

ISAM cookbook for Advanced API Protection

Mehrdad Abdi

Ebrahim Khalilzadeh

Version 1.0.1

February 2019



Release Date	Version	Authors	Comments
15 Dec 2018	1.0.0	Mehrdad Abdi Ebrahim Khalilzadeh	
1 Feb 2019	1.0.1	Mehrdad Abdi Ebrahim Khalilzadeh	

Table of Contents

1 Introduction	8
1.1 API definition	8
1.2 API security concepts	9
1.3 OAuth protocol	10
1.3.1 OAuth 2.0 workflow	12
1.3.1.1 Authorization code flow	12
1.3.1.2 Authorization code flow with refresh token	14
1.3.1.3 Implicit grant flow	15
1.3.1.4 Resource owner password credentials flow	16
1.3.1.5 JWT and SAML bearer grant type flows	17
1.3.1.6 Client credentials flow	17
1.4 Multi-factor authentication mechanisms	18
1.4.1 ISAM Mobile Multi-Factor Authenticator	19
1.5 API gateways and ISAM	19
1.5.1 API gateways – Resource Servers	19
1.5.2 ISAM – Authorization Server	20
2 Architecture	22
2.1 High level Architecture	22
2.2 Required Components	22
2.2.1 Access Manager Virtual Appliance ISO Image	22
2.2.2 Access Manager 9.0.5 Activation Codes	23
2.2.3 Mobile Device running IBM Verify App	23
2.2.4 Host machine running VMWare	23
2.2.5 VMWare Networking	23
2.2.6 Hosts file	24
2.3 Lab scenario overview	24
3 Create and configure backend servers	26
3.1 Hostnames	26
3.1.1 Add hostnames to windows hosts files on the host machine	26
3.2 Subnet Address Settings	27
3.3 Virtual Machines Creation	29
3.3.2 ISAM Virtual Machine Creation	29
3.3.2.1 Add hostnames to ISAM	29
3.3.3 Merchant Virtual machine creation	32
3.3.3.1 Set IP for Merchant virtual machine	32

3.3.3.2 Configure hosts file	33
3.3.3.3 Install git, python3 and pip3	34
3.3.3.4 Create and configure Merchant server	35
3.3.4 Apps Virtual machine creation	36
3.3.4.1 Set IP for Apps virtual machine	36
3.3.4.2 Configure hosts file	37
3.3.4.3 Install git, python3 and pip3	38
3.3.4.4 Create and configure API server	39
3.3.4.5 Create and configure customer dashboard	39
4 Create and configure ISAM reverse proxies	42
4.1 Create API Server reverse proxy	59
4.2 Create customer dashboard reverse proxy	59
4.3 Create Junctions	50
4.3.1 Create /api junction	50
4.3.1.1 Test /api Junction	53
4.3.2 Create /portal Junction	54
4.3.2.1 Test /portal Junction	57
4.4 Configuring AAC	59
4.4.1 Isamcfg Tool	59
4.4.2 Procedure	59
4.5 Modify Reverse Proxy Instance Configuration File	64
5 Configure OAuth, SCIM, and MMFA	67
5.1 Configure OAuth	67
5.1.1 Create Authenticator Definition	67
5.1.1.1 Create Client	69
5.1.2 Create AppsRegister Definition	71
5.2 Configure SCIM	74
5.2.1 Create an LDAP Server Connection	74
5.2.2 Configure SCIM	76
5.2.3 Create SCIM junction	78
5.2.4 Configure URL filtering for SCIM responses	82
5.2.5 Enable SCIM Demonstration Application	83
5.3 Configure MMFA	85
5.4 Configure Policy Administration	92
5.4.1 Create users	93
5.4.2 /scim ACLs	95

5.4.3 /mga ACLs	98
5.5 Create OAuth Client for Merchant	101
5.5.1 Test Merchant OAuth implementation	104
6 Simple API Protection using OAuth-EAS	108
6.1 Scenario	108
6.1.1 User access delegation	108
6.1.2 API Call	109
6.2 Attach API Protection Policy to APIs	110
6.3 Modify Reverse Proxy Instance Configuration File to enable OAuth-EAS	114
6.4 Test OAuth-EAS	115
6.5 Test Merchant API Call	119
6.6 Scenario drawbacks	121
7 Advanced API Protection	123
7.1 Scenario	123
7.2 Modify Reverse Proxy Instance Configuration File	124
7.3 Import template files	125
7.3.1 Edit metadata.xml	128
7.4 Import mapping rules	129
7.5 Create Authentication mechanisms	133
7.6 Create Authentication Policies	137
7.6.1 Create api_policy_otp	137
7.6.1.1 Create api_policy_otp_init policy	137
7.6.1.2 Create api_policy_otp_resp policy	138
7.6.2 Create api_policy_presence	140
7.6.2.1 Create api_policy_presence_init policy	140
7.6.2.2 Create api_policy_presence_resp policy	141
7.6.3 Create api_policy_finger	143
7.6.3.1 Create api_policy_otp_finger_init policy	143
7.6.3.2 Create api_policy_otp_finger_resp policy	145
7.7 Create Access Control policy	148
7.7.1 Attach Access Control Policies to APIs	152
7.8 Configure IBM Verify	156
7.8.1 Register Authenticator	157
7.9 Test API policies	161
7.9.1 Test Accounts API	162
7.9.2 Test Balance API	163

7.9.3 Test Transfer API	166
8 Context-Based API Protection	169
8.1 Modify Reverse Proxy Instance Configuration File	169
8.2 Define Attributes	170
8.3 OAuth Information Point	174
8.3.1 OAuth Attributes	176
8.4 Account List Information Point	180
8.4.1 AccountList Attribute	182
8.5 Define Context based Policies	184
8.5.1 Create cba_accounts policy	184
8.5.2 Create cba_balance policy	188
8.5.3 Create cba_transfer policy	192
8.5.4 Attach Context-Based Policies to Resources	198
8.5.5 Test Context-Based Access Control	200
8.5.5.1 Test Accounts API with permitted values	200
8.5.5.2 Test Accounts API with not permitted values	203
8.5.5.3 Test Balance API with permitted values	203
8.5.5.4 Test Balance API with not permitted values	206
8.5.5.5 Test Transfer API with Amount value lower than 1000	206
8.5.5.6 Test Transfer API with Amount value greater than 1000	208
8.6 Using AAC Policy to express OAuth Scope	210
8.6.1 Test Lack of Scope validation	210
8.6.2 Define policy to validate scope	214
8.6.3 Create Policy Sets	218
8.6.4 Attach Policy Sets to Resources	224
8.6.5 Test pset_accounts policy for Accounts API	227
8.6.6 Test pset_balance policy for Balance API	228
8.6.7 Test pset_transfer policy for Transfer API	229
9 A deep dive into our api_policy mapping rules	230
9.1 Branching Authentication Policy Using Cache mechanism	231
9.1.1 Pros and cons	233
9.1.2 Configure infomaps to Activate Cache Mechanism	233
9.2 Branching Authentication Policy Using an external service	235
9.2.1 Pros and cons	237
9.2.2 Configure the external service	237
9.2.3 Configure infomaps to Activate external service mechanism	238

9.3 Branching Authentication Policy Using a custom encryption mechanism	241
9.3.1 Pros and cons	242
9.3.2 Configure infomaps to Activate custom encryption mechanism	242
9.4 Branching Authentication Policy Using STS	245
9.4.1 Configuring Chain Templates	247
9.4.2 Configure Module Chains	252
9.4.3 Configure infomaps to Activate STS mechanism	260
10 Appendix I: API Developer Guide	263
10.1 OAuth APIs	263
10.2 Mobile APIs	264
10.3 Dashboard APIs	266
11 Appendix II: IBM Verify connectivity to ISAM by forwarding ports	269
11.1 VMware NAT port forwarding	269
11.2 PLINK remote port forwarding	275
12 Notices	278

About the authors



Software engineering and application development, combined with Information Security experiences, has introduced **Mehrdad Abdi** as a Software Security specialist. He has about 7 years experiences in security engineering, consist of penetration testing of web and mobile applications, developing tools related to software security analyze, secure development training, and secure architecture designing. He received a master of Information Security from Amirkabir University of Technology, Iran, Tehran in 2013.



In the role of software security manager **Ebrahim Khalilzadeh** is responsible for assuring the security of software developed by the company that he works for. Software security standard adoption and security architecture design are the two most important responsibilities that have been taken by him. He graduated from Amirkabir University of Technology in Iran with a major in Information security in 2009. **Ebrahim** has more than 10 years experience in the computer security industry and he held a variety of technical positions before receiving a management appointment. He has a profound knowledge of software development activities in terms of security and knows how to fit security activities in an SDLC in order to ensure the security of software.

1 Introduction

The explosion in consumer mobile adoption, computerization of goods and services, and an increase in data generation have driven a change in the way Internet-based businesses are built and consumed. The digital economy has prompted online organizations to facilitate the creation and exchange of information to new channels, partners, and developers with the goal of unlocking new business value for these consumers. Today, connecting businesses to their cross-channel customers is technically driven by Application Programming Interfaces, or APIs¹.

1.1 API definition

In computer programming, an application programming interface (API) is a set of subroutine definitions, communication protocols, and tools for building software. In general terms, it is a set of clearly defined methods of communication among various components. A good API makes it easier to develop a computer program by providing all the building blocks, which are then put together by the programmer².

Web APIs are the defined interfaces through which interactions happen between an enterprise and applications that use its assets, which also is a Service Level Agreement (SLA) to specify the functional provider and expose the service path or URL for its API users. An API approach is an architectural approach that revolves around providing a program interface to a set of services to different applications serving different types of consumers³.

When used in the context of web development, an API is typically defined as a set of specifications, such as Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages, usually in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format. An example might be a shipping company API that can be added to an eCommerce-focused website to facilitate ordering shipping services and automatically include current shipping rates, without the site developer having to enter the shipper's rate table into a web database. While "web API" historically virtually has been synonymous for web service, the recent trend (so-called Web 2.0) has been moving away from Simple Object Access Protocol (SOAP) based web services and service-oriented architecture (SOA) towards more direct representational state transfer (REST) style web resources and

¹<https://www.akamai.com/us/en/multimedia/documents/white-paper/akamai-strategies-for-api-security-white-paper.pdf>

² https://en.wikipedia.org/wiki/Application_programming_interface

³ http://www.hcltech.com/sites/default/files/apis_for_dsi.pdf

resource-oriented architecture (ROA)⁴. In the social media space, web APIs have allowed web communities to facilitate sharing content and data between communities and applications. In this way, content that is created in one place dynamically can be posted and updated to multiple locations on the web⁵. For example, Twitter's REST API allows developers to access core Twitter data and the Search API provides methods for developers to interact with Twitter Search and trends data.

1.2 API security concepts

Authenticating and Authorizing programs or users who are invoking an API are two most important security concerns. Currently, most RESTful applications leverage OAuth 2.0 for authentication and authorization purpose. Along with OAuth, JWT is the newcomer that is gaining more and more popularity with API developers.

As of December 2018 there is no standard for securing APIs, and providers may use their custom solution to secure APIs.

When used in the context of Account Servicing Payment Service Providers(ASPSPs), the PSD2 brings some guidelines but it doesn't specify exactly how open APIs should be secured. However, EU banks are likely to follow the Open Banking Standard set by the UK's Competition and Markets Authority (CMA). The CMA now requires the use of authorization framework OAuth 2.0 with the additional OpenID Connect authentication layer to protect UK bank open APIs.

OAuth 2.0 is a popular authorization framework in which a user can authorize a Third Party Provider(TPP) without revealing any credentials like account passwords.

Essentially, the third-party application must ask the user for an authorization grant, and use the authorization grant to receive an access token from the API. This access token can then be exchanged for a protected user resource, like bank account information.

While OAuth 2.0 is great at providing authorization, it doesn't provide any authentication. That is, while it allows users to give TPPs access to their bank accounts without giving them passwords, it does not allow TPPs to check who the user is.

That authentication step is crucial—otherwise hackers could potentially grant a TPP access to someone else's account.

⁴ Benslimane, Djamal; Schahram Dustdar; Amit Sheth (2008). "Services Mashups: The New Generation of Web Applications". IEEE Internet Computing, vol. 12, no. 5. Institute of Electrical and Electronics Engineers.

⁵ Parr, Ben. "The Evolution of the Social Media API". Mashable. Retrieved 26 July 2016.

OpenID Connect is an authentication layer which sits on top of the OAuth 2.0 protocol. Before a user can grant the TPP authorization, their identity is checked by an identity provider (IdP). If the user is authenticated, the IdP gives the TPP an identity token and usually access tokens as well which can then be exchanged for user resources. The TPP still doesn't gain access to a user's credentials—that remains with the IdP.

OpenID Connect will allow banks and other ASPSPs to incorporate authentication, but that authentication can't be as simple as a password anymore.

PSD2 requires ASPSPs to use Strong Customer Authentication (SCA), essentially Multi-Factor Authentication (MFA), whenever a user, “accesses its payment account online; initiates an electronic payment transaction; carries out any action through a remote channel which may imply a risk of payment fraud or other abuses.”

Authentication of a payment service user means authentication based on the use of two or more elements that are categorized as knowledge (something the user knows only), ownership (something that only the user has), and inference (something, the user is) and are independent in the sense that the violation of one element does not impair the reliability of the other elements, while being created in such a way as to protect the confidentiality of the authentication data.

The European Banking Authority (EBA) has authorized a few exemptions to this rule, such as when the user is only viewing the balance of a bank account, is performing a recurring transaction, or is paying a parking fare. However, most of the time ASPSPs will need to use SCA.

ASPSPs have an extra incentive to use SCA even when not required. In general, users must cover small losses from unauthorized payments due to lost, stolen, or misappropriated payment instruments (like a phone). Yet, if the ASPSP didn't have SCA, the user isn't liable and the provider must cover all losses.

1.3 OAuth protocol⁶

OAuth is an HTTP based authorization protocol that provides 3rd party applications scoped access to protected resources on behalf of the resource owner. This allows private resources to be shared between sites without obtaining a username and password.

6

https://www.ibm.com/support/knowledgecenter/SSPREK_9.0.5/com.ibm.isam.doc/config/concept/oauth.html

ISAM support OAuth 2.0 authentication and strictly follows the OAuth standards. The following concepts are generally used in OAuth 2.0.

Resource owner	An entity capable of authorizing access to a protected resource. When the resource owner is a person, it is called a user.
OAuth client	A third-party application that wants access to the private resources of the resource owner. The OAuth client can make protected resource requests on behalf of the resource owner after the resource owner grants it authorization. OAuth 2.0 introduces two types of clients: confidential and public. Confidential clients are registered with a client secret, while public clients are not.
OAuth server	Known as the Authorization server in OAuth 2.0. The server that gives OAuth clients scoped access to a protected resource on behalf of the resource owner. The server issues an access token to the OAuth client after it successfully does the following actions: <ul style="list-style-type: none">● Authenticates the resource owner.● Validates a request or an authorization grant.● Obtains resource owner authorization. An authorization server can also be the resource server.
Access token	A string that represents authorization granted to the OAuth client by the resource owner. This string represents specific scopes and durations of access. It is granted by the resource owner and enforced by the OAuth server.
Protected resource	A restricted resource that can be accessed from the OAuth server using authenticated requests.
Resource server	The server that hosts the protected resources. It can use access tokens to accept and respond to protected resource requests. The resource

	server might be the same server as the authorization server.
Authorization grant	A grant that represents the resource owner authorization to access its protected resources. OAuth clients use an authorization grant to obtain an access token. There are four authorization grant types: authorization code, implicit, resource owner password credentials, and client credentials.
Authorization code	A code that the Authorization server generates when the resource owner authorizes a request.
Refresh Token	<p>A string that is used to obtain a new access token.</p> <p>A refresh token is optionally issued by the authorization server to the OAuth client together with an access token. The OAuth client can use the refresh token to request another access token that is based on the same authorization, without involving the resource owner again.</p>

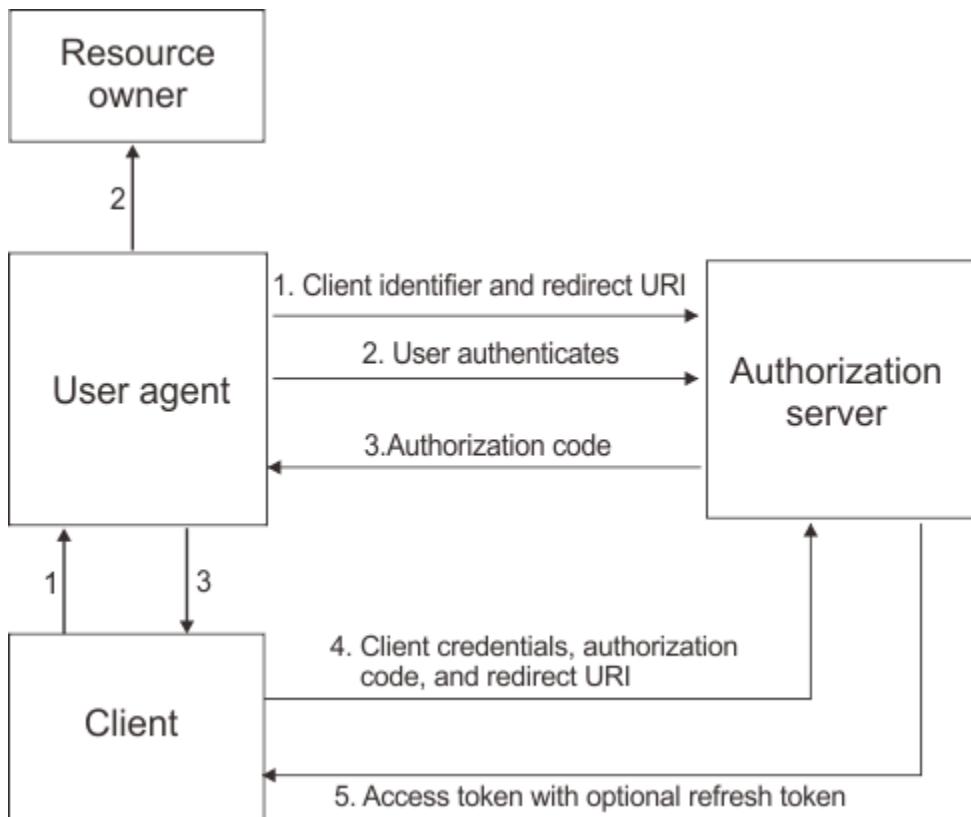
The OAuth 2.0 support in IBM Security Access Manager provides four different ways for an OAuth client to obtain access the protected resource.

1.3.1 OAuth 2.0 workflow

Advanced Access Control supports the following OAuth 2.0 workflows.

1.3.1.1 Authorization code flow

The authorization code grant type is suitable for OAuth clients that can keep their client credentials confidential when authenticating with the authorization server. For example, a client implemented on a secure server. As a redirection-based flow, the OAuth client must be able to interact with the user agent of the resource owner. It also must be able to receive incoming requests through redirection from the authorization server.

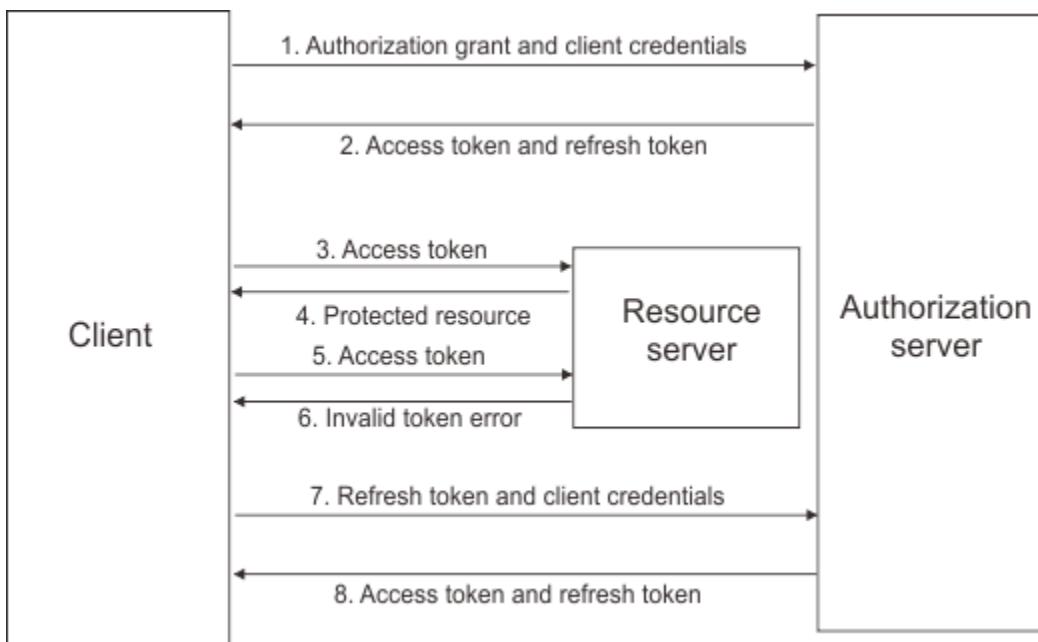


The authorization code workflow diagram involves the following steps:

1. The OAuth client initiates the flow when it directs the user agent of the resource owner to the authorization endpoint. The OAuth client includes its client identifier, requested scope, local state, and a redirection URI. The authorization server sends the user agent back to the redirection URI after access is granted or denied.
2. The authorization server authenticates the resource owner through the user agent and establishes whether the resource owner grants or denies the access request.
3. If the resource owner grants access, the OAuth client uses the redirection URI provided earlier to redirect the user agent back to the OAuth client. The redirection URI includes an authorization code and any local state previously provided by the OAuth client.
4. The OAuth client requests an access token from the authorization server through the token endpoint. The OAuth client authenticates with its client credentials and includes the authorization code received in the previous step. The OAuth client also includes the redirection URI used to obtain the authorization code for verification.
5. The authorization server validates the client credentials and the authorization code. The server also ensures that the redirection URI received matches the URI used to redirect the client in Step 3. If valid, the authorization server responds back with an access token.

The authorization server can be the same server as the resource server or a separate entity. A single authorization server can issue access tokens accepted by multiple resource servers.

1.3.1.2 Authorization code flow with refresh token



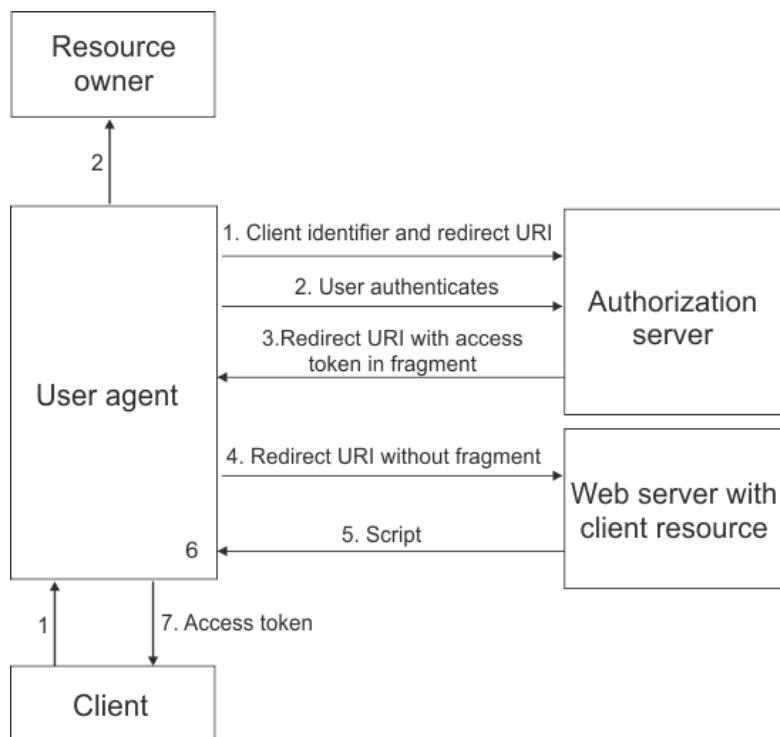
The authorization code workflow with refresh token diagram involves the following steps:

1. The OAuth client requests an access token by authenticating with the authorization server with its client credentials, and presenting an authorization grant.
2. The authorization server validates the client credentials and the authorization grant. If valid, the authorization server issues an access token and a refresh token.
3. The OAuth client makes a protected resource request to the resource server by presenting the access token.
4. The resource server validates the access token. If the access token is valid, the resource owner serves the request.
5. Repeat steps 3 and 4 until the access token expires. If the OAuth client knows that the access token has expired, skip to Step 7. Otherwise, the OAuth client makes another protected resource request.
6. If access token is not valid, the resource server returns an error.
7. The OAuth client requests a new access token by authenticating with the authorization server with its client credentials, and presenting the refresh token.
8. The authorization server validates the client credentials and the refresh token, and if valid, issues a new access token and a new refresh token.

1.3.1.3 Implicit grant flow

The implicit grant type is suitable for clients that are not capable of maintaining their client credentials confidential for authenticating with the authorization server. An example can be in the form of client applications that are in a user agent, typically implemented in a browser using a scripting language such as JavaScript.

As a redirection-based flow, the OAuth client must be able to interact with the user agent of the resource owner, typically a web browser. The OAuth client must also be able to receive incoming requests through redirection from the authorization server.



The implicit grant workflow diagram involves the following steps:

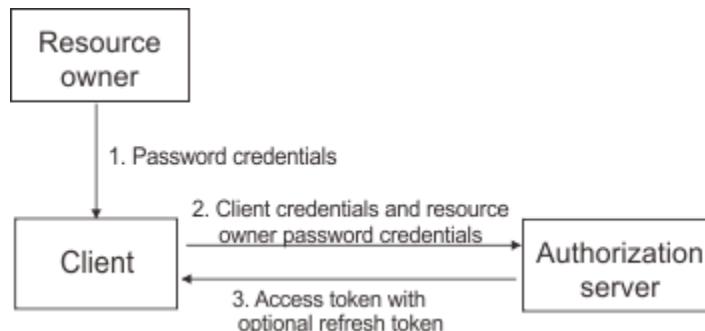
1. The OAuth client initiates the flow by directing the user agent of the resource owner to the authorization endpoint. The OAuth client includes its client identifier, requested scope, local state, and a redirection URI. The authorization server sends the user agent back to the redirection URI after access is granted or denied.
2. The authorization server authenticates the resource owner through the user agent and establishes whether the resource owner grants or denies the access request.

3. If the resource owner grants access, the authorization server redirects the user agent back to the client using the redirection URI provided earlier. The redirection URI includes the access token in the URI fragment.
4. The user agent follows the redirection instructions by making a request to the web server without the fragment. The user agent retains the fragment information locally.
5. The web server returns a web page, which is typically an HTML document with an embedded script. The web page accesses the full redirection URI including the fragment retained by the user agent. It can also extract the access token and other parameters contained in the fragment.
6. The user agent runs the script provided by the web server locally, which extracts the access token and passes it to the client.

1.3.1.4 Resource owner password credentials flow

The resource owner password credentials grant type is suitable in cases where the resource owner has a trust relationship with the client. For example, the resource owner can be a computer operating system of the OAuth client or a highly privileged application.

You can only use this grant type when the OAuth client has obtained the credentials of the resource owner. It is also used to migrate existing clients using direct authentication schemes by converting the stored credentials to an access token.



The resource owner password credentials workflow diagram involves the following steps:

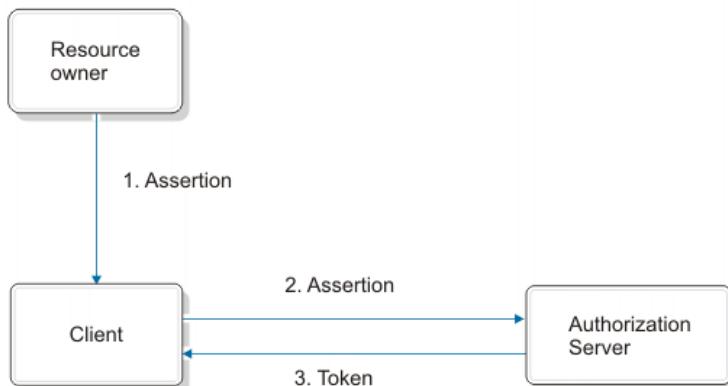
1. The resource owner provides the client with its username and password.
2. The OAuth client requests an access token from the authorization server through the token endpoint. The OAuth client authenticates with its client credentials and includes the credentials received from the resource owner.
3. After the authorization server validates the resource owner credentials and the client credentials, it issues an access token and optionally a refresh token.

1.3.1.5 JWT and SAML bearer grant type flows

The assertion bearer grant types are an extension to the OAuth 2.0 framework. In such flows, a client presents a JWT or SAML assertion to the token endpoint in exchange for tokens. The assertion that is presented must represent the resource owner for whom tokens will be issued to. See RFC 7522 and RFC 7523 for further details.

The assertions must be validated in the pre-token mapping rule. A callout to the STS is one way to validate a presented assertion.

The following diagram describes the steps in the assertion bearer grant type flows:

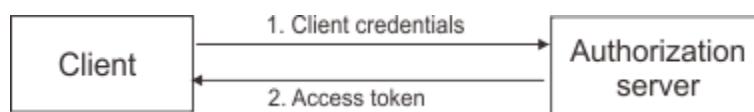


1. The client obtains a JWT or SAML assertion from the resource owner.
2. The client presents the assertion to the authorization server.
3. The authorization server validates the presented assertion through signature validation, decryption, or both. The issuer and subject are extracted and validated. If the issuer is trusted, tokens are issued to the subject that is captured in the assertion.

1.3.1.6 Client credentials flow

The client credentials flow is used when the OAuth client requests an access token using only its client credentials. This flow is applicable to one of the following situations:

- The OAuth client is requesting access to the protected resources under its control.
- The OAuth client is requesting access to a different protected resource, where authorization has been previously arranged with the authorization server.



The client credentials workflow diagram involves the following steps:

1. The OAuth client requests an access token from the token endpoint by authenticating with its client credentials.
2. After the authorization server validates the client credentials, it issues an access token.

1.4 Multi-factor authentication mechanisms⁷

Multi-factor authentication (MFA) is a method of confirming a user's claimed identity in which a computer user is granted access only after successfully presenting two or more pieces of evidence (or factors) to an authentication mechanism: knowledge (something the user and only the user knows), possession (something the user and only the user has), and inherence (something the user and only the user is).

Two-factor authentication (also known as 2FA) is a type, or subset, of multi-factor authentication. It is a method of confirming users' claimed identities by using a combination of *two* different factors: 1) something they know, 2) something they have, or 3) something they are.

A good example of two-factor authentication is the withdrawing of money from an ATM; only the correct combination of a bank card (something that the user possesses) and a PIN (something that the user knows) allows the transaction to be carried out.

Another example of two-factor authentication is being frequently used on gmail.com. Every fresh login would ask for the password & a system generated one-time password (OTP) sent on the registered mobile number or email id.

Two-step verification or two-step authentication is a method of confirming a user's claimed identity by utilizing something they know (password) and a second factor other than something they have or something they are. An example of a second step is the user repeating back something that was sent to them through an out-of-band mechanism. Or, the second step might be a six digit number generated by an app that is common to the user and the authentication system.

Mobile-phone two-step authentication involving devices such as mobile phones and smartphones was developed to provide an alternative method that would avoid such issues. To authenticate themselves, people can use their personal access-codes to the device (i.e. something that only the individual user knows) plus a one-time-valid, dynamic passcode, typically consisting of 4 to 6 digits. The passcode can be sent to their mobile device by SMS or push

⁷ https://en.wikipedia.org/wiki/Multi-factor_authentication

notification or can be generated by a one-time-passcode-generator app. In all three cases, the advantage of using a mobile phone is that there is no need for an additionally dedicated token, as users tend to carry their mobile devices around at all times.

1.4.1 ISAM Mobile Multi-Factor Authenticator

The IBM Security Access Manager Advanced Access Control component supports authenticator applications. Such support is built around the OAuth 2.0 protocol.

Authenticator applications are mobile-based applications that enable users to authenticate with minimal reliance on passwords. Mobile devices and biometric characteristics are used to support authentication and reduce the threat of unauthorized access to sensitive resources.

The *IBM Verify* application, which is available for download in major mobile application stores, is natively supported by the Advanced Access Control component. The authenticator application is built on the Mobile Access SDK, which is available for download from Fix Central. The Mobile Access SDK can also be used to create custom applications.

The authenticators' framework can be integrated with context-based access and authentication policies. Security Access Manager provides several predefined authentication policies to enable combinations of mobile and biometric mechanisms.

The authenticator application registration is built around an OAuth grant that is issued to the mobile device. The grant is used to identify the authenticator in future requests.

1.5 API gateways and ISAM⁸

ISAM has both an Authorization Server available in the form of API protection, as well as a resource server, the Web Reverse proxy. This section describes a deployment pattern in which ISAM plays both Authorization Server and API Protection roles.

1.5.1 API gateways – Resource Servers

An API gateway is responsible for fronting APIs and performing various roles, such as audit, validation, and security measures. The focus of this section is on the security aspects of API gateways. The API gateway, upon receiving a request to a protected API will check for the presence of an access token in the request and verify it. The result of this verification will be used to make a decision about whether the client performing the request is indeed allowed to access the protected API.

⁸ <https://www.ibm.com/blogs/security-identity-access/oauth-api-gateways-and-isam/>

API gateways are usually not capable of fine-grained authorization decisions beyond the basic validation of the presented OAuth access token. As compensation for this, API gateways often expose interfaces and features that enable protected resource servers to obtain additional token metadata. This metadata can enable finer-grained authorization decision at the resources. An example might be the evaluation of scopes which might have been permitted by the resource owner and associated with the access token.

The API gateway is a critical part of an OAuth deployment. The Gateway provides the front end point of contact for API calls performed by OAuth clients. To use the terminology defined by the OAuth authorization framework, this makes the API gateway the *Resource Server*, responsible for verifying a token and making a decision about whether or not a user is allowed access.

There are two common ways in which an API gateway will validate an access token depending on the type of token presented. Two common access token types exist:

- An opaque token acting as a reference to the grant which exists on the authorization server.
- An encapsulated token, the steps to validate the token will be agreed upon by the two entities in an out of band fashion.

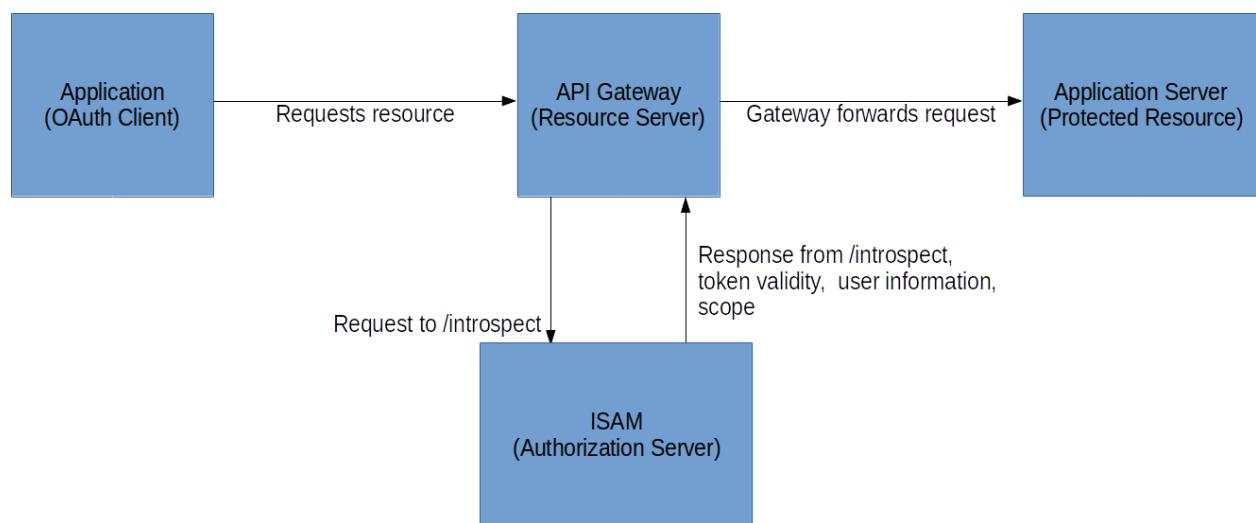
When using an opaque token, the *Resource Server* may communicate directly with the *Authorization Server* in order to validate an access token. In the initial specification for OAuth 2.0, no provisions were made for the mechanism for a *Resource Server* to request validation of an access token. OAuth introspection (RFC 7662) was created to provide a well-defined interface for clients and *Resource Servers* to discover the validity and metadata of an OAuth token from an *Authorization Server*.

1.5.2 ISAM – Authorization Server

ISAM can be configured to issue either opaque or encapsulated access tokens. Token Introspection which was added to ISAM in version 9.0.3.0 and the WS-Trust interface which has long been used by the ISAM Web Reverse Proxy to validate access tokens.

The preferred method for API gateways to validate tokens on ISAM is OAuth token introspection. This is a standardized interface. This is the core integration point between the two entities fulfilling OAuth roles.

On ISAM, both *Resource Servers* which are performing token introspection, and clients who are making requests to authorize on behalf of a user take the role of an API protection client. This API protection client can act in both capacities.

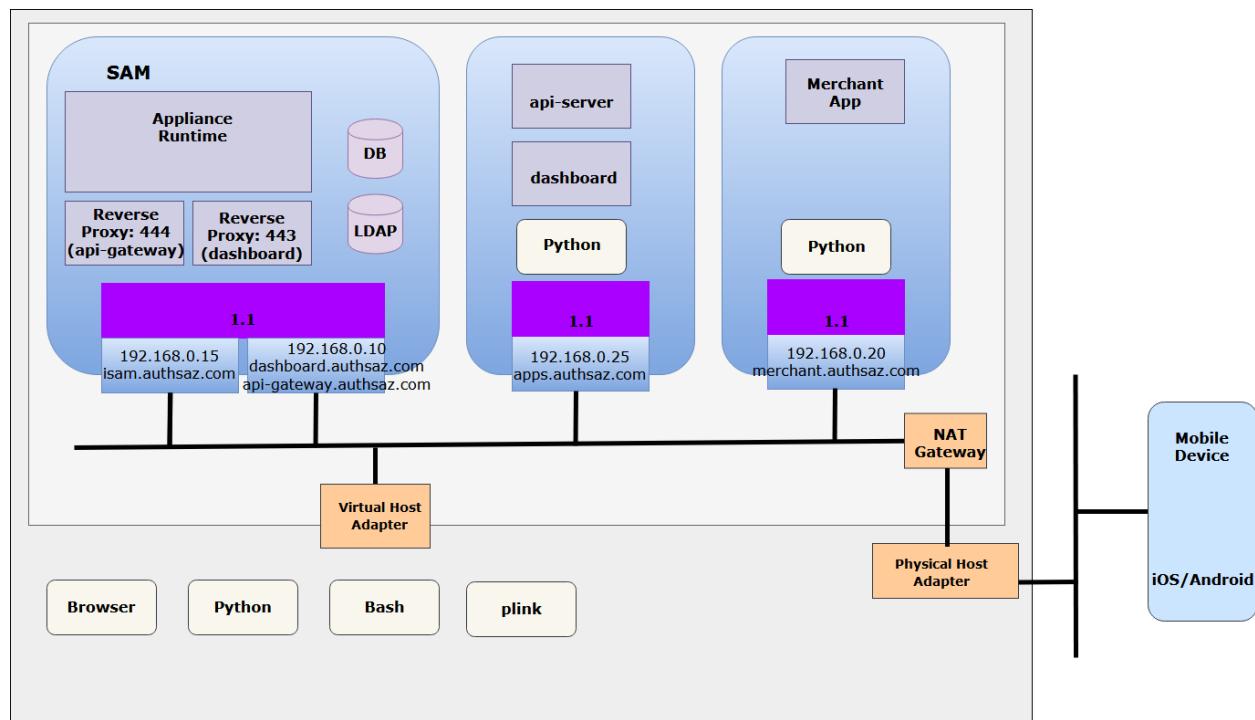


2 Architecture

This cookbook provides a step-by-step guide to configuring an IBM Security Access Manager Virtual Appliance to demonstrate API Protection scenarios. It was written to work with a fresh installation of IBM Security Access Manager 9.0.5.

2.1 High level Architecture

The high-level architecture and networking for the environment described in this document may be summarized as follows:



2.2 Required Components

2.2.1 Access Manager Virtual Appliance ISO Image

The Access Manager Virtual Appliance installation ISO image is required to create a Virtual Appliance from an empty Virtual Machine.

Access Manager version 9.0.5.0 can be downloaded from IBM Software Sellers Workplace (IBMer), IBM PartnerWorld (Authorized Partners), or Passport Advantage (Entitled Customers).

2.2.2 Access Manager 9.0.5 Activation Codes

Access Manager functionality is enabled using Activation Codes. To use this cookbook, you will need the Activation Codes for the Platform, Federation Runtime and the Advanced Access Control Add-on. Files containing these codes can be downloaded from IBM Software Sellers Workplace (IBMer), IBM PartnerWorld (Authorized Partners), or Passport Advantage (Entitled Customers).

You will need the files during configuration. make sure they are available on the same machine as the browser you will use to access the appliance.

2.2.3 Mobile Device running IBM Verify App

The Mobile Multi-Factor Authentication capability requires the use of a mobile application written to use the APIs in the IBM MMFA SDK (available for Android and iOS). IBM provides a pre-built application called *IBM Verify* for generic use. This can be found on the Apple App Store here:

<https://itunes.apple.com/app/ibm-verify/id1162190392> or on Google Play here:

<https://play.google.com/store/apps/details?id=com.ibm.security.verifyapp>.

Pre-requisite information on hardware and software required to run the *IBM Verify* application can be found at the application links above.

2.2.4 Host machine running VMWare

This guide assumes that the Hypervisor environment is VMWare Workstation (or Fusion for Mac). The host machine should have these minimum specifications:

- Good 64-bit processor (recommend dual-core i5 or better)
- 8GB memory (4GB for host OS + 4GB for Virtual Appliance)
- 20GB free disk space

2.2.5 VMWare Networking

This cookbook assumes NAT networking is used within VMWare and that the NAT network is configured for **192.168.0.0** subnet.

The *IBM Verify* App running on the mobile device must have connectivity to the SAM Reverse Proxy listening on port **444** of IP Address **192.168.0.10**. To achieve this, NAT port forwarding must be configured under VMWare to forward TCP packets received at Host port **444** to VM IP address **192.168.0.10** port **444** (Appendix II).

Internet connectivity is required for Network Time Protocol to be configured against an internet source. It is not otherwise required.

2.2.6 Hosts file

The hosts file on the host machine must include the following entries to allow it to resolve the hostnames used in this lab guide:

Address	Domain	Description
192.168.0.15	isam.authsaz.com	ISAM admin interface
192.168.0.10	api-gateway.authsaz.com dashboard.authsaz.com	ISAM interface for reverse proxies
192.168.0.25	apps.authsaz.com	A server for running backend services
192.168.0.20	merchant.authsaz.com	A server which a merchant is running on

2.3 Lab scenario overview

All the lab scenarios in this book have three entities:

1. **API Server:** this entity provides 3 simple banking APIs and the APIs would be accessed through ISAM WebSeal. We consider that there is no direct Access to API Server except ISAM WebSeal.

Below is the detail for these APIs.

API Name	API Description	API URI	API Inputs
Accounts	Get User Accounts	/api/nf/accounts	username
Balance	Get Account Balance	/api/nf/balance	account
Transfer	Transfer money between two accounts	/api/f/transfer	amount account account2

2. **Merchant App:** merchant App is a Third Party Application that consumes API Server endpoints. Merchant is considered to be an untrusted application and it requires OAuth access token for calling APIs. It has no direct access to API Server and access should be carried out through WebSeal. Merchant is considered to be as an OAuth Client.
3. **Dashboard Application:** this application is a trusted one that serve web interface for OAuth grant management, dynamic client registration and device registration. End-users and OAuth clients(merchant in our scenarios) can use this application.

We will create some users based on the table below. The Accounts and Balance for each account are hardcoded in API Server.

Username	Password	Accounts	Balance
user1	Passw0rd	101010 202020	10 20
user2	Passw0rd	303030	30
user3	Passw0rd	404040 505050	40 50
merchant	Passw0rd		

3 Create and configure backend servers

This book comes with two simple applications, One for API server and other for user dashboard application. These applications have been developed by the Python Flask framework.

In this chapter, we discuss the requirements and installation steps for running these demo applications.

The following sections discuss step by step installation of demo applications in VMware Workstation.

3.1 Hostnames

For the sake of simplicity, all the configuration in this book has been carried out based on domain names instead of IP addresses.

The hosts file on the Host machine must include some entries to allow it to resolve the hostnames used in this book.

3.1.1 Add hostnames to windows hosts files on the host machine

Following steps shows how to add required domain names in a windows 7 operating system as the Host machine:

1. Press the **Windows** key.
2. Type **Notepad** in the search field.
3. In the search results, right-click **Notepad** and select **Run as administrator**.
4. From Notepad, open the following file:

c:\Windows\System32\Drivers\etc\hosts

5. Add the following hostnames and IP addresses to the bottom of the hosts file.
192.168.0.15 isam.authsaz.com
192.168.0.10 api-gateway.authsaz.com
192.168.0.10 dashboard.authsaz.com
192.168.0.25 apps.authsaz.com
192.168.0.20 merchant.authsaz.com

```

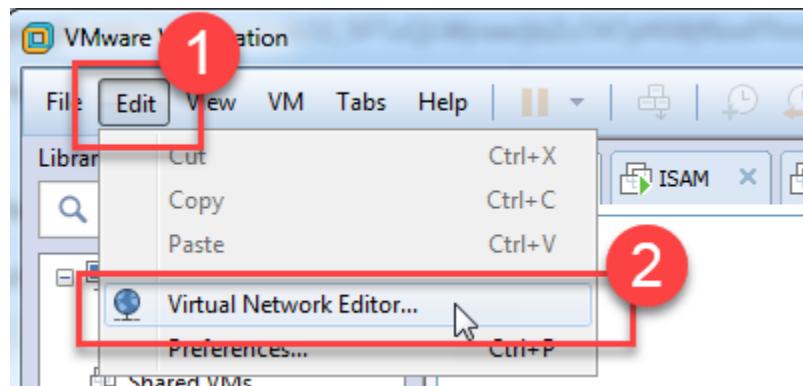
1 # Copyright (c) 1993-2009 Microsoft Corp.
2 #
3 # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
4 #
5 # This file contains the mappings of IP addresses to host names. Each
6 # entry should be kept on an individual line. The IP address should
7 # be placed in the first column followed by the corresponding host name.
8 # The IP address and the host name should be separated by at least one
9 # space.
10 #
11 # Additionally, comments (such as these) may be inserted on individual
12 # lines or following the machine name denoted by a '#' symbol.
13 #
14 # For example:
15 #
16 #      102.54.94.97    rhino.acme.com        # source server
17 #      38.25.63.10    x.acme.com            # x client host
18 #
19 # localhost name resolution is handled within DNS itself.
20 #   127.0.0.1        localhost
21 #   ::1              localhost
22 #
23 #
24
25 192.168.0.15    isam.authsaz.com
26 192.168.0.10    api-gateway.authsaz.com
27 192.168.0.10    dashboard.authsaz.com
28
29 192.168.0.25    apps.authsaz.com
30 192.168.0.20    merchant.authsaz.com

```

6. Select **File > Save** to save your changes.

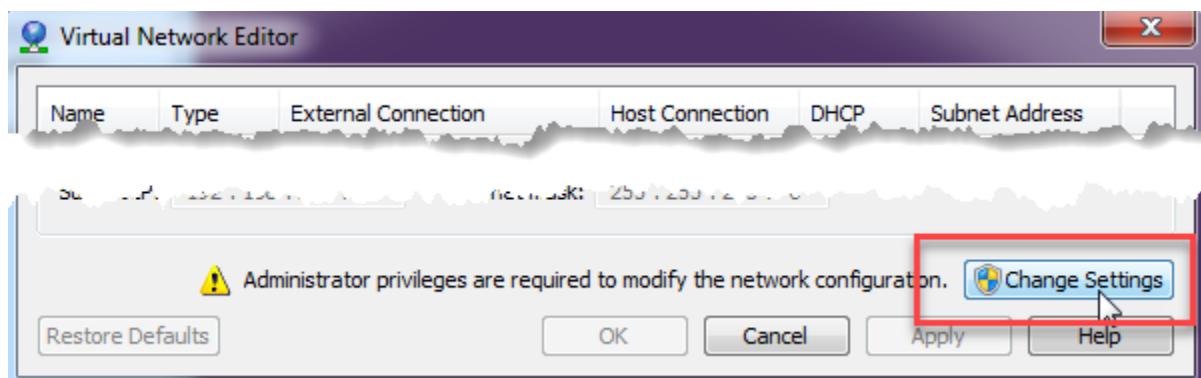
3.2 Subnet Address Settings

Run *VMware Workstation*.

