the policy list page click on the icon in the **Policy Used For** section, the following popup will be displayed:



3. Click on the **Download Plugin to Enforce Policy in Application** button to download the plugin ZIP file.
4. Extract the plugin ZIP into the root directory of the web application
5. In the main template/layout file of the web application enter a single server side include statement at the top of the page to enforce the policy for the web application. e.g

```
For PHP: <?php include 'rns/enforcePolicy.php';?>

For JSP: <%@include file="rns/enforcePolicy.jsp"%>

For Python: from rns import enforcePolicy
```

Access Web Application Trust Tag uses a cookie to keep track of policy evaluation context and the parameters for the policy. This cookie is created in your applications domain and is named as follows:

RNS.<app name>.Policy. The expiration of the cookie is set to the value specified for **Access expires in** field while creating policy. The policy will be enforced again when the expiration time is reached.

The plugin consists of the following four server scripts:

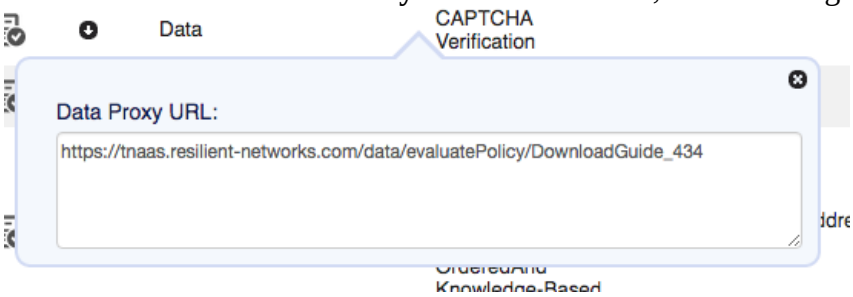
1. **enforcePolicy**: This is the entry point script that is referenced through the server side include. It checks if the Trust Tag cookie exists, if not, it will redirect to the login script to enforce the policy. If cookie exists, the application pages will be rendered. This script file also contains the javascript code that injects the widget consisting of the Logout button and policy parameters that you see in the top right corner of the application.
2. **login**: The only content in this file is the Trust Tag script tag embedded in <head> section. The policy evaluation UI is rendered in this page. If you wish to render your applications header/footer or other widgets during policy evaluation you may edit this file.
3. **postCredential**: This script will be invoked by Trust Tag when the policy evaluation is successful. This script received policy evaluation context information. The script creates the cookie and places the context information in it.
4. **logout**: Called when the Logout button is clicked. It removed the Trust Tag cookie and closes the Trust Network session and redirects to the home page of the application which will again redirect to the login page through the **enforcePolicy** script

Access Web Application Trust Tag uses Cross Origin Resource Sharing (CORS) to facilitate the secure execution of cross-domain Javascript code. This feature works in all modern browsers as shown in the Browser Support section of the Wikipedia link. In order to successfully execute Access Web Page Trust Tag, the web page should be served from the domain value specified in the **Application Hosting Domain** field including the HTTP scheme, e.g. ***https://www.example.com***

Data Proxy

Data Proxy allows you to control access to documents and data APIs through Resilient Access policies. The data API or document that is protected through the policy is referred to as Asset. The steps for setting up a **Data Proxy** policy is as follows:

1. Create a Policy in Resilient Access admin console referencing the authorities you wish to use to control access to the asset. Select the **Data** tab from the *Policy Used to Access* section. The *Data Source:* field has two options, **Web URL** and **File/Folder in Box**. This page covers the **Web URL** data source type, please refer to Box Integration for **File/Folder in Box** option. Enter the URL to the asset in the *Asset URL* field and select whether the asset should be downloaded or displayed in the *Asset Retrieval Type* field. Only a limited set of content-types can be displayed such as PDF files, popular image formats etc. If the asset cannot be displayed, a download option will be provided.
2. In the Policy list page click on the icon in the **Policy Used For** column, the following



   The URL displayed in the popup can be sent to the recipients, embedded in your web application with an iframe, etc. It will act as a proxy for your asset. It will first execute the policy and if it evaluates to a GRANT, the asset will be retrieved from the URL specified and either embedded in the *Content Viewer* panel of the **Data Proxy** result page for the **Display** option or can be downloaded for the **Download** option.