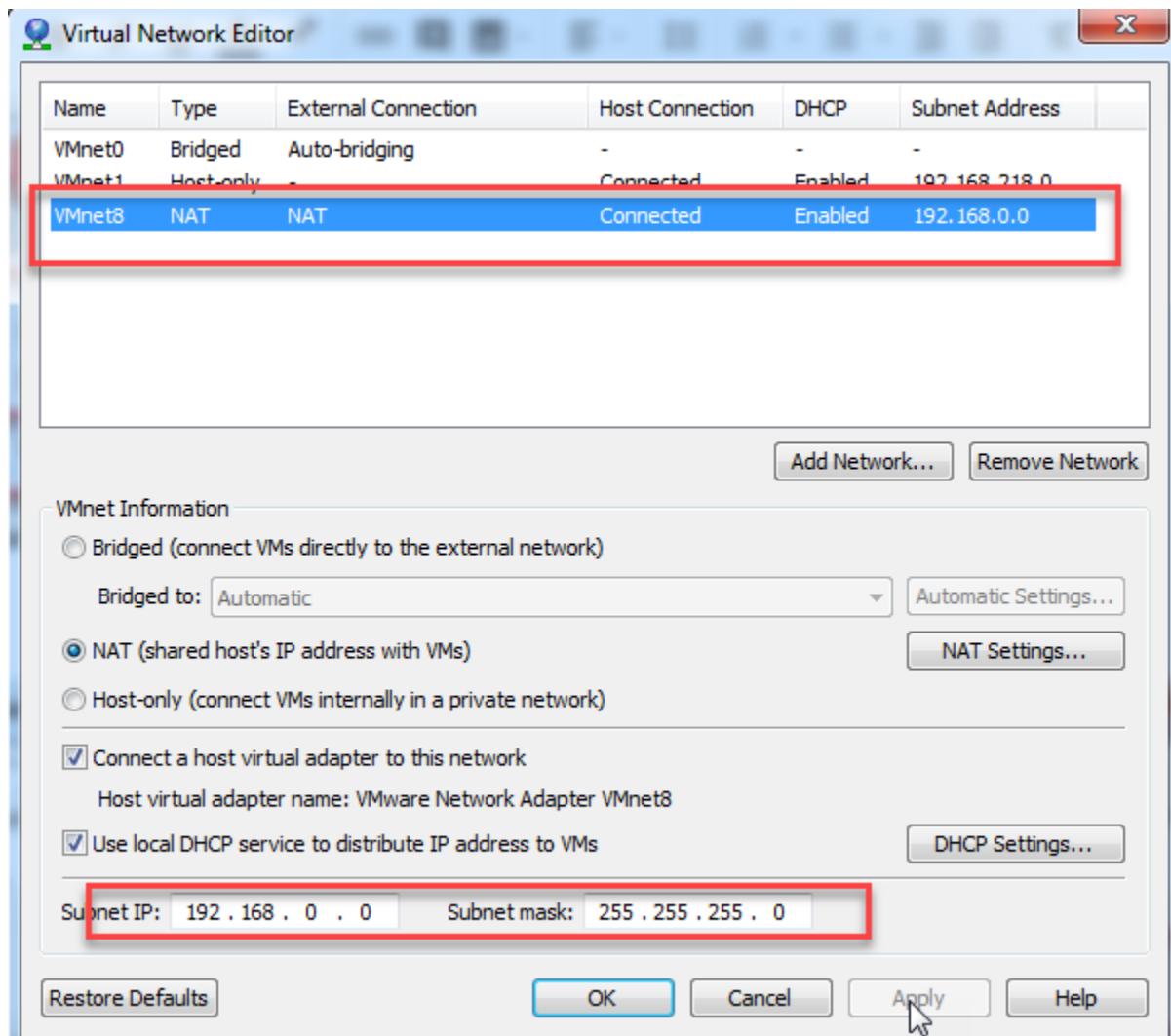


Navigate to **Edit** menu.

Click **Virtual Network Editor...** from the drop-down menu.



Click **Change Settings**.



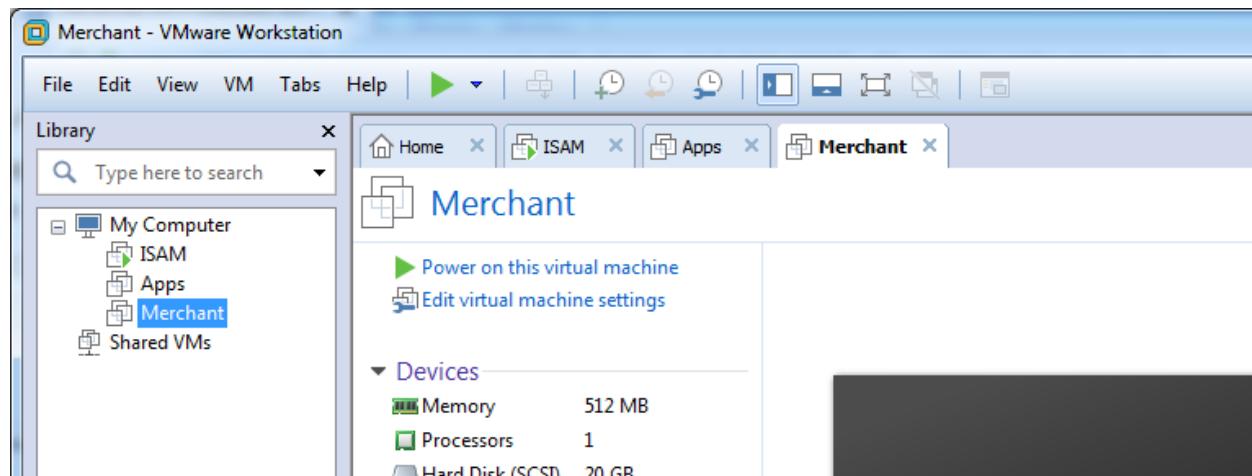
Enter **192.168.0.0** as *Subnet IP* and **255.255.255.0** as *Subnet mask*.

Click **Apply**.

3.3 Virtual Machines Creation

Three virtual machines are required to be created for this book:

- **ISAM**: this virtual machine hosts ISAM Virtual Appliance
- **Merchant**: this virtual machine hosts merchant web application
- **Apps**: this virtual machine hosts API Server and user dashboard applications



3.3.2 ISAM Virtual Machine Creation

This book assumes that readers are familiar with installing and configuring ISAM Virtual Appliance in VMware Workstation. We recommend to follow steps in chapter 2 of "**IBM Verify Cookbook Mobile Multi-Factor Authentication with IBM SAM**", For those who are not familiar with these steps.

During ISAM Virtual Machine configuration enter **192.168.0.10** as Reverse proxy interface and **192.168.0.15** as admin interface. Enter **PasswOrd** as **admin** and **sec_master** password.

3.3.2.1 Add hostnames to ISAM

From the top menu, select **Manage System Settings > Network Settings: Hosts File**. All current host records with their IP address and host names are displayed

The screenshot shows the IBM Security Access Manager interface. At the top, there are several navigation links: Home, Monitor, Secure Web Settings, Secure Access Control, Secure Federation, Connect IBM Cloud Identity, and Manage System Settings. The 'Manage System Settings' link is highlighted with a red box. Below the header, there's a sidebar with sections like Updates and Licensing, Network Settings, System Settings, and Secure Settings. Under 'System Settings', the 'Host File' entry is also highlighted with a red box.

Select the root level **Host Records** entry or do not select any entries. Then click **New**.

The screenshot shows the 'Manage Hosts File' interface. At the top, there are buttons for New, Delete, and Refresh. Below that, a tree view shows 'Host Records' expanded, with entries '127.0.0.1' and '::1'. A cursor is clicking the 'New' button.

On the **Create Host record** page, provide **192.168.0.15** as IP address and **isam.authsaz.com** as host name of the host record to add. Then click **Save**.

The screenshot shows the 'Create Host Record' dialog box. It has two input fields: 'Address *' containing '192.168.0.15' and 'Hostname *' containing 'isam.authsaz.com'. At the bottom, there are 'Save' and 'Cancel' buttons, with a cursor hovering over the 'Save' button.

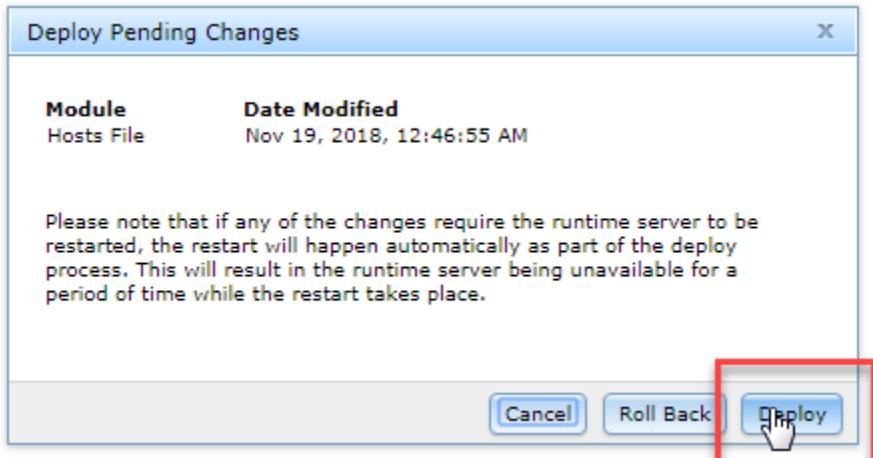
Repeat adding new host records for next host names as below.

The screenshot shows a web-based management interface titled "Manage Hosts File". At the top, there is a yellow warning bar with a yellow exclamation mark icon. The text in the bar reads: "There is currently one undeployed change. [Click here to review the changes or apply them to the system.](#)". A cursor arrow is pointing at the link. Below the bar, there are three buttons: "New" (green plus), "Delete" (red minus), and "Refresh" (blue circular arrow). Under these buttons, there is a section titled "Host Records" with a minus sign icon. The host records listed are: 127.0.0.1, 192.168.0.10 (with sub-records api-gateway.authsaz.com and dashboard.authsaz.com), 192.168.0.15 (with sub-records isam.authsaz.com), 192.168.0.20 (with sub-records merchant.authsaz.com), 192.168.0.25 (with sub-records apps.authsaz.com), and ::1.

To complete the host creation process we must deploy the changes we have made.

Click the **Click here to review the changes or apply them to the system** link in the warning message - as shown below.

This screenshot is similar to the one above, showing the "Manage Hosts File" interface. It includes the yellow warning bar with the undeployed change message and the same list of host records. However, the link in the warning bar is now highlighted with a thick red rectangular box. The rest of the interface, including the navigation buttons and the host record list, appears identical to the first screenshot.



3.3.3 Merchant Virtual machine creation

This book assumes that readers are familiar with installing an ubuntu distribution of linux in VMware Workstation. We skip the steps regarding to installing ubuntu operating system in VMware Workstation.

The minimum requirements for creating Merchant virtual machine is to create a virtual machine named **Merchant** with at least 512 MB memory.

3.3.3.1 Set IP for Merchant virtual machine

Run linux **bash**. Open **/etc/network/interfaces** using an editor like **nano**.

```
GNU nano 2.2.6                               File: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.0.20
    netmask 255.255.255.0
    gateway 192.168.0.2
```

Edit the file as picture above and save the file. Close the editor.

```
user@authsaz:~$ sudo ifdown eth0
user@authsaz:~$ sudo ifup eth0
user@authsaz:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 00:0c:29:c9:6d:b3
          inet addr:192.168.0.20 Bcast:192.168.0.255 Mask:255.255.255.0
           inet6 addr: fe80::20c:29ff:fe9:6db3/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
              RX packets:3802 errors:0 dropped:0 overruns:0 frame:0
              TX packets:3384 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:1000
              RX bytes:659717 (659.7 KB) TX bytes:1533349 (1.5 MB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
              UP LOOPBACK RUNNING MTU:65536 Metric:1
              RX packets:88 errors:0 dropped:0 overruns:0 frame:0
              TX packets:88 errors:0 dropped:0 overruns:0 carrier:0
              collisions:0 txqueuelen:0
              RX bytes:6744 (6.7 KB) TX bytes:6744 (6.7 KB)

user@authsaz:~$ _
```

Enter **sudo ifdown eth0**.

Enter **sudo ifup eth0**.

Enter **ifconfig** to make sure virtual machine ip has been changed correctly.

3.3.3.2 Configure hosts file

Open **/etc/hosts** using an editor like **nano**.

Add the following entries to the bottom of the file:

```
192.168.0.15  isam.authsaz.com
192.168.0.10  api-gateway.authsaz.com
192.168.0.10  dashboard.authsaz.com
192.168.0.25  apps.authsaz.com
192.168.0.20  merchant.authsaz.com
```

```
GNU nano 2.2.6                               File: /etc/hosts

127.0.0.1      localhost
127.0.1.1      authsaz

192.168.0.15    isam.authsaz.com
192.168.0.10    api-gateway.authsaz.com
192.168.0.10    dashboard.authsaz.com
192.168.0.25    apps.authsaz.com
192.168.0.20    merchant.authsaz.com

# The following lines are desirable for IPv6 capable hosts
::1      localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Save and close the file.

3.3.3.3 Install git, python3 and pip3

Enter the following commands in order to install git, python3 and pip3:

```
sudo apt-get update
sudo apt-get install git
sudo apt-get install python3 python3-pip
```

```
user@authsaz:~/isam-book$ python3
Python 3.4.3 (default, Nov 17 2016, 01:08:31)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Enter **python3** in *bash* to see python version.

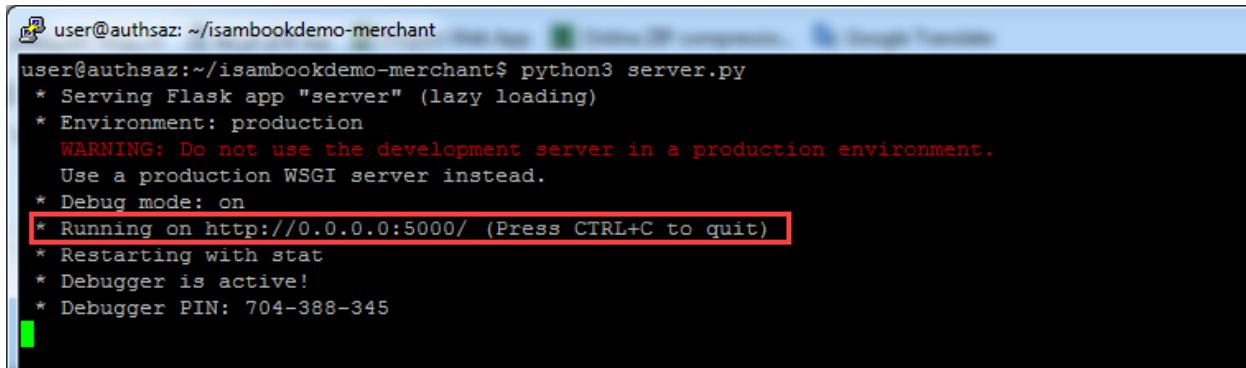
```
user@authsaz:~/isam-book$ git --version
git version 1.9.1
user@authsaz:~/isam-book$ █
```

Enter **git --version** in *bash* to see git version.

3.3.3.4 Create and configure Merchant server

Run the following commands in Merchant's linux *bash*:

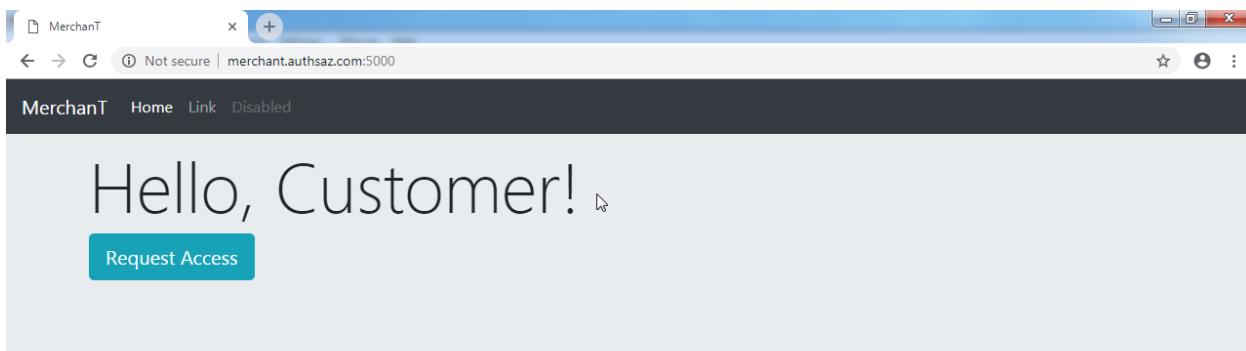
```
git clone https://github.com/authsaz/isambookdemo-merchant.git  
cd isambookdemo-merchant  
sudo pip3 install -r requirements.txt  
python3 server.py
```



```
user@authsaz: ~/isambookdemo-merchant  
user@authsaz:~/isambookdemo-merchant$ python3 server.py  
* Serving Flask app "server" (lazy loading)  
* Environment: production  
  WARNING: Do not use the development server in a production environment.  
  Use a production WSGI server instead.  
* Debug mode: on  
* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)  
* Restarting with stat  
* Debugger is active!  
* Debugger PIN: 704-388-345
```

The server will be started on port **5000**. Open a browser on Host machine and type **http://merchant.authsaz.com:5000** as *url*.

You should see merchant site as shown below.



3.3.4 Apps Virtual machine creation

This book assumes that readers are familiar with installing an Ubuntu distribution of Linux in VMware Workstation. We skip the steps regarding installing Ubuntu operating system in VMware Workstation.

The minimum requirements for creating Apps virtual machine is to create a virtual machine named **Apps** with at least 512 MB memory.

3.3.4.1 Set IP for Apps virtual machine

Run Linux **bash**.

Open **/etc/network/interfaces** using an editor like **nano**.

```
GNU nano 2.2.6          File: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.0.25
    netmask 255.255.255.0
    gateway 192.168.0.2
```

Edit the file as the picture above and save the file. Close the editor.

```
user@authsaz:~$ sudo ifdown eth0
user@authsaz:~$ sudo ifup eth0
user@authsaz:~$ ifconfig
eth0      Link encap:Ethernet HWaddr 00:0c:29:86:9f:6f
          inet addr:192.168.0.25  Bcast:192.168.0.255  Mask:255.255.255.0
                      inet6 addr: fe80::20c:29ff:fe86:9f6f/64 Scope:Link
                         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
                         RX packets:30460 errors:0 dropped:0 overruns:0 frame:0
                         TX packets:14209 errors:0 dropped:0 overruns:0 carrier:0
                         collisions:0 txqueuelen:1000
                         RX bytes:41991727 (41.9 MB)  TX bytes:907970 (907.9 KB)

lo       Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
                         UP LOOPBACK RUNNING  MTU:65536  Metric:1
                         RX packets:0 errors:0 dropped:0 overruns:0 frame:0
                         TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
                         collisions:0 txqueuelen:0
                         RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

user@authsaz:~$
```

Enter **sudo ifdown eth0**.

Enter **sudo ifup eth0**.

Enter **ifconfig** to make sure virtual machine IPhas been changed correctly.

3.3.4.2 Configure hosts file

Open **/etc/hosts** using an editor like **nano**.

Add the following entries to the bottom of the file:

```
192.168.0.15  isam.authsaz.com
192.168.0.10   api-gateway.authsaz.com
192.168.0.10   dashboard.authsaz.com
192.168.0.25   apps.authsaz.com
192.168.0.20   merchant.authsaz.com
```

```
GNU nano 2.2.6                               File: /etc/hosts

127.0.0.1      localhost
127.0.1.1      authsaz

192.168.0.15    isam.authsaz.com
192.168.0.10    api-gateway.authsaz.com
192.168.0.10    dashboard.authsaz.com
192.168.0.25    apps.authsaz.com
192.168.0.20    merchant.authsaz.com

# The following lines are desirable for IPv6 capable hosts
::1      localhost ip6-localhost ip6-loopback
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters
```

Save and close the file.

3.3.4.3 Install git, python3 and pip3

Enter the following commands to install git, python3, and pip3:

```
$ sudo apt-get update
$ sudo apt-get install git
$ sudo apt-get install python3 python3-pip
```

```
user@authsaz:~/isam-books$ python3
Python 3.4.3 (default, Nov 17 2016, 01:08:31)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Enter **python3** in *bash* to see python version.

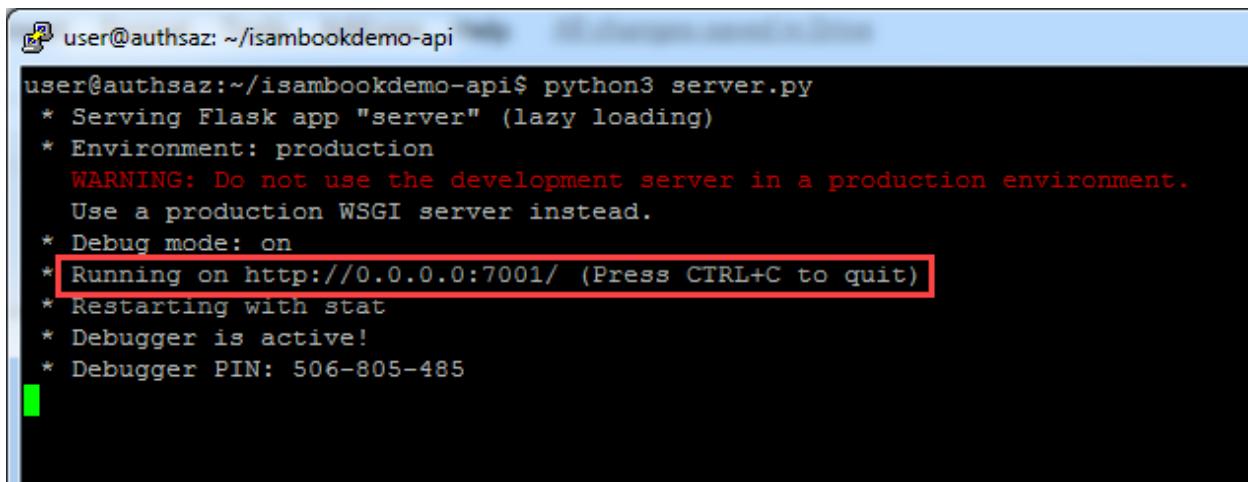
```
user@authsaz:~/isam-books$ git --version
git version 1.9.1
user@authsaz:~/isam-books$ █
```

Enter **git --version** in *bash* to see git version.

3.3.4.4 Create and configure API server

Run the following commands in Linux bash:

```
git clone https://github.com/authsaz/isambookdemo-api.git  
cd isambookdemo-api  
sudo pip3 install -r requirements.txt  
python3 server.py
```



```
user@authsaz:~/isambookdemo-api$ python3 server.py  
 * Serving Flask app "server" (lazy loading)  
 * Environment: production  
   WARNING: Do not use the development server in a production environment.  
   Use a production WSGI server instead.  
 * Debug mode: on  
 * Running on http://0.0.0.0:7001/ (Press CTRL+C to quit)  
 * Restarting with stat  
 * Debugger is active!  
 * Debugger PIN: 506-805-485
```

The server will be started on port 7001. Open a browser and type **http://apps.authsaz.com:7001** as URL.

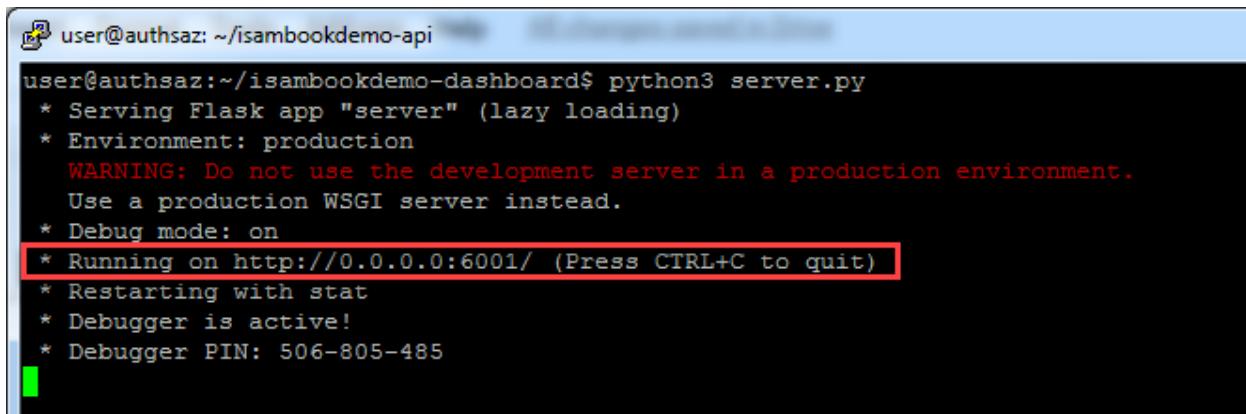
You should see the API server as shown below.



3.3.4.5 Create and configure customer dashboard

Run the following commands in Linux bash:

```
git clone https://github.com/authsaz/isambookdemo-dashboard.git  
cd isambookdemo-dashboard  
sudo pip3 install -r requirements.txt  
python3 server.py
```



```
user@authsaz: ~/isambookdemo-dashboard$ python3 server.py  
 * Serving Flask app "server" (lazy loading)  
 * Environment: production  
 WARNING: Do not use the development server in a production environment.  
 Use a production WSGI server instead.  
 * Debug mode: on  
 * Running on http://0.0.0.0:6001/ (Press CTRL+C to quit)  
 * Restarting with stat  
 * Debugger is active!  
 * Debugger PIN: 506-805-485
```

The server will be started on port **6001**. Open a browser and type <http://apps.authsaz.com:6001/portal/user> as URL.

You should see the dashboard site as shown below.

The screenshot shows a web application interface with two modal dialogs. The top dialog is titled "Authorization Grant" and displays the message "apps.authsaz.com:6001 says failure" with an "OK" button. The bottom dialog is titled "Authenticators" and shows a table header with columns: Id, Device Name, Device Type, OS Version, oAuth Grant, Enabled, and Remove. It also features a "New" button.

For now, don't worry about the errors and failures. They will be fixed when the app is served behind a reverse proxy.

4 Create and configure ISAM reverse proxies

In this chapter, we will create two reverse proxies. One will be used as the contact point for API Server and the other will be used as a contact point for the dashboard application.

At the end of this chapter, we will discuss the steps required to configure the *Advanced Access Control* module in ISAM.

4.1 Create API Server reverse proxy

In the Host machine open a browser and navigate to <https://isam.authsaz.com>.

Login to ISAM Management Interface using user **admin** and password **PasswOrd**.

The screenshot shows the IBM Security Access Manager Management Interface. The top navigation bar includes links for Home, Monitor, Secure Web Settings (which is highlighted with a red box), and Secure Access Control. The left sidebar has a 'Manage' section with options like Runtime Component, Reverse Proxy (also highlighted with a red box), Authorization Server, Distributed Session Cache, and Policy Administration. The main content area is divided into three columns: Global Settings (containing URL Mapping, Junction Mapping, Client Certificate Mapping, User Name Mapping, Password Strength, Forms Based Single Sign-on, HTTP Transformation, Kerberos Configuration, and RSA SecurID Configuration) and Global Keys (containing SSO Keys and LTPA Keys). A cursor arrow points towards the 'Junction Mapping' link under Global Settings.

In the top menu panel, select **Secure Web Settings > Manage: Reverse Proxy**, as indicated above.

The screenshot shows the IBM Security Access Manager interface. At the top, there's a navigation bar with links for Home, Monitor, Secure Web Settings, Secure Access Control, and Secure Federation. Below the navigation bar, the main content area has a title 'Reverse Proxy'. In the top left of this area, there's a toolbar with several icons: New (highlighted with a red box), Edit, Delete, Start, Stop, Restart, Refresh, and Manage. Below the toolbar is a table header with columns for 'Instance Name' and 'State'. Underneath the header, there's a search bar with the placeholder 'No filter applied' and a pagination control showing '0 item' and '10 | 25 | X'.

Click the **New** button to open the Reverse Proxy creation dialog.

The screenshot shows the 'New Reverse Proxy Instance' dialog. It has tabs for Instance, IBM Security Access Manager, and Transport, with the Instance tab selected. Inside the dialog, there are four required fields: 'Instance Name *' containing 'api-gateway', 'Host name *' containing 'isam.authsaz.com', 'Listening Port *' containing '7234', and 'IP Address for the Primary Interface *' containing '192.168.0.10'. At the bottom right of the dialog, there are buttons for Previous, Next, Finish, and Cancel, with the 'Next' button highlighted with a red box.

Enter **api-gateway** as the *Instance Name* and select the IP address associated with the non-management interface we configured earlier (**192.168.0.10**) from the *IP Address for the Primary Interface* pull-down list.

Ensure the *Host name* and *Listening Port* default correctly to the values shown above.

Click **Next** to progress to the next configuration panel.

New Reverse Proxy Instance

Instance IBM Security Access Manager Transport

Administrator Name *
sec_master

Administrator Password *

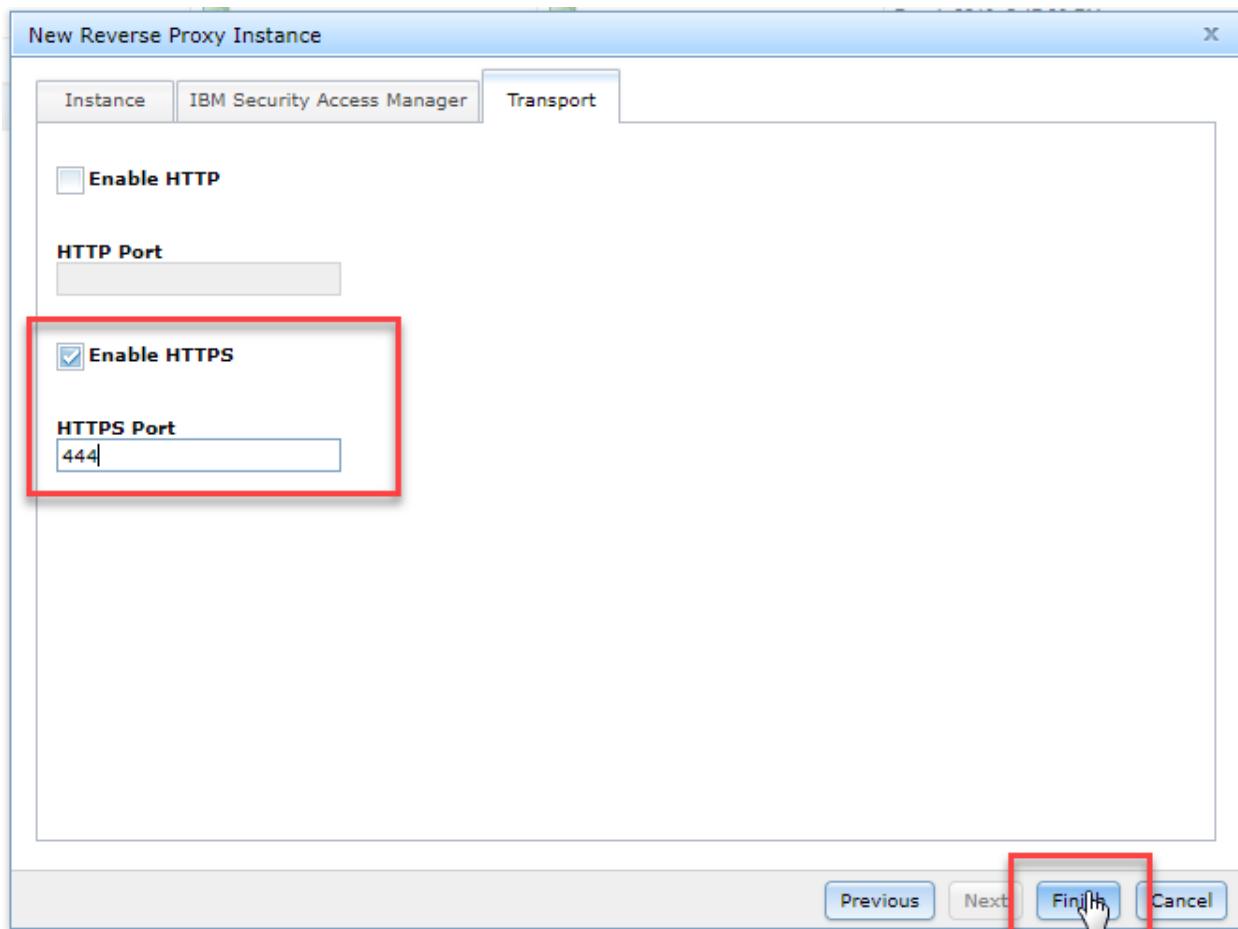
Domain *
Default

Previous Next Finish Cancel

The screenshot shows a configuration dialog titled "New Reverse Proxy Instance". The "IBM Security Access Manager" tab is active. It contains three input fields: "Administrator Name *" with value "sec_master", "Administrator Password *" with value "*****" (redacted), and "Domain *" with value "Default". Below the form is a row of buttons: "Previous", "Next", "Finish", and "Cancel".

Enter **Passw0rd** as the (ISAM) *Administrator Password*. Ensure the other fields default correctly as shown above.

Click **Next** to progress to the next configuration panel.



Select the checkbox for *HTTPS* and ensure the “*HTTPS Port*” is set to **444**. Click **Finish** to create the Reverse Proxy instance.

The Reverse Proxy instance is now configured and started.

The screenshot shows the Appliance Dashboard with the 'Reverse Proxy' section selected. At the top, there is a 'System Notification' box with the message 'Successfully configured a new proxy instance.' Below the notification, there is a toolbar with buttons for 'New', 'Edit', 'Delete', 'Start', 'Stop', 'Restart', 'Refresh', and 'Manage'. There is also a filter dropdown with the text 'No filter applied'. A table below the toolbar lists proxy instances. The first row, which contains the entry 'api-gateway' and the status 'Started', is highlighted with a red box. At the bottom of the table, there are pagination controls: '1 - 1 of 1 item', '10 | 25 | 50 | 100 | All', and a 'Manage' dropdown.

Instance Name	State
api-gateway	Started

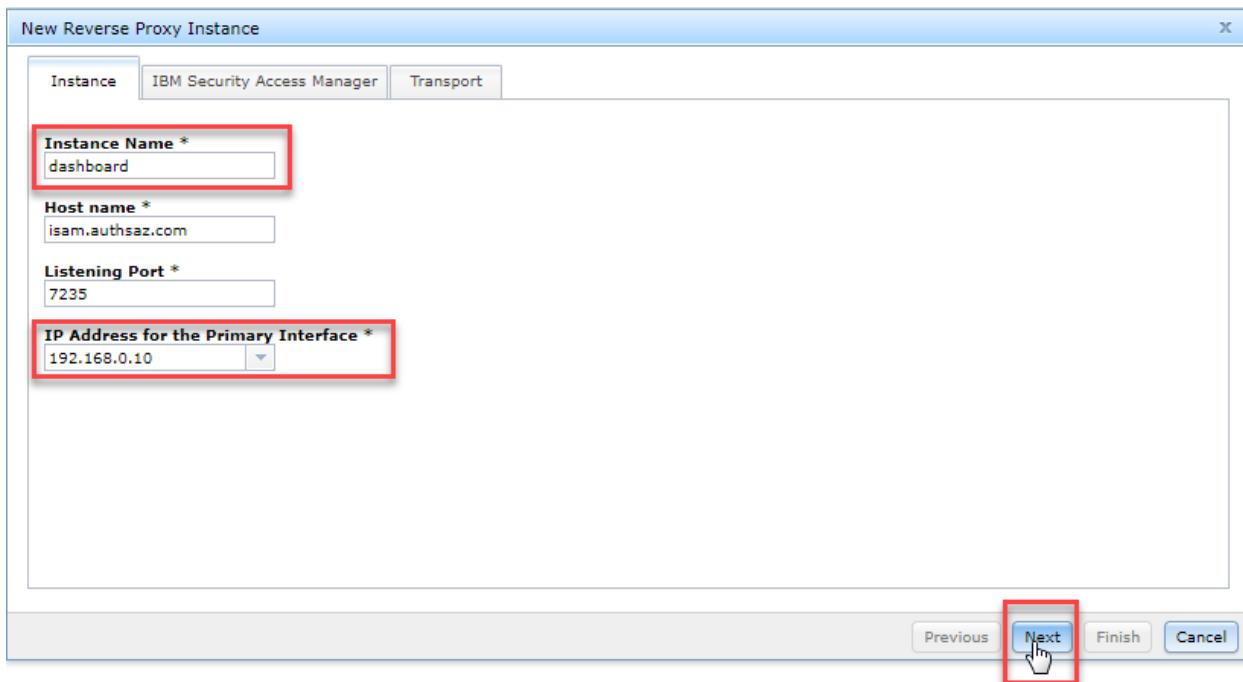
4.2 Create customer dashboard reverse proxy

The screenshot shows the IBM Security Access Manager interface. The top navigation bar includes links for Home, Monitor, Secure Web Settings (which is highlighted with a red box), and Secure Access Control. The main content area has three tabs: Manage, Global Settings, and Global Keys. The Manage tab is selected and displays a list of components under Runtime Component, including Reverse Proxy (which is also highlighted with a red box). Other items in the list include Application Server, Distributed Session Cache, and Policy Administration.

In the top menu panel, select **Secure Web Settings > Manage: Reverse Proxy**, as indicated above.

The screenshot shows the Reverse Proxy configuration page. The top navigation bar includes Home, Monitor, Secure Web Settings, Secure Access Control, and Secure Federation. The main content area is titled "Reverse Proxy" and features a toolbar with buttons for New, Edit, Delete, Start, Stop, Restart, Refresh, and Manage. A "New" button is highlighted with a red box and a cursor icon. Below the toolbar, there is a table header with columns for "Instance Name" and "State".

Click the **New** button to open the Reverse Proxy creation dialog.



Enter **dashboard** as the *Instance Name* and select the IP address associated with the non-management interface we configured earlier (**192.168.0.10**) from the *IP Address for the Primary Interface* pull-down list.

Ensure the *Host name* and *Listening Port* default correctly to the values shown above.

Click **Next** to progress to the next configuration panel.

New Reverse Proxy Instance

Instance IBM Security Access Manager Transport

Administrator Name *
sec_master

Administrator Password *

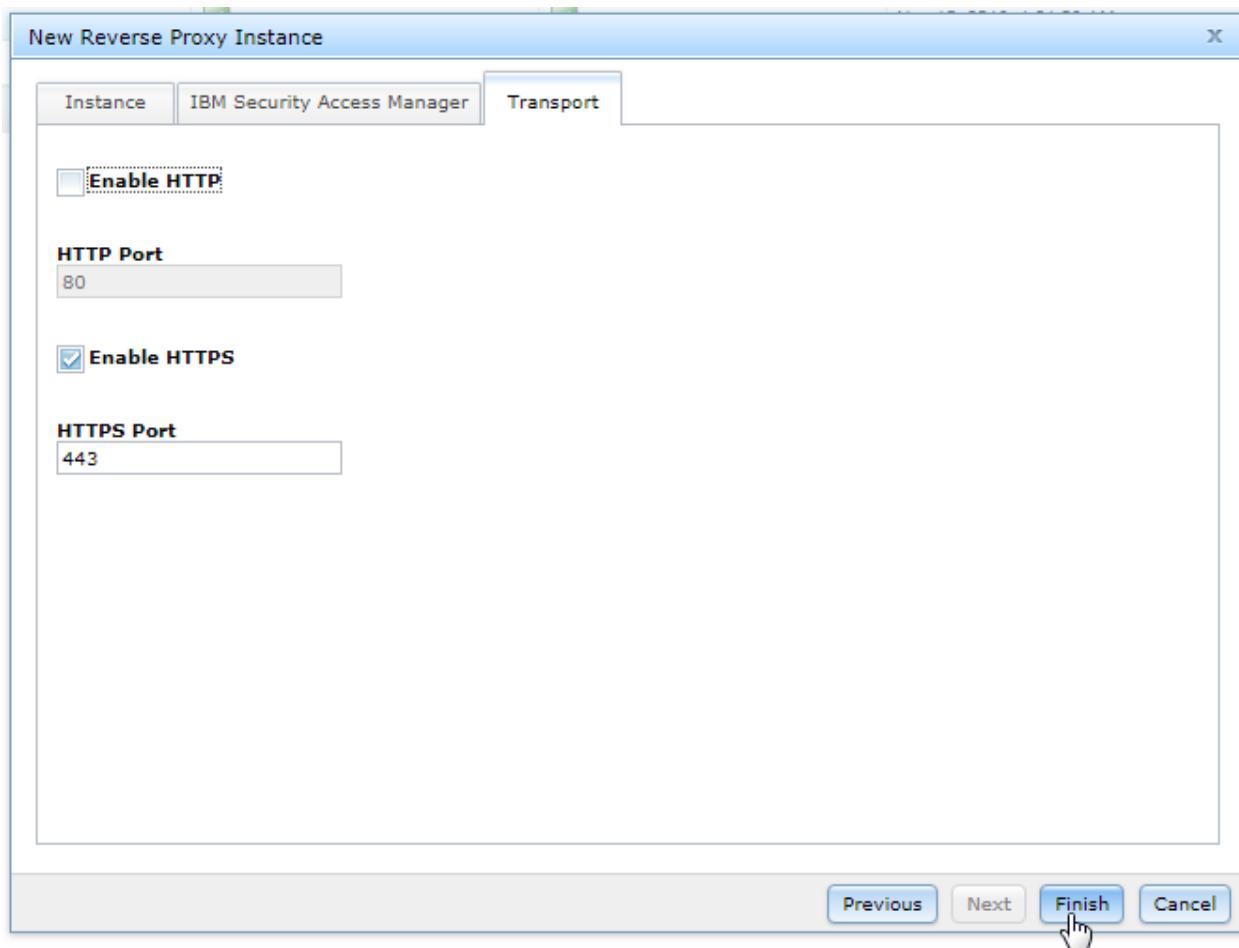
Domain *
Default

Previous Next Finish Cancel

The screenshot shows a configuration dialog titled "New Reverse Proxy Instance". The "IBM Security Access Manager" tab is active. It contains three input fields: "Administrator Name *" with value "sec_master", "Administrator Password *" with a redacted value, and "Domain *" with value "Default". At the bottom are buttons for "Previous", "Next", "Finish", and "Cancel".

Enter **Passw0rd** as the (ISAM) *Administrator Password*. Ensure the other fields default correctly as shown above.

Click **Next** to progress to the next configuration panel.



Select the checkbox for *HTTPS* and ensure the “*HTTPS Port*” is set to **443**. Click **Finish** to create the Reverse Proxy instance.

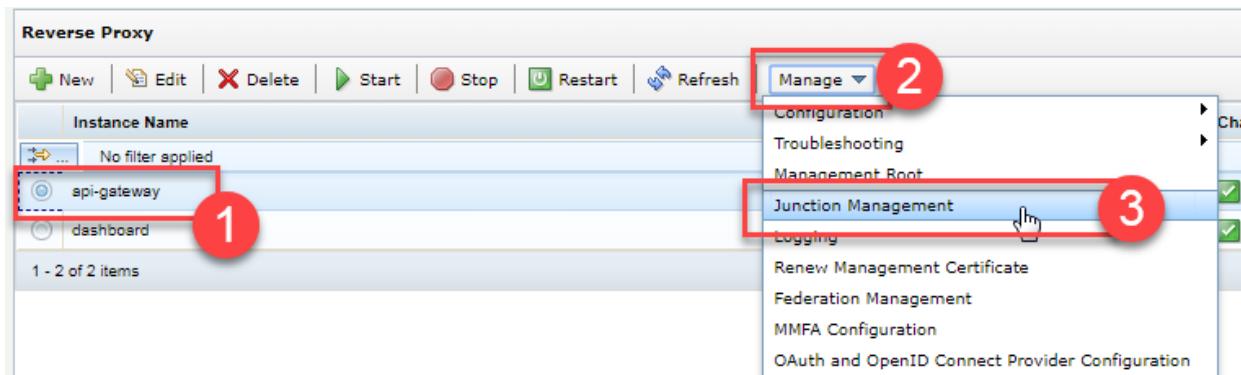
The Reverse Proxy instance is now configured and started.

Reverse Proxy	
New Edit Delete Start Stop Restart Refresh Manage ▾	
Instance Name	State
api-gateway	<input checked="" type="checkbox"/> Started
dashboard	<input checked="" type="checkbox"/> Started

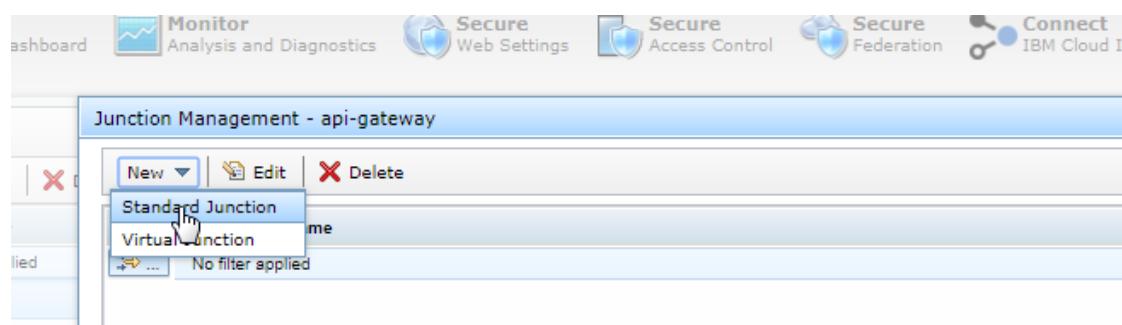
4.3 Create Junctions

4.3.1 Create /api junction

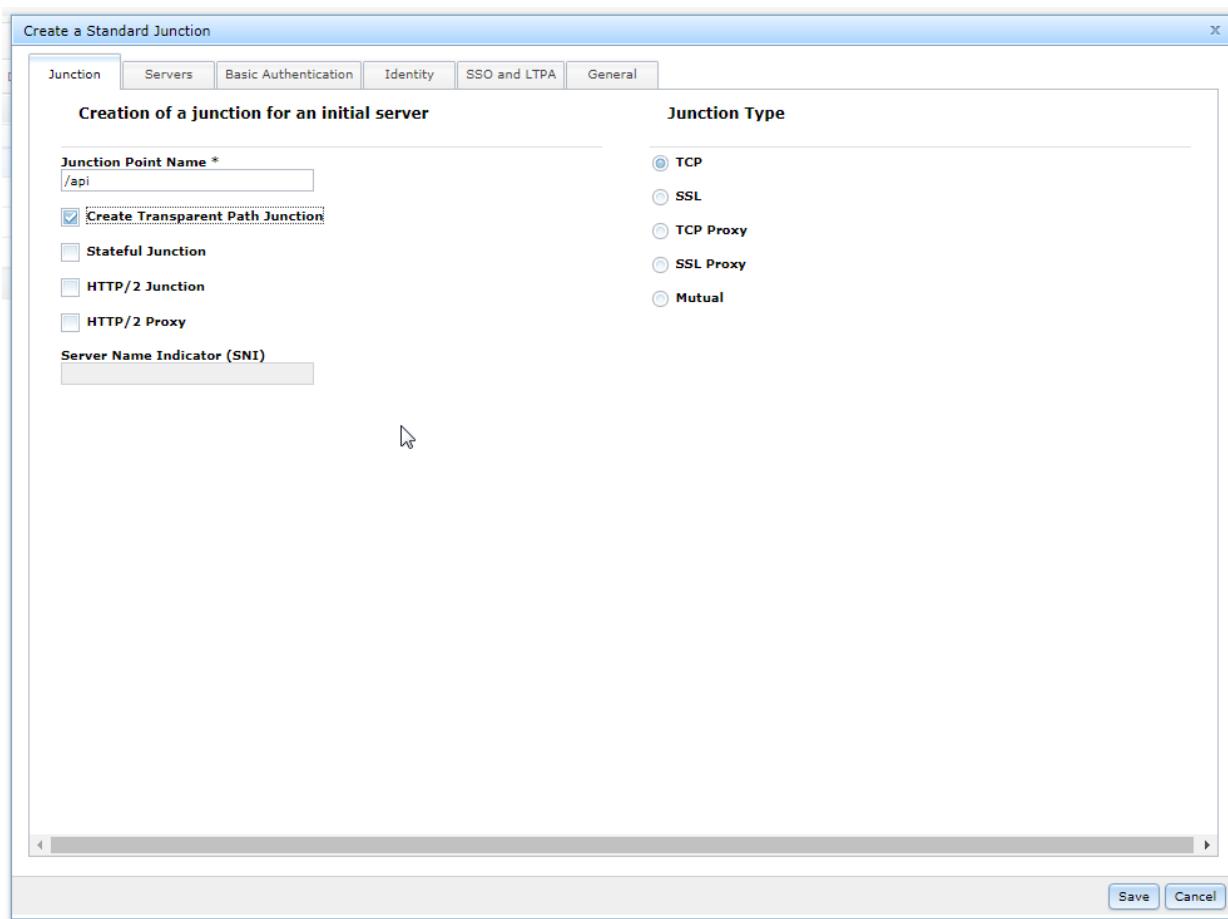
Use the Junction Management page to create junctions required for **api-gateway**. From the top menu, select **Secure Web Settings > Manage: Reverse Proxy**.



Select the **api-gateway** reverse proxy to manage its junctions. Then select **Manage > Junction Management**.



Click **New > Standard Junction**.

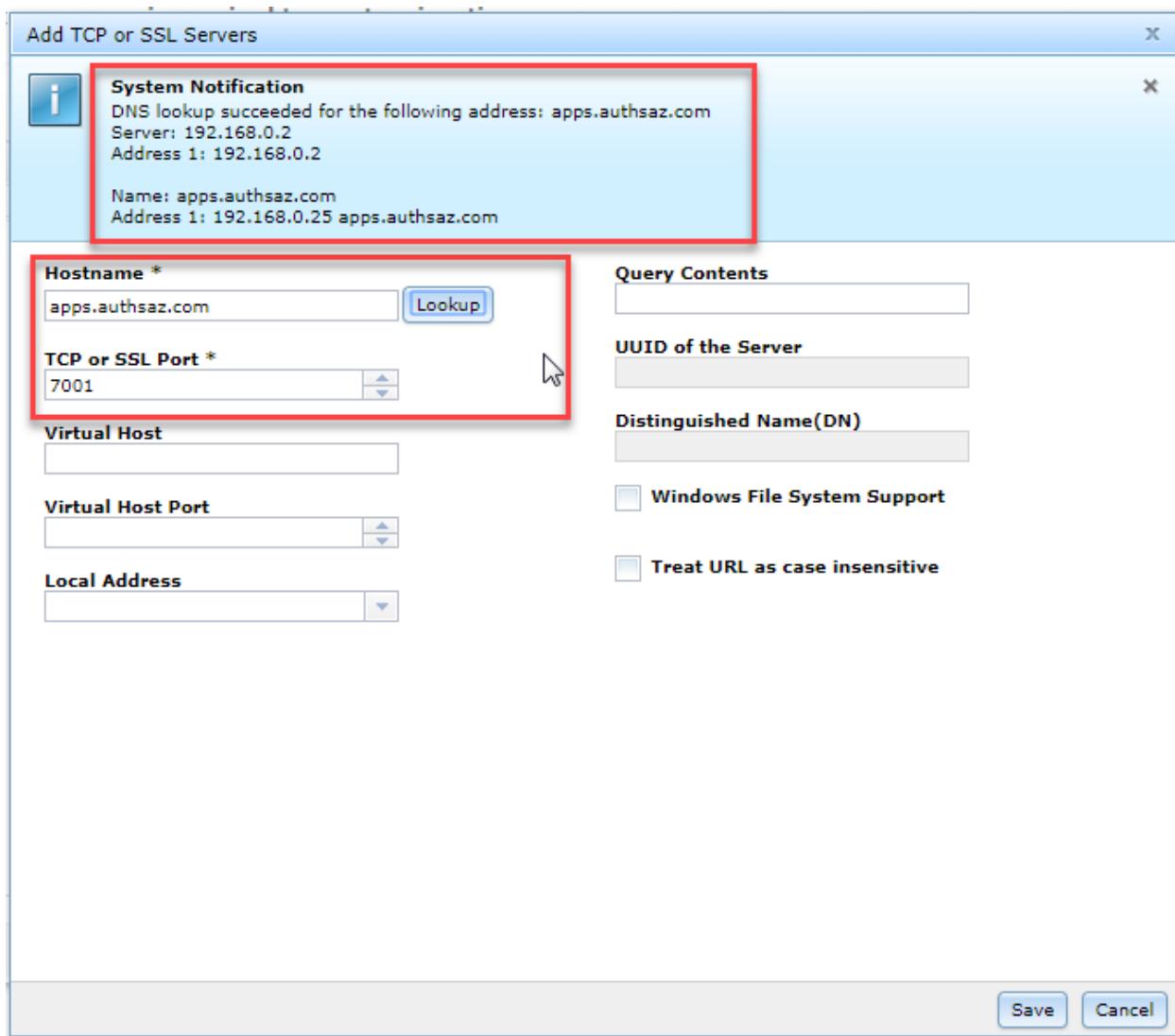


On the Junction tab page:

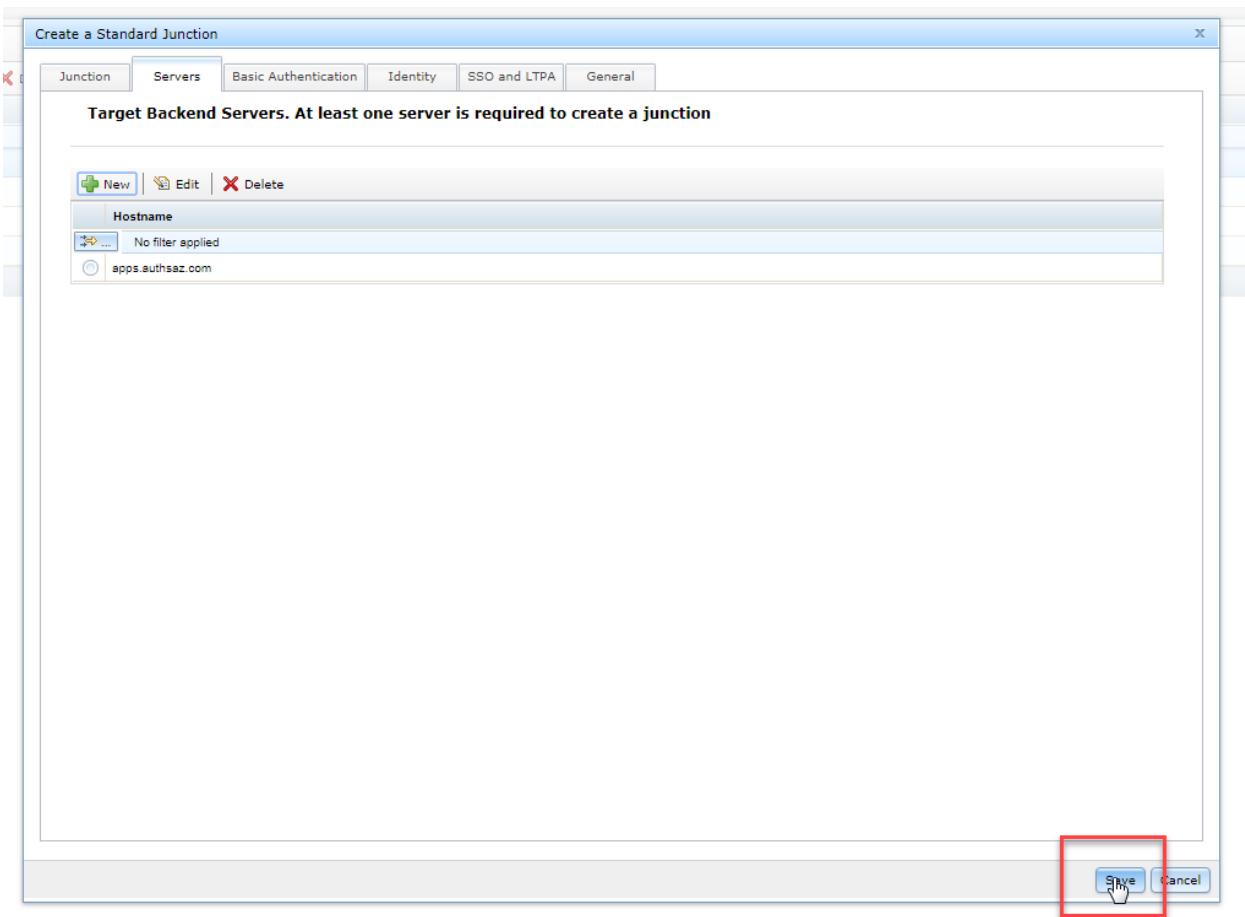
- Enter **/api** as the *junction point name*.
- Select the *Create Transparent Path Junction* checkbox.
- Select **TCP** as *junction type* from the listed options



Select the **Servers** tab and click **New**.



Enter **apps.authsaz.com** and **7001** as *Hostname* and *TCP or SSL Port*. You can verify the availability of host by pressing **Lookup** button. click **Save**.



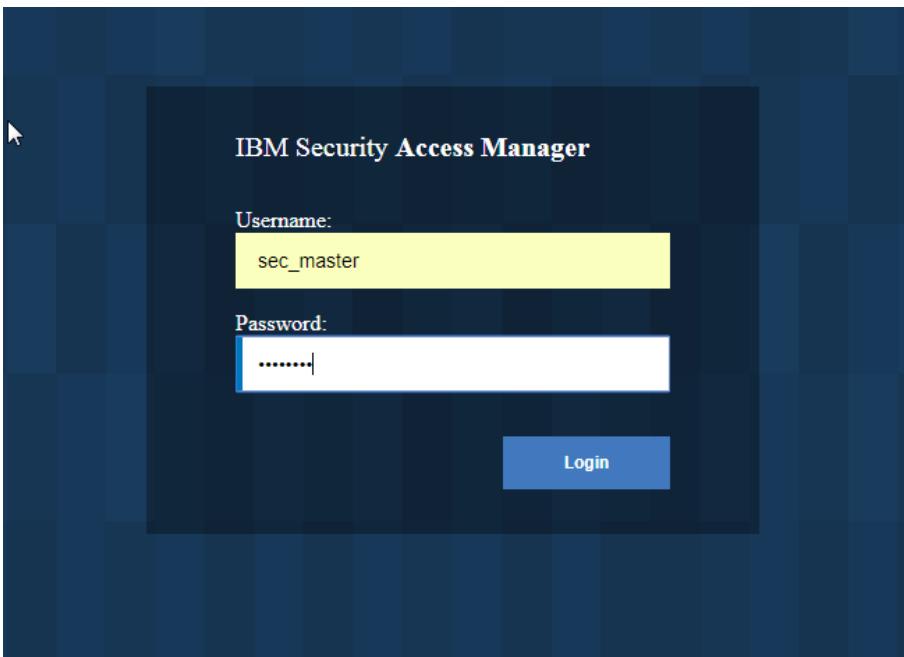
Save and close the window.

4.3.1.1 Test /api Junction

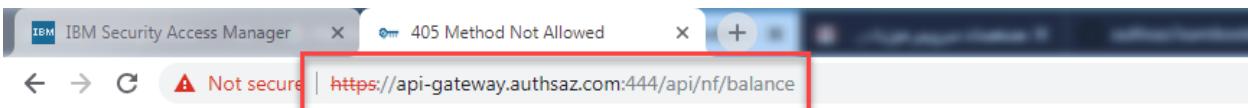
In your favorite web browser navigate to <https://api-gateway.authsaz.com:444/api/nf/balance>.



ISAM default form-based login page will be displayed. For now login with user **sec_master** and **Passw0rd**.



If you get **Method Not Allowed** error, you have configured the junction correctly.

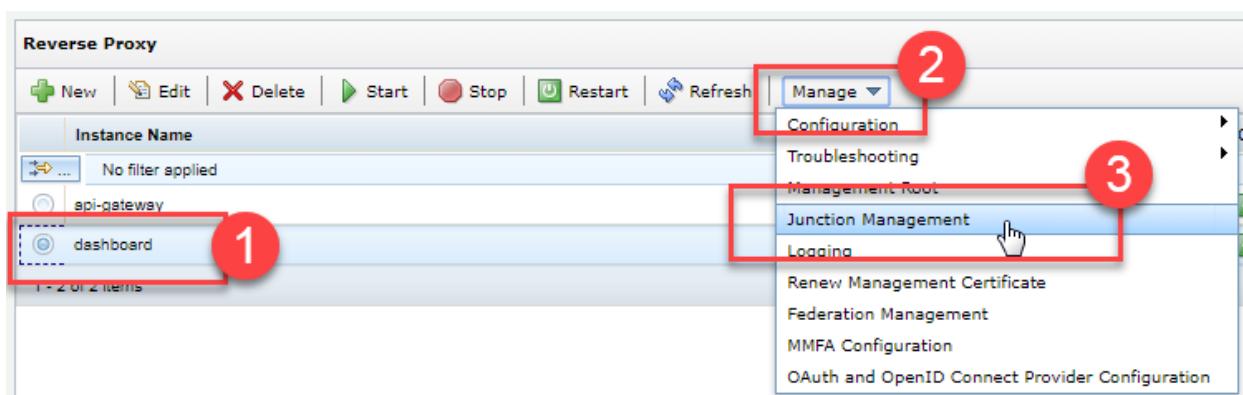


Method Not Allowed

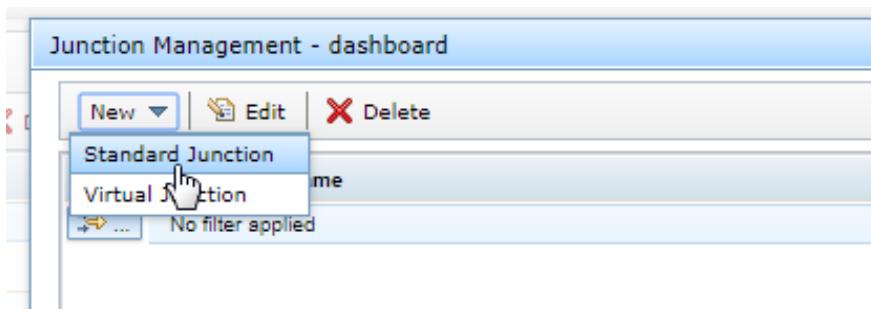
The method is not allowed for the requested URL.

4.3.2 Create /portal Junction

Use the Junction Management page to create junctions required for the **dashboard**. From the top menu, select **Secure Web Settings > Manage: Reverse Proxy**.



Select **dashboard** reverse proxy to manage its junctions. Then select **Manage > Junction Management**.

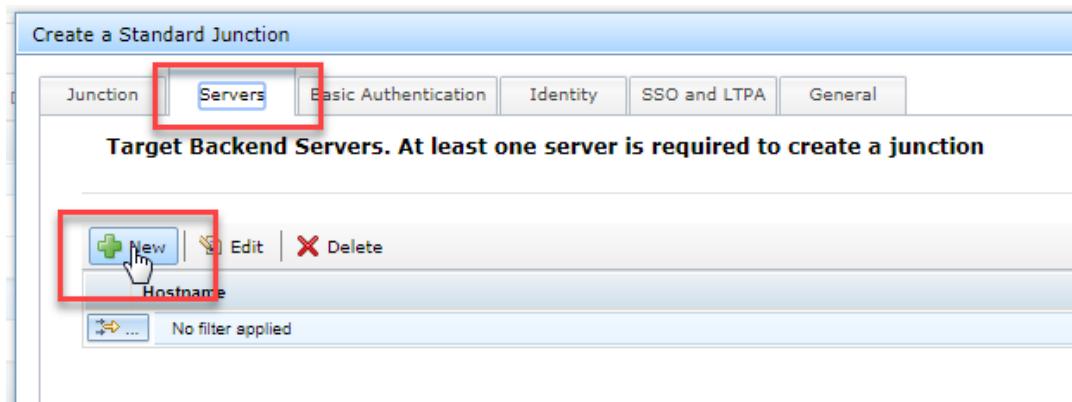


Click **New > Standard Junction**.

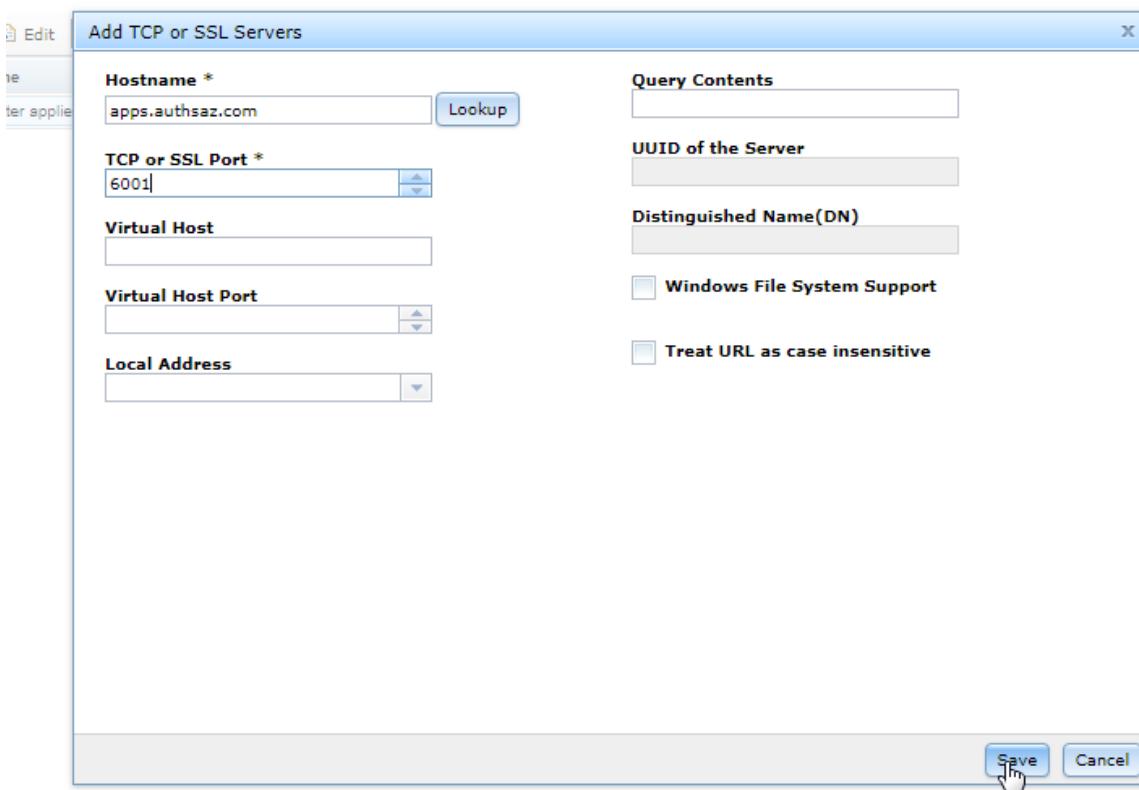
The screenshot shows the 'Create a Standard Junction' configuration page. The 'Junction' tab is selected. A red box highlights the 'Junction Point Name' field containing '/portal' and the checked 'Create Transparent Path Junction' checkbox. Another red box highlights the 'Server Name Indicator (SNI)' field. To the right, a 'Junction Type' section lists several options with radio buttons: TCP (selected), SSL, TCP Proxy, SSL Proxy, and Mutual.

On the Junction tab page:

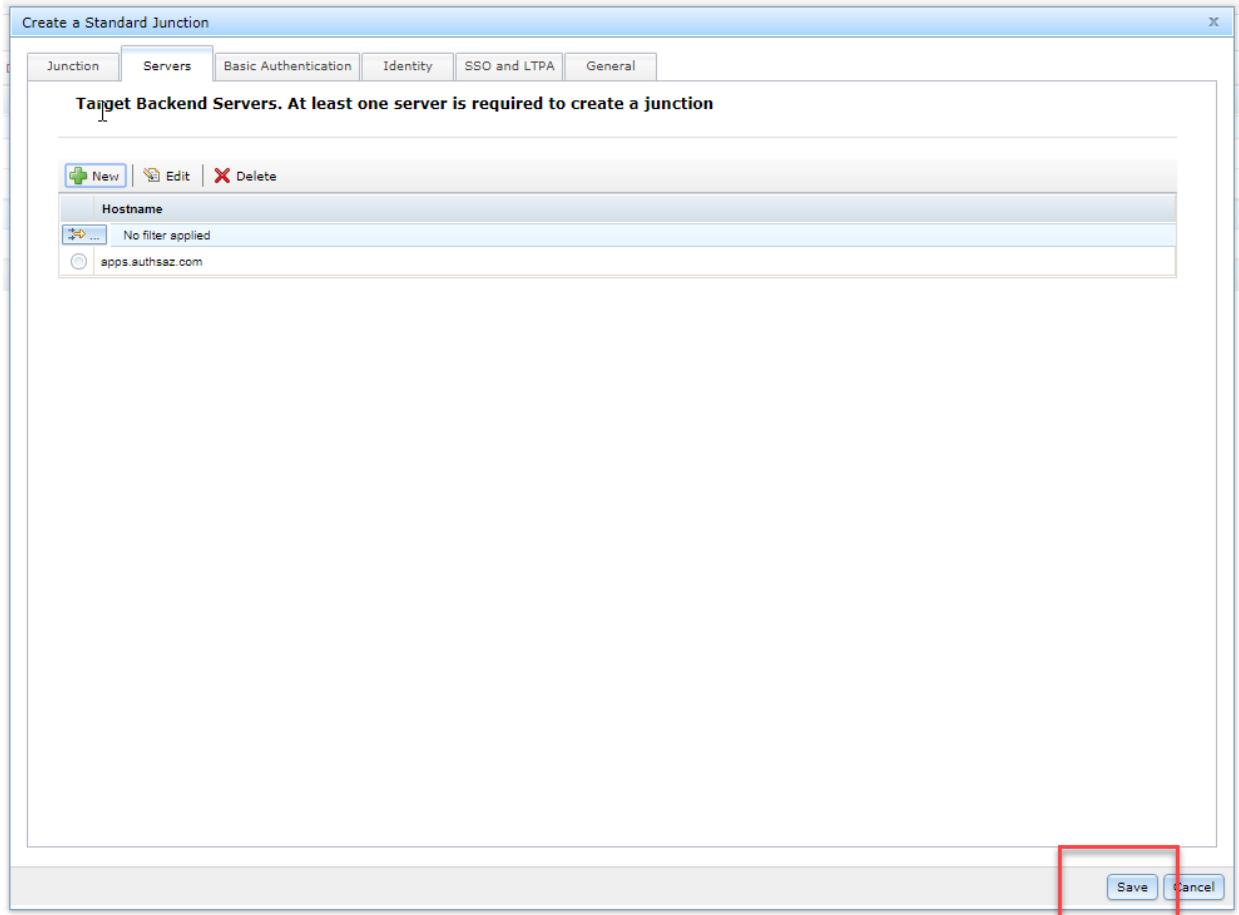
- Enter **/portal** as the *junction point name*.
- Select the *Create Transparent Path Junction* checkbox.
- Select **TCP** as *junction type* from the listed options



Select the **Servers** tab and click **New**.



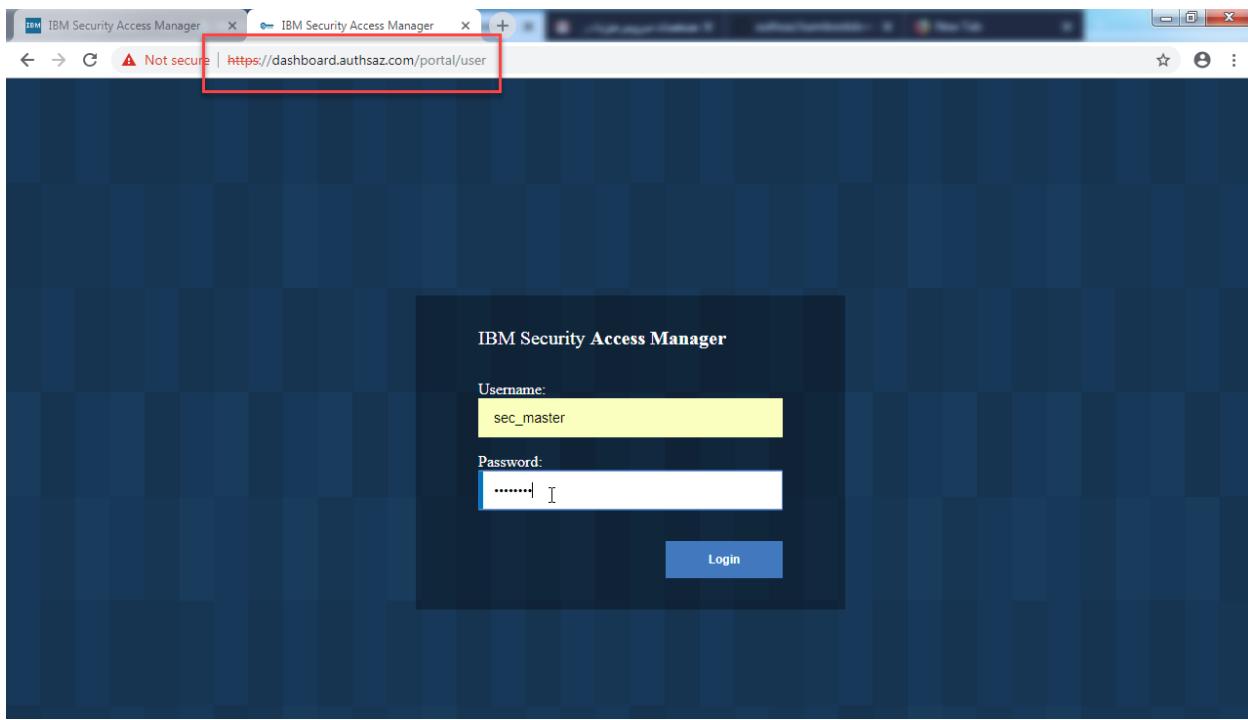
Enter **apps.authsaz.com** and **6001** as *Hostname* and *TCP or SSL Port* and then click **Save**.



Save and close the Junction Management window.

4.3.2.1 Test /portal Junction

In your favorite browser, navigate to <https://dashboard.authsaz.com/portal/user>. Login as **sec_master** using **Passw0rd** password.



You should see the dashboard page as below. Yet, don't worry about errors on the page.

A screenshot of the IBM Security Access Manager dashboard. The URL in the address bar is https://dashboard.authsaz.com/portal/user. A red box highlights the address bar. The dashboard shows sections for 'Authorization Grant' and 'Authenticators'. A modal dialog box is open in the center, displaying the message 'dashboard.authsaz.com says failure' with an 'OK' button. The 'Authorization Grant' section has a table with columns: Id, IsEnabled, ClientId, and Remove. The 'Authenticators' section has a table with columns: Id, Device Name, Device Type, OS Version, oAuth Grant, Enabled, and Remove. Buttons for 'New' and 'Refresh' are visible at the bottom of each section.

4.4 Configuring AAC

4.4.1 lsamcfg Tool

Various features of *Advanced Access Control* can be configured with the **lsamcfg** tool. The **lsamcfg** tool helps automate configuration of the following software and appliances:

- WebSEAL
- Security Access Manager appliance
- Security Access Manager for Web appliance
- Security Access Manager for Web software version
- Security Web Gateway appliance

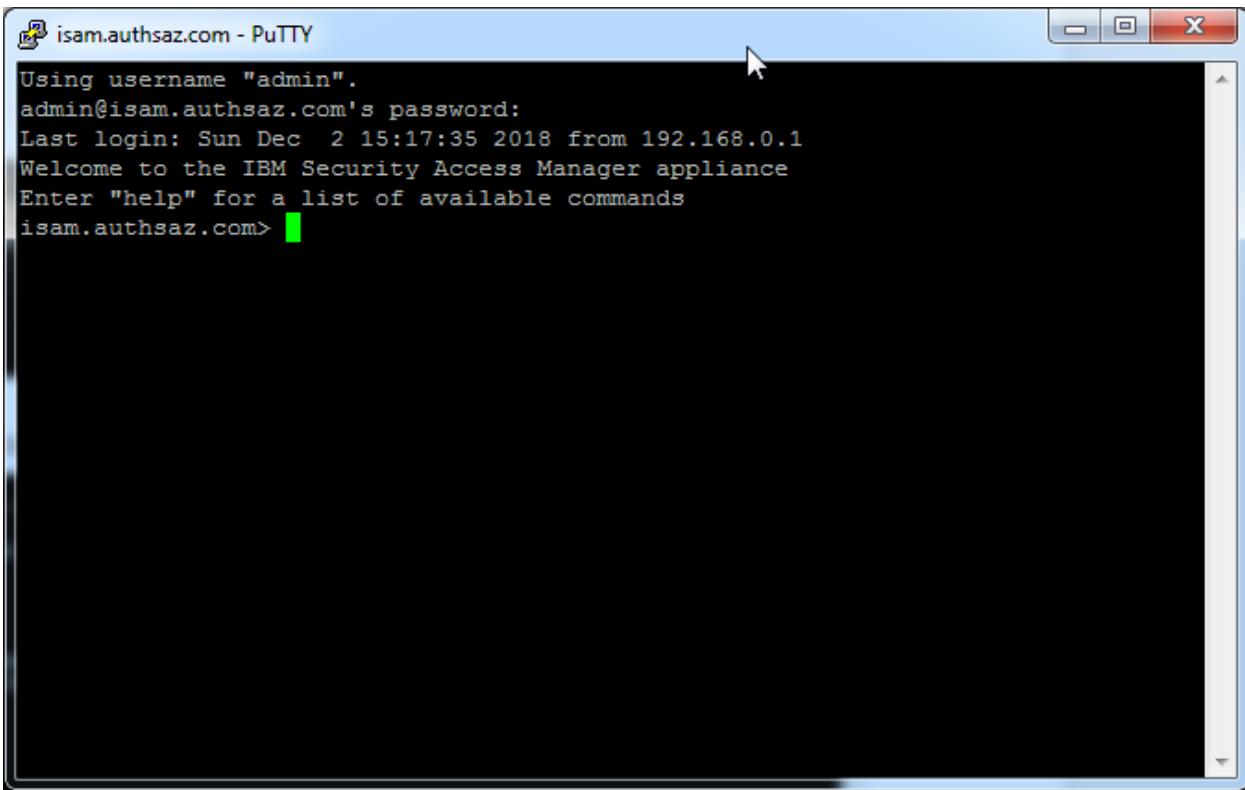
The tool helps with:

- Creating a junction that points to the *Advanced Access Control* runtime endpoint.
- Creating a *Security Access Manager* POP used by *Advanced Access Control* to attach a policy.
- Configuring the instance of the appliance that supplies the web function, such as WebSEAL, with the instance of the appliance that provides the authorization server for Advanced Access Control.
- Configuring SSL, sets up key stores, trust stores, and authentication configuration between *Advanced Access Control* and the web function.
- Configuring a set of default obligation-to-URL mappings for the authentication service.
- Modifying the *Security Access Manager* appliance authentication configuration to support the authentication service.

4.4.2 Procedure

Log in to the appliance command-line interface as an administrator.

Use an ssh client like putty and connect to **isam.authsaz.com** using **admin** username and **Passw0rd** password.



```
Using username "admin".
admin@isam.authsaz.com's password:
Last login: Sun Dec 2 15:17:35 2018 from 192.168.0.1
Welcome to the IBM Security Access Manager appliance
Enter "help" for a list of available commands
isam.authsaz.com>
```

Run the *Security Access Manager* configuration utility by using the command **isam aac config**.

Follow steps as below:

```
isam.authsaz.com> isam aac config
Security Access Manager Autoconfiguration Tool Version 9.0.5.0 [20180530-2317]

Select/deselect the capabilities you would like to configure by typing its number.
Press enter to continue:
[ X ] 1. Context-based Authorization
[ X ] 2. Authentication Service
[ X ] 3. API Protection
Enter your choice: ENTER
Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
Advanced Access Control Local Management Interface hostname: isam.authsaz.com
Advanced Access Control Local Management Interface port [443]: ENTER
Advanced Access Control administrator user ID [admin]: ENTER
Advanced Access Control administrator password: Passw0rd
Testing connection to https://isam.authsaz.com:443/.
SSL certificate information:
  Issuer DN: CN=isam.authsaz.com
  Subject DN: CN=isam.authsaz.com
SSL certificate fingerprints:
  MD5: BC:D1:1B:5D:93:08:AE:43:B3:4F:66:13:F0:DF:46:97
  SHA1: F3:85:91:70:9C:DA:C1:40:CD:C3:41:1A:4F:A4:A7:9F:FA:F3:48:DE
  SHA256:
  DB:6C:AE:33:F9:74:12:30:16:F2:20:1B:0E:97:D4:B2:0F:86:38:00:D8:65:A2:46:31:18:2F:D9:
  5A:F9:F8:B5
```

```

SSL certificate data valid (y/n): y
Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
Security Access Manager Local Management Interface hostname:
isam.authsaz.com
Security Access Manager Local Management Interface port [443]: ENTER
Security Access Manager Appliance administrator user ID [admin]: ENTER
Security Access Manager Appliance administrator password: PasswOrd
Testing connection to https://isam.authsaz.com:443/.

SSL certificate information:
  Issuer DN: CN=isam.authsaz.com
  Subject DN: CN=isam.authsaz.com
SSL certificate fingerprints:
  MD5: BC:D1:1B:5D:93:08:AE:43:B3:4F:66:13:F0:DF:46:97
  SHA1: F3:85:91:70:9C:DA:C1:40:CD:C3:41:1A:4F:A4:A7:9F:FA:F3:48:DE
  SHA256:
DB:6C:AE:33:F9:74:12:30:16:F2:20:1B:0E:97:D4:B2:0F:86:38:00:D8:65:A2:46:31:18:2F:D9:
5A:F9:F8:B5

SSL certificate data valid (y/n): y
Instance to configure:
  1. api-gateway
  2. dashboard
  3. Cancel
Enter your choice [1]: 1
Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
Security Access Manager administrator user ID [sec_master]: ENTER
Security Access Manager administrator password: PasswOrd
Security Access Manager Domain Name [Default]: ENTER
Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
Advanced Access Control runtime listening interface hostname: localhost
Advanced Access Control runtime listening interface port: 443
Select the method for authentication between WebSEAL and the Advanced Access Control
runtime listening interface:
  1. Certificate authentication
  2. User-id/password authentication
Enter your choice [1]: 2
Advanced Access Control runtime listening interface user ID: easuser
Advanced Access Control runtime listening interface password: PasswOrd
Testing connection to https://localhost:443.
Connection completed.

SSL certificate information:
  Issuer DN: CN=isam, O=ibm, C=us
  Subject DN: CN=isam, O=ibm, C=us
SSL certificate fingerprints:
  MD5: CE:E2:B5:1C:A7:69:EA:8E:DF:63:29:B7:92:B1:6E:03
  SHA1: 04:7F:62:7E:7D:47:EA:30:DF:4E:1C:E6:A9:FF:82:2D:24:40:1E:AD
  SHA256:
78:54:C5:96:1D:1F:A7:A9:E3:B9:F2:1C:86:20:22:C4:7B:C3:B9:32:65:8B:29:BC:F3:28:68:E6:
76:E4:BA:4D

SSL certificate data valid (y/n): y
Automatically add CA certificate to the key database (y/n): y
Restarting the WebSEAL server...
Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
Gathering properties to automatically configure: RTSS cluster, RBA POP
Gathering properties to automatically configure: Authentication service mappings
Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
The following files are available on the Security Access Manager Appliance. Choose
one for the '400 Bad Request' response page.

```

```

1. oauth_template_rsp_400_bad_request.html
2. oauth_template_rsp_401_unauthorized.html
3. oauth_template_rsp_502_bad_gateway.html
Enter your choice [1]: ENTER
The following files are available on the Security Access Manager Appliance. Choose one for the '401 Unauthorized' response page.
1. oauth_template_rsp_400_bad_request.html
2. oauth_template_rsp_401_unauthorized.html
3. oauth_template_rsp_502_bad_gateway.html
Enter your choice [2]: ENTER
The following files are available on the Security Access Manager Appliance. Choose one for the '502 Bad Gateway' response page.
1. oauth_template_rsp_400_bad_request.html
2. oauth_template_rsp_401_unauthorized.html
3. oauth_template_rsp_502_bad_gateway.html
Enter your choice [3]: ENTER
Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
The junction /mga contains endpoints that require Authorization HTTP header to be forwarded to the backend server.
Do you want to enable this feature? [y|n]? y

URLs allowing unauthenticated access:
https://api-gateway.authsaz.com:444/mga/sps/oauth/oauth20/authorize
https://api-gateway.authsaz.com:444/mga/sps/oauth/oauth20/introspect
https://api-gateway.authsaz.com:444/mga/sps/static
https://api-gateway.authsaz.com:444/mga/sps/mga/user/mgmt/html/common.js
URLs allowing all authenticated users access:
https://api-gateway.authsaz.com:444/mga/sps/ac
https://api-gateway.authsaz.com:444/mga/sps/xauth
https://api-gateway.authsaz.com:444/mga/sps/mga/user/mgmt/html
https://api-gateway.authsaz.com:444/mga/sps/oauth/oauth20/clients
https://api-gateway.authsaz.com:444/mga/sps/oauth/oauth20/logout
https://api-gateway.authsaz.com:444/mga/sps/common/qr
https://api-gateway.authsaz.com:444/mga/sps/mga/user/mgmt/otp
https://api-gateway.authsaz.com:444/mga/sps/mga/user/mgmt/device
https://api-gateway.authsaz.com:444/mga/sps/mga/user/mgmt/questions
https://api-gateway.authsaz.com:444/mga/sps/mga/user/mgmt/grant
URLs used for authentication:
https://api-gateway.authsaz.com:444/mga/sps/oauth/oauth20/session

Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: 1
-----
Planned configuration steps:

A junction to the Security Access Manager server will be created at /mga.

The POP oauth-pop will be created.
The POP rba-pop will be created.

ACLs denying access to all users will be attached to:
/WebSEAL/isam.authsaz.com-api-gateway/mga

ACLs allowing access to all users will be attached to:
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/authsvc
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/xauth
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/authservice/authentication
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/oauth/oauth20/authorize
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/oauth/oauth20/introspect
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/static

```

```
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/mga/user/mgmt/html/common.js  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/oauth/oauth20/session  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/oauth/oauth20/token  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/apiauthsvc
```

ACLs allowing access to all authenticated users will be attached to:

```
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/auth  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/ac  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/xauth  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/mga/user/mgmt/html  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/oauth/oauth20/clients  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/oauth/oauth20/logout  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/common/qr  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/mga/user/mgmt/otp  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/mga/user/mgmt/device  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/mga/user/mgmt/questions  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/mga/user/mgmt/grant
```

EAI authentication will be enabled for the endpoints:

```
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/oauth/oauth20/session  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/auth  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/authservice/authentication  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/authsvc  
/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/apiauthsvc
```

Certificate authentication will be disabled.

HTTP-Tag-Value header insertion will be configured for the attributes:
user_session_id=user_session_id

Press 1 for Next, 2 for Previous, 3 to Repeat, C to Cancel: **1**

Beginning configuration...

```
Attaching ACLs.  
Creating ACL isam_mobile_nobody.  
Creating ACL isam_mobile_unauth.  
Creating ACL isam_mobile_rest.  
Creating ACL isam_mobile_rest_unauth.  
Creating ACL isam_mobile_anyauth.  
Creating junction /mga.
```

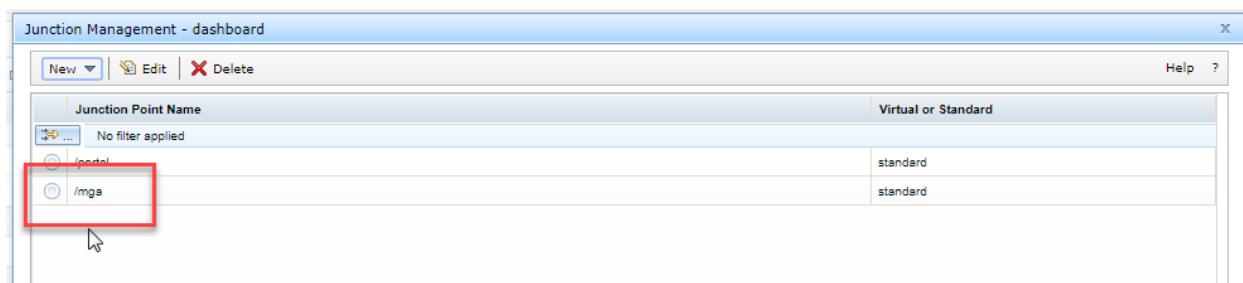
Editing configuration file...

```
Restarting the WebSEAL server...  
Configuration complete.  
isam.authsaz.com>
```

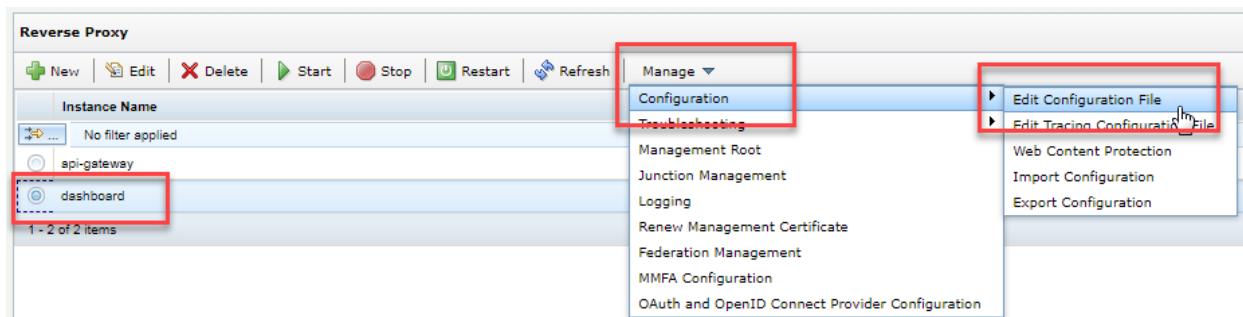
You should see a newly created junction **/mga** in **api-gateway** junctions list:



Repeat above steps for **dashboard** reverse proxy. You should see **/mga** in **dashboard** junctions list, as well.



4.5 Modify Reverse Proxy Instance Configuration File



Select the checkbox for the **dashboard** Reverse Proxy instance. Click on **Manage** and select **Configuration > Edit Configuration File** from the pop-up menu.

This will open the configuration file where we need to make a number of changes.

To find a location in this file, use the browser's search function. In Firefox this is activated using **Ctrl-f**.

First we configure Websocket support. This configuration is needed so that clients can connect to the WebSocket service that MMFA uses for notification of transaction completion.

```
[websocket]
```

```
max-worker-threads = 20
jct-read-inactive-timeout = 300
clt-read-inactive-timeout = 300
jct-write-blocked-timeout = 300
clt-write-blocked-timeout = 300
```

Click **Save** to save the updated configuration.

The screenshot shows the 'Reverse Proxy' management interface. At the top, there is a yellow warning bar with the text: 'There is currently one undeployed change. Click here to review the changes or apply them to the system.' Below the bar is a toolbar with buttons for New, Edit, Delete, Start, Stop, Restart, Refresh, and Manage. A dropdown menu for 'Manage' is open. The main area displays a table with two rows: 'api-gateway' and 'dashboard'. The 'api-gateway' row has a 'Started' status and a 'True' value next to it. The 'Changes are Active' indicator is also present. The 'Restart' button in the toolbar and the 'False' status in the table are highlighted with red boxes.

Deploy Pending changes.

This screenshot shows the same 'Reverse Proxy' interface after deploying pending changes. The 'Restart' button in the toolbar is highlighted with a red box. In the table, the 'api-gateway' row now has a 'False' status next to it, which is also highlighted with a red box. The 'Changes are Active' indicator is still present.

Restart Proxy instance.

This screenshot shows the 'Reverse Proxy' interface with the 'api-gateway' instance selected in the list. A context menu is open over the 'api-gateway' row, with 'Manage' selected. A submenu is shown with 'Configuration' highlighted. The 'Edit Configuration File' option in this submenu is highlighted with a red box. Other options like 'Edit Tracing Configuration File', 'Web Content Protection', 'Import Configuration', and 'Export Configuration' are also visible.

Now, select the checkbox for the **api-gateway** Reverse Proxy instance. Click on **Manage** and select **Configuration > Edit Configuration File** from the pop-up menu.

As API consumers will only authenticate with OAuth, we disable form-based login

```
[forms]
forms-auth = none
```

To configure websocket support for **api-gateway** reverse proxy, edit **websocket** stanza as below:

```
[websocket]
max-worker-threads = 20
```

```
jct-read-inactive-timeout = 300  
clt-read-inactive-timeout = 300  
jct-write-blocked-timeout = 300  
clt-write-blocked-timeout = 300
```

Add the following line to **junction:/mga** stanza:

```
[junction:/mga]  
...  
managed-cookies-list = *JSESSIONID*,PD_*
```

5 Configure OAuth, SCIM, and MMFA

As we mentioned in chapter 1, using OAuth along with an out of band authentication method could provide a strong authentication mechanism to protect APIs. IBM has introduced Mobile Multi-Factor Authentication (MMFA) in ISAM since 9.0.2.1 version.

MMFA uses the SCIM interface to provide access to read and update information about the authentication applications a user has registered against their account and to read and update information related to pending MMFA "transactions".

In this chapter, we will configure OAuth, SCIM and MMFA modules.

5.1 Configure OAuth

In our deployment, we require two OAuth definitions. One for MMFA Authenticator application (e.g *IBM Verify*) and the other for applications registered by merchants.

5.1.1 Create Authenticator Definition

Authenticator definition will be used to provide OAuth to MMFA authenticator applications. In this book we use *IBM Verify* as the authenticator application.

The screenshot shows the IBM Security Access Manager dashboard. The top navigation bar includes links for Home, Monitor, Secure Web Settings, and Secure Access Control. A red box highlights the 'Secure Access Control' link. The main content area has three columns: Policy, Manage, and Global Settings. A red box highlights the 'Policy' column under 'OpenID Connect and API Protection'. The 'Manage' column lists various options like Devices, Grants, Database Maintenance, etc. The 'Global Settings' column lists options like Advanced Configuration, User Registry, Runtime Parameters, etc.

Policy	Manage	Global Settings
Access Control	Devices	Advanced Configuration
Authentication	Grants	User Registry
Risk Profiles	Database Maintenance	Runtime Parameters
Attributes	SCIM Configuration	Template Files
Obligations	Push Notification Providers	Mapping Rules
OpenID Connect and API Protection	MMFA Configuration	Distributed Session Cache
Information Points	Attribute Source	Server Connections
Extensions		Point of Contact
		Access Policies

Navigate to **Advanced Access Control > Policy: OpenID Connect and API Protection**.

The screenshot shows the IBM Security Access Manager web interface. At the top, there is a navigation bar with links for Home, Monitor, Secure Web Settings, Secure Access Control, Secure Federation, and Connected IBM Cloud. Below the navigation bar, there is a sub-navigation bar for OpenID Connect and API Protection, with tabs for Definitions, Resources, Clients, and Mapping Rules. The Definitions tab is highlighted with a red box. In the main content area, there is a section titled "API Definition" with a "Create" button, which is also highlighted with a red box.

Click **Create** to create a new API Protection Definition.

The screenshot shows the "Create API Protection Definition" form. At the top, there are "Save" and "Cancel" buttons. The form has several sections:

- Name:** Authenticator (highlighted with a red box)
- Description:** MFA Authenticator (highlighted with a red box)
- Access Policy:** (dropdown menu)
- Grant Types:** A section with checkboxes:
 - Authorization code (highlighted with a red box)
 - Resource owner password
 - Client credentials
 - Implicit
 - JWT Bearer
 - SAML 2.0 Bearer
 - Device Grant
- Token Management:** A section with input fields:
 - Access token lifetime (seconds): 3,600
 - Access token length: 20
 - Enforce single-use authorization grant (highlighted with a red box)
 - Authorization code lifetime (seconds): 300
 - Authorization code length: 30

Enter **Authenticator** as the *Name* and provide a *Description*.

Select checkbox for the **Authorization code**.

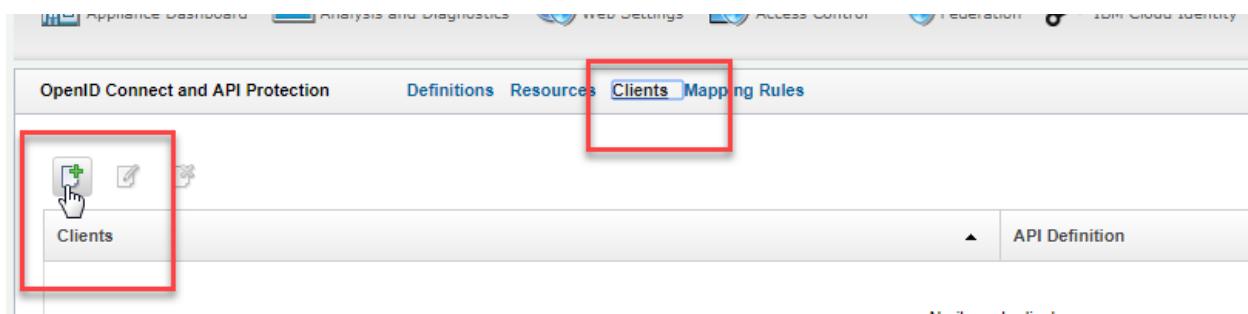
Expand the *Token Management* section and check the check-box for *Enforce single-use authorization grant*.

Click **Save** at the top of the screen.

Deploy changes using the link in the yellow warning message.

5.1.1.1 Create Client

Now that an API Protection Definition has been created (which defines a set of OAuth services) we can create an API Protection Client Definition and associate it with that definition.



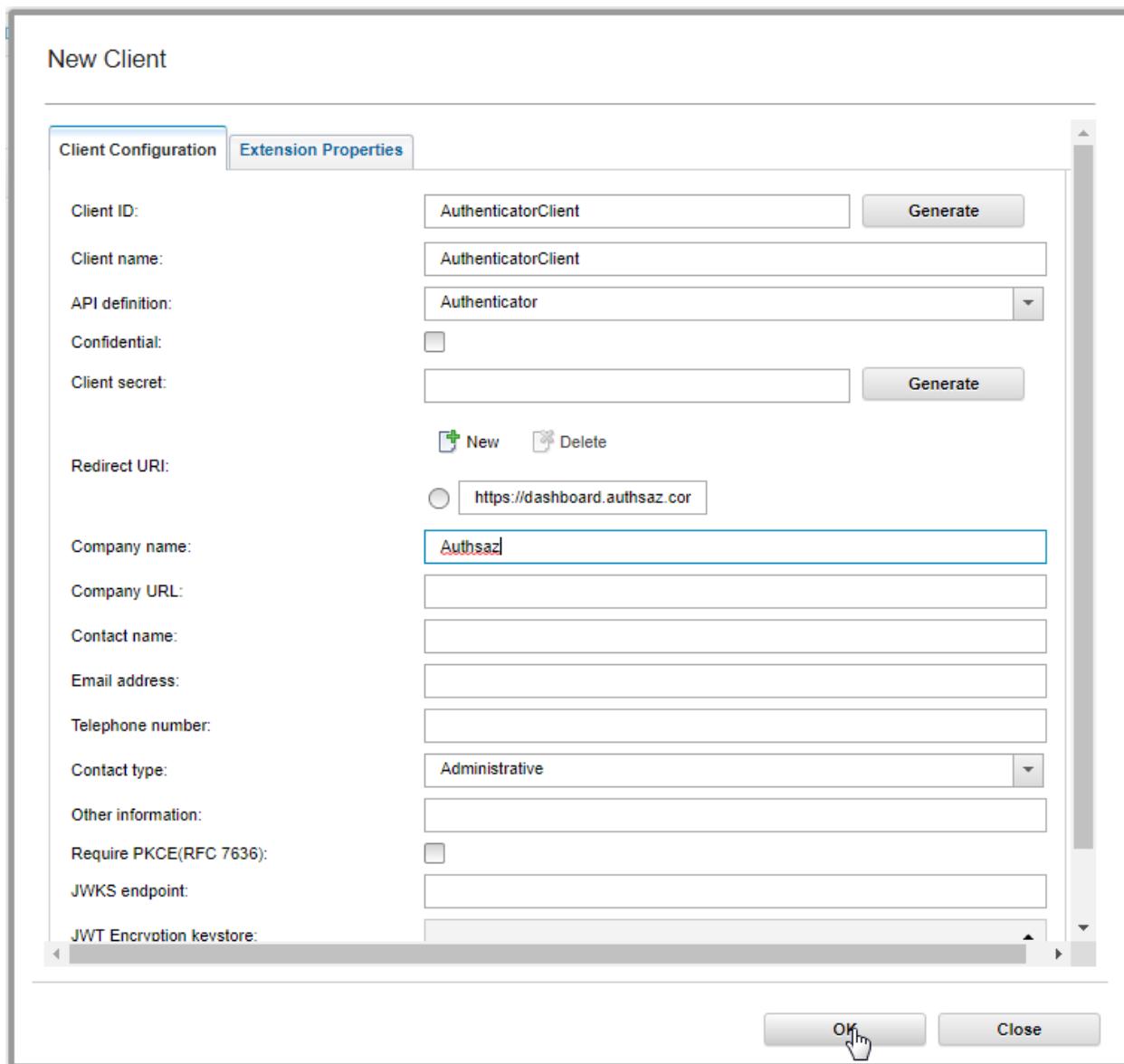
Select the **Clients** tab under API Protection and then click the **Create** button.

New Client

Client Configuration Extension Properties

Client ID:	AuthenticatorClient	Generate
Client name:	AuthenticatorClient	
API definition:	Authenticator	
Confidential:	<input type="checkbox"/>	
Client secret:		Generate
Redirect URI:	https://dashboard.authsaz.cor	New Delete
Company name:	Authsaz	
Company URL:		
Contact name:		
Email address:		
Telephone number:		
Contact type:	Administrative	
Other information:		
Require PKCE(RFC 7636):	<input type="checkbox"/>	
JWKS endpoint:		
JWT Encryption keystore:		

OK  Close



By default, the *Client ID* is a randomly generated string but that complexity is not required here.

Enter **AuthenticatorClient** as the *Client ID*.

Enter **AuthenticatorClient** as the *Client name*.

Select **Authenticator** from the drop-down list as the *API Definition*.

Clear the **Confidential** checkbox. An MMFA client is generally running on in an untrusted environment (i.e. a mobile device) and so a client secret has little value (and in any case, we only want to allow a specific redirect).

Enter the following as the *Redirect URI*:

https://dashboard.authsaz.com/portal/callback?client=AuthenticatorClient

Enter **Authsaz** as the *Company name* and click OK.

The screenshot shows a software interface titled "OpenID Connect and API Protection". At the top, there are tabs for "Definitions", "Resources", "Clients", and "Mapping Rules", with "Clients" being the active tab. A yellow warning bar at the top displays a yellow exclamation mark icon and the text "There is currently one undeployed change. Click here to review the changes or apply them to the system." A cursor is hovering over the "Click here" link. Below the bar are standard window control buttons (minimize, maximize, close).

Deploy the changes using the link in the yellow warning message.

5.1.2 Create AppsRegister Definition

AppsRegister definition will be used to provide OAuth to applications registered by merchants.

The screenshot shows the same software interface as the previous one, but now with a red box highlighting the "Create" button (a plus sign icon) in the toolbar. Below the toolbar, the text "API Definition" is visible. In the main pane, there is a list with one item: "Authenticator" under "MMFA Authenticator".

Click **Create** to create a new API Protection Definition.

OpenID Connect and API Protection Definitions Resources Clients Mapping Rules

Save Cancel

Name: AppsRegister

Description: Apps Register Definition

Access Policy:

Grant Types

- Authorization code
- Resource owner username password
- Client credentials

Token Management

Access token lifetime (seconds): 3,600

Access token length: 20

Enforce single-use authorization grant

Authorization code lifetime (seconds): 300

Trusted Clients and Consent

- Always prompt
- Never prompt
- Prompt once and remember

OpenID Connect Provider

Enable OpenID Connect

Issuer Identifier*: https://isam.authsaz.com

Point of Contact Prefix*: https://isam.authsaz.com/mga

Metadata URI: https://isam.authsaz.com/mga/sp/sps/oauth/oauth20/metadata/

id_token Lifetime*: 3,600

Enable client registration

Issue client secret

Enter **AppsRegister** as the *Name* and provide a *Description*.

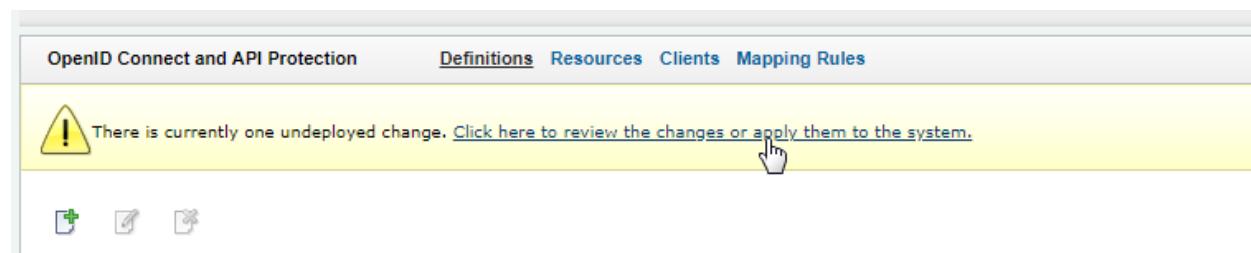
Select checkbox for the **Authorization code**.

Expand the *Token Management* section and check the check-box for *Enforce single-use authorization grant*.

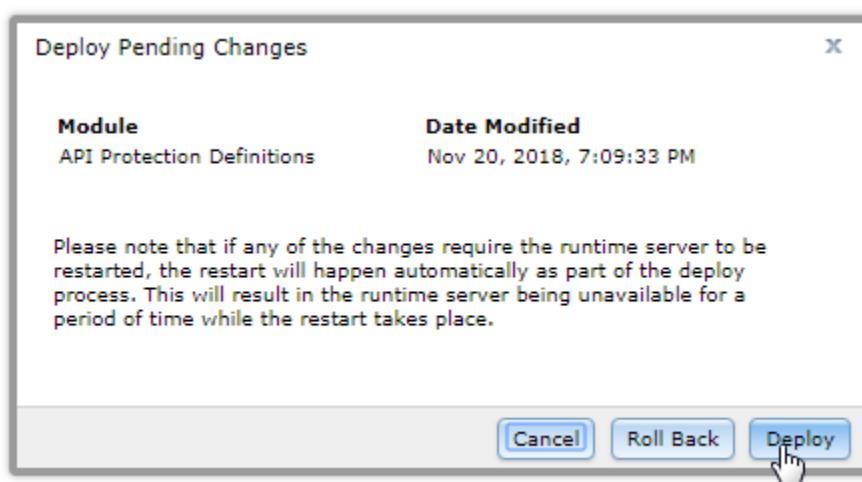
In *Trusted Clients and Consent* section select radio button for **Always prompts**.

In *OpenID Connect Provider* section enable the checkbox for **Enable OpenID Connect** and enter **https://isam.authsaz.com** as *Issuer Identifier* and **https://isam.authsaz.com/mga** as *Point Of Contact Prefix*. Enable checkbox for **Enable client registration** and then enable the checkbox for **Issue client secret**.

Click **Save** at the top of the screen.



Deploy changes using the link in the yellow warning message.



Click **Deploy**.

5.2 Configure SCIM

We will now configure the SCIM interface for our SAM Appliance.

5.2.1 Create an LDAP Server Connection

The SCIM configuration uses LDAP Server Connections to connect to the LDAP servers where user information is stored. This includes both ISAM-specific information and standard LDAP user objects. In this environment, we are using the embedded LDAP server to store both ISAM-specific data and user objects so only a single LDAP Server Connection is required (which we will use twice when we configure SCIM).

The screenshot shows the IBM Security Access Manager web interface. At the top, there is a navigation bar with several tabs: Home, Monitor, Secure Web Settings, Secure Access Control (which is highlighted with a red box), and Secure Federation. Below the navigation bar is a main content area divided into three columns: Policy, Manage, and Global Settings. The Global Settings column contains links for Advanced Configuration, User Registry, Runtime Parameters, Template Files, Mapping Rules, Distributed Session Cache, and Server Connections. A red box highlights the 'Server Connections' link under Global Settings. On the left side of the main content area, there is a sidebar with a tree view of various management categories like Access Control, Authentication, and SCIM Configuration.

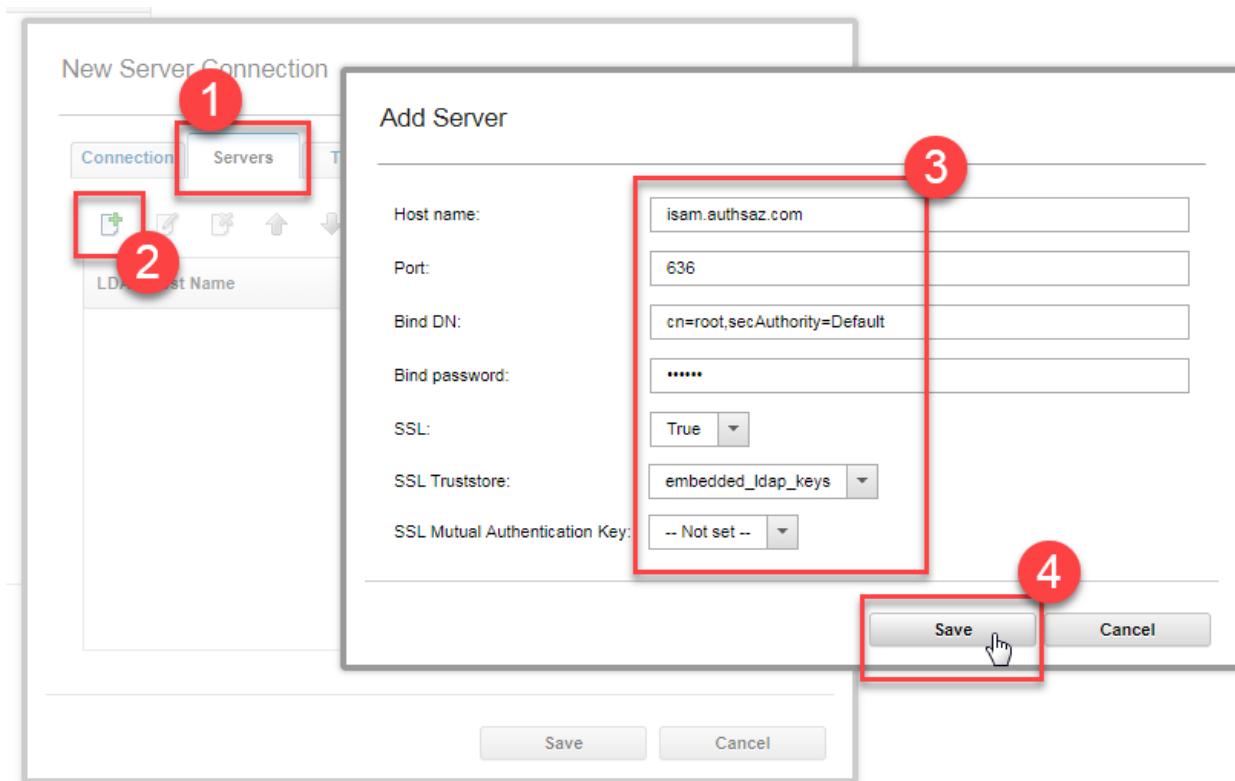
Navigate to **Secure Access Control > Global Settings: Server Connections**.

The screenshot shows the 'Server Connections' page in the IBM Security Access Manager interface. On the left, there's a sidebar with icons for Home, Monitor, Secure Web Settings, Secure Access Control, and Secure Federation. The main area has a title 'Server Connections' and a toolbar with icons for New, Edit, Delete, and Filter. A dropdown menu is open, showing options like Oracle, DB2, Solid DB, PostgreSQL, LDAP, SMTP, Web Service, Cloud Identity, and ISAM Runtime. The 'LDAP' option is highlighted with a red box and a cursor icon.

Click the **New** button and then select **LDAP** from the drop-down list.

The screenshot shows the 'New Server Connection' dialog box. At the top, there are tabs for 'Connection', 'Servers' (which is selected), and 'Tuning'. Below the tabs, there are fields for 'Name' (set to 'localldap'), 'Connection ID' (set to 'To be defined'), 'Description' (an empty text area), and 'Type' (set to 'LDAP'). At the bottom right, there are 'Save' and 'Cancel' buttons.

On the Connection tab, enter **localldap** as the **Name** and give a **Description**.



Select the **Servers** tab and click the **New** button to add a new server.

Enter **isam.authsaz.com** as the *Host name*.

Enter **636** as *Port*.

Enter **cn=root,secAuthority=Default** as *Bind DN*.

Select **True** for *SSL*.

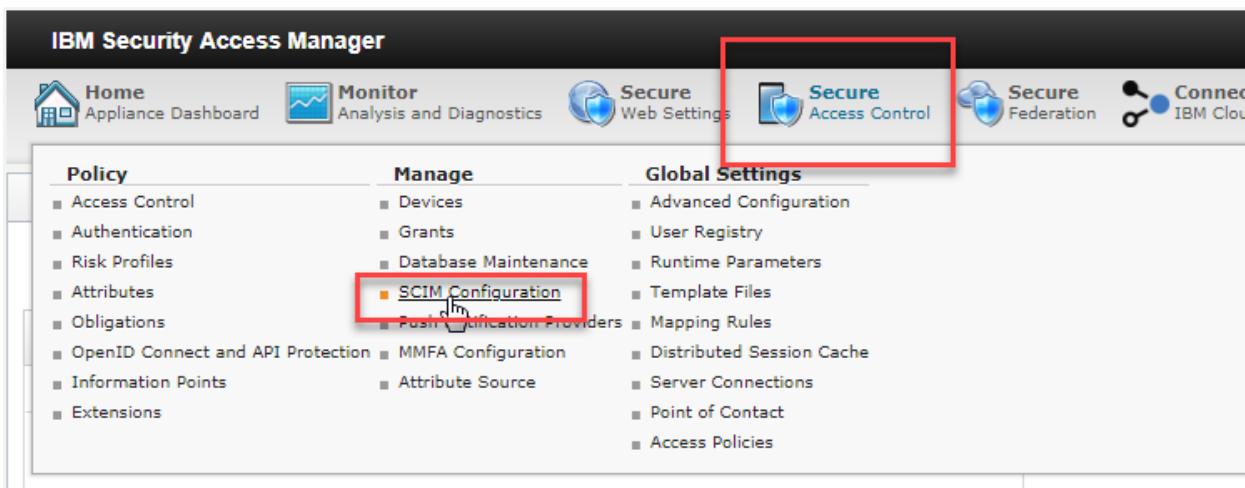
Select **embedded_ldap_keys** for *SSL Truststore*.

Click **Save** to save the Server Connection.

Deploy pending changes.

5.2.2 Configure SCIM

Now that the LDAP Server Connection is defined we can configure the SCIM interface. SCIM is part of the *Advanced Access Control* add-on.



Navigate to **Secure Access Control > Manage: SCIM Configuration**.

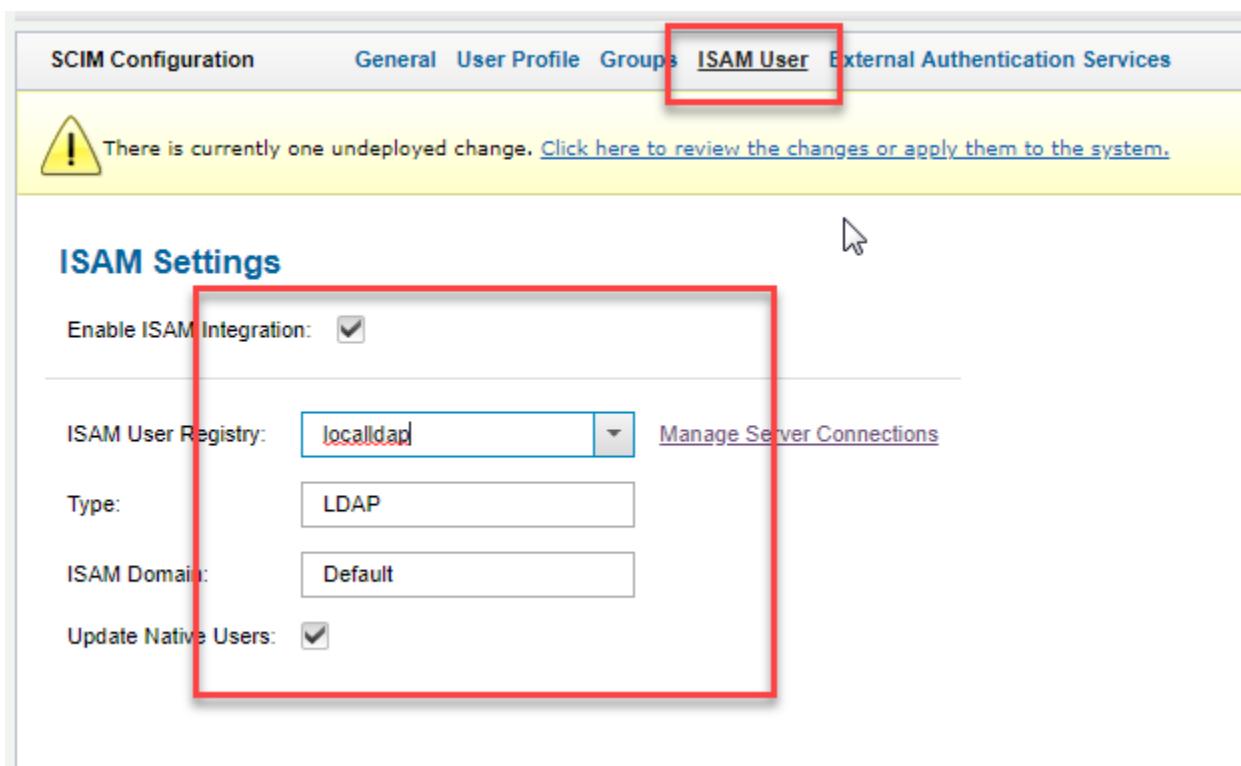
The screenshot shows the SCIM Configuration page. The top navigation tabs include SCIM Configuration, General, User Profile (which is highlighted with a red box), Groups, ISAM User, and External Authentication. The main content area is titled "LDAP Server". It includes fields for Server Connection (set to "localldap"), Type (set to "LDAP"), Enforce Password Policy (unchecked), Search Suffix ("dc=iswga"), User Suffix ("dc=iswga"), and User DN Attribute ("cn"). To the right, there is an "Attribute" panel with a list of attributes: active, addresses, costCenter, department, and displayNan. At the bottom, there is a link for "LDAP User Related Object Classes".

Select the **User Profile** tab.

Select **localldap** from the Server Connection drop-down list. This points the SCIM function at the LDAP server that contains user profile (i.e. `inetOrgPerson`) objects.

Enter **dc=iswga** as the Search Suffix and User Suffix. These tell the SCIM function where to look for user objects and where new users should be created.

Click **Save** at the bottom of the page (settings in each tab must be saved before moving on).



Select the **ISAM User** tab and click the checkbox to **Enable ISAM Integration**. This tells the SCIM functionality that it should look for ISAM-specific information (e.g. account-valid, password-valid) for the users that it finds in LDAP.

Select **localldap** as the *ISAM User Registry*.

Select the checkbox for **Update Native Users**. This tells the SCIM system to update ISAM-specific information when LDAP standard information is updated.

Click **Save**.

Deploy changes using the link in the yellow warning message.

5.2.3 Create SCIM junction

To allow applications (such as authentication devices) running outside the SAM appliance to access the SCIM interface, we will make it available via the Reverse Proxies. We want the SCIM interface to be available via both Reverse Proxies because it will be accessed by both browser and mobile systems. This means creating a Transparent Path Junction and making some configuration changes on each Reverse Proxy.

The screenshot shows the IBM Security Access Manager dashboard. The top navigation bar includes links for Home, Monitor, Secure Web Settings, and Secure Access. The main menu on the left is titled "Manage" and lists several options: Runtime Component, Reverse Proxy, Authorization Server, Distributed Session Cache, Policy Administration, Global Settings, Global Keys, URL Mapping, Junction Mapping, Client Certificate Mapping, User Name Mapping, Password Strength, Forms Based Single Sign-on, HTTP Transformation, Kerberos Configuration, RSA SecurID Configuration, SSO Keys, and LTPA Keys. The "Reverse Proxy" option is highlighted.

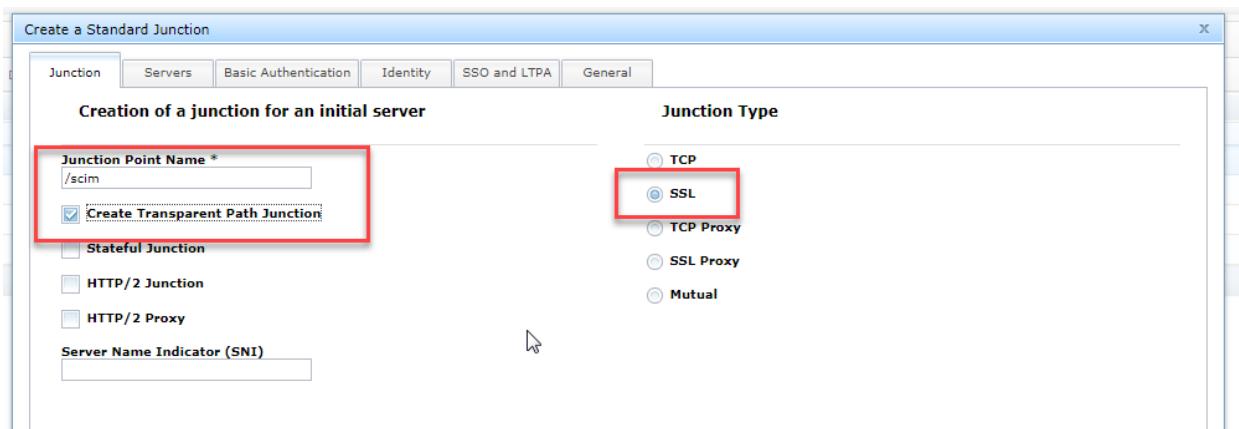
Navigate to **Secure Web Settings > Manage: Reverse Proxy**

This screenshot shows the "Reverse Proxy" management interface. On the left, a list of instances shows "api-gateway" selected and highlighted with a red box. On the right, a "Manage" dropdown menu is open, also highlighted with a red box. The menu items include Configuration, Troubleshooting, Management Root, Junction Management (which is being selected), and Logging. Other options like Renew Management Certificate, Federation Management, MMFA Configuration, and OAuth and OpenID Connect Provider Configuration are also listed.

Select the radio button for the **api-gateway** Reverse Proxy instance. Click **Manage** and then select **Junction Management** from the drop-down menu.

This screenshot shows the "Junction Management - api-gateway" interface. At the top, there is a toolbar with New, Edit, and Delete buttons. Below the toolbar, a list of junction types is shown, with "Standard Junction" highlighted and selected, also indicated by a red box. Other options include Virtual Junction and No filter applied.

Click **New** and select **Standard Junction** from the drop-down list.

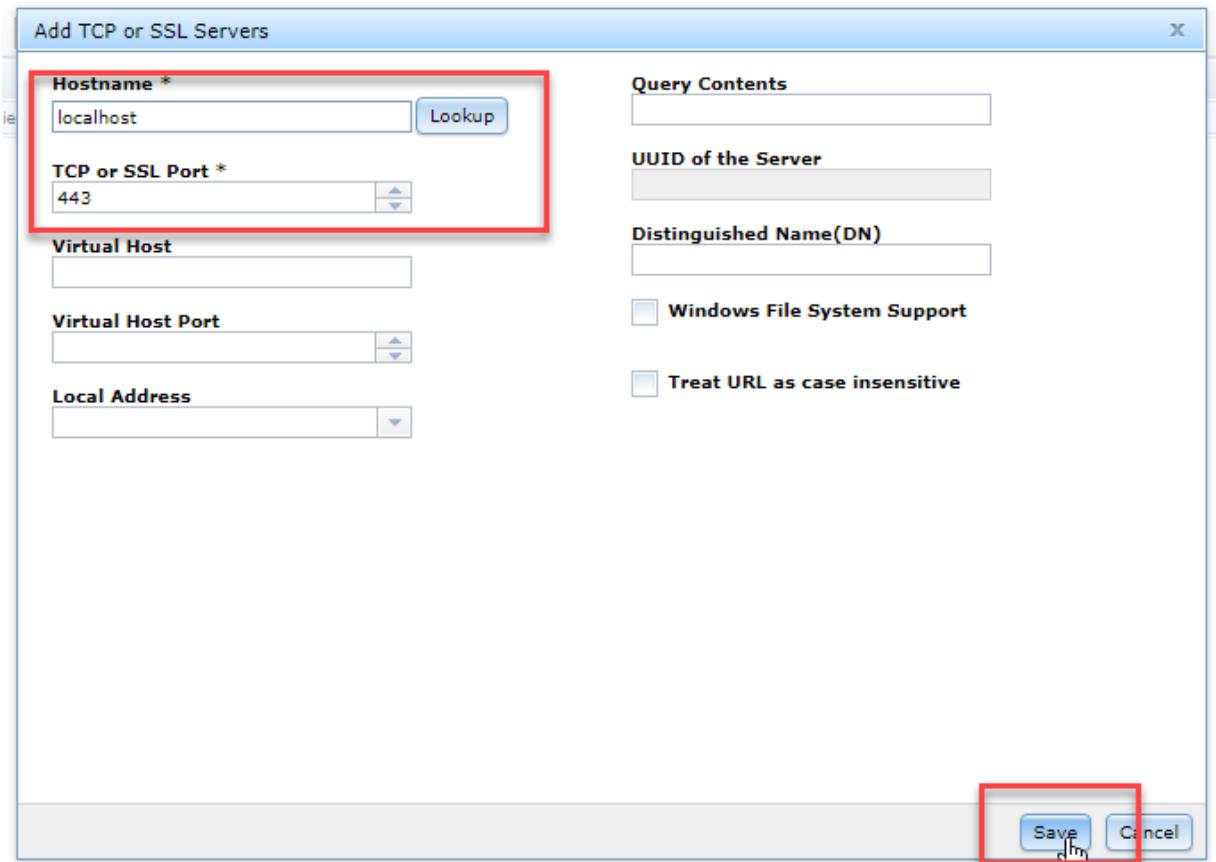


Enter **/scim** as the *Junction Point Name* and select check-box for *Create Transparent Path Junction*.

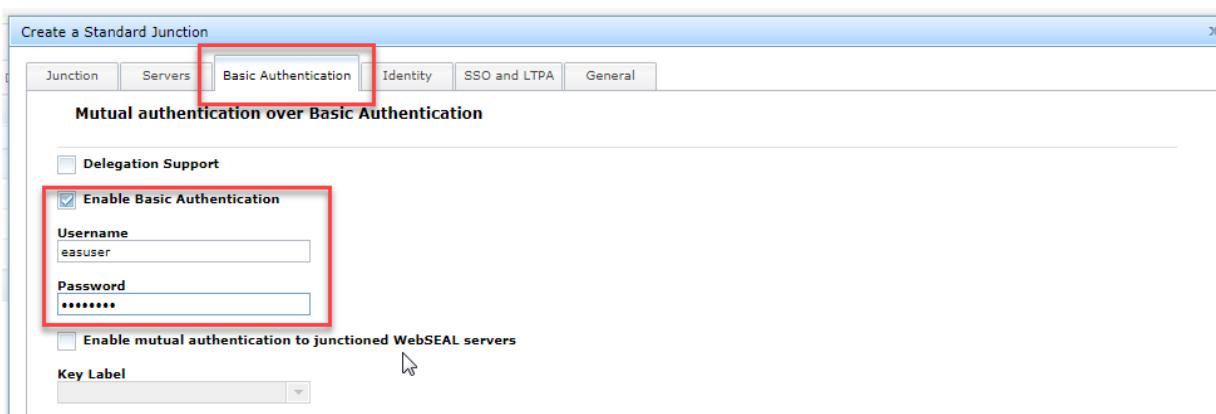
Select **SSL** radio-button for *Junction Type*.



Select the **Servers** tab and click **New**.

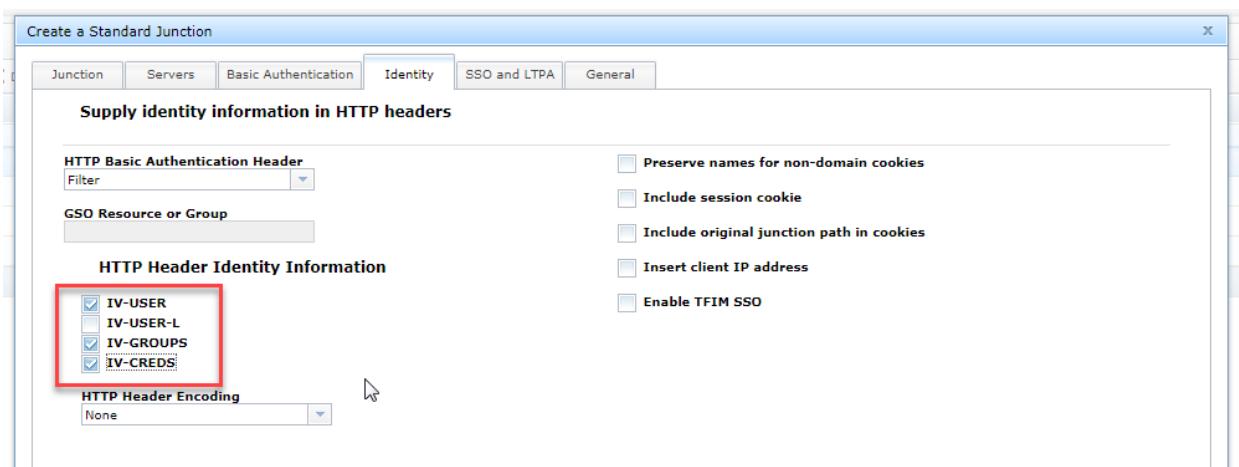


Enter localhost as the Hostname and click Save.



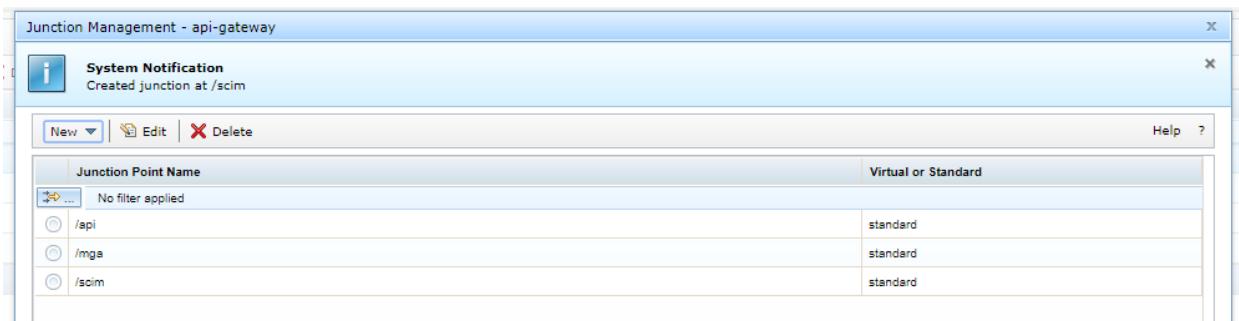
Select the **Basic Authentication** tab and check the checkbox for *Enable Basic Authentication*.

Enter **easuser** as the *Username* and enter **Passw0rd** as the *Password*.



Select the **Identity** tab.

Check the checkboxes for *IV-USER*, *IV-GROUPS*, and *IV-CREDS*. Then click **Save** to create the junction.

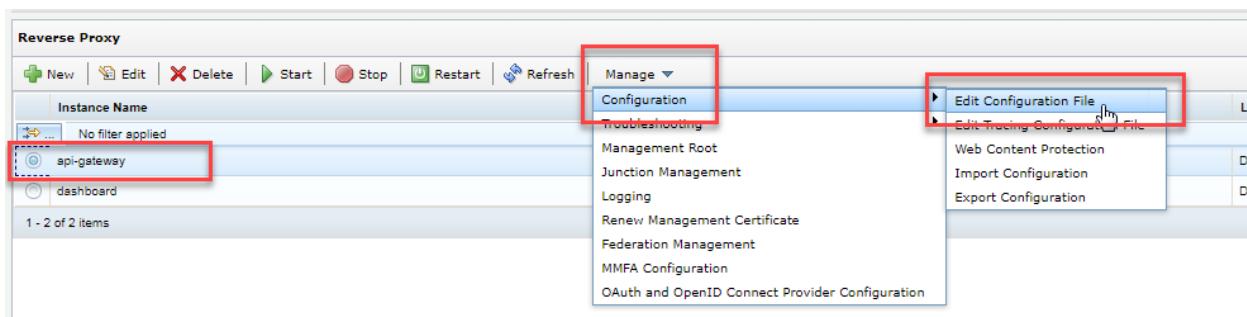


Click **Close** to close the Junction Management window.

5.2.4 Configure URL filtering for SCIM responses

URLs generated by the SCIM interface will reference the hostname where the AAC Runtime is listening (e.g. localhost). We need to configure the Reverse Proxy so that it will recognize these URLs and replace the hostname with its own. This requires some changes in the Reverse Proxy configuration file.

Navigate to **Secure Web Settings > Manage: Reverse Proxy**



Select the radio button for the **api-gateway** Reverse Proxy instance. Click on Manage and select **Configuration > Edit Configuration File** from the pop-up menu.

In the [filter-content-types] stanza add the following entry highlighted in red:

```
[filter-content-types]
...
type = application/scim+json
```

In the [script-filtering] stanza, enable script filtering and replacing absolute URLs with absolute URLs:

```
[script-filtering]
...
script-filter = yes
...
#rewrite-absolute-with-absolute = no
rewrite-absolute-with-absolute = yes
```

Click **Save**.

Deploy changes and **Restart** the Reverse Proxy instance.

Now, repeat *Create SCIM junction* and *Configure URL filtering for SCIM responses* sections for creating and configuring SCIM junction for **dashboard** reverse proxy.

5.2.5 Enable SCIM Demonstration Application

The SAM appliance includes a built-in demonstration application which is useful for validating SCIM functionality. However, it must be enabled before it is available.

The screenshot shows the IBM Security Access Manager (ISAM) interface. At the top, there's a navigation bar with links like Home, Monitor, Secure Web Settings, Secure Access Control, Secure Federation, Connect IBM Cloud Identity, and Manage System Settings (which is highlighted with a red box). Below the navigation bar is a main menu with several sections: Updates and Licensing, Network Settings, System Settings, and Secure Settings. Under System Settings, there's a sub-menu with items like General, DNS, Interfaces, Static Routes, Test Connection, Front End Load Balancer, Hosts File, Packet Tracing, Cluster Configuration, Date/Time, Administrator Settings, Management Authentication, Management Authorization, Management SSL Certificate, Account Management, SSL Certificates, File Downloads, Silent Configuration, Support Files, System Alerts, SNMP Monitoring, Restart or Shut down, and Application Locale. A specific item, "Advanced Tuning Parameters", is highlighted with a red box.

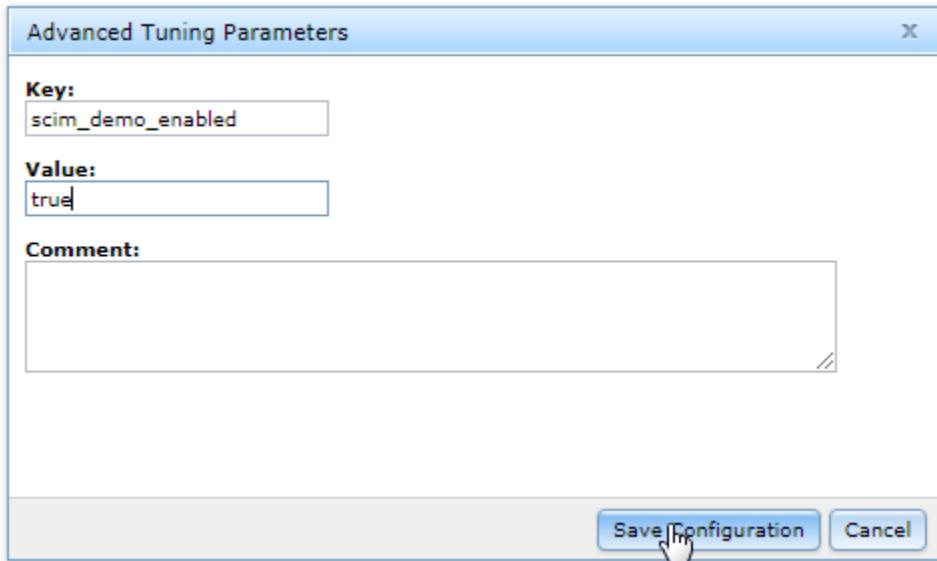
Navigate to **Manage System Settings > System Settings: Advanced Tuning Parameters**.

The screenshot shows the "Advanced Tuning Parameters" configuration page. The title bar says "Advanced Tuning Parameters (Change advanced tuning parameter values only under the supervision of IBM Customer Support.)". Below the title is a toolbar with "New", "Edit", and "Delete" buttons. The main area is a table with columns: Key, Value, and Comment. There are three entries:

Key	Value	Comment
nist.sp800-131a.strict	false	Advanced tuning for the FIPS and NIST SP800-131a configuration. This will control whether T
wga.rte.embedded.ldap.ssl.port	636	The port on which the embedded LDAP server will listen for LDAPS requests. The ISAM RTE
wga_rte.embedded.ldap.include.in.snapshot	false	Include the user registry data from the embedded user registry in the appliance snapshot files.

At the bottom left, it says "1 - 3 of 3 items". At the bottom right, there are pagination links: 10 | 25 | 50 | 100 | 200.

Click **New**.



Enter **scim_demo_enabled** as the *Key* and **true** as the *Value*. Click **Save Configuration**.

Deploy changes using the link in the yellow warning message.

5.3 Configure MMFA

We now need to configure MMFA endpoint and client configuration information that will be shared with the client via QR code.

This can be configured in the LMI.

The screenshot shows the IBM Security Access Manager Home page. The top navigation bar includes links for Home, Monitor, Secure Web Settings, Secure Access Control, and Secure Federation. The main menu is organized into three columns: Policy, Manage, and Global Settings. The Policy column lists Access Control, Authentication, Risk Profiles, Attributes, Obligations, OpenID Connect and API Protection, Information Points, and Extensions. The Manage column lists Devices, Grants, Database Maintenance, SCIM Configuration, Push Notification Providers, and Attribute Source. The Global Settings column lists Advanced Configuration, User Registry, Runtime Parameters, Template Files, Mapping Rules, Distributed Session Cache, Server Connections, Point of Contact, and Access Policies. At the bottom, there are two status indicators: 'In' with version 1.1 and 'In' with version 1.2.

Navigate to **Secure Access Control > Manage: MMFA Configuration**.

The screenshot shows the Mobile Multi-factor Authentication Configuration page under the Secure Access Control section. The top navigation bar is identical to the Home page. The main content area has tabs for General, Discovery Mechanisms, Custom QR Code Options, and a selected tab for Wizard. Below the tabs is a toolbar with Edit, Wizard (highlighted with a cursor), and Clear Configuration buttons. A table below the toolbar shows a single row with 'Key' and 'Value' columns, both currently empty. A message at the bottom right states 'No items to display'.

Click on **Wizard** to begin configuration.

Initial Properties

Client ID:

AuthenticatorClient

Reverse Proxy URL:

<https://api-gateway.authsaz.com:444>

Next

Cancel

Set the *Client ID* to **AuthenticatorClient**, and the *Reverse Proxy URL* to **https://api-gateway.authsaz.com:444**.

Then press **Next**.

AAC Endpoints

Discovery Endpoint:

<https://api-gateway.authsaz.com:444/mga/sps/mmfa/user/mgmt/details>

Token Endpoint:

<https://api-gateway.authsaz.com:444/mga/sps/oauth/oauth20/token>

HOTP Shared Secret Endpoint:

<https://api-gateway.authsaz.com:444/mga/sps/mga/user/mgmt/otp/hotp>

TOTP Shared Secret Endpoint:

<https://api-gateway.authsaz.com:444/mga/sps/mga/user/mgmt/otp/totp>

Previous

Next

Cancel

The generated defaults for the *AAC Endpoints* should be fine. Just press **Next**.

SCIM Endpoints

Enrollment Endpoint:

Transaction Endpoint:

Previous **Next**  **Cancel**

The *SCIM Endpoints* defaults should also be ok. Press **Next**.

Mobile Channel Endpoints

Endpoint Prefix:

 **Previous** **Save** **Cancel**

The default *Endpoint Prefix* does not need to be changed. Click **Save**.

On Saving, all configuration endpoints should be displayed in the table:

Mobile Multi-factor Authentication Configuration [General](#) [Discovery Mechanisms](#) [Custom QR Code Options](#)

 There is currently one undeployed change. [Click here to review the changes or apply them to the system.](#)

[Edit](#) [Wizard](#) [Clear Configuration](#)

Key	Value
Client ID	AuthenticatorClient
Discovery Endpoint	https://api-gateway.authsaz.com:444/mga/sp/sps/mmfa/user/mgmt/details
Endpoint Prefix	https://api-gateway.authsaz.com:444/mga
Enrollment Endpoint	https://api-gateway.authsaz.com:444/scim/Me
HOTP Shared Secret Endpoint	https://api-gateway.authsaz.com:444/mga/sp/sps/mga/user/mgmt/otp/hotp
TOTP Shared Secret Endpoint	https://api-gateway.authsaz.com:444/mga/sp/sps/mga/user/mgmt/otp/totp
Token Endpoint	https://api-gateway.authsaz.com:444/mga/sp/sps/oauth/oauth20/token
Transaction Endpoint	https://api-gateway.authsaz.com:444/scim/Me?attributes=urn:ietf:params:scim:schemas:core:2.0:User

Deploy the changes using the link in the yellow warning message.

Successfully deployed 0 pending changes.

Appliance Dashboard Analysis and Diagnostics Web Settings Access Control Federation IBM Cloud IDE

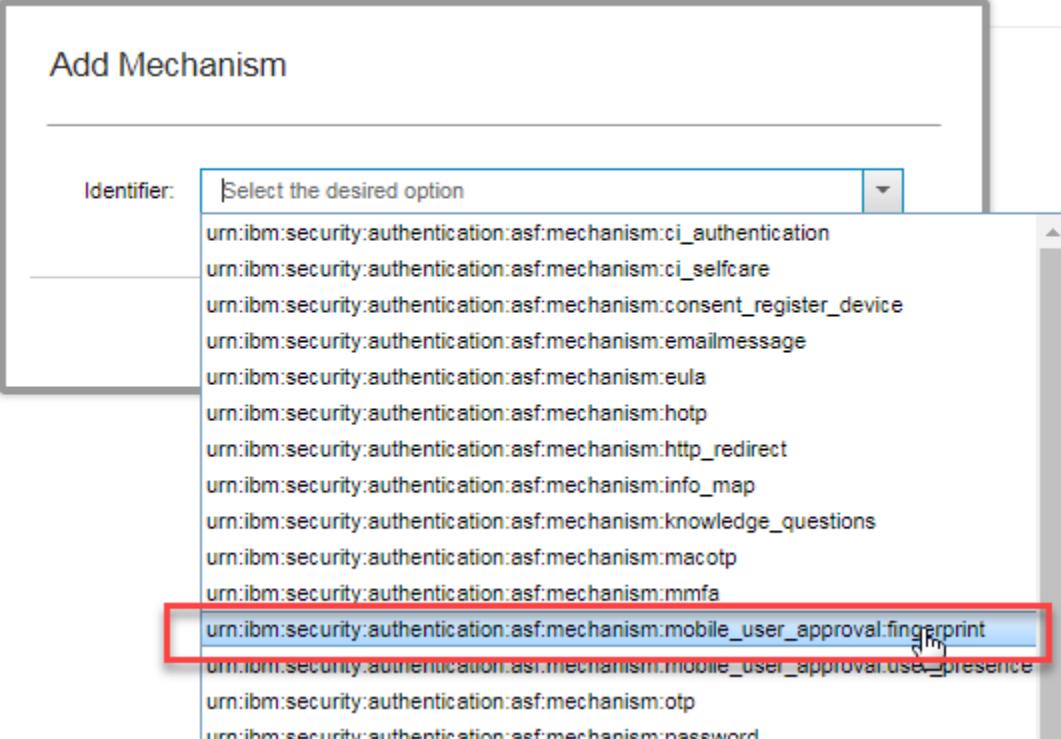
Mobile Multi-factor Authentication Configuration [General](#) **Discovery Mechanisms** [Custom QR Code Options](#)

[Add](#) [Remove](#)

Mechanisms

No items to display

Navigate to the **Discovery Mechanisms** tab and click **Add**.



Select **urn:ibm:security:authentication:ASF:mechanism:mobile_user_approval:fingerprint** from the drop-down list and click **Save**.

Repeat this step to add the following additional mechanisms:

- **urn:ibm:security:authentication:ASF:mechanism:mobile_user_approval:user_presence**
- **urn:ibm:security:authentication:ASF:mechanism:totp**

The list should now look like this:

Mobile Multi-factor Authentication Configuration General Discovery Mechanisms Custom QR Code Options

Mechanisms

Name: Fingerprint Approval
Identifier: urn:ibm:security:authentication:ASF:mechanism:mobile_user_approval:fingerprint

Name: User Presence Approval
Identifier: urn:ibm:security:authentication:ASF:mechanism:mobile_user_approval:user_presence

Name: TOTP One-time Password
Identifier: urn:ibm:security:authentication:ASF:mechanism:totp

This will indicate to the *IBM Verify* application that it should enroll TOTP, user-presence, and fingerprint authentication methods.

Deploy the changes using the link in the yellow warning message.

Mobile Multi-factor Authentication Configuration General Discovery Mechanisms **Custom QR Code Options**

Key ▲ **Value**

No items to display

Navigate to the **Custom QR Code Options** menu, and click **Add**.

Add Option

Key: ignoreSslCerts

Value: true

Save Cancel

Enter **ignoreSslCerts** as the *Key* and **true** as the *Value*.

This option will be added to registration QR codes and instructs *IBM Verify* that it is OK to connect to SSL endpoints which do not host a valid SSL certificate. You would not set this in a production environment but we need it for our test system.

Click **Save**.

5.4 Configure Policy Administration

IBM Security Access Manager

Home Appliance Dashboard Monitor Analysis and Diagnostics Secure Web Settings Secure Access Control Secure Federation

Manage Global Settings Global Keys

- Runtime Component
- Reverse Proxy
- Authorization Server
- Distributed Session Cache
- **Policy Administration**

■ URL Mapping ■ Junction Mapping ■ Client Certificate Mapping ■ User Name Mapping
■ Password Strength ■ Forms Based Single Sign-on ■ HTTP Transformation ■ Kerberos Configuration
■ RSA SecurID Configuration

Navigate to **Secure Web Settings > Manage: Policy Administration**.

Policy Administration

Task List	Security Access Manager Sign On
	Secure Domain <input type="text"/> *User Id sec_master *Password <password>.....</password> <input type="button" value="Sign On"/>

Log in to the **Policy Administration** using the administrator user id **sec_master** and the default password **PasswOrd**.

5.4.1 Create users

Policy Administration

Task List	Create User
User Search Users Create User 2 Import User Show Global User Policy Change My Password ▶ Group ▶ Object Space ▶ ACL ▶ POP ▶ AuthzRule ▶ GSO Resource ▶ Secure Domain	*User Id user1 3 *Common Name user1 *Surname user1 *Password <password>.....</password> *Confirm Password <password>.....</password> Description <input type="text"/> *Registry UID cn=user1,dc=iswga <input checked="" type="checkbox"/> Account Valid <input checked="" type="checkbox"/> GSO User <input checked="" type="checkbox"/> Password Valid <input type="checkbox"/> No Password Policy <input style="background-color: #0070C0; color: white; border: 1px solid #0070C0; border-radius: 5px; padding: 5px; width: fit-content; height: fit-content; font-weight: bold; font-size: 1em; margin-right: 10px;" type="button" value="Create"/> <input type="button" value="Cancel"/> 4

Expand the **User** from the **Task List** panel and click on the **Create User** link.

On the right panel, enter **user1** as *User Id*, **user1** as *Common Name and Surname*, **PasswOrd** as *Password*, and **cn=user1,dc=iswga** as *Registry UID*. Click **Create**.

Repeat steps above to create **user2**, **user3**, and **merchant** users.

The screenshot shows the Policy Administration interface with the 'User Search' task selected. The search criteria are set to find 100 users. The results table displays 8 users matching the search criteria. The users listed are: api-gateway-webseald/isam.authsaz.com, dashboard-webseald/isam.authsaz.com, ivmgrd/master, merchant, sec_master, user1, user2, and user3. The 'merchant' and 'user1' entries are highlighted with red boxes.

Select	User Id
<input type="checkbox"/>	api-gateway-webseald/isam.authsaz.com
<input type="checkbox"/>	dashboard-webseald/isam.authsaz.com
<input type="checkbox"/>	ivmgrd/master
<input type="checkbox"/>	merchant
<input type="checkbox"/>	sec_master
<input type="checkbox"/>	user1
<input type="checkbox"/>	user2
<input type="checkbox"/>	user3

5.4.2 /scim ACLs

The screenshot shows the Policy Administration interface with two panels: Task List and Browse Object Space.

Task List Panel:

- User
- Group
- Object Space** (highlighted with a red box and numbered 1)
- Browse Object Space (highlighted with a red box and numbered 1)
- Copy/Paste Object Space
- Create Object
- Import Object
- Create Object Space
- ACL
- POP
- AuthzRule
- GSO Resource
- Secure Domain

Browse Object Space Panel:

Path	ACL	POP	AuthzRule
/	default-root		
+ Management	default-management		
+ WebSEAL	default-webseal		
+ isam.authsaz.com-api-gateway			
+ isam.authsaz.com-dashboard			
+ favicon.ico	favicon		
+ icons			
+ index.html			
+ mga	isam_mobile_nobody		
+ pics			
+ portal			
+ scim			

Red boxes and numbers indicate the navigation steps:

- Object Space (Task List)
- WebSEAL (Browse Object Space)
- isam.authsaz.com-dashboard (Browse Object Space)
- scim (Browse Object Space)

Expand the **Object Space** from the **Task List** panel and click on the **Browse Object Space** link.

Expand **WebSEAL** from the **Browse Object Space** in the right-hand panel.

Select **isam.authsaz.com-dashboard** from the list and expand it to find **scim**.

Click on **scim**.

Protected Object Properties

[General](#) [Extended Attributes](#)

Object Name
/WebSEAL/isam.authsaz.com-dashboard/scim

Description
Object from host isam.authsaz.com.

Can Policy be attached to this object

ACL Attached
 [Detach](#)

POP Attached
 [Attach...](#)

AuthzRule Attached
 [Attach...](#)

[Create Child Object...](#)

[Apply](#) [Delete](#) [Export](#) [Cancel](#)

Click **Attach** button next to *ACL Attached* textbox to show the list of defined ACLs. Select **isam_mobile_rest** ACL.

Click **Apply**.

Repeat this step to attach **isam_mobile_rest** to **scim** object in **isam.authsaz.com-api-gateway**.

Browse Object Space				
	Path	ACL	POP	AuthzRule
[+]	/	default-root		
[+]	Management	default-management		
[+]	WebSEAL	default-webseal		
[+]	isam.authsaz.com-api-gateway			
[+]	api			
[+]	favicon.ico	favicon	favicon	
[+]	icons			
[+]	index.html			
[+]	mga	isam_mobile_nobody		
[+]	pics			
[+]	scim	isam_mobile_rest		
[+]	isam.autnsaz.com-dashboard			
[+]	favicon.ico	favicon	favicon	
[+]	icons			
[+]	index.html			
[+]	mga	isam_mobile_nobody		
[+]	pics			
[+]	portal			
[+]	scim	isam_mobile_rest		

After doing these steps **isam_mobile_rest** ACL will be attached to **scim** for both **api-gateway** and **dashboard** reverse proxy.

5.4.3 /mga ACLs

The screenshot shows the IBM Security Access Manager Policy Administration interface. On the left, the Task List sidebar is open, showing various management options. The 'ACL' section is expanded, and the 'Search ACLs' option is highlighted with a red box and the number '1'. In the main pane, the 'Search ACLs' page is displayed. It features a search bar with fields for 'ACL Name' (containing an asterisk) and 'Maximum Results' (set to 100), with a 'Search' button highlighted with a red box and the number '2'. Below the search bar, a message states '15 ACLs matched the search criteria'. A table lists the matching ACL names, each preceded by a checkbox. The 'isam_mobile_rest' entry is highlighted with a red box and the number '3'. The table includes buttons for 'Create...', 'Delete', 'Export', 'Options', and 'Filters' at the top.

Select	ACL Name
<input type="checkbox"/>	default-config
<input type="checkbox"/>	default-domain
<input type="checkbox"/>	default-gso
<input type="checkbox"/>	default-management
<input type="checkbox"/>	default-management-proxy
<input type="checkbox"/>	default-policy
<input type="checkbox"/>	default-replica
<input type="checkbox"/>	default-root
<input type="checkbox"/>	default-webseal
<input type="checkbox"/>	favicon
<input type="checkbox"/>	isam_mobile_anyauth
<input type="checkbox"/>	isam_mobile_nobody
<input type="checkbox"/>	isam_mobile_rest
<input type="checkbox"/>	isam_mobile_st_unauth
<input type="checkbox"/>	isam_mobile_unauth

Expand the *ACL* menu from the *Task List*.

Click *Search ACLs*.

Click on **isam_mobile_rest** ACL.

The screenshot shows the IBM Security Access Manager interface. In the top navigation bar, there are links for Home, Appliance Dashboard, Monitor, Secure Web Settings, Secure Access Control, Secure Federation, and Connect IBM Cloud Identity. Below the navigation bar, the title "IBM Security Access Manager" is displayed, followed by "Policy Administration". On the left, a "Task List" sidebar lists various administrative tasks: User, Group, Object Space, ACL (with sub-options: Search ACLs, Create ACL, Import ACL, Export All ACLs), POP, AuthzRule, and GSO Resource. The main panel is titled "ACL Properties" and contains tabs for General, Attach, and Extended Attributes. The General tab is selected. It includes fields for "ACL Name" (set to "isam_mobile_rest") and "Description" (set to "ISAM autoconfiguration tool ACL"). A "Set" button is also present. Below these fields is a section labeled "ACL Entries".

In ACL Properties panel navigate to **Attach**.

This screenshot is similar to the previous one, showing the "ACL Properties" panel with the "Attach" tab selected. The "General" tab is highlighted with a red box. The "Attach" tab is also highlighted with a red box. The "Extended Attributes" tab is visible but not selected. The "ACL Name" field is set to "isam_mobile_rest". Below it, a message states "The ACL is attached to these objects". Underneath this message, there are two buttons: "Attach" and "Detach", with "Attach" being highlighted with a red box. There are also "Select" and "Projected Object" buttons. A list of attached objects is shown, each with a checkbox and a link:

- /WebSEAL/isam-test-dashboard/mga/sps/mga/user/mgmt/otp
- /WebSEAL/isam-test-dashboard/mga/sps/mga/user/mgmt/device
- /WebSEAL/isam-test-dashboard/mga/sps/mga/user/mgmt/questions
- /WebSEAL/isam-test-dashboard/mga/sps/mga/user/mgmt/grant

Click **Attach**.

The screenshot shows the IBM Security Access Manager interface. In the top navigation bar, there are several tabs: Home (Appliance Dashboard), Monitor (Analysis and Diagnostics), Secure Web Settings, Secure Access Control, Secure Federation, and Connect (IBM Cloud Identity). The main content area is titled "Policy Administration". On the left, a "Task List" sidebar includes options like User, Group, Object Space, and ACL (with sub-options: Search ACLs, Create ACL, Import ACL, Export All ACLs, List Action Groups, Create Action Group, and a "..." button). The right panel is titled "Attach ACL" and contains fields for "ACL Name" (set to "isam_mobile_rest") and "Protected Object Path" (set to "/WebSEAL/isam-test-dashboard/mga/sps/mga/user/mgmt/clients"). At the bottom of this panel are "Attach" and "Cancel" buttons.

Enter **/WebSEAL/isam.authsaz.com-dashboard/mga/sps/mga/user/mgmt/clients** as *Protected Object Path*.

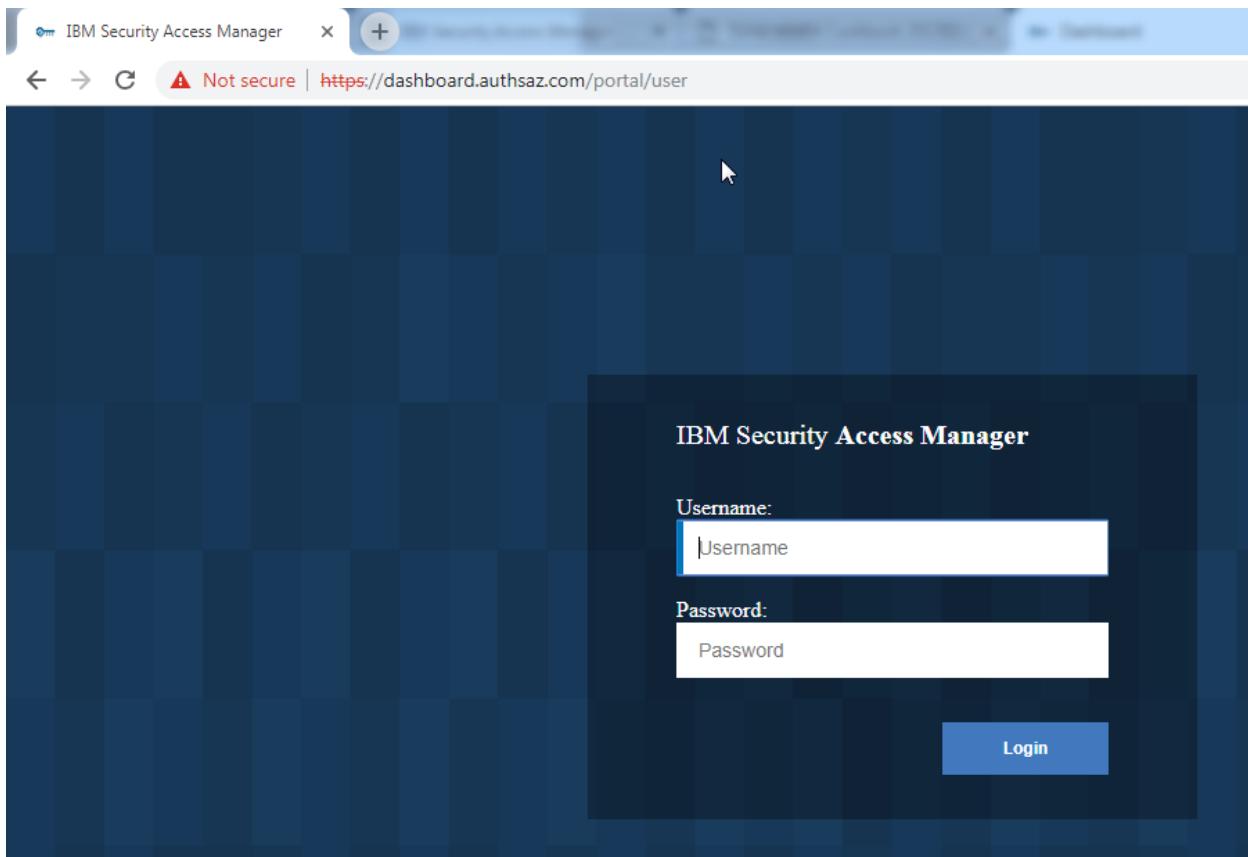
Click **Attach**.

Repeat the same steps to add following ACLs:

ACL	Protected Object Path
isam_mobile_rest	/WebSEAL/isam.authsaz.com-dashboard/mga/sps/mmfa/user/mgmt/authenticators /WebSEAL/isam.authsaz.com-dashboard/mga/sps/mmfa/user/mgmt/auth_methods /WebSEAL/isam.authsaz.com-dashboard/mga/sps/mmfa/user/mgmt/qr_code /WebSEAL/isam.authsaz.com-dashboard/mga/sps/mmfa/user/mgmt/transactions /WebSEAL/isam.authsaz.com-dashboard/mga/sps/oauth/oauth20/register
isam_mobile_rest_unauth	/WebSEAL/isam.authsaz.com-api-gateway/mga/websock/mmfa-wss
isam_mobile_unauth	/WebSEAL/isam.authsaz.com-api-gateway/mga/sps/mmfa/user/mgmt/details

5.5 Create OAuth Client for Merchant

In this section, we will create a dynamic OAuth client for our Merchant application and reconfigure Merchant application to use this newly created client for connecting to the OAuth endpoints and calling APIs.



Log in to <https://dashboard.authsaz.com/portal/user> with the **merchant** as *Username* and **PasswOrd** as *Password*.

At this time there should be no error and you should see the dashboard page as below.

The screenshot shows the Authsaz dashboard interface. At the top, there's a navigation bar with links for Dashboard, User, Apps, ISAM Devices, ISAM Scim, and Logout. Below the navigation bar are two main sections:

- Authorization Grant**: A table with columns Id, IsEnabled, ClientId, and Remove. It displays a message: "No matching records found". A yellow "Refresh" button is at the bottom left.
- Authenticators**: A table with columns Id, Device Name, Device Type, OS Version, oAuth Grant, Enabled, and Remove. It displays a message: "No matching records found". It includes a green "New" button and a yellow "Refresh" button.

Navigate to **Apps** menu and register a new app:

The screenshot shows the "Register New App" form. It has fields for App Name and Redirect URL. The App Name field contains "merchantApp". The Redirect URL field contains "http://merchant.authsaz.com:5000/callback". Below the fields, there's a note: "Use [ENTER] for multiple URLs." and a blue "Submit" button with a hand cursor icon.

Provide **merchantApp** as *App Name* and **http://merchant.authsaz.com:5000/callback** as *Redirect URL*.

Click **Submit**.

Current Apps				
Client Name	Client Id	Definition Id	Definition Name	Remove
merchantApp	jSNhWHOviWPzcqlB3u81	2	AppsRegister	<button>Remove</button>
	<button>Refresh</button>			

A row will be inserted into the **Current Apps** table. Click on the link in **Client Id** column.

App Detail

client_secret_expires_at: 0

owner_username: merchant

registration_client_uri: [https://isam.authsaz.com/mga/sps/oauth/oauth20/register/AppsRegister?
client_id=jSNhWHOviWPzcqlB3u81](https://isam.authsaz.com/mga/sps/oauth/oauth20/register/AppsRegister?client_id=jSNhWHOviWPzcqlB3u81)

client_secret: 8jNcmGcjpyTN0HmL1aG8

client_id_issued_at: 1543759312

redirect_uris: http://merchant.authsaz.com:5000/callback

client_name: merchantApp

client_id: jSNhWHOviWPzcqlB3u81

Close

A popup window will be opened and the required information for a client will be shown.

Copy **client_id** and **client_secret** values.

SSH to Merchant Virtual machine and stop **server.py** script.

Go to **isambookdemo-merchant** folder and find the **server.py** file.

Edit **server.py** with your favorite editor and replace **CLIENT_ID** and **SECRET_KEY** values by copied values from the previous step respectively.

```

GNU nano 2.2.6                               File: server.py

from flask import Flask
from flask import session, redirect, url_for, render_template, Response, request
import json
import requests
from urllib.parse import urlencode, quote_plus

CLIENT_ID = "CHANGE ME"
SECRET_KEY = "CHANGE ME"
FLASK_SESSION_KEY = "CHANGE ME"
MY_ADDRESS = "http://merchant.authsaz.com:5000"
REDIRECT_URL = MY_ADDRESS + "/callback"
API_CALLBACK_URL = MY_ADDRESS + "/apicallback"

SERVER_NAME_API =
# More information is available at:
#   https://www.ibm.com/support/knowledgecenter/SSPREK_9.0.5/com.ibm.isam.doc/config/concept/OAuthEndpoints.$
SERVER_NAME_OAUTH_AUTHORIZE = "https://api-gateway.authsaz.com/mga/sps/oauth/oauth20/authorize"
SERVER_NAME_OAUTH_TOKEN = "https://api-gateway.authsaz.com/mga/sps/oauth/oauth20/token"
SERVER_NAME_OAUTH_INTROSPECT = "https://api-gateway.authsaz.com/mga/sps/oauth/oauth20/introspect"
SERVER_NAME_OAUTH_USERINFO = "https://api-gateway.authsaz.com/mga/sps/oauth/oauth20/userinfo"
SERVER_NAME_OAUTH_REVOC = "https://api-gateway.authsaz.com/mga/sps/oauth/oauth20/revoke"
SERVER_NAME_OAUTH_LOGOUT = "https://api-gateway.authsaz.com/mga/sps/oauth/oauth20/logout"

# SERVER_NAME_MGT = SERVER_NAME_API + "/mgt"

```

Save changes and run the following command to start the server again on port 5000:

```
python3 server.py
```

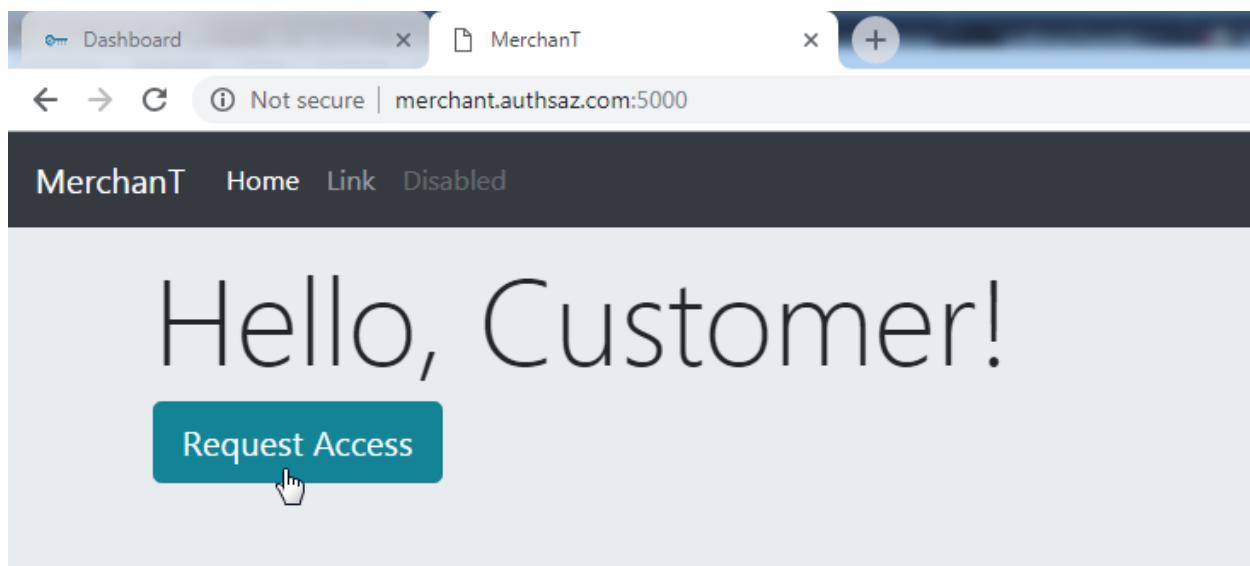
```

user@authsaz:~/isambookdemo-merchant
user@authsaz:~/isambookdemo-merchant$ nano server.py
user@authsaz:~/isambookdemo-merchant$ python3 server.py
 * Serving Flask app "server" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat

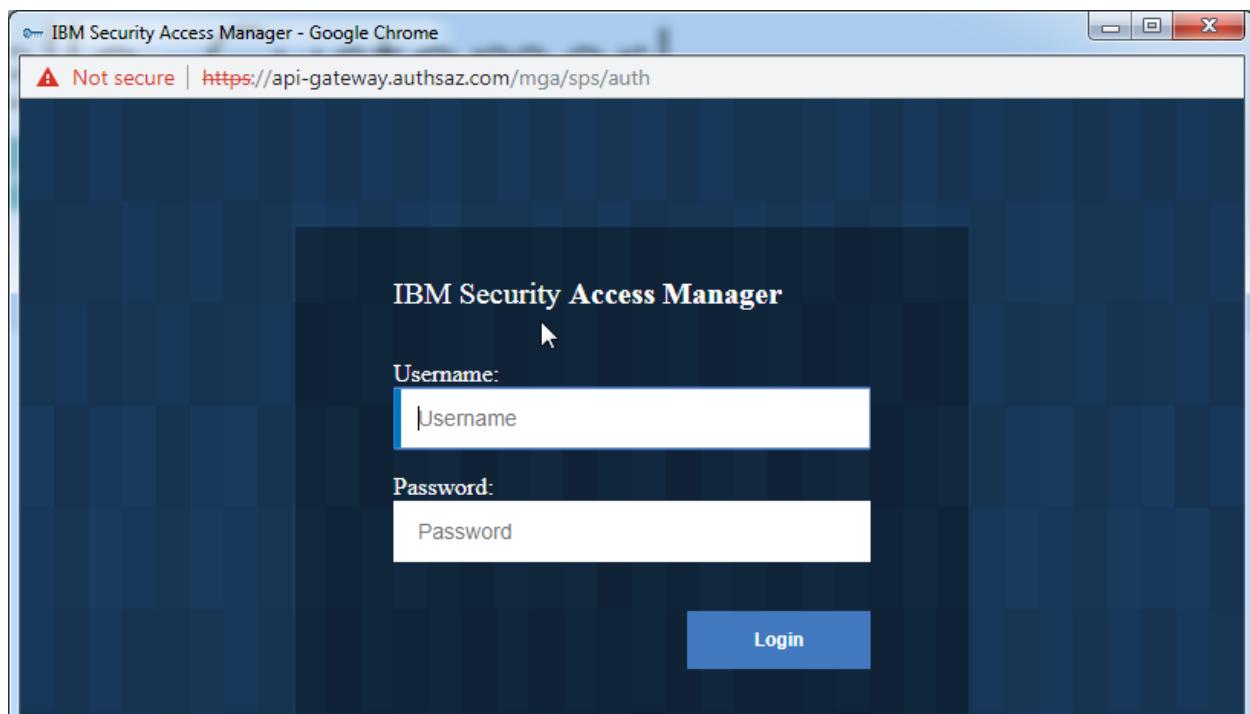
```

5.5.1 Test Merchant OAuth implementation

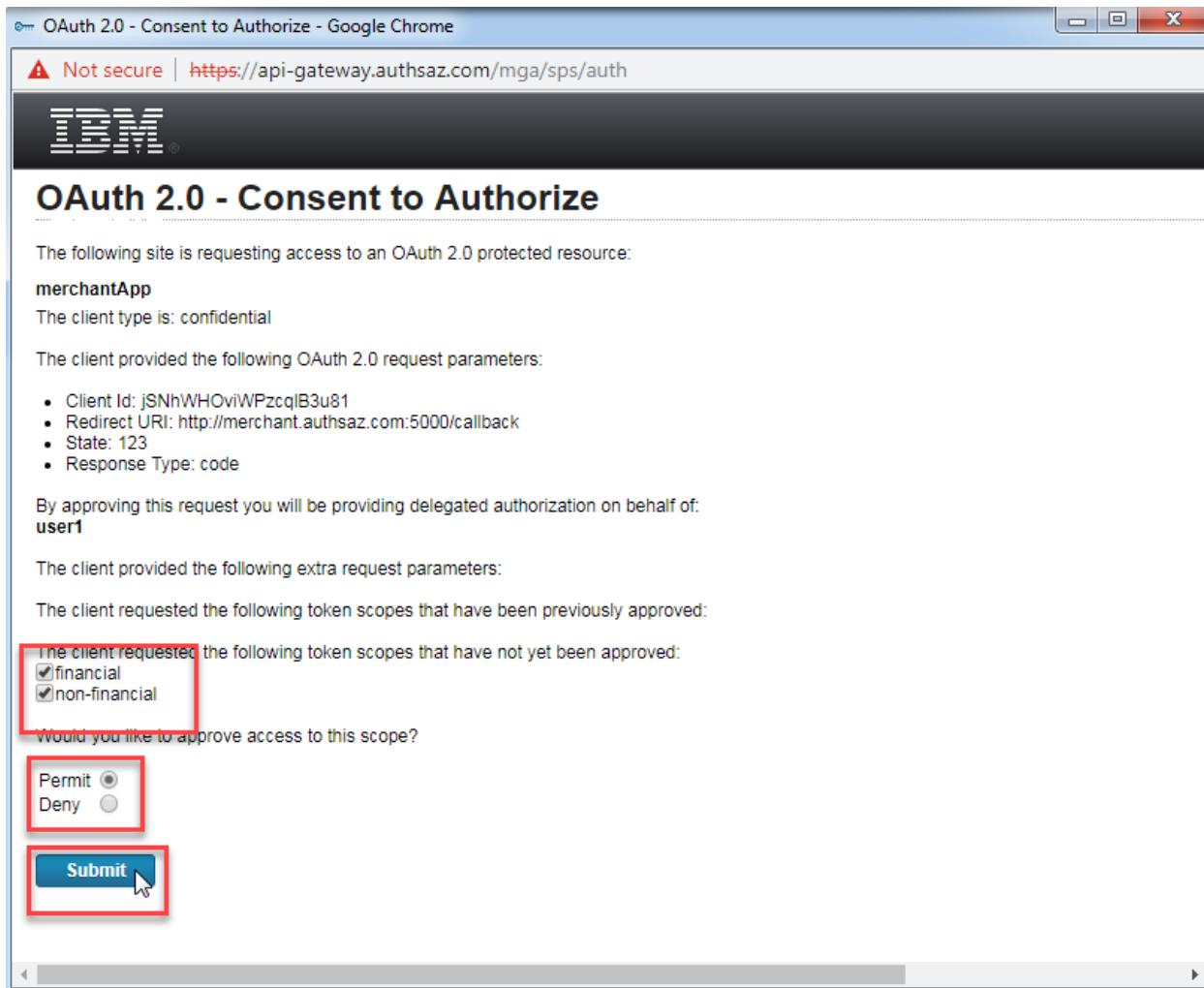
Open a browser and go to <http://merchant.authsaz.com:5000>. You should see the Merchant site as below:



Click **Request Access**.



Enter **user1** as *Username* and **PasswOrd** as *Password*.



You can see the user consent page. Select checkboxes for **financial** and **non-financial**.

Select the radio button for **Permit**.

Click **Submit**.

The screenshot shows a web browser window with three tabs: 'Dashboard', 'MerchanT', and 'Merchant authsaz.com:5000/#'. The 'MerchanT' tab is active, displaying the 'Merchant' home page. The page features a large 'Hello, Customer!' header. Below it, a red box highlights a section containing access token details:

- Access token: Dtx5aq67Hhnt6nyGKQWQb
- Refresh token: L288c61q6daqnUI2heHvTw0ZtqgFMXF85dM2lfe
- Expires in: 3599
- Token type: bearer
- Scope: financial non-financial

Below these details are three buttons: 'Revoke Access And Logout' (red), 'Refresh Token' (yellow), and 'Query Info' (yellow). A large blue button labeled 'Request Access' is also present. Further down, another red box highlights the 'Accounts', 'Balance', and 'Transfer' sections:

- Accounts:** Includes a 'User name' input field and a 'View Accounts' button.
- Balance:** Includes an 'Account Id' input field and a 'View Balance' button.
- Transfer:** Includes 'From Account Id' and 'To Account Id' input fields, an 'Amount' input field, and a 'Transfer Money' button.

After a successful login and accepting the consent, you should see **Access token** information on top of merchant Home page.

Accounts API, Balance API, and Transfer API will be shown on this page.

A modal window is displayed, containing the following content:

Buttons at the top: 'Revoke Access And Logout' (red), 'Refresh Token' (yellow), and 'Query Info' (yellow, highlighted with a red box).

JSON token response:

```
{"scope":"financial non-financial","active":true,"token_type":"bearer","exp":1543833131,"iat":1543829531,"client_id":"jSNhWHOviWPzcqlB3u81","username":"user1"}
```

Large blue 'Request Access' button at the bottom.

Click **Query Info** to show more information about the access token that has been given in the previous step.

6 Simple API Protection using OAuth-EAS

In this chapter, we use OAuth-EAS feature in ISAM to protect APIs. The following section describes a scenario that we use to protect APIs using OAuth-EAS.

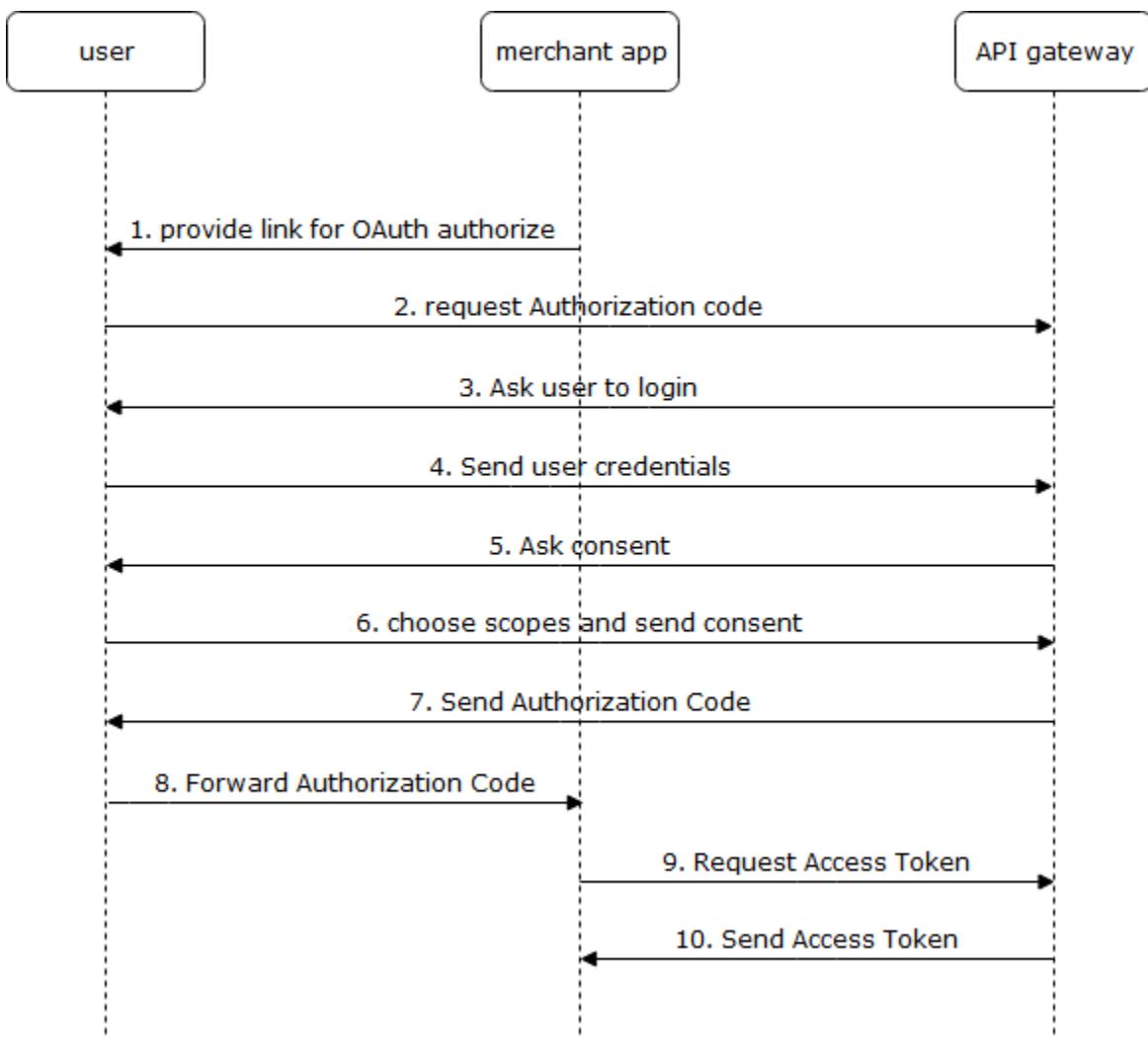
6.1 Scenario

In this scenario, user delegate a scoped access to Merchant using the OAuth 2.0 protocol. By receiving an access token, Merchant can call every APIs that belong to access token scope.

6.1.1 User access delegation

Delegating access to Merchant is based on OAuth 2.0 protocol and the following steps will be required for a successful access delegation:

1. The Merchant provides a link to OAuth authorize endpoint, including its own client_id, redirect_uri, scopes, and other parameters.
2. The user starts an OAuth authorization code flow by sending an OAuth authorization request to API gateway
3. API gateway redirects the user to the login page.
4. User Enter his Username and Password in the login page
5. Consent will be sent back to the user
6. User selects the scopes that he wants to delegate and sends a permit request to API gateway
7. API gateway creates an authorization code and sends it to the user. The user automatically will be redirected to Merchant Callback URL with a request containing authorization code.
8. By receiving the authorization code, Merchant request an access token from API gateway
9. API gateway sends back corresponding access token to the Merchant
10. Merchant saves the access token for this user

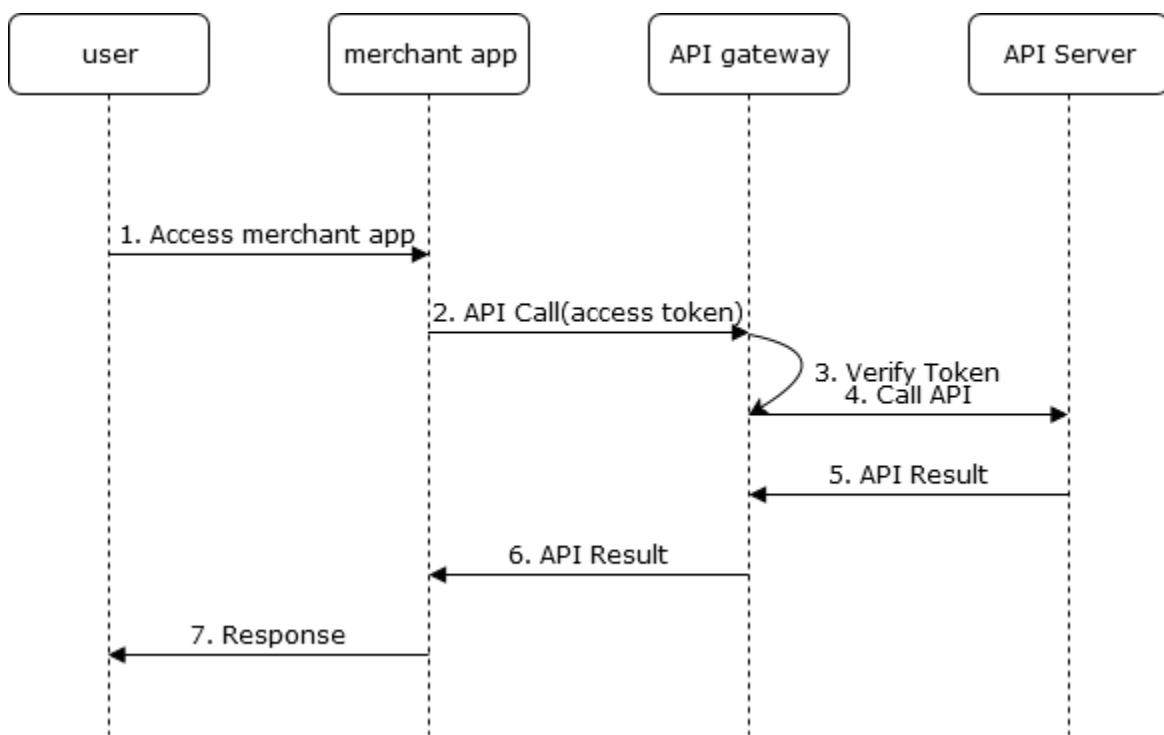


6.1.2

API Call

After a successful access delegation, Merchant can call APIs that are belong to access token scope. The following steps should be carried out in an API call:

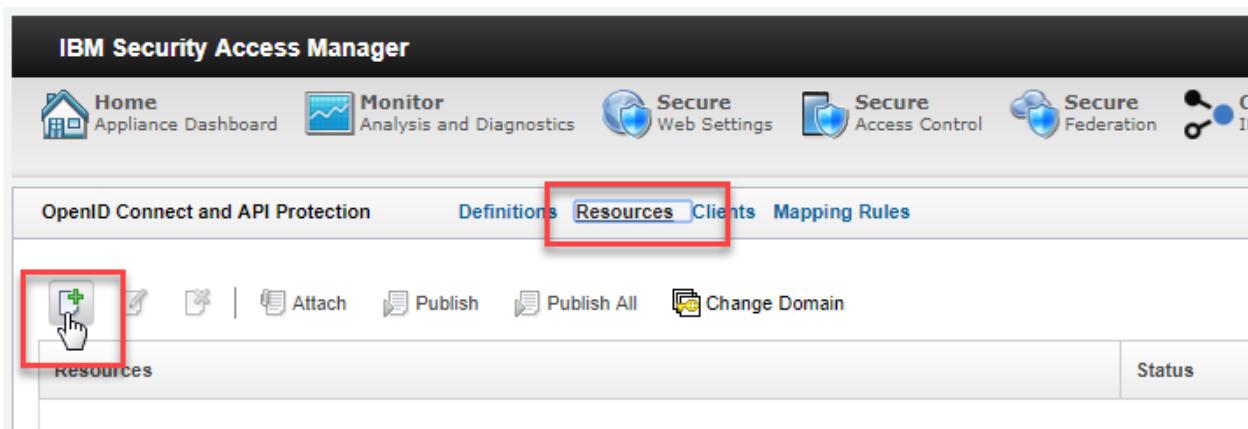
1. User access Merchant web application and do an action on it
2. Behind the action in Merchant application, there is an API call. Merchant application calls an API on behalf of User by sending its access token.
3. API gateway validates the access token sent by Merchant request
4. API gateway sends an API call to API server
5. API server returns the result for this call
6. API gateway forwards the result to the Merchant application
7. Merchant application creates an appropriate response based on the result of the API call



6.2 Attach API Protection Policy to APIs

The screenshot shows the IBM Security Access Manager interface. The top navigation bar includes Home, Monitor, Secure Web Settings, **Secure Access Control** (which is highlighted with a red box), and Secure Federation. The main content area has three columns: Policy, Manage, and Global Settings. The Policy column lists: Access Control, Authentication, Risk Profiles, Attributes, Obligations, **OpenID Connect and API Protection** (which is highlighted with a red box), and Extensions. The Manage column lists: Devices, Grants, Database Maintenance, SCIM Configuration, Push Notification Providers, MMFA Configuration, Attribute Source. The Global Settings column lists: Advanced Configuration, User Registry, Runtime Parameters, Template Files, Mapping Rules, Distributed Session Cache, Server Connections, Point of Contact, and Access Policies. At the bottom, there are two 'In' buttons with version numbers 1.1 and 1.2.

Navigate to **Secure Access Control > Policy: OpenID Connect and API Protection**.



Navigate to the **Resources** menu, and click **Add**.

The dialog box is titled 'Policy Server Login'. It contains three input fields: 'Administrator Username' with the value 'sec_master', 'Password' with the value 'Passw0rd', and 'Secure Domain' which is empty. At the bottom are 'Save' and 'Cancel' buttons.

* Administrator Username:	sec_master
* Password:
Secure Domain:	

Save Cancel

If this is the first time you visit this page, a *Policy Server Login* prompt will appear. Enter **sec_master** as *Administrator Username* and **Passw0rd** as *Password*. Click **Save**.

Add Resource

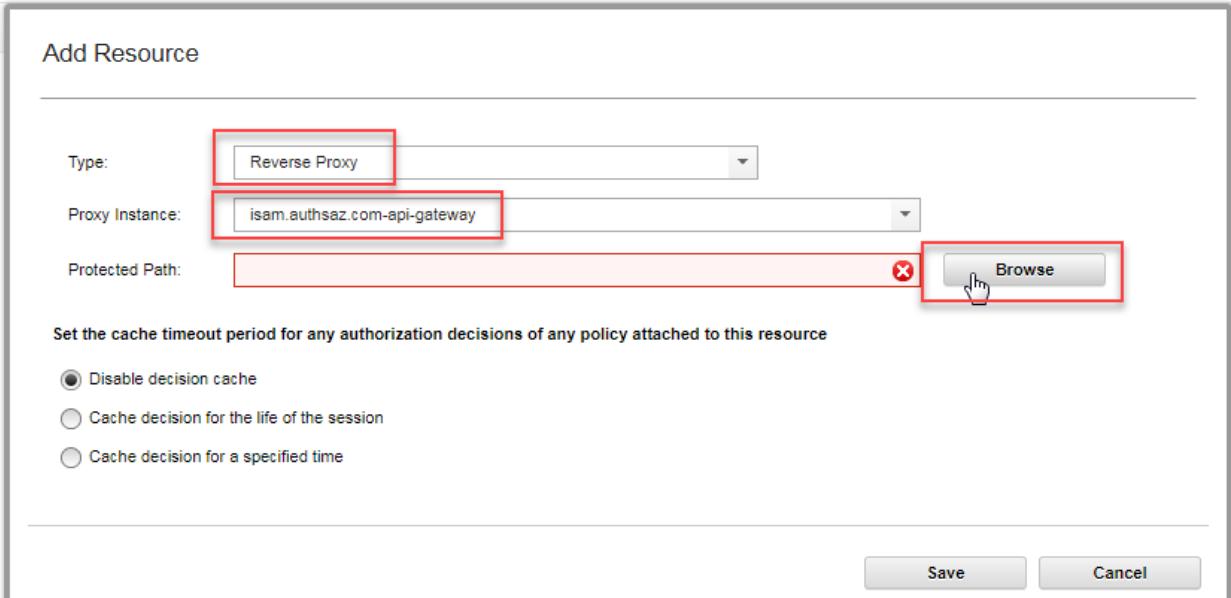
Type: **Reverse Proxy**

Proxy Instance: **isam.authsaz.com-api-gateway**

Protected Path: **/api**

Set the cache timeout period for any authorization decisions of any policy attached to this resource

Disable decision cache
 Cache decision for the life of the session
 Cache decision for a specified time



Select **Reverse Proxy** from *type* drop-down list.

Select **isam.authsaz.com-api-gateway** from *Proxy Instance* drop-down list.

Add Resource

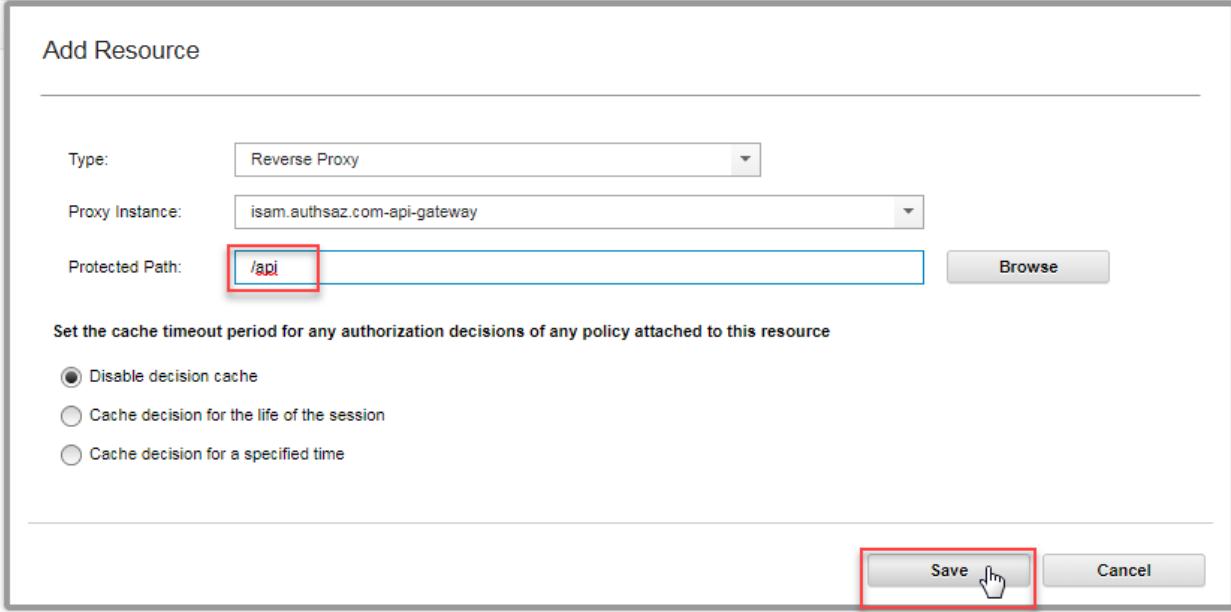
Type: **Reverse Proxy**

Proxy Instance: **isam.authsaz.com-api-gateway**

Protected Path: **/api**

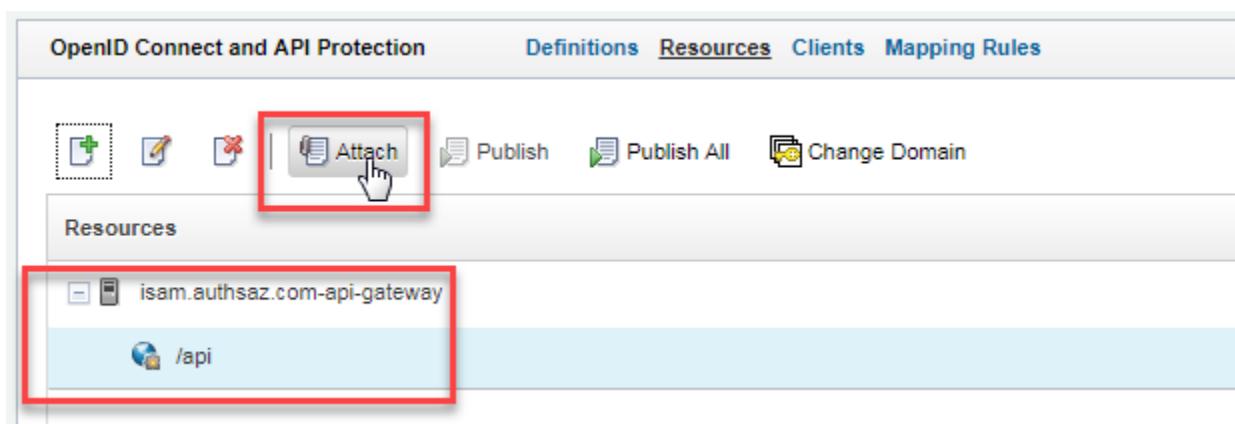
Set the cache timeout period for any authorization decisions of any policy attached to this resource

Disable decision cache
 Cache decision for the life of the session
 Cache decision for a specified time

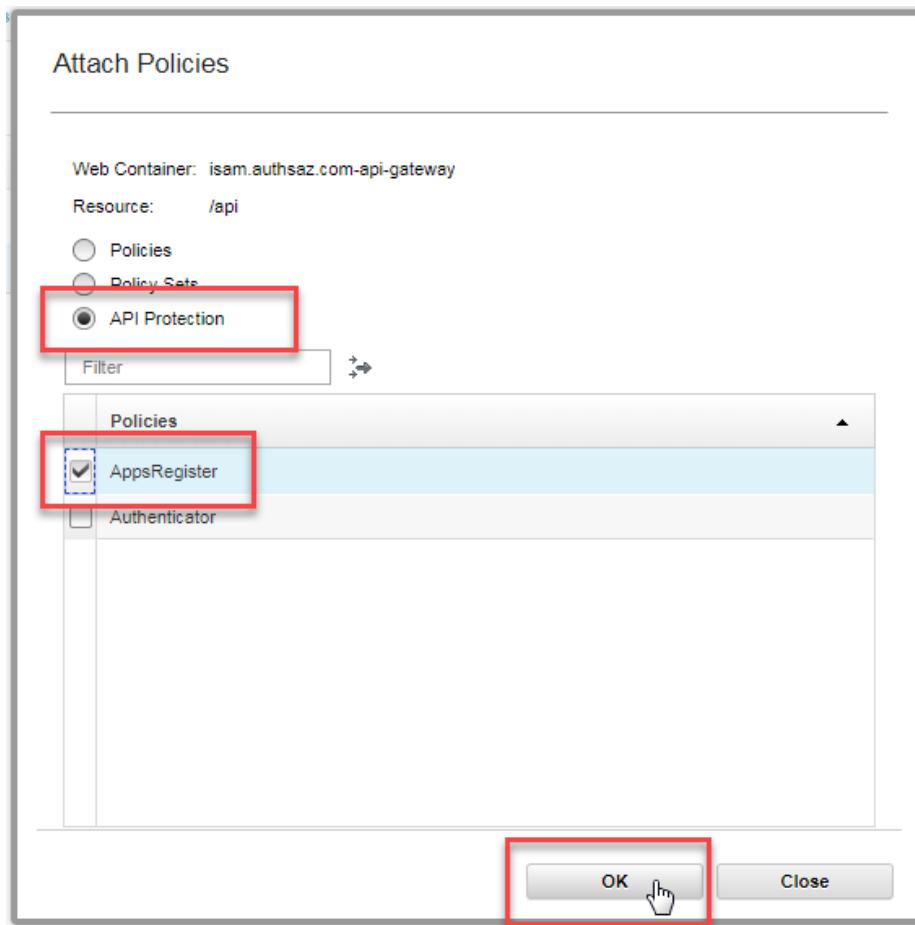


Enter **/api** as *Protected Path*.

Click **Save**.



Select the newly added resource and click on the **Attach** button.



Select radio button for **API Protection** and select checkbox for **AppsRegister**.

Click **Ok**.

The screenshot shows the 'Resources' tab in the IBM Security Access Manager interface. At the top, there are several icons: a green plus sign, a blue checkmark, a red minus sign, an 'Attach' icon, a 'Publish' icon, a 'Publish All' button (which is highlighted with a red box), and a 'Change Domain' icon. Below the toolbar, a table lists resources. One resource, '/api', is selected and has a status bar at the bottom right corner that says 'Publish required' with a warning icon.

Click **Publish All** button to publish newly attached policy.

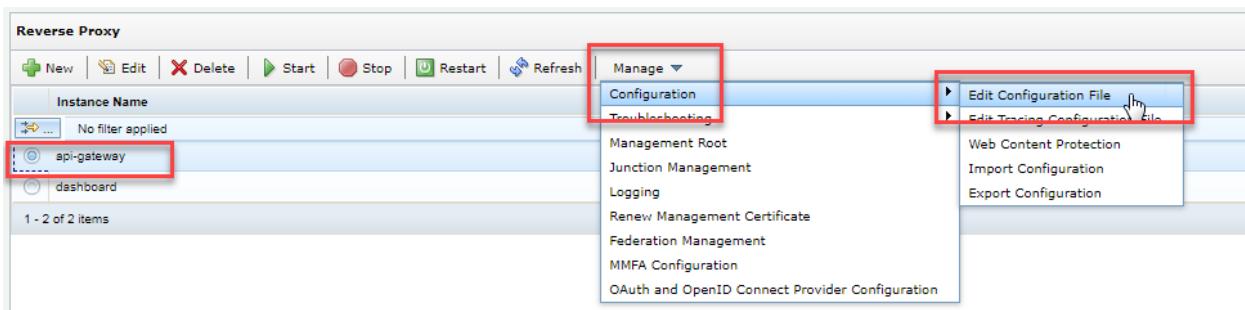


Click **Publish**.

6.3 Modify Reverse Proxy Instance Configuration File to enable OAuth-EAS

The screenshot shows the IBM Security Access Manager dashboard. At the top, there is a navigation bar with icons for Home, Monitor, Secure Web Settings (which is highlighted with a red box), and Secure Access Control. Below the navigation bar, there are three main sections: 'Manage' (with sub-options like Runtime Component, Reverse Proxy, Authorization Server, etc.), 'Global Settings' (with sub-options like URL Mapping, Junction Mapping, etc.), and 'Global Keys' (with sub-options like SSO Keys, LTPA Keys, etc.). A red box also highlights the 'Reverse Proxy' link under the 'Manage' section.

Navigate to **Secure Web Settings > Manage: Reverse Proxy**.



Select the radio button for the **api-gateway** Reverse Proxy instance. Click on **Manage** and select **Configuration > Edit Configuration File** from the pop-up menu.

Disable OAuth authentication:

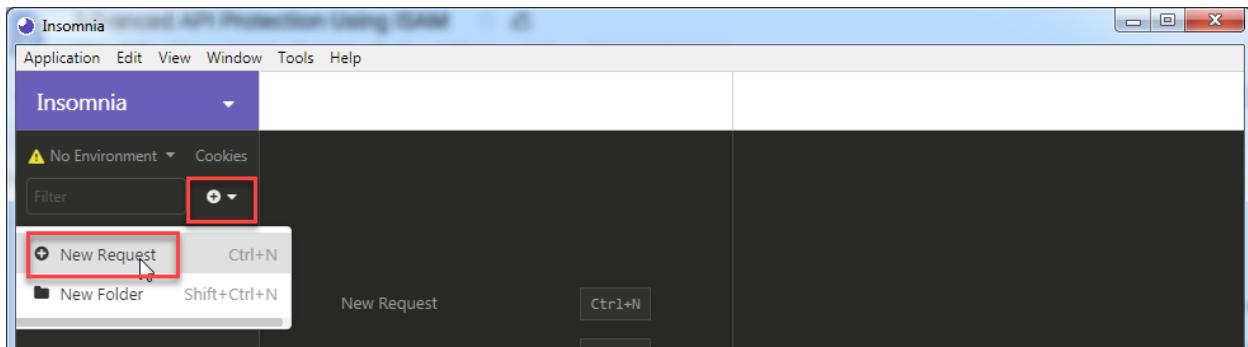
```
[oauth]
oauth-auth = none
```

Enable OAuth EAS:

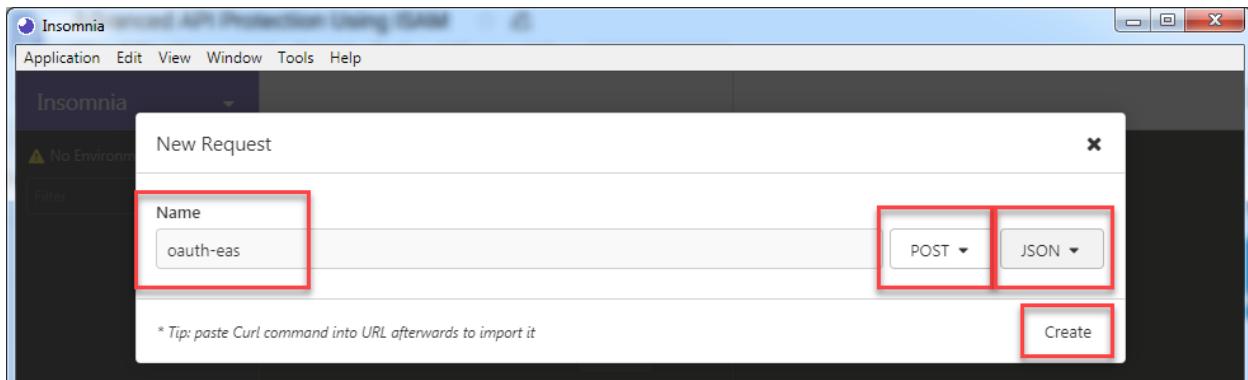
```
[oauth-eas]
eas-enabled = true
```

6.4 Test OAuth-EAS

Open your favorite request editor to check whether API Protection Policy is enforced or not. We used **Insomnia** as a request editor.



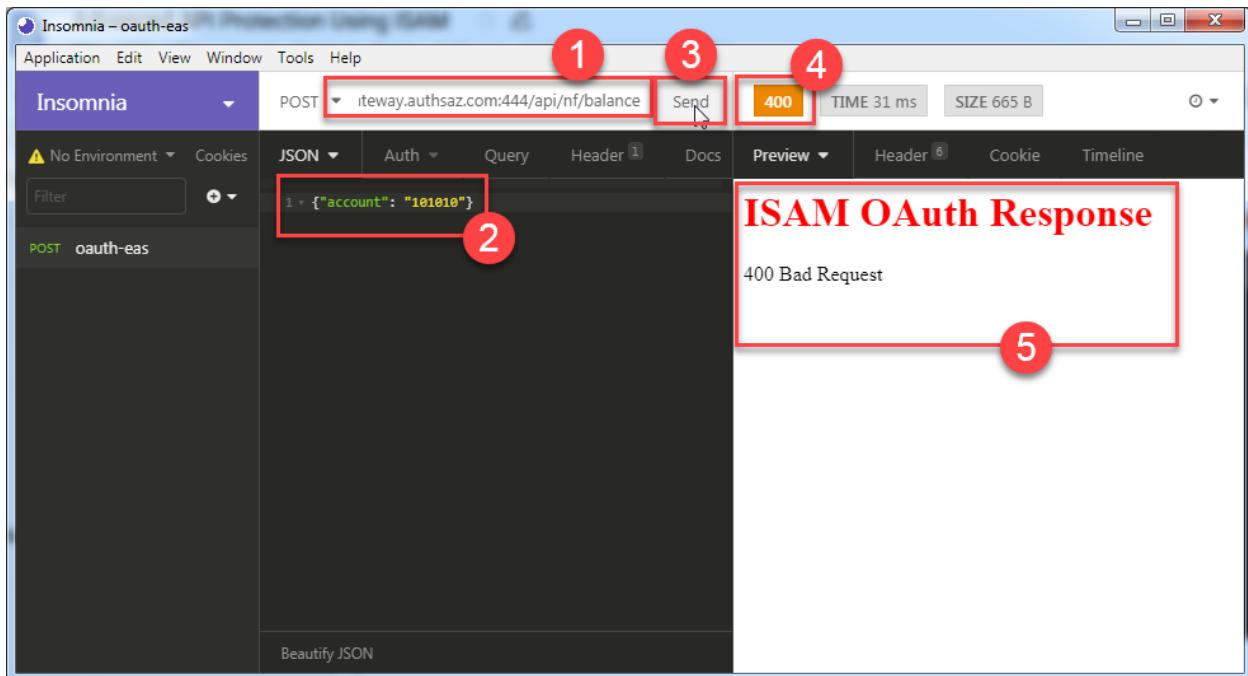
In the left panel, click **add** icon. Click **New Request** from the drop-down menu.



Enter **oauth-eas** as request Name.

Select **POST** from first drop-down list and **JSON** from the second drop-down list.

Click **Create**.

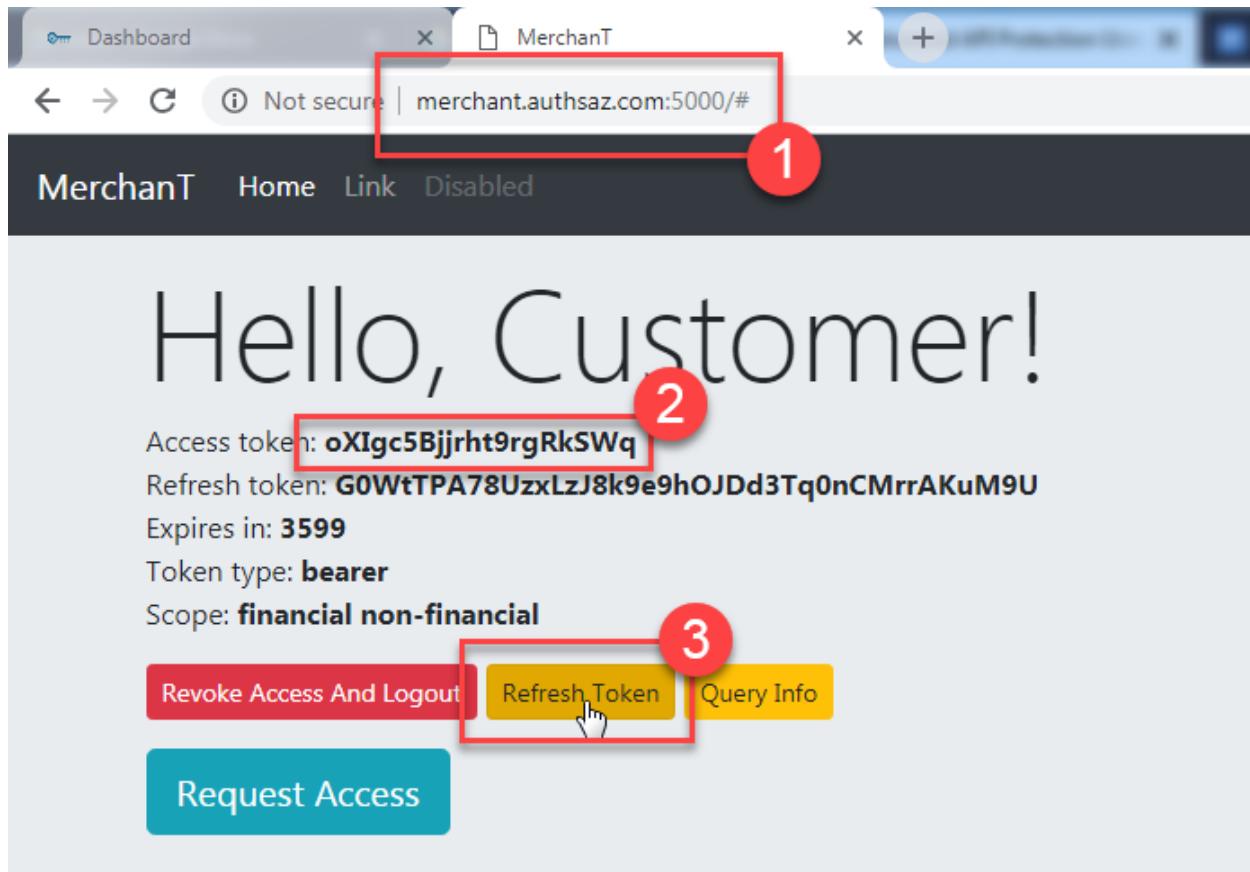


Enter <https://api-gateway.authsaz.com:444/api/nf/balance> in URL bar.

Provide `{"account": "101010"}` as the request body.

Click **Send**.

The response code should be **400 Bad Request** (It's because we haven't yet provided access token for this request⁹).



Open a browser.

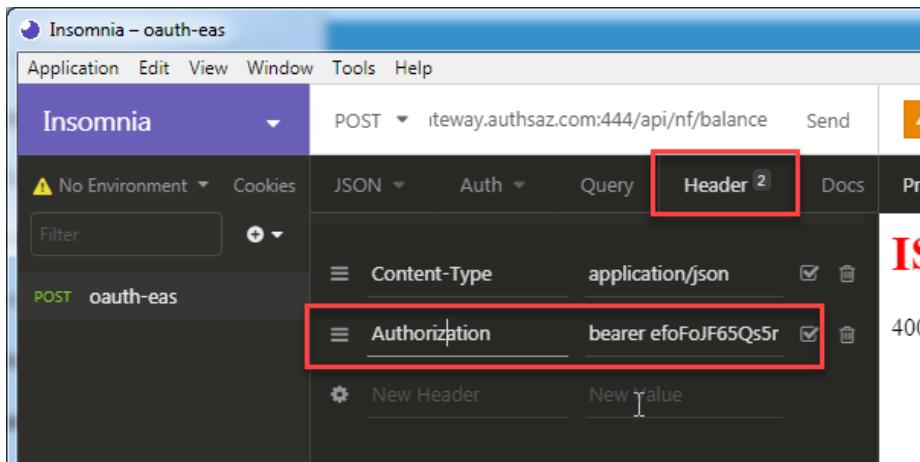
Go to <http://merchant.authsaz.com:5000/>.

If you have requested an access token before, you can just copy the **Access Token** value provided at the first line. Otherwise, request an access token by clicking **Request Access**.

In case you have an access token but you are not sure that this token is valid you can refresh the token by clicking **Refresh Token**.

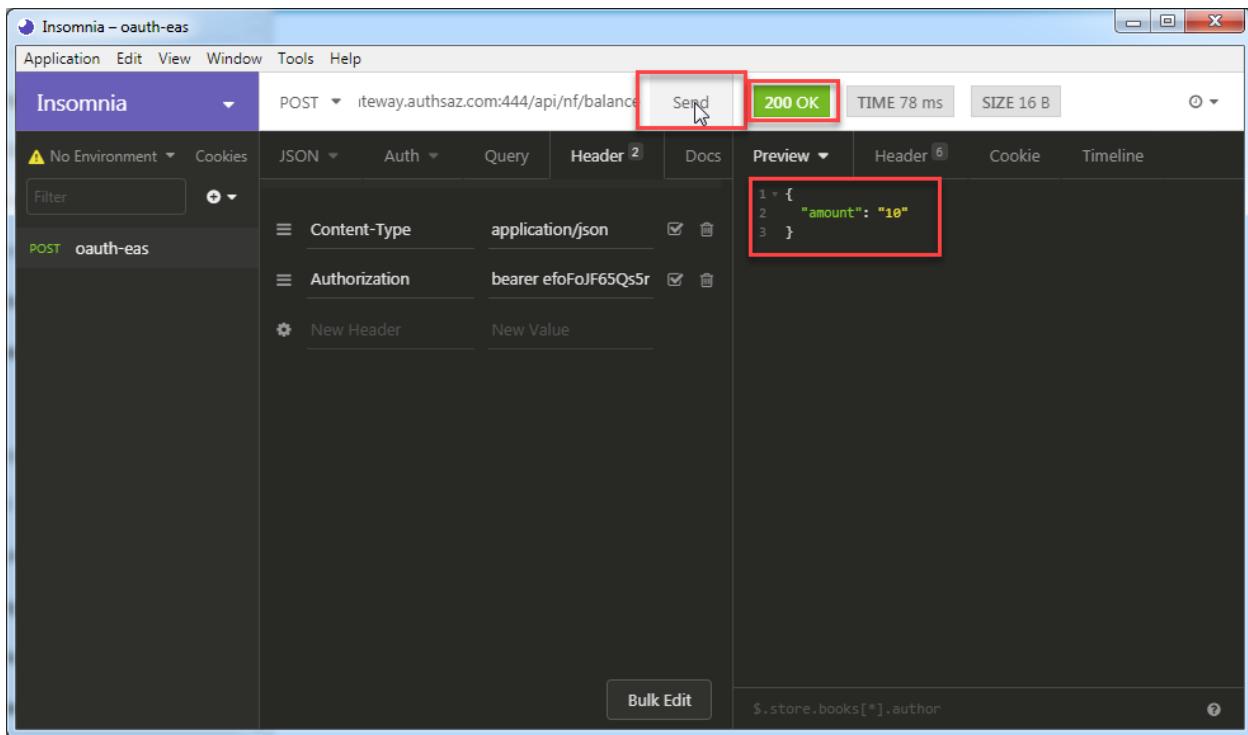
⁹

https://www.ibm.com/support/knowledgecenter/SSELE6_8.0.1.3/com.ibm.isam.doc/wrp_config/concept/con_oauth_eas_overvw.html



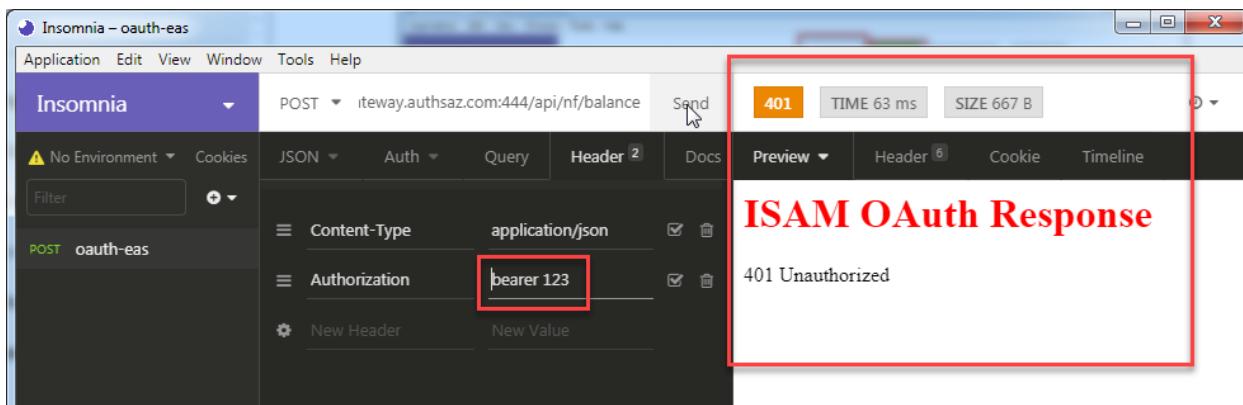
Go back to Insomnia and navigate to **Header** tab.

Create a new header with name **Authorization** and value **bearer efoFoJF65Qs5mgci4Lev** which the string in front of **bearer** is the access token that has been requested in the previous step.



Click **Send**.

If everything is ok you should receive a response that shows you the value of **amount** in json format.



If you provide Authorization header in your request but your token is not valid you should receive a **401 Unauthorized** response code.

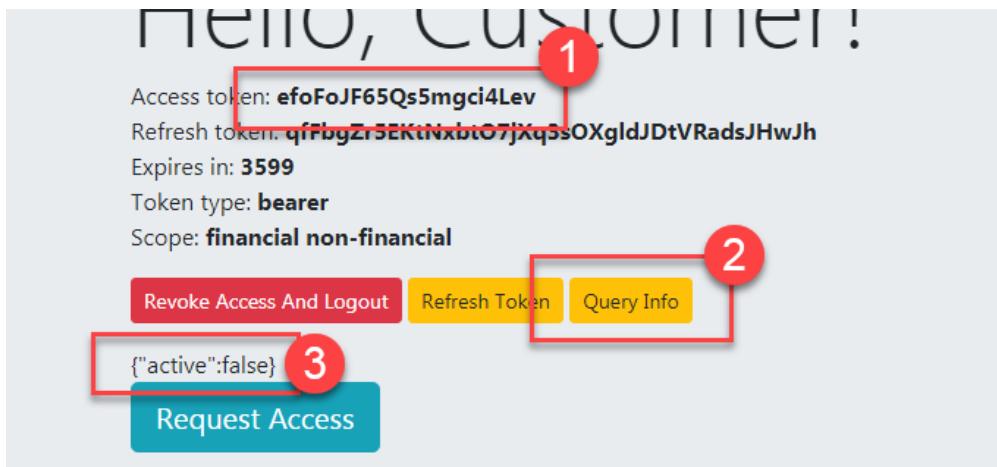
6.5 Test Merchant API Call

The screenshot shows a browser window titled "Merchant".
1. URL bar: `Not secure | merchant.authsaz.com:5000/#`
2. Access token value: `oXIgc5Bjjrht9rgRkSWq`
3. Refresh Token button highlighted with a red circle.

Open a browser. Go to <http://merchant.authsaz.com:5000/>.

If you have requested an access token before, you can see the value of **Access Token** at the first line. Otherwise, request an access token by clicking **Request Access**.

In case you have an access token but you are not sure that this token is valid you can refresh the token by clicking **Refresh Token**. To see which user access token created for click **Query Info**.

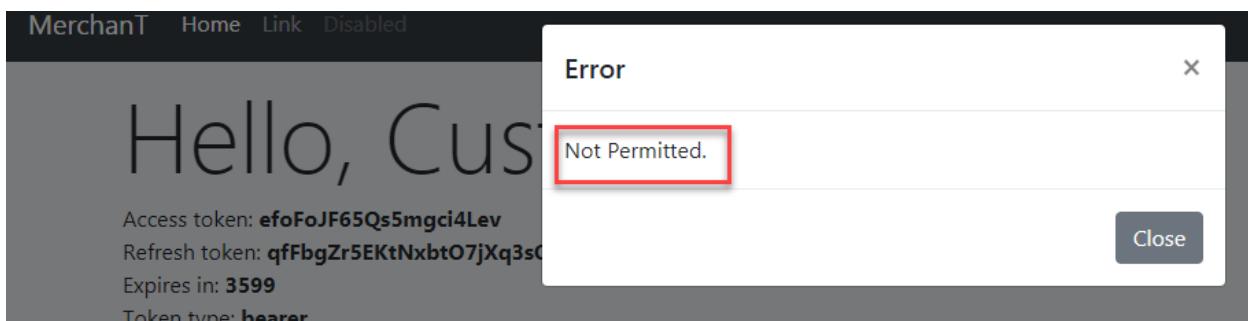


In case the **Access Token** is not valid the response from **Query Info** returns **{"Active":false}**.

The screenshot shows a user interface with the following elements:

- A modal dialog titled "Result" containing the JSON object **{"status":"1","account":["404040","505050"]}** (highlighted with a red box)
- Background information:
 - Access token: **0yWf2c**
 - Refresh token: **DCymL**
 - Expires in: **3599**
 - Token type: **bearer**
 - Scope: **financial non-financial**
- Action buttons: **Revoke Access And Logout**, **Refresh Token**, and **Query Info**
- A message box with the value **{"scope":"financial non-financial","active":true,"token_type":"bearer","exp":1543833131,"iat":1543829531,"client_id":"jSNhWHOviWPzcqJB3u8j","username":"user1"}** (highlighted with a red box)
- Below the modal:
 - Accounts**: User name **user3** (highlighted with a red box), **View Accounts** button
 - Balance**: Account Id input field, **View Balance** button
 - Transfer**: From Account Id input field, To Account Id input field

Enter **user1** as the *User name* in *Accounts* form. If everything is ok it will return **user1** account information as a JSON object.

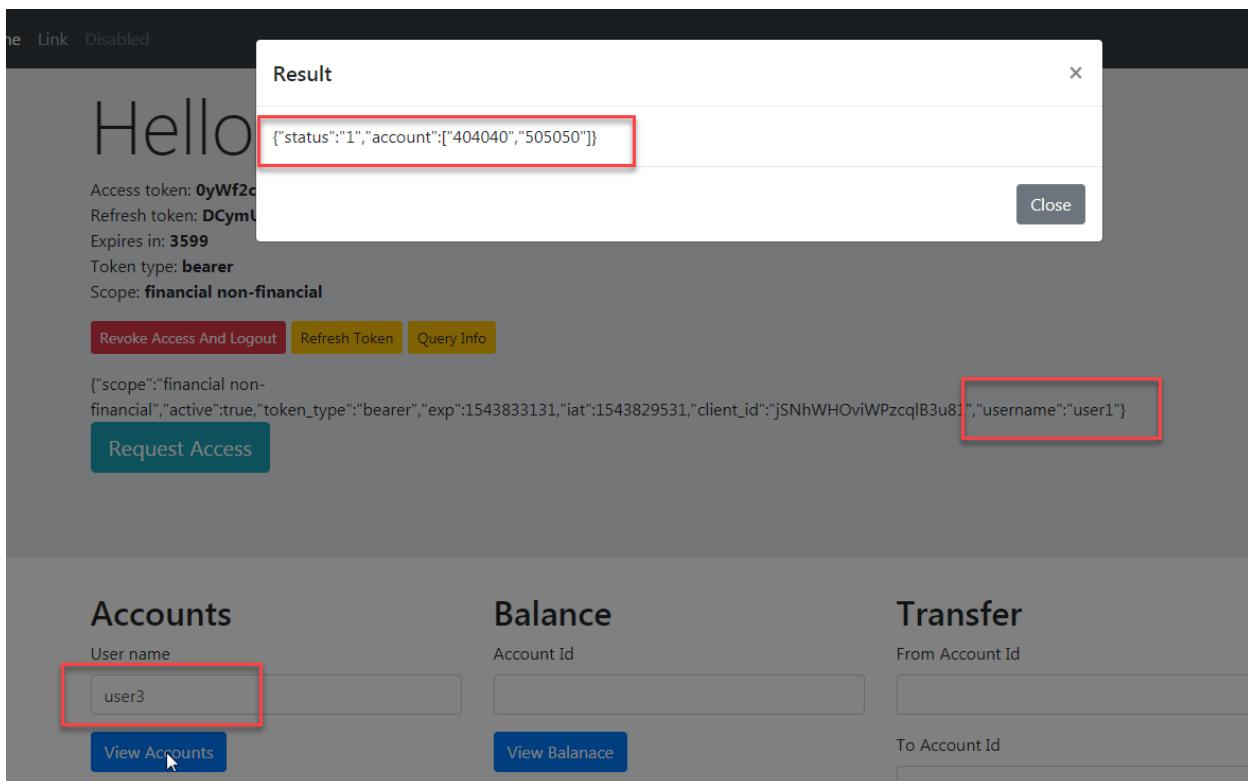


By calling *Accounts* API with an invalid **Access Token** the response would be “**Not Permitted**”.

6.6 Scenario drawbacks

The OAuth-EAS scenario has some limitations. In some cases calling API requires more security controls. As an example If you want to provide payment APIs to the merchants, in addition to OAuth Access Token, it is required that users verify every transaction via a second-factor authentication mechanism. By attaching an **API Protection** policy to a resource we can't attach any other access control policies.

The picture below shows that calling *Accounts* API with username **user3** while the **Access Token** created for **user3** successfully returns account information for **user1**.



The next chapter is about a scenario in which we can enforce more sophisticated access control policies along with OAuth authentication mechanism.

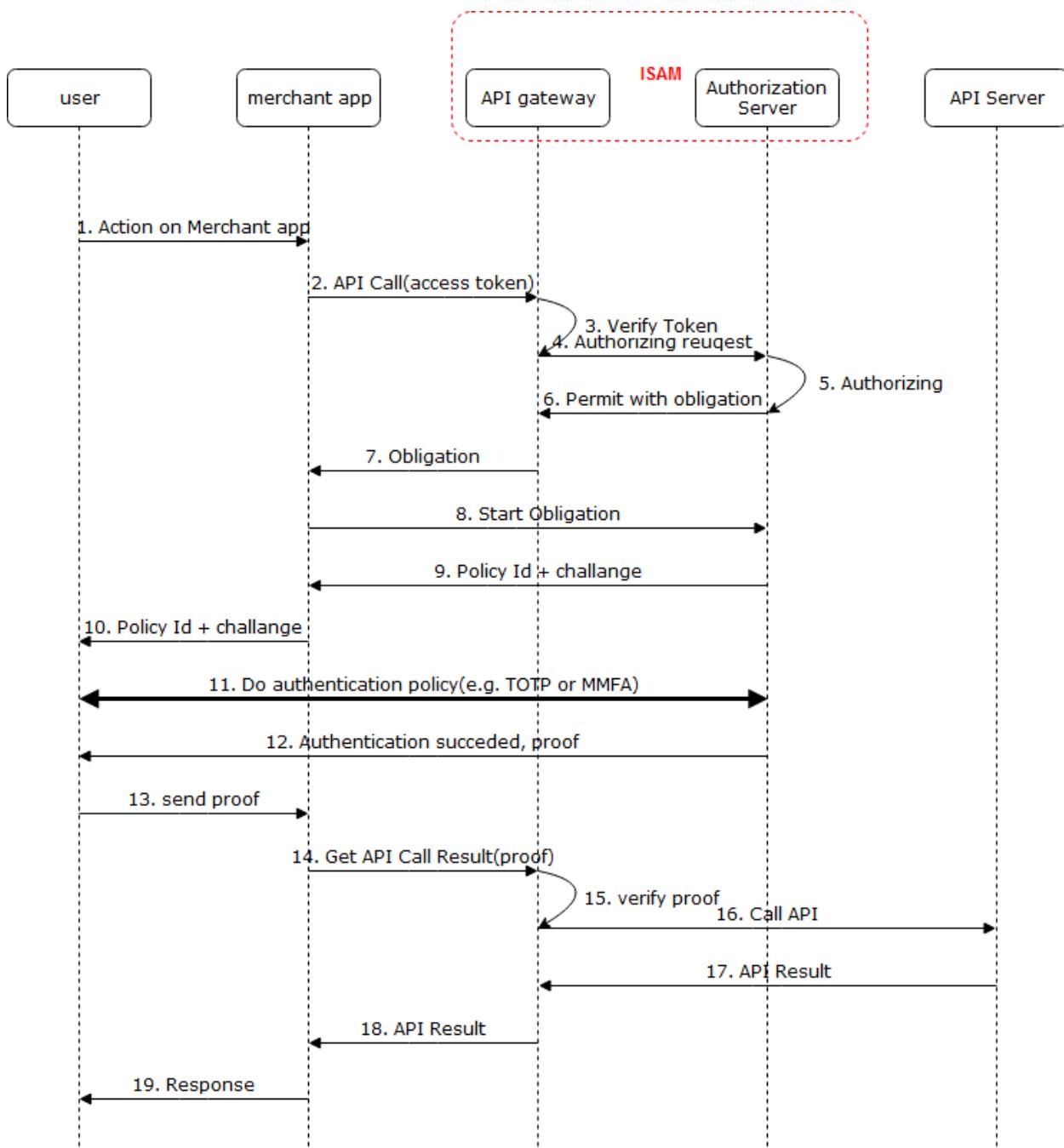
7 Advanced API Protection

As we mentioned in the previous chapter, real-world APIs require more sophisticated protection mechanisms. In this chapter, we discuss a scenario in which we protect APIs using OAuth along with MMFA.

7.1 Scenario

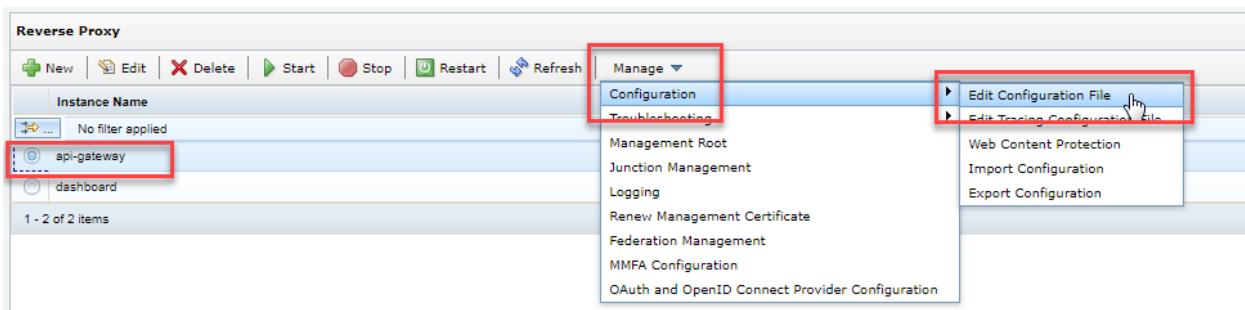
After a successful access delegation, Merchant can call APIs belonging to access token scope. The following steps should be carried out in an API call:

1. User access Merchant web application and do an action on it
2. Behind the action in Merchant application, there is an API call. Merchant application calls an API on behalf of User by sending its access token.
3. API gateway validates the access token sent by merchant request
4. To authorize this API call, API gateway sends an authorization request to Authorization Server
5. Authorization Server authorize this API call
6. Authorization server responds “**Permit With Obligation**” and ask the user to have more authentication steps (Obligation) based on the authorization policy defined for this API
7. API gateway send authentication policy id to the Merchant
8. Merchant start authentication flow regarding policy id against Authorization Server
9. Authorization Server provides a challenge along with a new authentication policy id to the Merchant
10. Merchant send back authentication policy id and challenge to the user
11. User does the authentication steps based on the authentication policy id.
12. After successful authentication, The Authorization Server sends a proof of the successful authentication to the user.
13. User send the proof to the Merchant
14. Merchant ask to get API call Response by sending the proof to the API gateway
15. API gateway verifies the proof
16. API gateway passes the API call to API server
17. API server returns the result for this call
18. API gateway forwards the result to the Merchant application
19. Merchant application creates an appropriate response based on the result of the API call



7.2 Modify Reverse Proxy Instance Configuration File

Navigate to **Secure Web Settings > Manage: Reverse Proxy**.



Select the radio button for the **api-gateway** Reverse Proxy instance. Click on **Manage** and select **Configuration > Edit Configuration File** from the pop-up menu.

Enable OAuth authentication:

```
[oauth]
oauth-auth = https
```

Disable OAuth EAS:

```
[oauth-eas]
eas-enabled = false
```

7.3 Import template files

Clone required infomaps in Host machine using the following command:

```
git clone https://github.com/authsaz/isambookdemo-infomaps.git
```

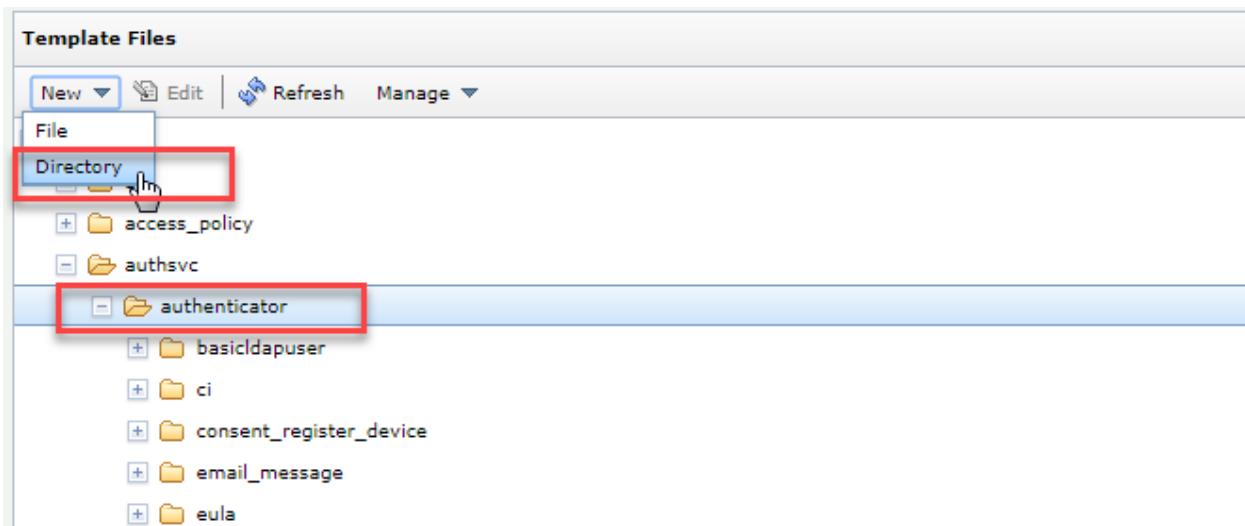
Name	Date modified	Type	Size
.git	12/18/2018 2:45 PM	File folder	
AccountListPip.js	12/5/2018 8:43 AM	JScript Script File	3 KB
api_policy_cache.js	12/18/2018 8:58 AM	JScript Script File	2 KB
api_policy_config.js	12/18/2018 1:58 PM	JScript Script File	1 KB
api_policy_init.js	12/18/2018 10:10 ...	JScript Script File	2 KB
api_policy_JWT.js	12/18/2018 1:59 PM	JScript Script File	2 KB
api_policy_resp_finish.js	12/18/2018 8:53 AM	JScript Script File	1 KB
api_policy_resp_init.js	12/18/2018 9:46 AM	JScript Script File	2 KB
api_policy_STS.js	12/18/2018 2:40 PM	JScript Script File	5 KB
api_policy_WebService.js	12/18/2018 10:00 ...	JScript Script File	3 KB
initiate_api_policy.json	12/17/2018 5:33 PM	JSON File	1 KB
jsrsasign.js	12/16/2018 3:00 PM	JScript Script File	264 KB
LICENSE	12/2/2018 5:57 PM	File	2 KB
OAuthPip.js	12/4/2018 4:55 PM	JScript Script File	8 KB
README.md	12/2/2018 5:57 PM	MD File	1 KB

After successful cloning you should see these files.

The screenshot shows the IBM Security Access Manager dashboard. A red box highlights the "Secure Access Control" icon in the top navigation bar. Another red box highlights the "Template Files" link under the "Global Settings" section of the main menu.

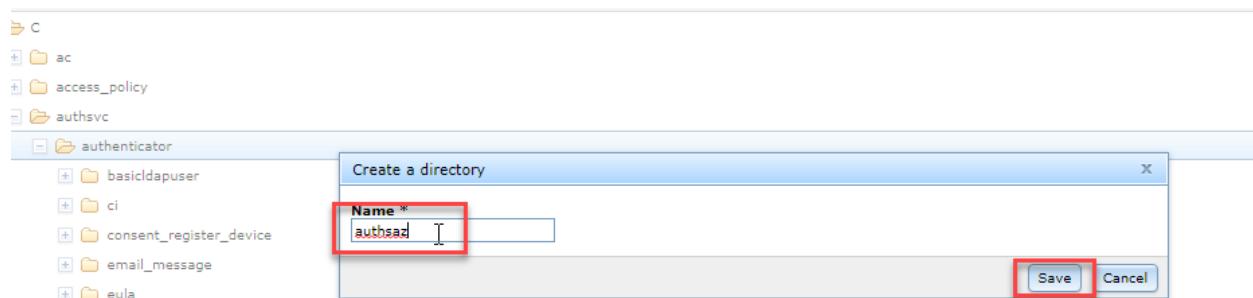
Policy	Manage	Global Settings
Access Control	Devices	Advanced Configuration
Authentication	Grants	User Registry
Risk Profiles	Database Maintenance	Runtime Parameters
Attributes	SCIM Configuration	Template Files
Obligations	Push Notification Providers	Mapping Rules
OpenID Connect and API Protection	MMFA Configuration	Distributed Session Cache
Information Points	Attribute Source	Server Connections
Extensions		Point of Contact
		Access Policies

Navigate to **Secure Access Control > Global Settings: Template Files**

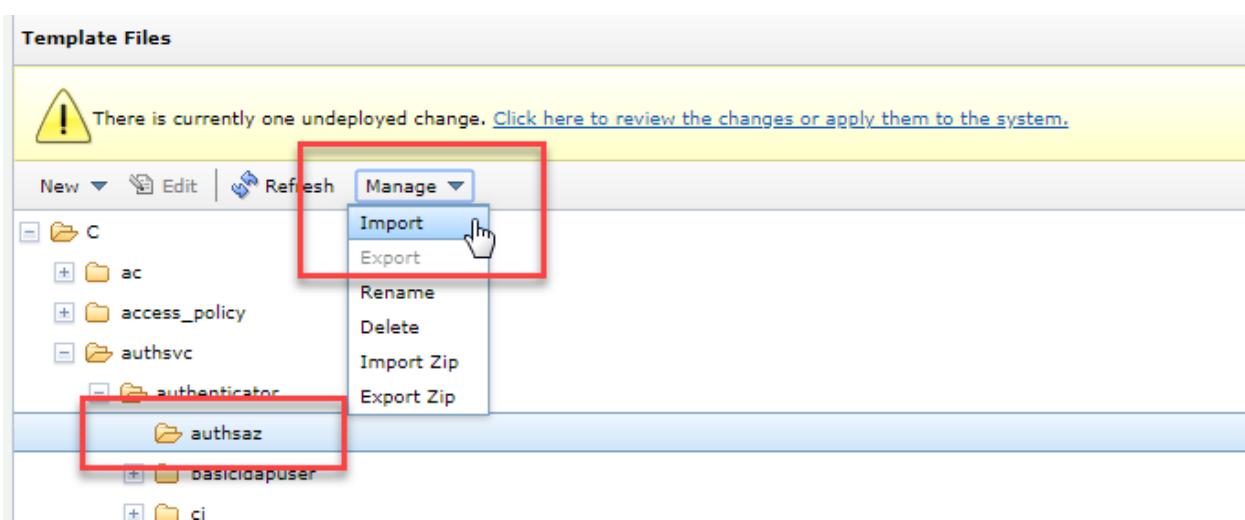


Expand **authsvc** directory. Select **authenticator**.

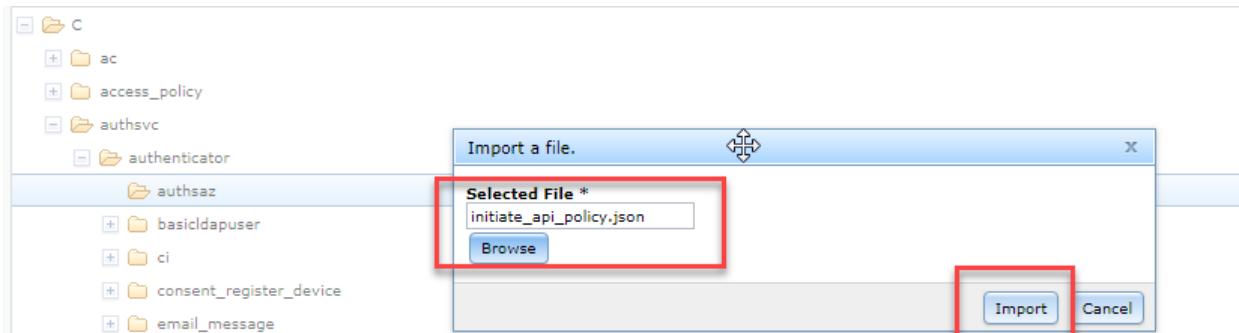
Click **New**. from drop-down menu select **Directory**.



Enter **authsaz** as the Name for *directory*. Click **Save**.



Click **Manage**. From drop-down menu click **Import**.

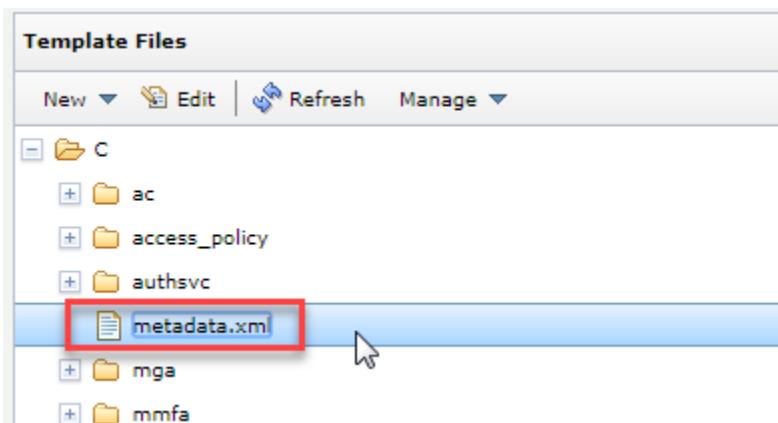


Click **Browse**. From the **isambookdemo-infomaps** directory select **initiate_api_policy.json** file.

Click **Import**.

Deploy the changes using the link in the yellow warning message.

7.3.1 Edit metadata.xml



From *Template Files* tree double click on **metadata.xml** file.

```
<!-- make sure all json pages get sent back as application/json -->
<meta:Metadata>
    <meta:Identifiers>
        <meta:IdentifierPattern>
            <meta:Include>.*\.json</meta:Include>
        </meta:IdentifierPattern>
    </meta:Identifiers>
    <meta:Parameters>
        <meta:Parameter name="setHeader.Content-Type">application/json</meta:Parameter>
```

```

</meta:Parameters>
</meta:Metadata>

```

Copy above content at the end of the file.

```

<meta:Include>*.\css</meta:Include>
</meta:IdentifierPattern>
</meta:Identifiers>
<meta:Parameters>
    <meta:Parameter name="setHeader.Content-Type">text/css</meta:Parameter>
</meta:Parameters>
</meta:Metadata>

<!-- make sure all json pages get sent back as application/json -->
<meta:Metadata>
    <meta:Identifiers>
        <meta:IdentifierPattern>
            <meta:Include>*.\json</meta:Include>
        </meta:IdentifierPattern>
    </meta:Identifiers>
    <meta:Parameters>
        <meta:Parameter name="setHeader.Content-Type">application/json</meta:Parameter>
    </meta:Parameters>
</meta:Metadata>

</meta:MetadataCollection>

```

Click **Save**.

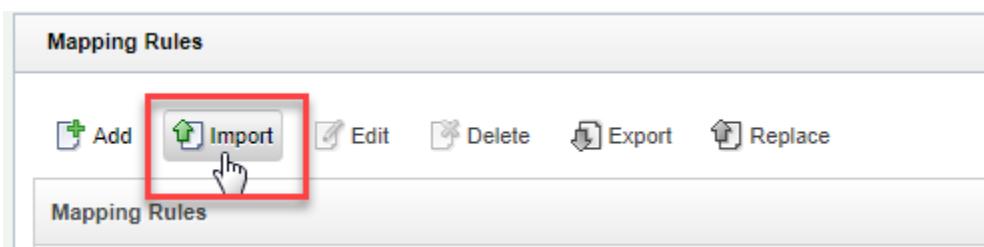
Deploy the changes using the link in the yellow warning message.

7.4 Import mapping rules

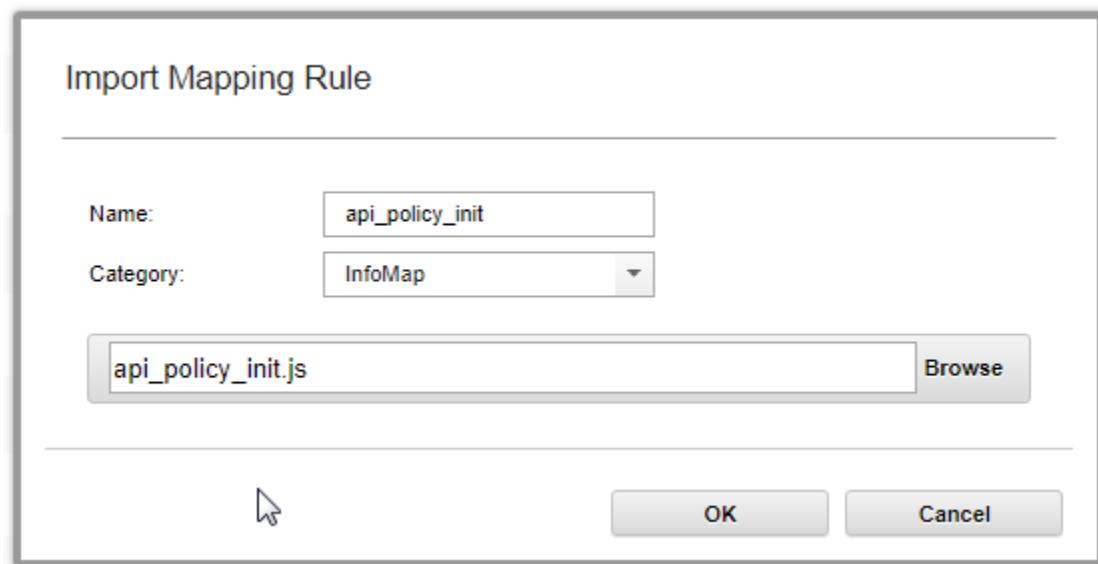
In this section, we will import some mapping rules required to implement our scenarios. In chapter 9 we have a deep delve into these mapping rules.

The screenshot shows the IBM Security Access Manager dashboard. The top navigation bar includes links for Home, Monitor, Secure Web Settings, Secure Access Control (which is highlighted with a red box), Secure Federation, and Connect. The main content area has three columns: Policy (with links like Access Control, Authentication, Risk Profiles, etc.), Manage (with links like Devices, Grants, Database Maintenance, etc.), and Global Settings (with links like Advanced Configuration, User Registry, Runtime Parameters, etc.). A red box highlights the 'Mapping Rules' link under the Global Settings column.

Navigate to **Secure Access Control > Global Settings: Mapping Rules**.



Click **Import**.



Enter **api_policy_init** as *Name*.

Select **InfoMap** as *Category*.

Click **Browse**.

From the **isambookdemo-infomaps** directory select **api_policy_init.js** file

Click **Ok**.

Repeat these step to add the following additional infomaps:

- **api_policy_config.js**
- **api_policy_resp_init.js**
- **api_policy_resp_finish.js**
- **api_policy_cache.js**
- **api_policy_JWT.js**
- **api_policy_STS.js**

- **api_policy_WebService.js**
- **jsrsasign.js**

The screenshot shows a table titled "Mapping Rules". At the top, there are buttons for Add, Import, Edit (which is highlighted with a red box), Delete, Export, and Replace. Below the buttons, the table lists several entries:

- USC_PasswordReset_Success Category: InfoMap
- api_policy_JWT Category: InfoMap
- api_policy_STS Category: InfoMap
- api_policy_WebService Category: InfoMap
- api_policy_cache Category: InfoMap
- api_policy_config** Category: InfoMap (This row is highlighted with a dashed red box)
- api_policy_init Category: InfoMap
- api_policy_resp_finish Category: InfoMap
- api_policy_resp_init Category: InfoMap
- jsrsasign Category: InfoMap

After importing all infomaps, select **api_policy_config** infomap from *Mapping Rules* table.

Click **Edit**.

Mapping Rules - api_policy_config

```
var secret = "CHANGE_ME";
var expiresInSeconds = 60;

/*
implementation:
1    cache
2    WebService
3    JWT
4    STS
*/
var implementation = 1;

switch(implementation){
    case 1: // cache
        importMappingRule("api_policy_cache");
        conf = {};
    case 2: // WebService
        importMappingRule("api_policy_WebService");
        conf = {
            webservice: "http://apps.authsaz.com:7001/internal/token"
        };
    case 3: // JWT
        importMappingRule("api_policy_JWT");
        conf = {secret: secret};
    case 4: // STS
}
```

Name:

Category:

Modify **api_policy_config** mapping rule by assigning **1** to variable **implementation**.

Click **Save**.

Deploy the changes using the link in the yellow warning message.

7.5 Create Authentication mechanisms

The screenshot shows the IBM Security Access Manager interface. At the top, there is a navigation bar with icons for Home, Monitor, Secure Web Settings, Secure Access Control (which is highlighted with a red box), and Secure Federation. Below the navigation bar is a main content area divided into three columns: Policy, Manage, and Global Settings. The Policy column contains links for Access Control, Authentication (which is highlighted with a red box), Risk Profiles, Attributes, Obligations, OpenID Connect and API Protection, Information Points, and Extensions. The Manage column contains links for Devices, Grants, Database Maintenance, SCIM Configuration, Push Notification Providers, MMFA Configuration, Attribute Source, and Attribute Source. The Global Settings column contains links for Advanced Configuration, User Registry, Runtime Parameters, Template Files, Mapping Rules, Distributed Session Cache, Server Connections, Point of Contact, and Access Policies.

Navigate to **Secure Access Control > Policy: Authentication.**

The screenshot shows a sub-menu for 'Authentication'. It includes tabs for Authentication, Policies, Mechanisms (which is highlighted with a red box), and Advanced. Below the tabs are several icons: a green plus sign for adding, a pencil for edit, a copy icon, a checkmark, a minus sign, and a refresh icon. A dropdown menu is open, showing options: Info Map Authentication (which is highlighted with a red box and has a cursor over it), Email Message, SCIM Config, FIDO Universal 2nd Factor, and Cloud Identity JavaScript. The text 'Select authentication mechanism' is visible at the bottom of the dropdown.

Navigate to **Mechanisms.**

The screenshot shows the 'Mechanisms' tab selected in the 'Authentication' sub-menu. The 'Info Map Authentication' option is highlighted with a red box and has a cursor over it. Other listed mechanisms include Email Message, SCIM Config, FIDO Universal 2nd Factor, and Cloud Identity JavaScript. The text 'Select authentication mechanism' is visible at the bottom of the list.

Click **add** icon. From the drop-down menu select **Info Map Authentication**.

New Authentication Mechanism

General Properties

Name: **api_policy_init**

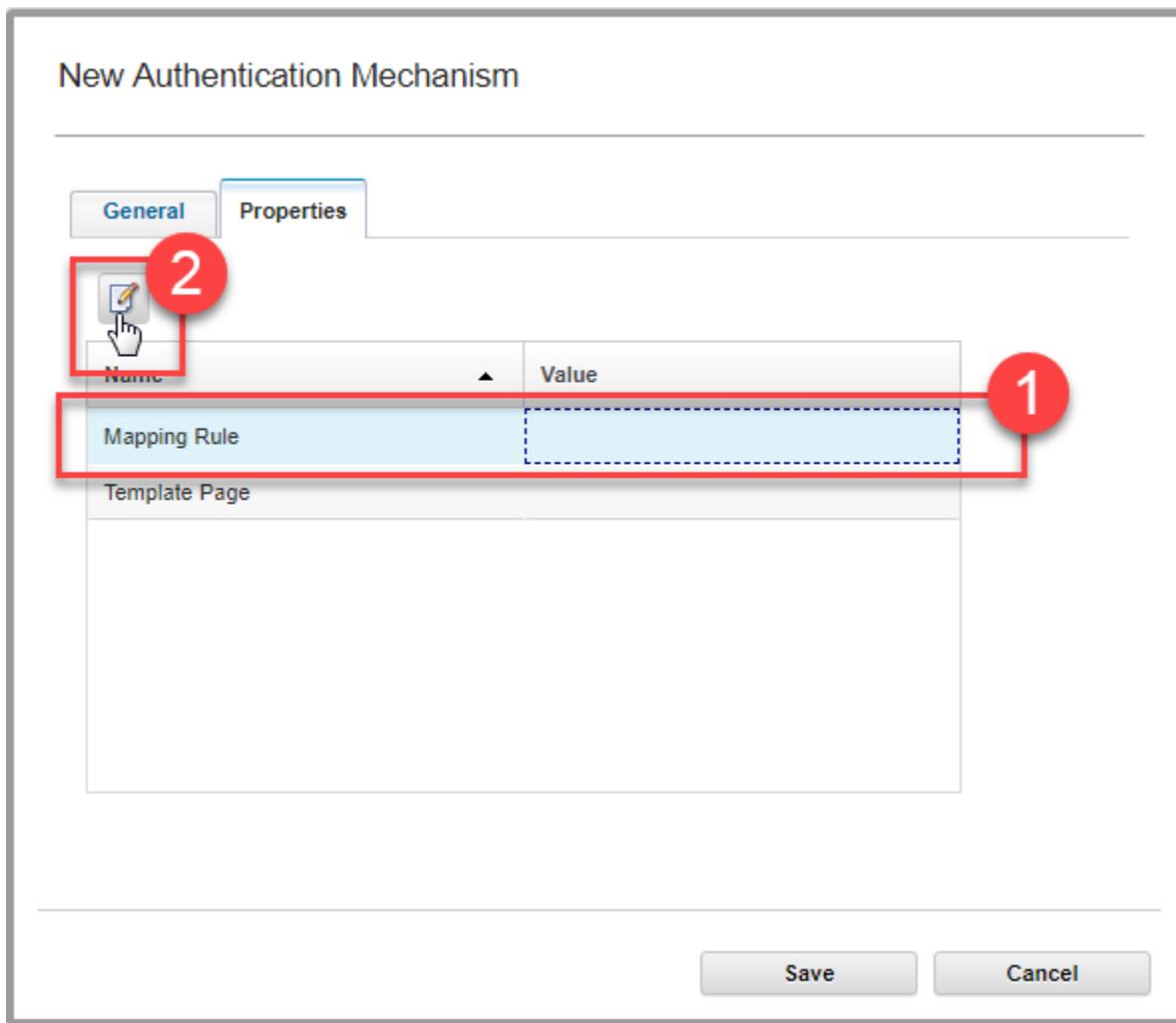
Identifier: **urn:ibm:security:authentication:asf:mechanism:
api_policy_init**

Description:

Type: Info Map Authentication

Save Cancel

Enter **api_policy_init** as *Name* and *Identifier*.



Navigate to **Properties** tab.

Select **Mapping Rules**. Click **Edit** icon.

New Authentication Mechanism

General Properties



Name	Value
Mapping Rule	api_policy_init
Template Page	/authsvc/authenticator/authsaz/initiate_api_

Save Cancel

Enter **api_policy_init** as value for *Mapping Rule*.

Select **Template Page**. Click **Edit** icon.

Enter **/authsvc/authenticator/authsaz/initiate_api_policy.json** as the value of *Template Page*.

Click **Save**.

Repeat steps above to add the following additional infomaps:

Name: api_policy_resp_init

Identifier: api_policy_resp_init

Mapping Rule: api_policy_resp_init

Template Page: *Empty*

Name: api_policy_resp_finish

Identifier: api_policy_resp_finish

Mapping Rule: api_policy_resp_finish

Template File: *Empty*

Deploy the changes using the link in the yellow warning message.

7.6 Create Authentication Policies

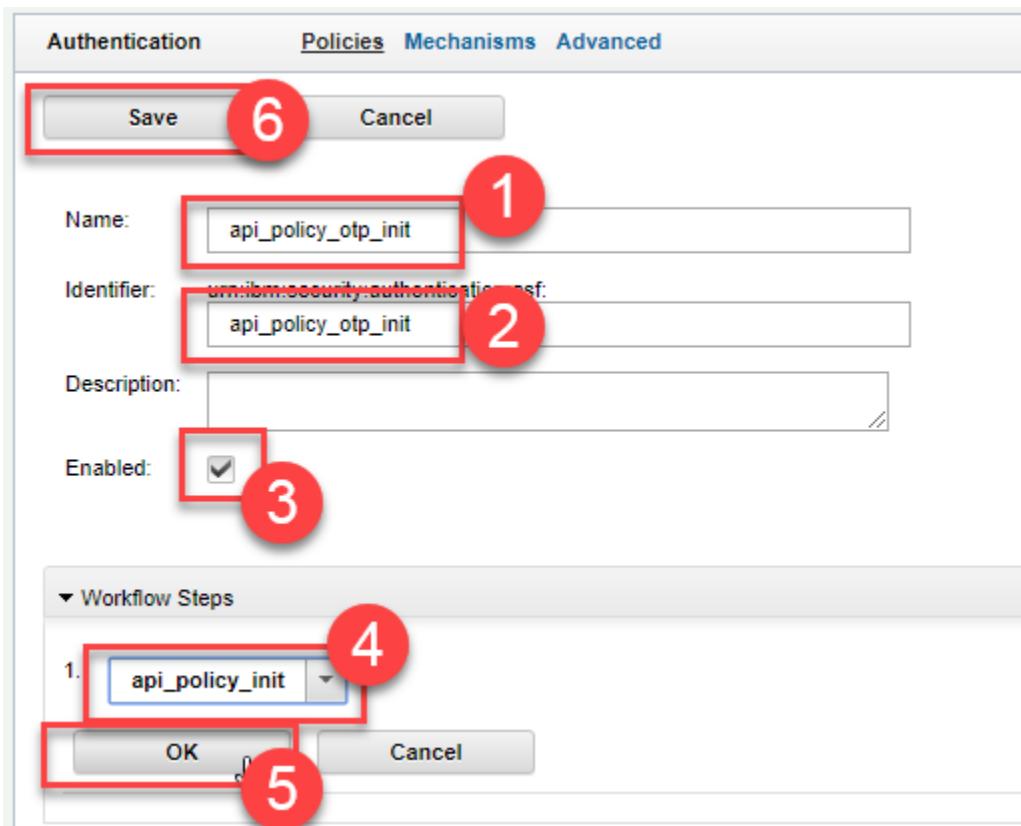
If not already there, navigate to **Secure Access Control > Policy: Authentication**.

7.6.1 Create api_policy_otp

7.6.1.1 Create api_policy_otp_init policy



Select the **Policies** tab click **New** button to create a new authentication policy.



Enter **api_policy_otp_init** as the *Name*.

Enter **api_policy_otp_init** in the *Identifier* box. This will be appended to the text already shown which makes the full identifier: **urn:ibm:security:authentication:asf:api_policy_otp_init**.

Enter a *Description*.

Check *Enabled*.

Click **Add Step** button to add the first step.

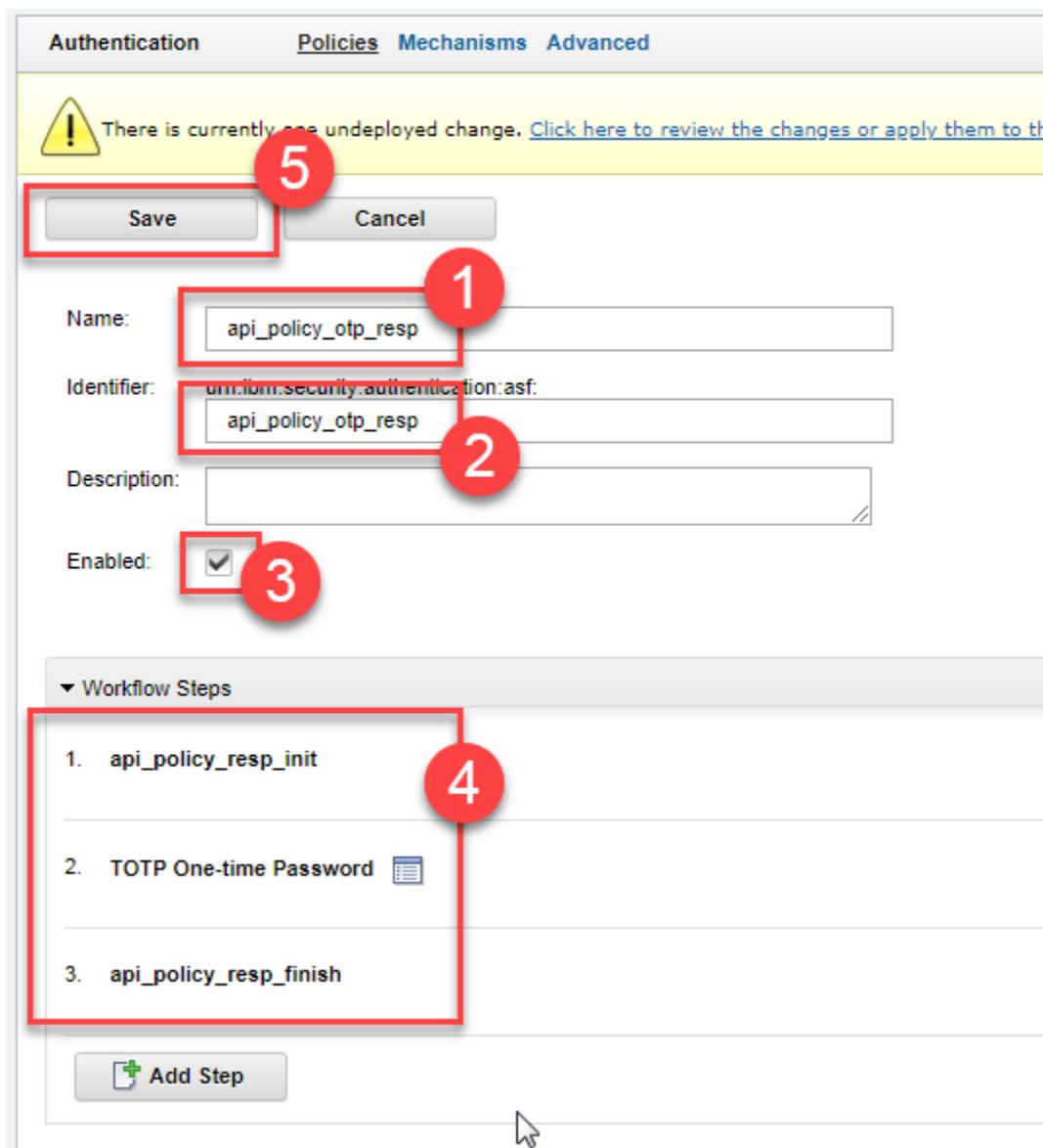
Select **api_policy_init** mechanism from the drop-down list. This is the InfoMap mechanism we created in the previous step. Then click **OK**.

Click **Save**.

Deploy the changes using the link in the yellow warning message.

7.6.1.2 Create api_policy_otp_resp policy

Again, select the Policies tab and click the **New** button to create another authentication policy.



Enter **api_policy_otp_resp** as the *Name*.

Enter **api_policy_otp_resp** in the *Identifier* box. This will be appended to the text already shown which makes the full identifier: **urn:ibm:security:authentication:asf:api_policy_otp_resp**.

Enter a *Description*.

Check *Enabled*.

Click **Add Step** button to add the first step.

Select **api_policy_resp_init** mechanism from the drop-down list. Then click **OK**.

Again, click **Add Step** button to add the second step. Select **TOTP One-time Password** mechanism from the drop-down list. Then click **OK**.

Again, click **Add Step** button to add the third step. Select **api_policy_resp_finish** mechanism from the drop-down list. Then click **OK**.

Click **Save**.

Deploy the changes using the link in the yellow warning message.

7.6.2 Create api_policy_presence

7.6.2.1 Create api_policy_presence_init policy

Select the **Policies** tab. Click the **New** button to create a new authentication policy.

The screenshot shows the 'Policies' tab selected in the top navigation bar. A new policy is being created with the following details:

- Name:** api_policy_presence_init
- Identifier:** urn:ibm:security:authentication_asf:api_policy_presence_init
- Description:** (empty)
- Enabled:**

The 'Workflow Steps' section shows one step listed: 1. api_policy_init. There is a red box around this step and another red box around the 'Add Step' button below it.

Enter **api_policy_presence_init** as the *Name*.

Enter **api_policy_presence_init** in the *Identifier* box

Enter a *Description*.

Check **Enabled**.

Click **Add Step** button to add the first step.

Select **api_policy_init** mechanism from the drop-down list. Then click **OK**.

Click **Save**.

Deploy the changes using the link in the yellow warning message.

7.6.2.2 Create api_policy_presence_resp policy

The screenshot shows the 'Create Policy' dialog box. At the top are 'Save' and 'Cancel' buttons. Below them are fields for 'Name' (set to 'api_policy_presence_resp'), 'Identifier' (set to 'urn:ibm:security:authentication:asf:api_policy_presence_resp'), 'Description' (empty), and 'Enabled' (checked). A large section below titled 'Workflow Steps' lists three steps: 1. 'api_policy_resp_init', 2. 'MMFA Authenticator' (with a red box and cursor icon highlighting it), and 3. 'api_policy_resp_finish'. At the bottom is an 'Add Step' button.

Name:	api_policy_presence_resp
Identifier:	urn:ibm:security:authentication:asf: api_policy_presence_resp
Description:	(empty)
Enabled:	<input checked="" type="checkbox"/>

Workflow Steps

1. api_policy_resp_init
2. MMFA Authenticator
3. api_policy_resp_finish

Add Step

Enter **api_policy_presence_resp** as the *Name*.

Enter **api_policy_presence_resp** in the *Identifier* box.

Enter a *Description*.

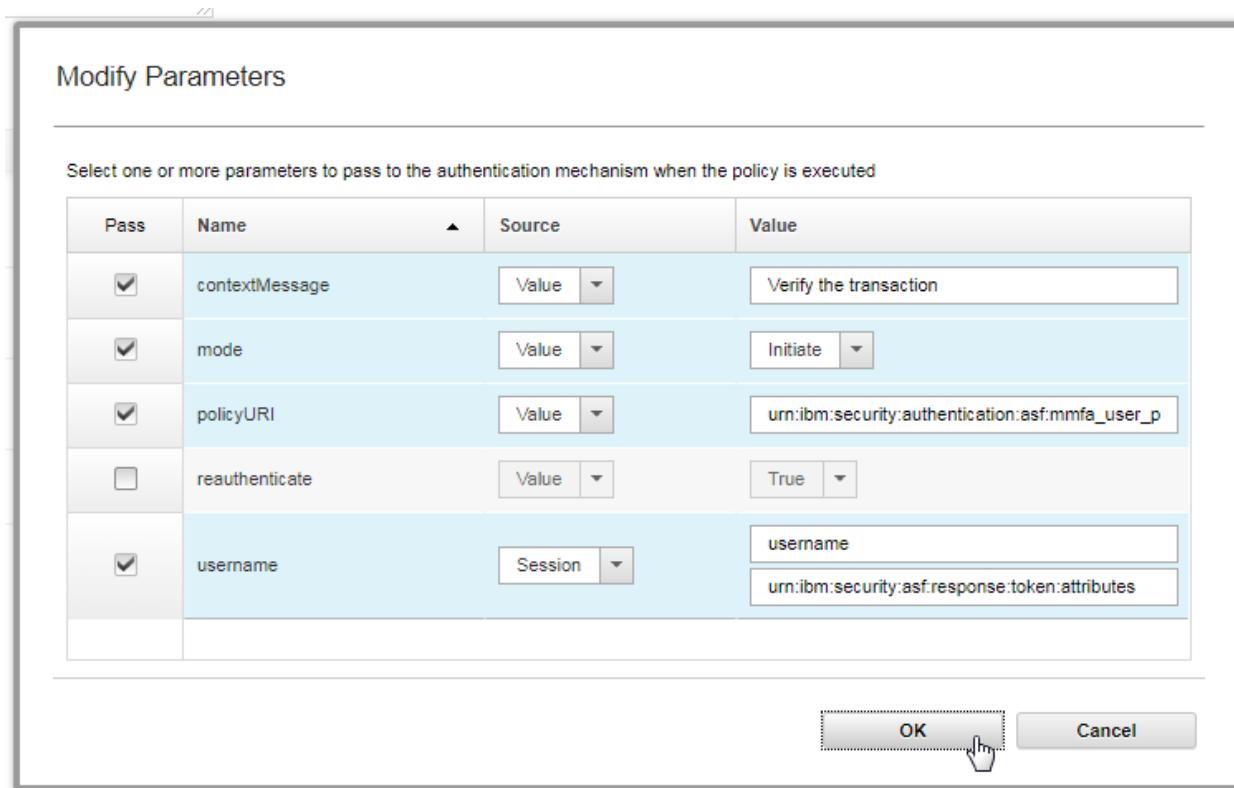
Check **Enabled**.

Click **Add Step** button to add the first step. Select **api_policy_resp_init** mechanism from the drop-down list. Then click **OK**.

Click **Add Step** button to add the second step. Select **MMFA Authenticator** mechanism from the drop-down list. Then click **OK**.

Click **Add Step** button to add the third step, select **api_policy_resp_finish** mechanism from the drop-down list. Then click **OK**.

Now, Click the icon next to **MMFA Authenticator** step in workflow steps table(second step).



Select the checkbox next to *contextMessage* and set the value to **Verify the transaction**. This is the text that will be displayed to the user in the Authenticator Client application.

Select the checkbox next to *mode*. The default value of **Initiate** is what we want so don't change it. This tells the mechanism that it is initiating the challenge to the Authenticator Client.

Select the checkbox next to *policyURI*. Set the value to:

urn:ibm:security:authentication:asf:mmfa_user_presence_response

Select the checkbox next to the *username*. Select **Session** from the drop-down list for Source. This indicates that we will read from the session.

Enter **urn:ibm:security:asf:response:token:attributes** as the Namespace for the value.

Enter **username** as the *Attribute Id*.

Click **OK** to close the parameters window.

Click **Save**.

Deploy the changes using the link in the yellow warning message.

7.6.3 Create api_policy_finger

7.6.3.1 Create api_policy_otp_finger_init policy

Select the **Policies** tab. Click the **New** button to create a new authentication policy.

The screenshot shows the 'Policies' tab selected in the top navigation bar. A new policy is being created with the following details:

- Name:** api_policy_otp_finger_init
- Identifier:** urn:ibm:security:authentication:asf:
api_policy_otp_finger_init
- Description:** (empty)
- Enabled:** checked

The 'Workflow Steps' section contains one step: 1. api_policy_init. An 'Add Step' button is available below it.

Enter **api_policy_otp_finger_init** as the *Name*.

Enter **api_policy_otp_finger_init** in the *Identifier* box

Enter a *Description*.

Check *Enabled*.

Click **Add Step** button to add the first step.

Select **api_policy_init** mechanism from the drop-down list. Then click **OK**.

Click **Save**.

Deploy the changes using the link in the yellow warning message.

7.6.3.2 Create api_policy_otp_finger_resp policy

Screenshot of the 'Policies' tab in a configuration interface, showing the creation of a new policy named 'api_policy_otp_finger_resp'. The 'Workflow Steps' section is expanded, displaying four steps: 1. api_policy_resp_init, 2. TOTP One-time Password, 3. MMFA Authentication (with a red box highlighting the icon), and 4. api_policy_resp_finish. An 'Add Step' button is visible at the bottom.

Name:	api_policy_otp_finger_resp
Identifier:	urn:ibm:security:authentication:asf: api_policy_otp_finger_resp
Description:	(empty)
Enabled:	<input checked="" type="checkbox"/>

Workflow Steps

1. api_policy_resp_init
2. TOTP One-time Password
3. MMFA Authentication
4. api_policy_resp_finish

Add Step

Enter **api_policy_otp_finger_resp** as the *Name*.

Enter **api_policy_otp_finger_resp** in the *Identifier* box.

Enter a *Description*.

Check *Enabled*.

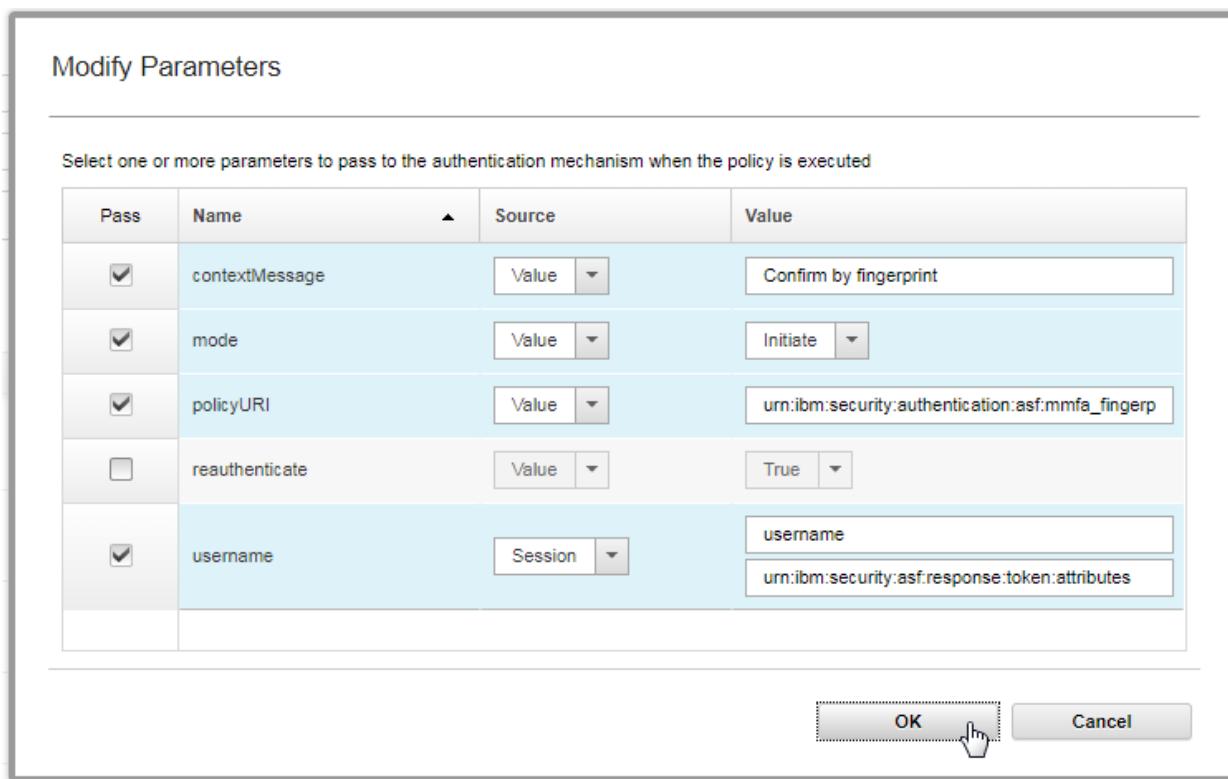
Click **Add Step** button to add the first step. Select **api_policy_resp_init** mechanism from the drop-down list. Then click **OK**.

Click **Add Step** button to add the second step. Select **TOTP One-time Password** mechanism from the drop-down list. Then click **OK**.

Click **Add Step** button to add the third step. Select **MMFA Authenticator** mechanism from the drop-down list. Then click **OK**.

Click **Add Step** button to add the fourth step, select **api_policy_resp_finish** mechanism from the drop-down list. Then click **OK**.

Now, Click the icon next to **MMFA Authenticator** step in workflow steps table(third step).



Select the checkbox next to *contextMessage* and set the value to **Confirm by fingerprint**. This is the text that will be displayed to the user in the Authenticator Client application.

Select the checkbox next to *mode*. The default value of *Initiate* is what we want so don't change it. This tells the mechanism that it is initiating the challenge to the Authenticator Client.

Select the checkbox next to *policyURI*. Set the value to:

urn:ibm:security:authentication:asf:mmfa_fingerprint_response

Select the checkbox next to the *username*. Select **Session** from the drop-down list for Source. This indicates that we will read from the session.

Enter **urn:ibm:security:asf:response:token:attributes** as the Namespace for the value.

Enter **username** as the *Attribute Id*.

Click **OK** to close the parameters window.

Click **Save**.

Deploy the changes using the link in the yellow warning message.

The screenshot shows the Policies tab in the IBM Security Policy Manager. The top navigation bar includes tabs for Authentication, Policies (which is selected), Mechanisms, and Advanced. Below the navigation is a toolbar with icons for creating, editing, deleting, and saving policies. The main area is titled "Authentication" and contains a table with the following rows:

	api_policy_otp_finger_init
	api_policy_otp_finger_resp
	api_policy_otp_init
	api_policy_otp_resp
	api_policy_presence_init
	api_policy_presence_resp

A red box highlights the first two rows of the table. At the bottom of the table, there is a note: "Consent Register Device Consent to device registration authentication policy."

After creating all policies, the **Authentication** table in **Policies tab** should be something like the picture above.

7.7 Create Access Control policy

The screenshot shows the IBM Security Access Manager dashboard. The top navigation bar includes links for Home, Monitor, Secure Web Settings, **Secure Access Control** (which is highlighted with a red box), Secure Federation, and Connected IBM Cloud. Below the navigation bar, there are three main sections: Policy, Manage, and Global Settings. The Policy section has a sub-section for Access Control, which is also highlighted with a red box. Other items in the Policy section include Authentication, Risk Profiles, Attributes, Obligations, OpenID Connect and API Protection, Information Points, and Extensions. The Manage section contains links for Devices, Grants, Database Maintenance, SCIM Configuration, Push Notification Providers, MMFA Configuration, Attribute Source, and others. The Global Settings section contains links for Advanced Configuration, User Registry, Runtime Parameters, Template Files, Mapping Rules, Distributed Session Cache, Server Connections, Point of Contact, and Access Policies.

Navigate to **Secure Access Control > Policy: Access control**.

The screenshot shows the Access Control screen within the IBM Security Access Manager. The top navigation bar is identical to the previous dashboard. Below it, there are tabs for Access Control, Policies, Resources, Attributes, and Obligations. The Policies tab is highlighted with a red box. On the left, there is a sidebar with icons for All Policies and Policy Sets. On the right, there is a toolbar with various buttons, one of which is a green 'New' button with a plus sign, also highlighted with a red box. Below the toolbar, there is a list area labeled 'All Policies'.

Select the **Policies** tab and click the **New** button to create a new Access Control policy.

IBM Security Access Manager

Home Appliance Dashboard Monitor Analysis and Diagnostics Secure Web Settings

Access Control Policies Resources Attributes Obligations

Save Cancel

Name: api_totp

Description:

▼ Subjects

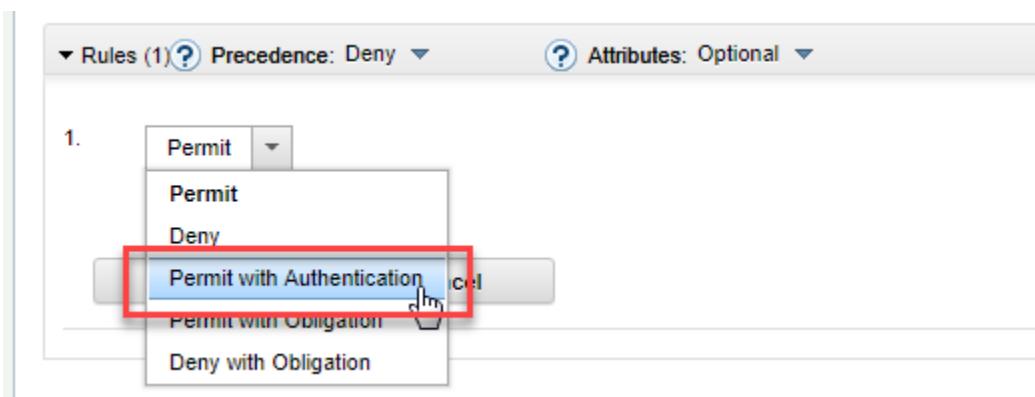
▼ Rules Precedence: Deny Attributes: Optional

Conditional rule Unconditional rule

The screenshot displays the IBM Security Access Manager web interface. At the top, there's a navigation bar with links for Home, Monitor, and Secure Web Settings. Below that is a secondary navigation bar with tabs for Access Control, Policies (which is selected), Resources, Attributes, and Obligations. Under the Policies tab, there are 'Save' and 'Cancel' buttons. The main form area starts with a 'Name:' field containing 'api_totp', which is enclosed in a red box. Below it is a 'Description:' field with an empty text area. A 'Subjects' section follows, featuring an 'Add Subject' button. The 'Rules' section is open, showing 'Add Rule' and two rule types: 'Conditional rule' and 'Unconditional rule'. The 'Unconditional rule' option is also highlighted with a red box.

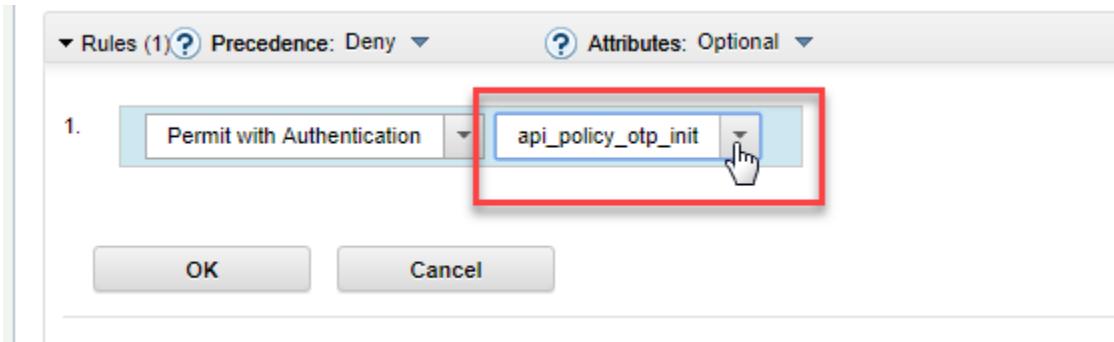
Enter **api_totp** as the *Name*. Enter a description.

Click **Add Rule**. Then select **Unconditional rule**.



Click on the drop-down menu.

Select **Permit with Authentication**.



Select **api_policy_otp_init** from second drop-down menu.

Click **OK**.

IBM Security Access Manager

Home Appliance Dashboard Monitor Analysis and Diagnostics Secure Web Settings Secure Access Cor

Access Control Policies Resources Attributes Obligations

Save Cancel

Name: api_totp

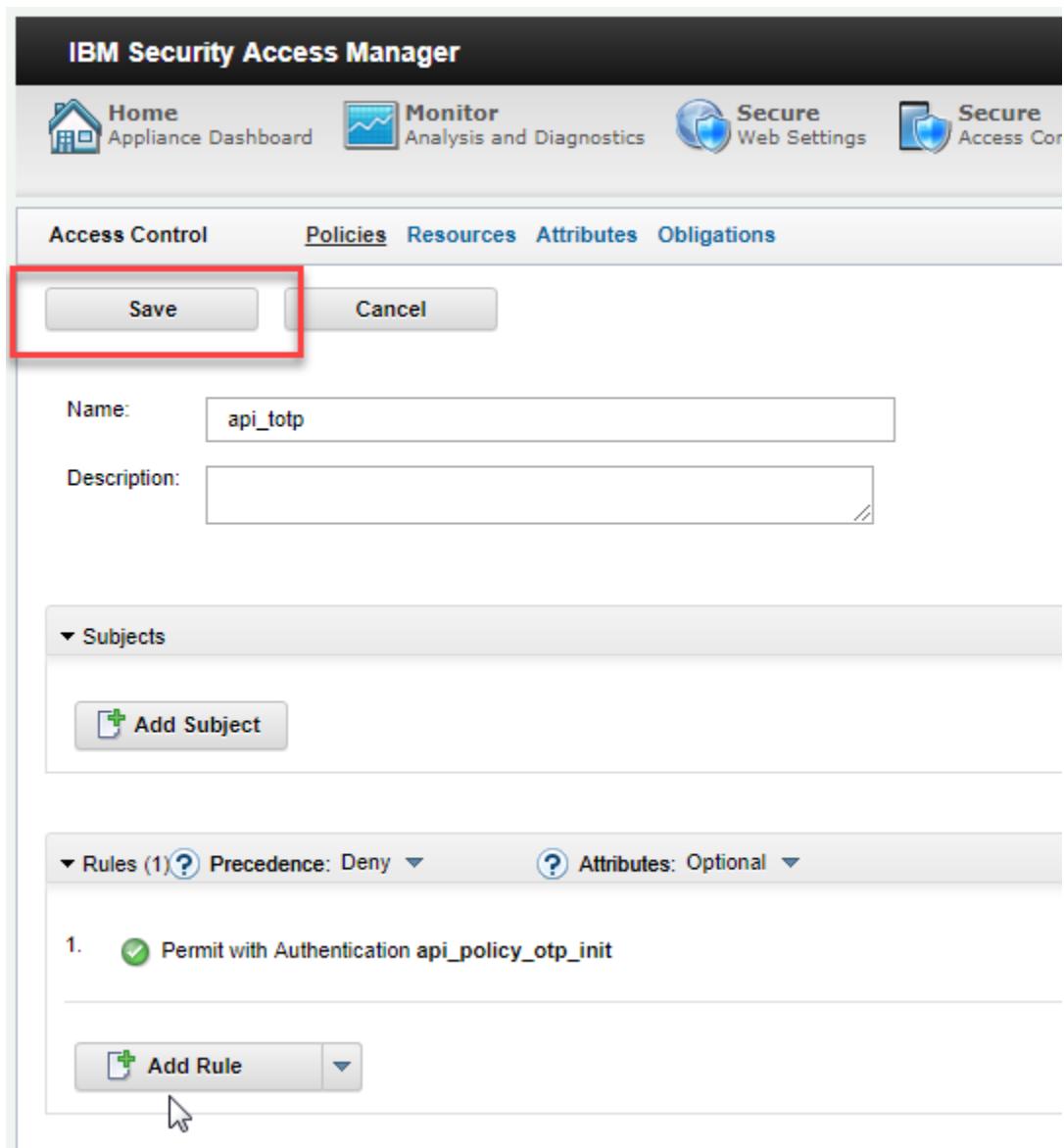
Description:

▼ Subjects **Add Subject**

▼ Rules (1) Precedence: Deny Attributes: Optional

1. Permit with Authentication api_policy_otp_init

Add Rule



Click **Save**.

Repeat steps above to create new policies with **access control policy name** and **authentication policy name** according to the table below.

access control policy name	authentication policy name
api_totp	api_policy_otp_init
api_presence	api_policy_presence_init
api_otp_finger	api_policy_otp_finger_init

The screenshot shows a software interface with a top navigation bar containing 'Policies', 'Resources', 'Attributes', and 'Obligations'. Below this is a table titled 'All Policies' with three entries: 'api_presence', 'api_otp_finger', and 'api_totp'. A red box highlights the 'api_presence' row, and a cursor is positioned over it. Above the table is a toolbar with icons for creating, editing, deleting, and adding to other sections.

After creating all policies, the **All Policies** table should be something like the picture above.

7.7.1 Attach Access Control Policies to APIs

The screenshot shows a software interface with a top navigation bar containing 'Access Control', 'Policies', 'Resources' (which is highlighted with a red box), 'Attributes', and 'Obligations'. Below this is a table titled 'All Policies' with three entries: 'api_presence', 'api_otp_finger', and 'api_totp'. To the left is a sidebar with 'All Policies' and 'Policy Sets' options. Above the table is a toolbar with icons for creating, editing, deleting, and adding to other sections.

Navigate to the **Resources** tab.

The screenshot shows a software interface with a top navigation bar containing 'Access Control', 'Policies', 'Resources' (highlighted with a red box), 'Attributes', and 'Obligations'. Below this is a table titled 'Resources' with two entries: '/api' and 'AppsRegister'. The '/api' entry has a 'Remove' icon highlighted with a red box. The 'AppsRegister' entry is highlighted with a red box. Above the table are buttons for 'Attach', 'Publish', 'Publish All', and 'Change Domain'. A status bar at the bottom right indicates 'Published: Dec 10'.

If **AppsRegister** policy is attached to **/api** resource select it. This policy has been added in the previous chapter and we don't need it anymore. Then click **Remove** icon.

The screenshot shows the ISAM (Identity and Access Management) interface. At the top, there are tabs: Access Control, Policies, **Resources**, Attributes, and Obligations. Below the tabs is a toolbar with several icons: a plus sign for adding, a pencil for edit, a delete, Attach, Publish, Publish All, and Change Domain. The main area is titled "Resources" and shows a tree structure. Under "isam-test-api-gateway", there is a node for "/api".

Click **add** button.

The dialog box is titled "Add Resource". It contains the following fields:
Type: **Reverse Proxy**
Proxy Instance: **isam.authsaz.com-api-gateway**
Protected Path: **/api/nf/accounts** (highlighted with a red box)
Below these fields is a section for cache timeout:
● Disable decision cache (selected)
● Cache decision for the life of the session
● Cache decision for a specified time
At the bottom right are "Save" and "Cancel" buttons, with "Save" highlighted by a red box.

Select **Reverse Proxy** as *Type*.

Select **isam.authsaz.com-api-gateway** as *Proxy Instance*.

Enter **/api/nf/accounts** as *Protected Path*.

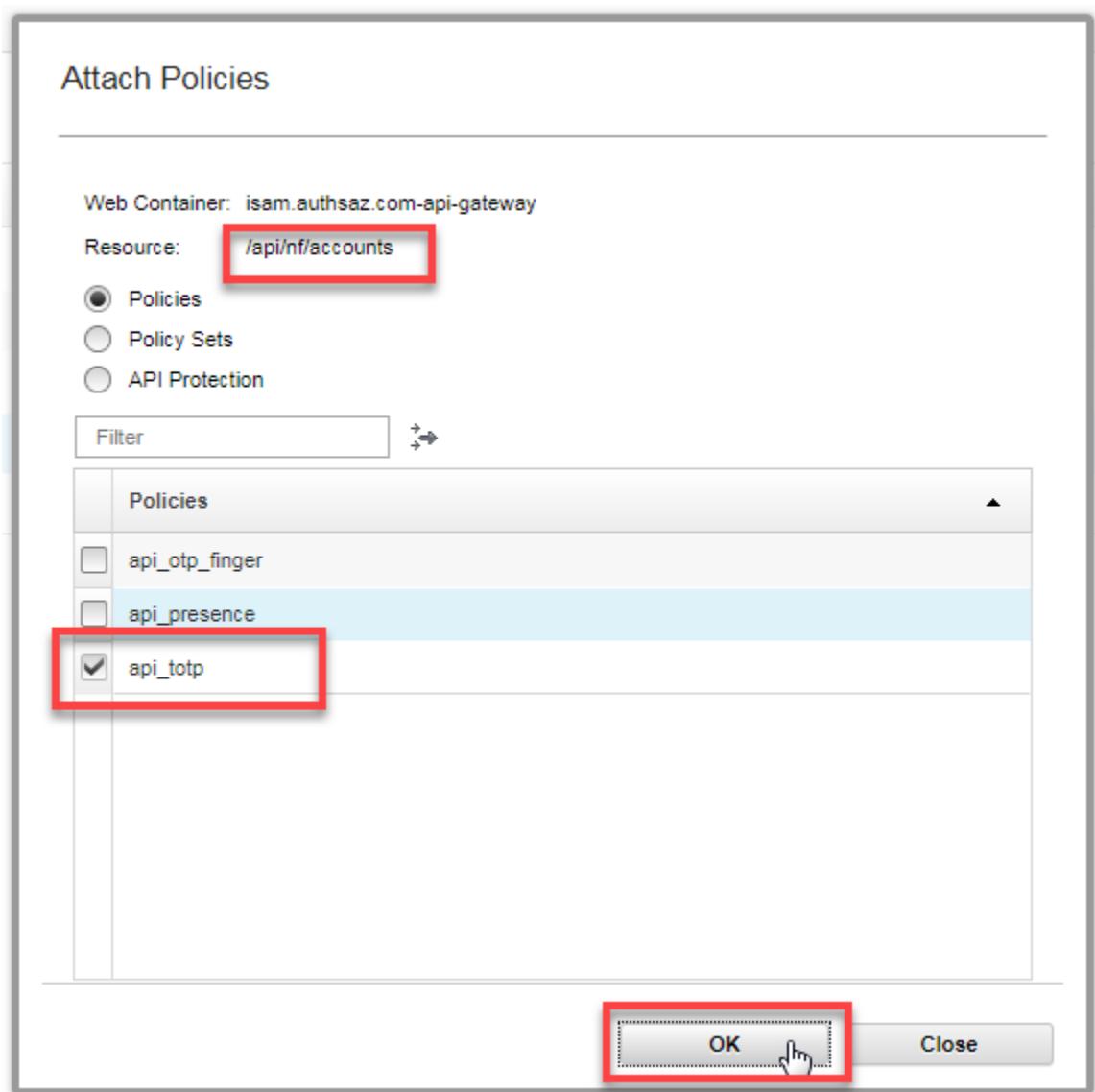
Click **Save**.

Repeat these steps to create the following paths, as well:

- **/api/nf/balance**
- **/api/f/transfer**

The screenshot shows a software interface for managing access control resources. At the top, there are tabs for 'Access Control', 'Policies', 'Resources' (which is selected and underlined), 'Attributes', and 'Obligations'. Below the tabs, there are several icons: a green plus sign, a pencil, a red delete, and a blue 'Attach' button, which is highlighted with a red box and a hand cursor icon. To the right of these are 'Publish', 'Publish All', and 'Change Domain' buttons. The main area is titled 'Resources' and shows a tree structure. Under the root node 'isam.authsaz.com-api-gateway', there is a folder named '/api'. Inside '/api', three resources are listed: '/api/ft/transfer', '/api/nf/accounts', and '/api/nf/balance'. The resource '/api/nf/accounts' is highlighted with a red box. A dashed blue rectangle surrounds the entire '/api' folder.

Now, Select the row **/api/nf/accounts** and click **Attach**.



Select radio button for **Policies** and select checkbox for **api_totp**.

Click **Ok**.

Repeat steps above to attach policies to other resources according to the table below:

Resource	Policy
----------	--------

/api/nf/accounts	api_totp
/api/nf/balance	api_presence
/api/f/transfer	api_otp_finger

In the end, the *Resource* table should be like this:

The screenshot shows the IBM API Gateway's Resource management interface. The 'Resources' tab is active. In the main pane, under the '/api' category, the '/api/f/transfer' resource is highlighted with a red box. The toolbar at the top has several buttons: '+', 'Edit', 'Delete', 'Attach', 'Publish', 'Publish All' (which is highlighted with a red box and a cursor icon), and 'Change Domain'. Below the toolbar, there is a table with columns 'Resources' and 'Status'. The '/api/f/transfer' row shows 'api_otp_finger' and a yellow warning icon with the text 'Publish required'. Other rows for '/api/nf/accounts', '/api/nf/balance', and '/api/f/transfer' also show 'Publish required' status.

Click **Publish All**.

7.8 Configure IBM Verify

As we mentioned earlier IBM provides a pre-built application called *IBM Verify* for generic use. Install *IBM Verify* app on your smartphone.

In your favorite browser, navigate to <https://dashboard.authsaz.com/portal/user>. Login to the **dashboard** as **user1**.

The screenshot shows the ISAM Devices interface. At the top, there are links for ISAM Devices, ISAM Scim, and [Logout]. Below this, the 'Authorization Grant' section is displayed with a table:

ID	IsEnabled	ClientId	Remove
uuid6f428b76-0167-1c2e-8372-e65b1c4c9d7e	true	jSNhWHOviWPzcqlB3u81	Remove

A yellow box highlights the ID column. Below the table are 'Refresh' and 'New' buttons.

Below the Authorization Grant section is the 'Authenticators' section, which currently shows no matching records found:

ID	Device Name	Device Type	OS Version	oAuth Grant	Enabled	Remove
No matching records found						

Below the table are 'New' and 'Refresh' buttons.

There is a record in **Authorization Grant** table. This record is about the grant token that has been delegated to **merchant.authsaz.com**. If you want to see more information about grant details click the link in **Authorization Grant** table.

7.8.1 Register Authenticator

The screenshot shows the 'Authenticators' section with a table:

ID	Device Name	Device Type	OS Version	oAuth Grant	Enabled	Remove
No matching records found						

A green box highlights the 'New' button, which is being clicked, as indicated by a mouse cursor icon. Below the table are 'New' and 'Refresh' buttons.

In **Authenticators** section, click **New**.

New Authenticator



Client Id: AuthenticatorClient

Code: qQDv5T5OE3vexJFuuL4phdi8Qm5GF0

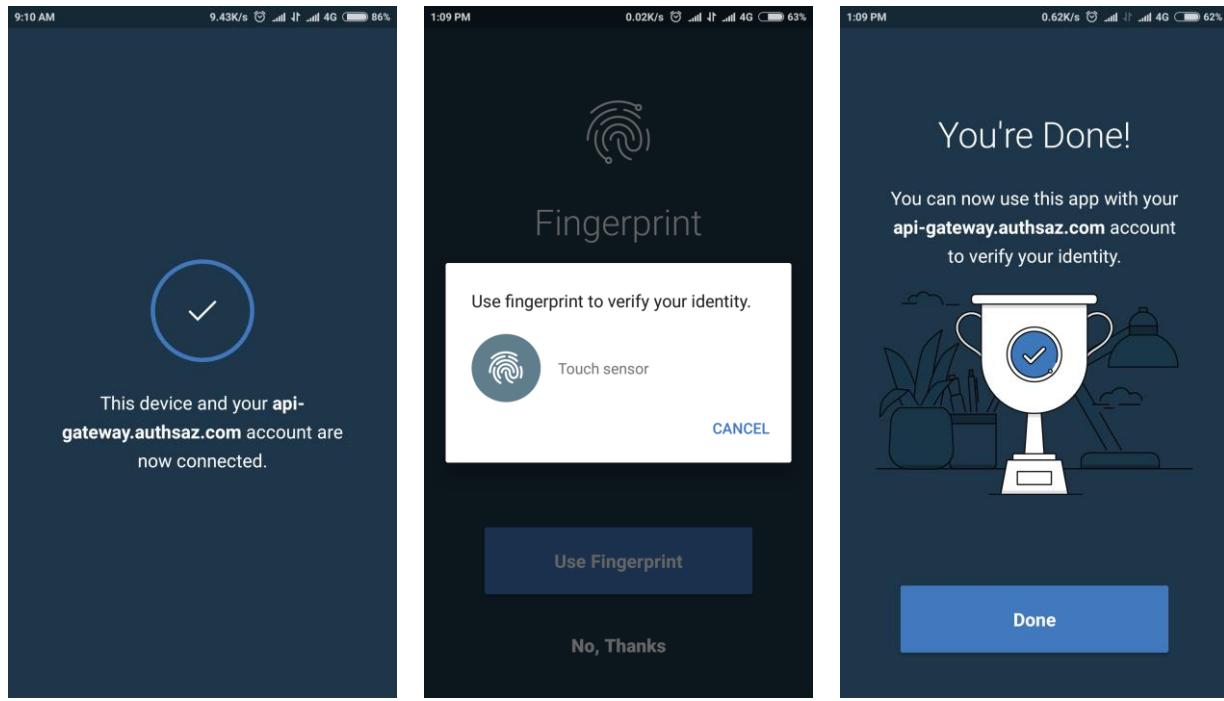
Details Url: <https://api-gateway.authsaz.com:444/mga/sps/mmfa/user/mgmt/details>

Options: ignoreSslCerts=true

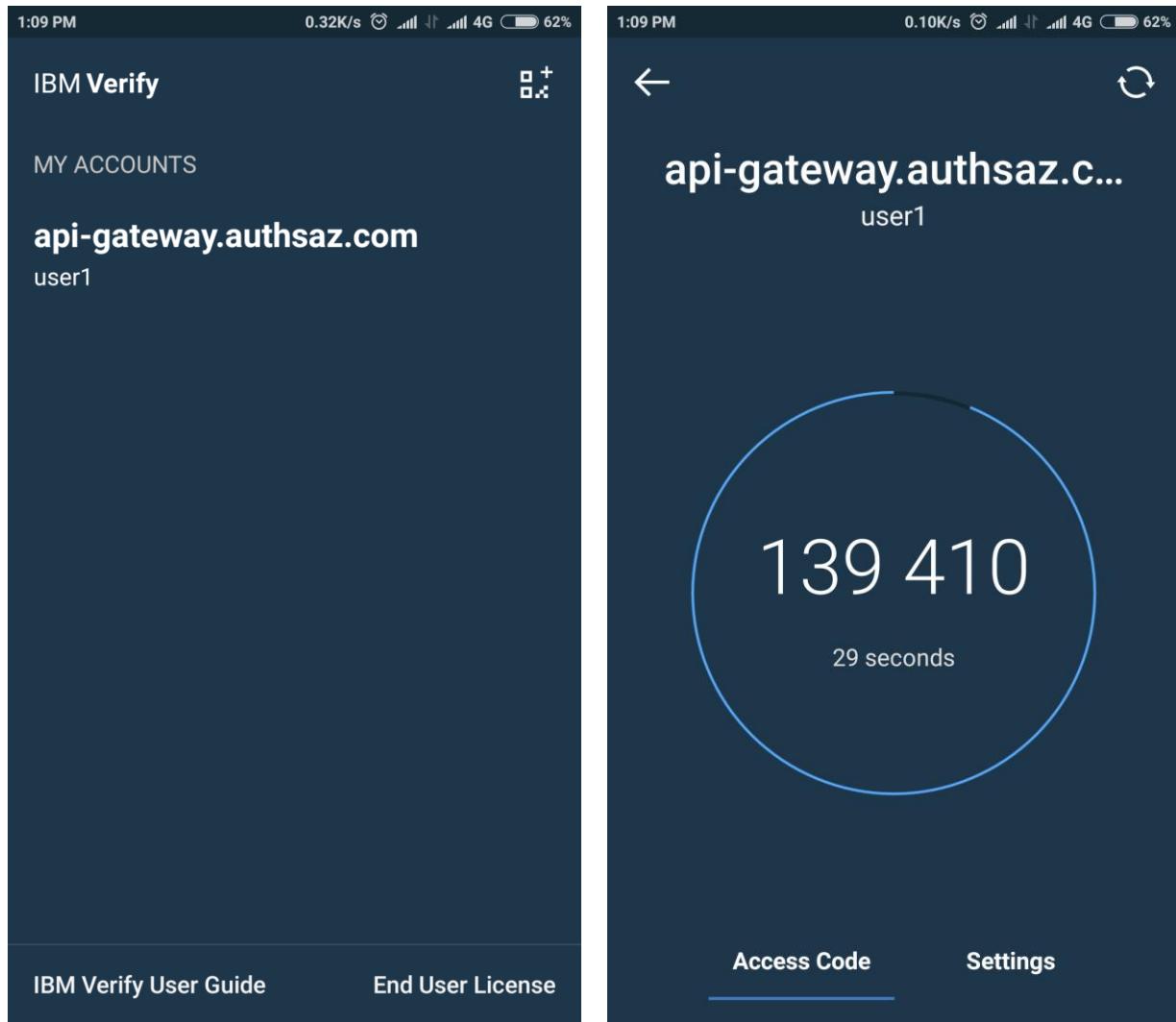
Scan the QR code with *IBM Verify*.

Hint: You may encounter a problem registering your device when working with VMWare in the test environment. In case of the problem, you can use instructions in **Appendix II**. It's because your smartphone network cannot route *IBM Verify* requests to ISAM since they are not in the same network.

Follow steps as shown below to register your mobile device:



Click **Done**. You now see the home page of the *IBM Verify App* with the new account shown (you can read chapter 10 of "**IBM Verify Cookbook Mobile Multi-Factor Authentication with IBM SAM**" book for more details about registering *IBM Verify*).



You now see the home page of the *IBM Verify* App with the new account shown.

Touch the account and it is opened. You can see the current TOTP code here.

Authorization Grant			
Id	IsEnabled	ClientId	Remove
uuid6f428b76-0167-1c2e-8372-e65b1c4c9d7e	true	jSNhWHOviWPzcqlB3u81	<button>Remove</button>
uuid7294c57f-0167-1f41-9c22-e65b1c4c9d7e	true	AuthenticatorClient	<button>Remove</button>
<button>Refresh</button>			

Authenticators						
Id	Device Name	Device Type	OS Version	oAuth Grant	Enabled	Remove
uuid29a136a6-3e02-4a47-807d-cea7ecc91af2	[REDACTED]	[REDACTED]	6.0.1	uuid7294c57f-0167-1f41-9c22-e65b1c4c9d7e	true	<button>Remove</button>
<button>New</button> <button>Refresh</button>						

After successful registration, a new row will be added to **Authenticators** table in the dashboard page. If you want to see more details click the link in the newly added row in the **Authenticators** table.

7.9 Test API policies

Open a new browser window and navigate to: <http://merchant.authsaz.com:5000>.



Make sure that your token has not expired by clicking **Query Info**.

7.9.1 Test Accounts API

Accounts

User name

View Accounts

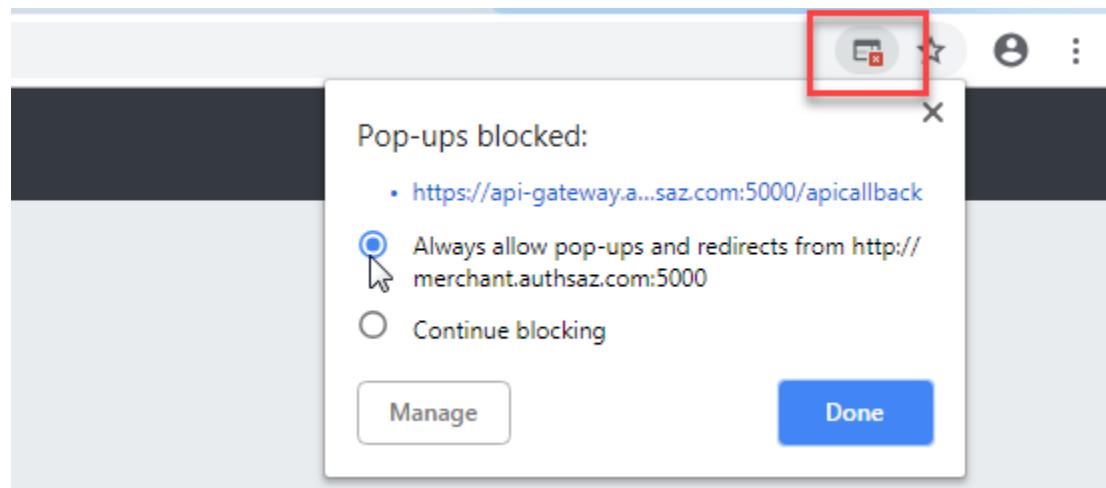
Balance

Account Id

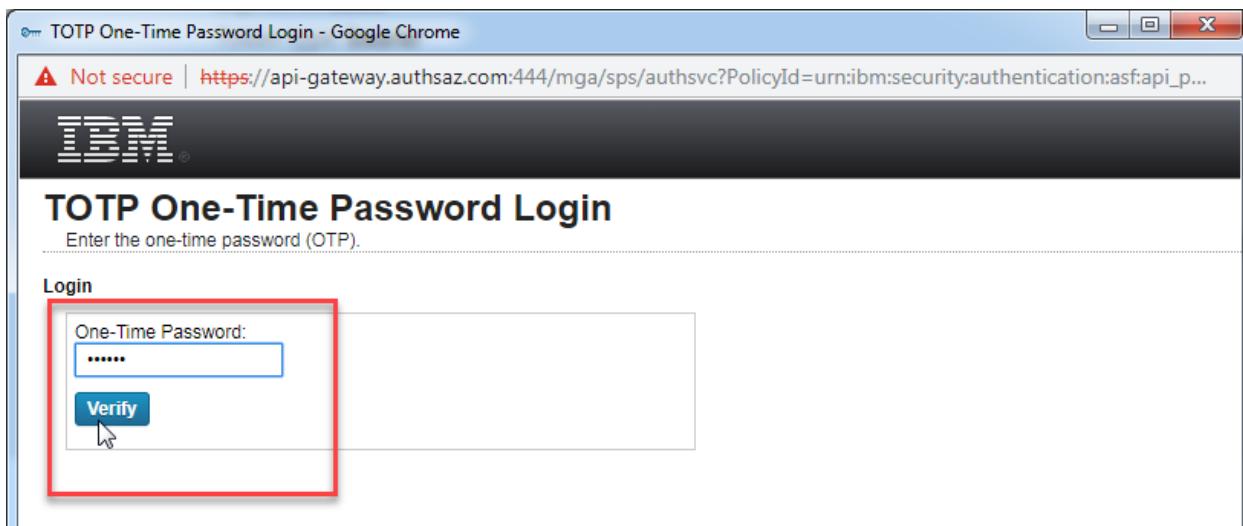
View Balance

Enter **user1** as *User name* in the **Accounts** form.

Click **View Accounts**. It will open a pop-up on your browser.



The browser may block pop-ups for some security reasons. To allow pop-ups On google chrome, click on the icon in right corner of the URL bar. Select radio button for **Always allow pop-ups and redirect from http://merchant.authsaz.com:5000**. Click **Done**.



The **TOTP One-Time Password Login** popup will appear. Read TOTP value from *IBM Verify* and submit the form.



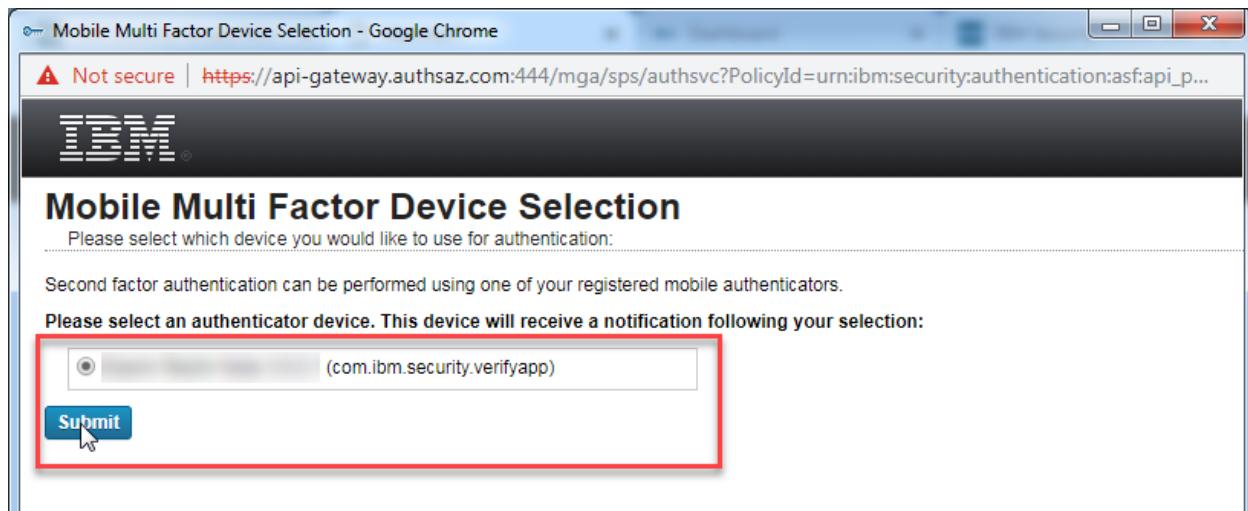
The account numbers of the user should be present.

7.9.2 Test Balance API

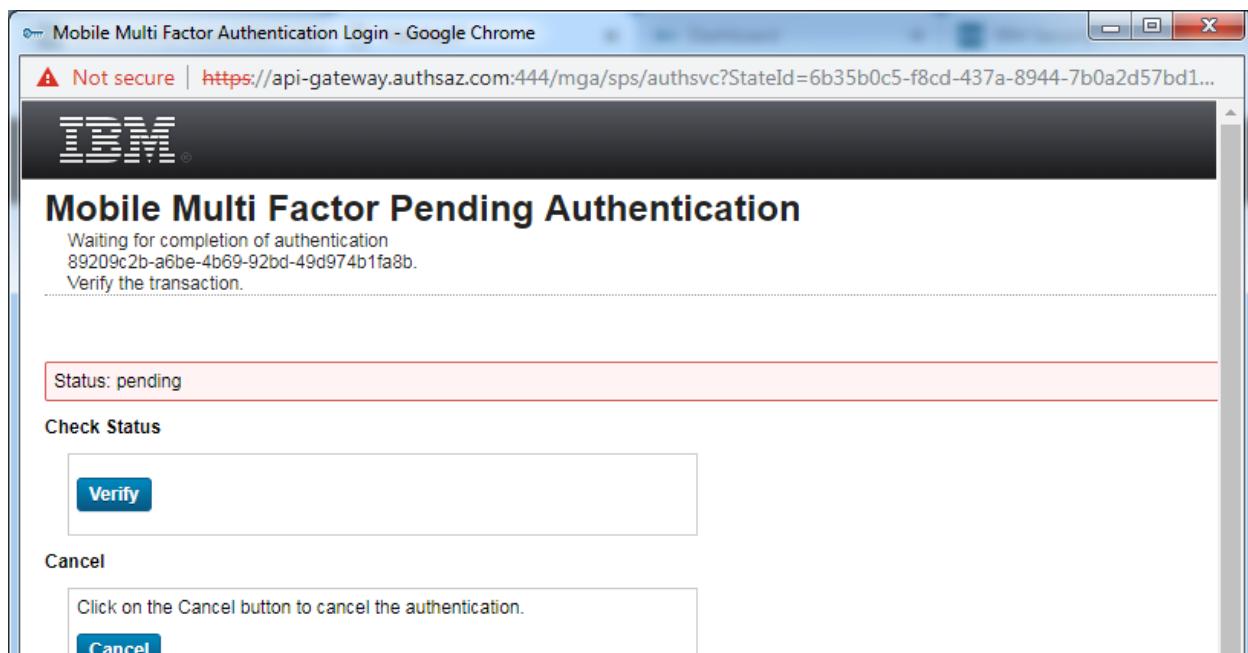
Use one of the account numbers was returned from *Account API* call and enter it as Account Id in the **Balance** form. We used **101010**:

A screenshot of a web application interface. On the left, there is an "Accounts" section with a "user name" input field containing "user1" and a "View Accounts" button. In the center, there is a "Balance" form with an "Account Id" input field containing "101010" and a "View Balance" button. Both the "Account Id" field and the "View Balance" button are highlighted with a red box. On the right, there is a "Transfer" section with "From Account Id" and "To Account Id" input fields, and an "Amount" input field below them.

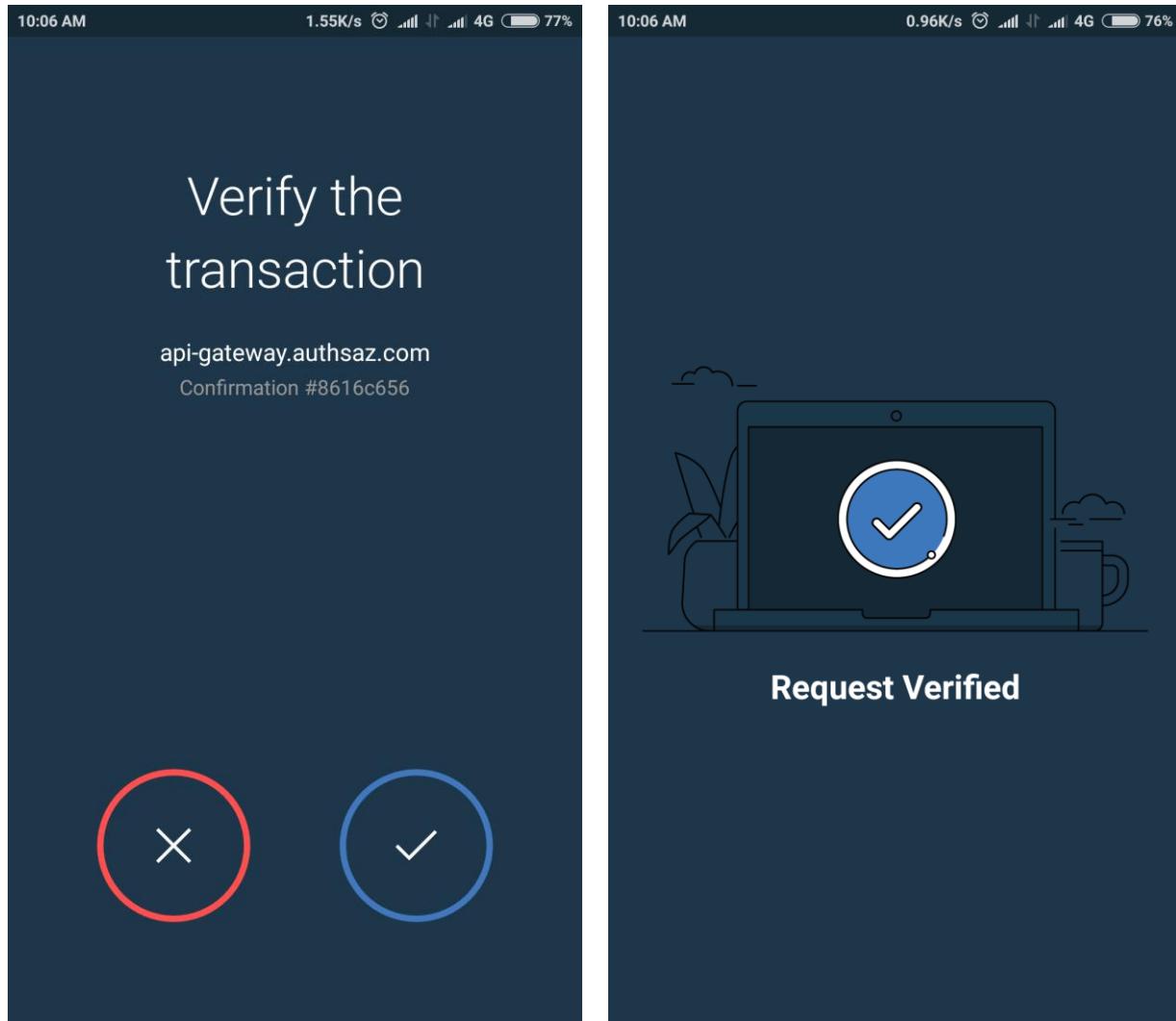
Click **View Balance**. The **Mobile Multi Factor Device Selection** popup will appear.



Choose your registered device, and submit the form.



Open the *IBM Verify* application and touch the refresh icon in the top right corner. This tells the application to poll Access Manager (using its Access Token to authenticate to the SCIM interface) and to retrieve any pending transactions. Repeat this refresh action until you see a blue dot next to the account indicating there is a pending transaction.

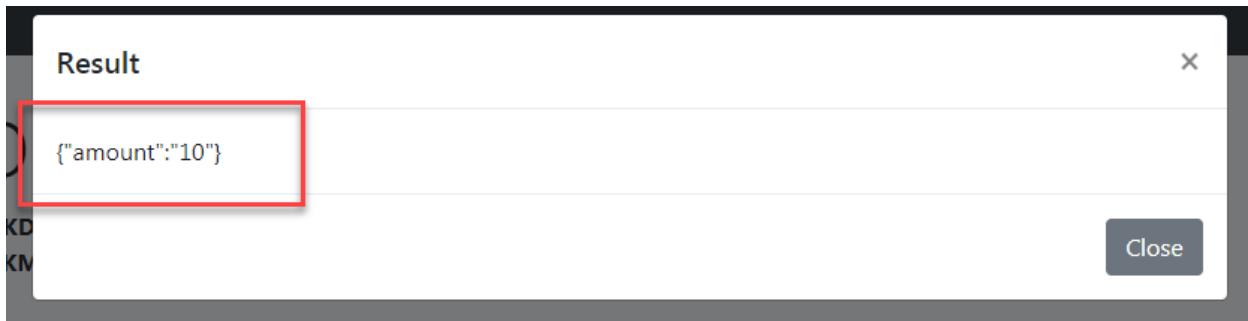


Once you see the blue dot on *IBM Verify*, touch the account to open it. You should immediately be prompted with the pending transaction. You can see the message that was hard-coded into the "initiate" policy.

Since this is a user_presence policy, no fingerprint is required. Touch the tick to verify that you are holding the device and that you are attempting to login.

At this point the *IBM Verify* App signs the challenge it received and returns it to the MMFA Authentication Service where it is validated in the "response" policy. Assuming validation is successful the MMFA Authenticator mechanism marks the transaction as complete and returns success to *IBM Verify*.

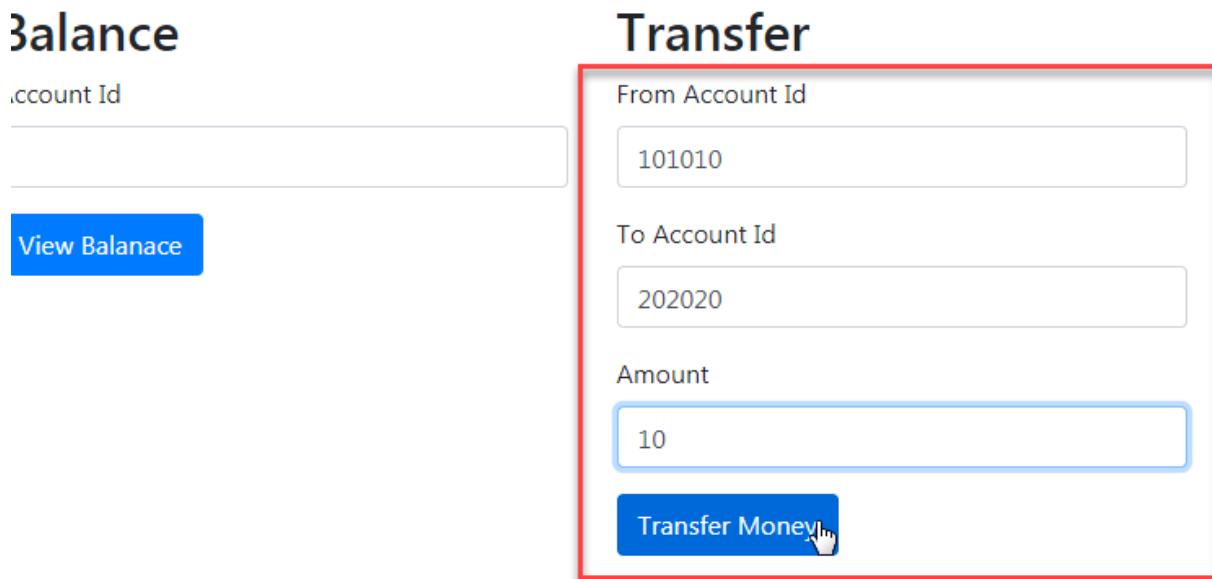
IBM Verify shows a **Request Verified** message and then returns to the accounts list.



At this point the browser is notified via the transaction status service Web Socket connection that the transaction has been processed. It redirects the browser to the MMFA "initiate" policy where user authentication is completed.

The result of calling API will be show in a pop-up window.

7.9.3 Test Transfer API



The image displays two screenshots of a web application interface. The left screenshot shows a 'Balance' page with a 'View Balance' button. The right screenshot shows a 'Transfer' form with fields for 'From Account Id' (101010), 'To Account Id' (202020), and 'Amount' (10). The 'Transfer Money' button is highlighted with a red box.

Go to *Transfer* form.

Enter **101010** as *From Account Id*.

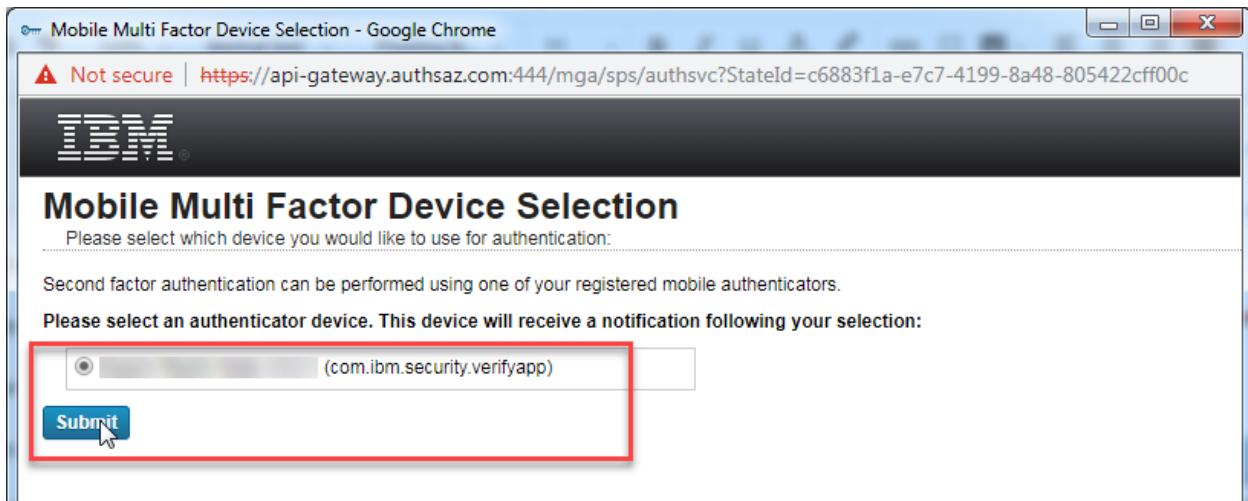
Enter **202020** as *To Account Id*.

Enter **10** as *Amount*.

Click **Transfer Money**.

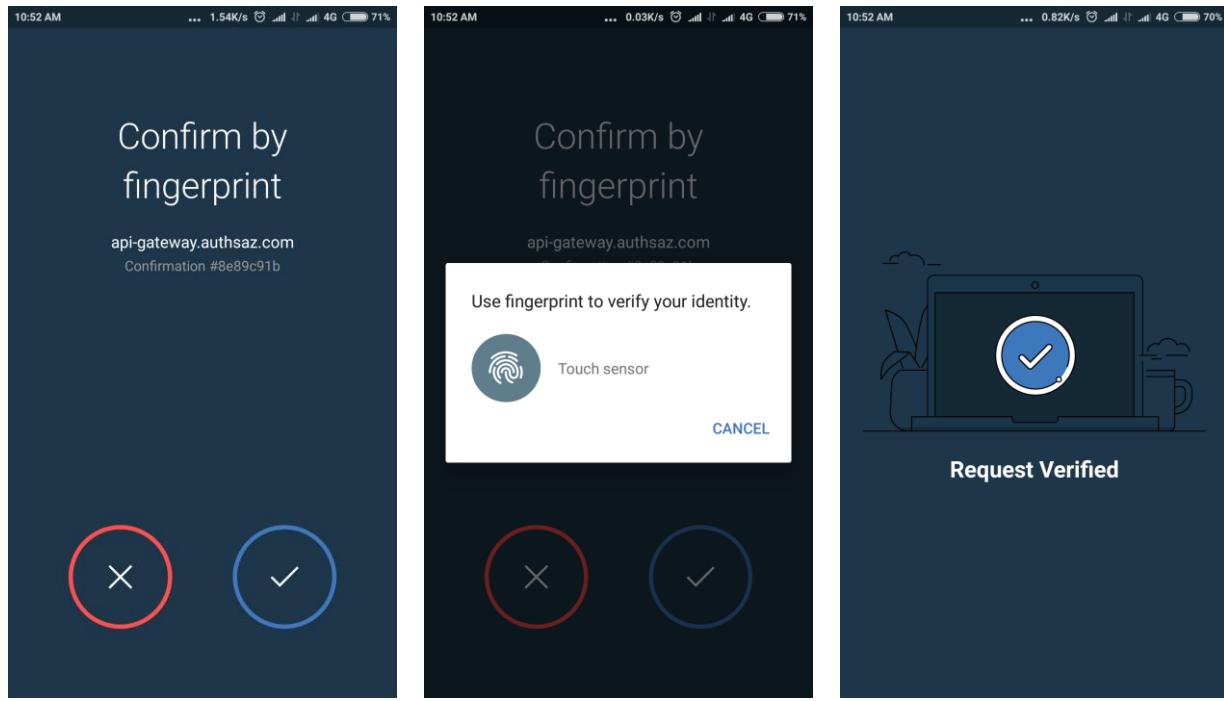


The **TOTP One-Time Password Login** popup will appear. Read TOTP value from *IBM Verify* and submit the form.



The **Mobile Multi Factor Device Selection** page will appear. Choose your registered device, and submit the form.

Open the *IBM Verify* application and touch the refresh icon in the top right corner.



Once you see the blue dot on *IBM Verify*, touch the account to open it. You should immediately be prompted with the pending transaction.

Use your fingerprint to verify the transaction.



After successful authentication, the API result will be shown in a pop-up window.

8 Context-Based API Protection

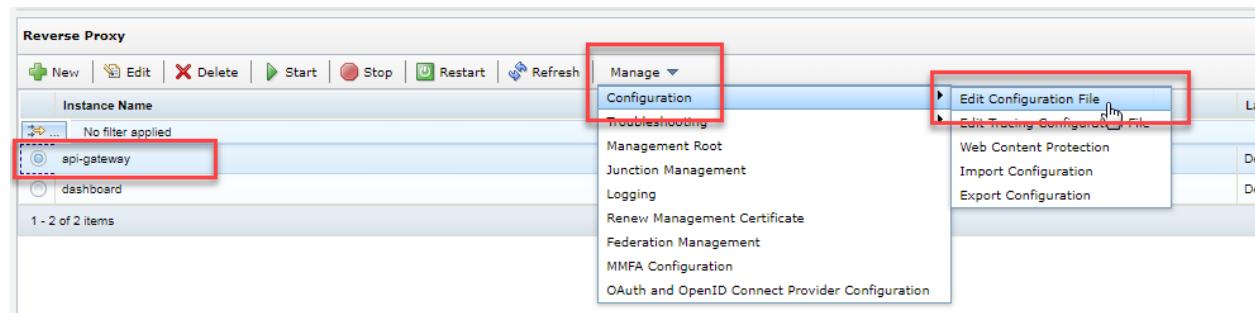
As we discussed in previous chapters, we can define a static access control policy and apply it to an API. Most of the access scenarios in the real world require more granularity for doing an access decision. As an example in our *Balance API*, the Merchant must not be allowed to get the balance of **user2** with **user1** Access Token or merchant with a **non-financial** Access Token must not be allowed to access **financial** APIs. For implementing these access policies ISAM has introduced context-based access control policies.

In this chapter, we focus on context based access control policies to protect APIs. These scenarios will be discussed in this chapter:

- Fine-grained access control based on the *Access Token* username
- Fine-grained access control based on the *Account* value
- Fine-grained access control based on the *Amount* value
- Fine-grained access control based on *OAuth Scope*.

8.1 Modify Reverse Proxy Instance Configuration File

Navigate to **Secure Web Settings > Manage: Reverse Proxy**



Select the radio button for the **api-gateway** Reverse Proxy instance. Click on Manage and select **Configuration > Edit Configuration File** from the pop-up menu.

In the [azn-decision-info] stanza add the following entry highlighted in red:

```
[azn-decision-info]
urn:authsaz:request:username = post-data:"username"
urn:authsaz:request:account = post-data:"account"
urn:authsaz:request:amount = post-data:"amount"
```

In the [user-attribute-definitions] stanza add the following entry highlighted in red:

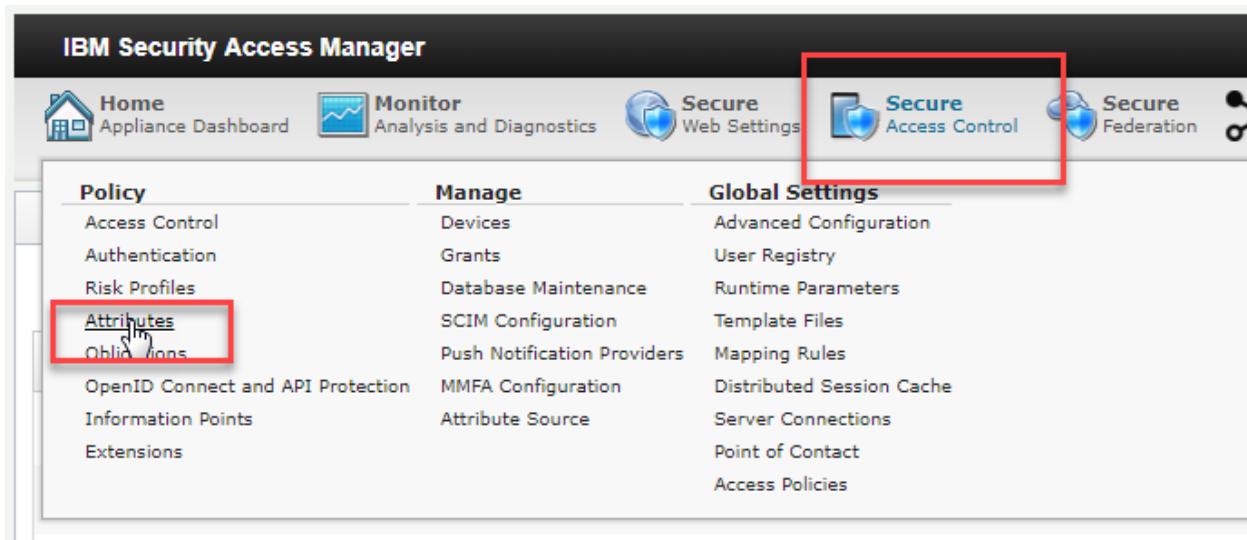
```
[user-attribute-definitions]
```

```
urn:authsaz:request:username.datatype = string
urn:authsaz:request:username.category = Subject
urn:authsaz:request:account.datatype = string
urn:authsaz:request:account.category = Subject
urn:authsaz:request:amount.datatype = double
urn:authsaz:request:amount.category = Subject
```

Click **Save**.

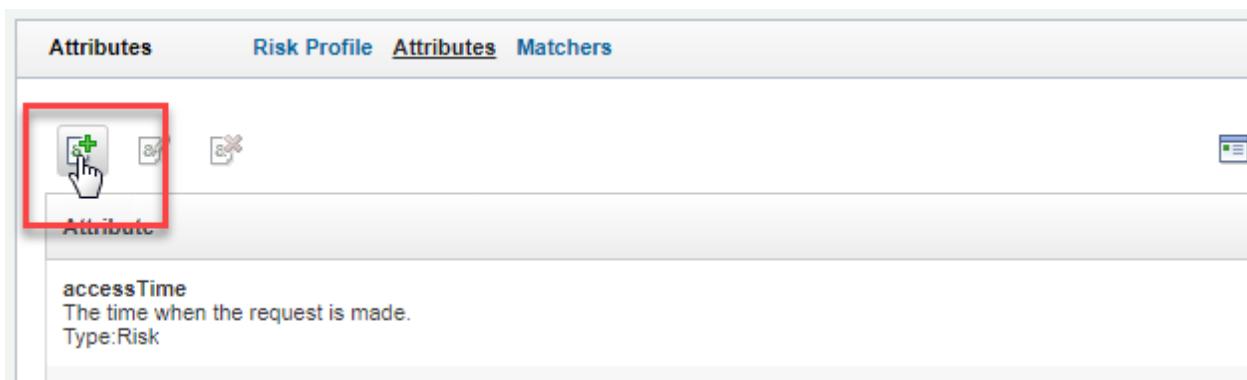
Deploy changes and **Restart** the Reverse Proxy instance.

8.2 Define Attributes



The screenshot shows the IBM Security Access Manager interface. At the top, there are several navigation links: Home (Appliance Dashboard), Monitor (Analysis and Diagnostics), Secure Web Settings, Secure Access Control (which is highlighted with a red box), and Secure Federation. Below the navigation bar, there are three main sections: Policy, Manage, and Global Settings. The Policy section contains links for Access Control, Authentication, Risk Profiles, Attributes (which is also highlighted with a red box), OAuth 2.0, OpenID Connect and API Protection, Information Points, and Extensions. The Manage section contains links for Devices, Grants, Database Maintenance, SCIM Configuration, Push Notification Providers, MMFA Configuration, Attribute Source, and Global Settings. The Global Settings section contains links for Advanced Configuration, User Registry, Runtime Parameters, Template Files, Mapping Rules, Distributed Session Cache, Server Connections, Point of Contact, and Access Policies.

Navigate to **Secure Access Control > Policy: Attributes**.



The screenshot shows the 'Attributes' tab in the Secure Access Control interface. At the top, there are tabs for Attributes (which is highlighted with a red box), Risk Profile, Attributes, and Matchers. Below the tabs, there is a toolbar with icons for New Attribute (highlighted with a red box), Edit, and Delete. A table lists attributes, with one row shown in detail: accessTime, The time when the request is made, Type:Risk. There is a 'New Attribute' button at the bottom of the list.

Select the “**New Attribute**” icon as indicated above.

New Attribute

Name:	input_username
Identifier:	urn:authsaz:request:username
Description:	
Issuer:	
Category:	Subject
Data Type:	String
Matcher:	exact_match
Type:	<input checked="" type="checkbox"/> Policy <input type="checkbox"/> Risk
Storage Domain:	<input type="checkbox"/> Device <input type="checkbox"/> Session <input type="checkbox"/> Behavior
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Enter **input_username** as the *Name*. This is how the attribute will be displayed in the console.

Enter **urn:authsaz:request:username** as the *Identifier*, select **Subject** as the *Category* and **String** as the *Data Type*. These values need to match what was configured in the Reverse Proxy.

Select the **Policy** checkbox. This indicates that this attribute should be made available for use in the policy editor.

Press the **Save** button to create the new attribute.

Repeat steps above to add the following attributes:

New Attribute

Name:

Identifier:

Description:

Issuer:

Category:

Data Type:

Matcher:

Type: Policy Risk

Storage Domain: Device Session Behavior

Enter **input_account** as the *Name*. Enter **urn:authsaz:request:account** as the *Identifier*, select **Subject** as the *Category* and **String** as the *Data Type*. Select the **Policy** checkbox.

Press the **Save** button to create the new attribute.

New Attribute

Name:	input_amount
Identifier:	urn:authsaz:request:amount
Description:	
Issuer:	
Category:	Subject
Data Type:	Double
Matcher:	exact_match
Type:	<input checked="" type="checkbox"/> Policy <input type="checkbox"/> Risk
Storage Domain:	<input type="checkbox"/> Device <input type="checkbox"/> Session <input type="checkbox"/> Behavior
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Enter **input_amount** as the *Name*. Enter **urn:authsaz:request:amount** as the *Identifier*, select **Subject** as the *Category* and **Double** as the *Data Type*. Select the **Policy** checkbox.

Press the **Save** button to create the new attribute.

Finally, **Deploy** the changes to the runtime. This needs to be done so that the attributes will be available in the policy editor.

8.3 OAuth Information Point

We need to extract some data from provided access token. Philip Nye has a blog post¹⁰ about how extracting the OAuth values of scope and supplies them to the CBA engine as a multi-valued attribute. We customized his PIP code to extract OAuth Scope, client-id and username from an access token.

Here are the steps to create our Information Point:

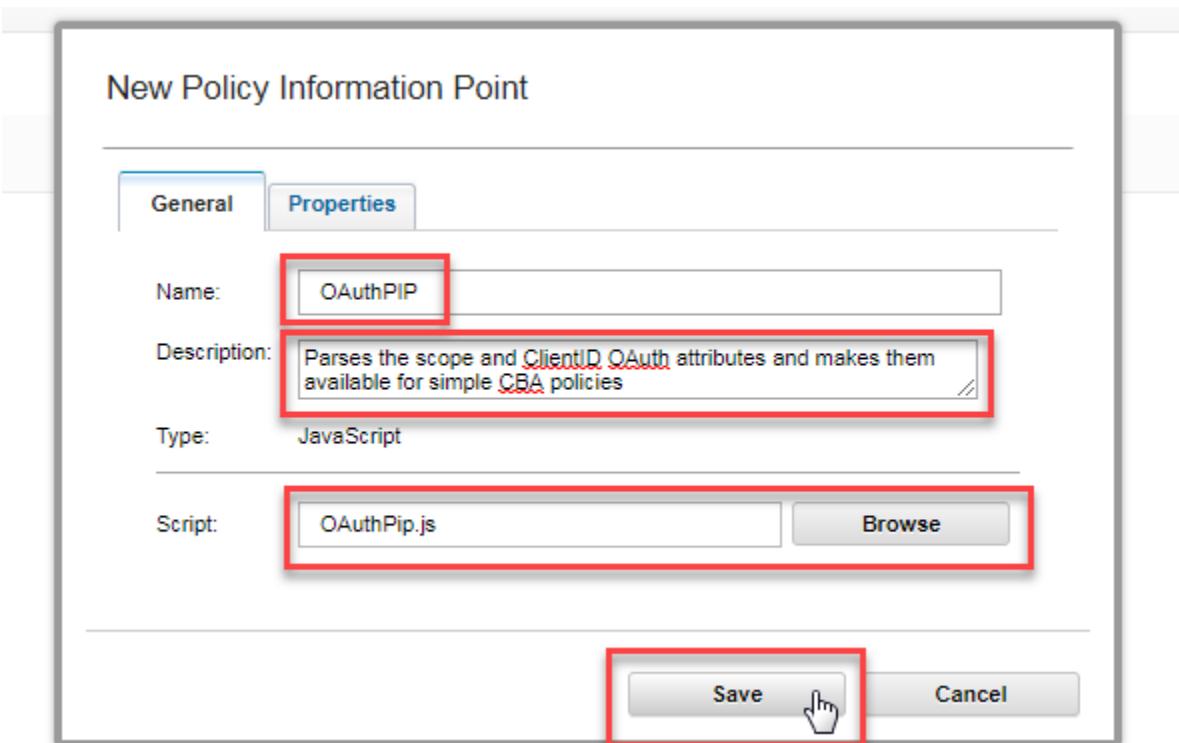
The screenshot shows the IBM Security Access Manager interface. At the top, there is a navigation bar with links for Home, Monitor, Secure Web Settings, Secure Access Control (which is highlighted with a red box), and Federate. Below the navigation bar is a main menu with three columns: Policy, Manage, and Global Settings. The Policy column contains links for Access Control, Authentication, Risk Profiles, Attributes, Obligations, OpenID Connect and API Protection, Information Points (which is highlighted with a red box and has a mouse cursor over it), and Extensions. The Manage column contains links for Devices, Grants, Database Maintenance, SCIM Configuration, Push Notification Providers, MMFA Configuration, Attribute Source, and Global Settings. The Global Settings column contains links for Advanced Configuration, User Registry, Runtime Parameters, Template Files, Mapping Rules, Distributed Session Cache, Server Connections, Point of Contact, and Access Policies.

Navigate to **Secure Access Control > Policy: Information Points**.

The screenshot shows the 'Policy Information Points' screen. At the top, there is a title bar with the text 'Policy Information Points'. Below the title bar is a toolbar with icons for New, Edit, Delete, and Import/Export. A dropdown menu is open under the 'New' icon, showing options: RESTful Web Service, JavaScript (which is highlighted with a blue selection bar and has a mouse cursor over it), Database, LDAP, FiberLink MaaS360, and QRadar UBA. The main area of the screen displays a table with columns for Name, Description, and Action. There are two rows visible in the table.

Click on **New** icon and select **JavaScript** from the drop-down menu.

¹⁰ <https://philipnye.com/2015/05/15/isam-context-based-access-pip-for-oauth/>



Enter **OAuthPIP** as *Name* and provide a *Description*.

Click **Browse**. From **isambookdemo-infomaps** directory select **OAuthPip.js**.

Click **Save**.

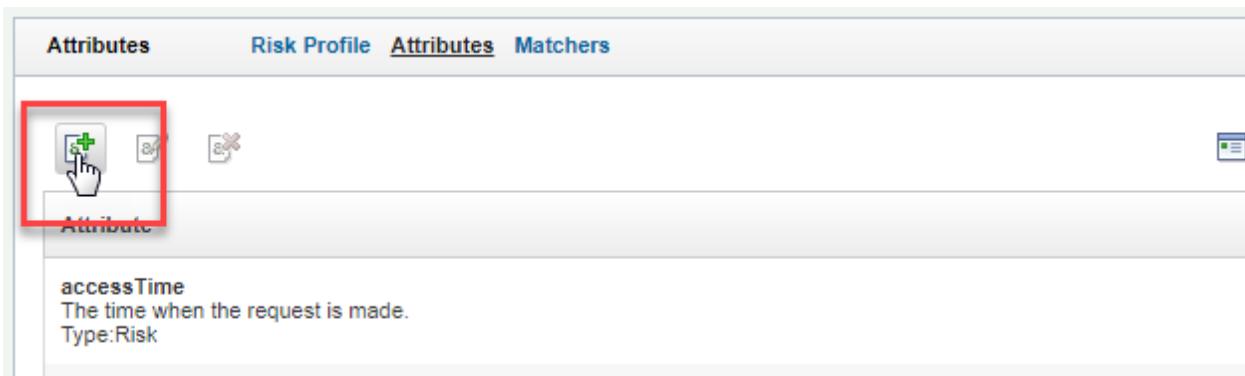
Deploy pending changes.

8.3.1 OAuth Attributes



The screenshot shows the IBM Security Access Manager dashboard. At the top, there are several navigation links: Home, Monitor, Secure Web Settings, Secure Access Control (which is highlighted with a red box), and Secure Federation. Below the navigation bar, there are three main sections: Policy, Manage, and Global Settings. The Policy section contains links for Access Control, Authentication, Risk Profiles, Attributes (which is highlighted with a red box), Obligations, OpenID Connect and API Protection, Information Points, and Extensions. The Manage section contains links for Devices, Grants, Database Maintenance, SCIM Configuration, Push Notification Providers, MMFA Configuration, Attribute Source, and Advanced Configuration. The Global Settings section contains links for User Registry, Runtime Parameters, Template Files, Mapping Rules, Distributed Session Cache, Server Connections, Point of Contact, and Access Policies.

Navigate to **Secure Access Control > Policy: Attributes**.



The screenshot shows the 'Attributes' tab in the Secure Access Control interface. At the top, there are tabs for Attributes, Risk Profile, Attributes, and Matchers. Below the tabs, there is a toolbar with icons for New Attribute (highlighted with a red box), Edit, and Delete. A table lists attributes, with one row visible: 'accessTime' (Type:Risk). The 'accessTime' row includes a 'Description' column with the text 'The time when the request is made.' and a 'Type' column with 'Type:Risk'.

Select the “**New Attribute**” icon as indicated above.

New Attribute

Name:	OAuth Scope
Identifier:	urn:ibm:security:iam:oauth:scope
Description:	(empty)
Issuer:	OAuthPIP
Category:	Subject
Data Type:	String
Matcher:	exact_match
Type:	<input checked="" type="checkbox"/> Policy <input type="checkbox"/> Risk
Storage Domain:	<input type="checkbox"/> Device <input type="checkbox"/> Session <input type="checkbox"/> Behavior
<input style="border: 1px solid red;" type="button" value="Save"/> <input type="button" value="Cancel"/>	

Enter **OAuth Scope** as the *Name*.

Enter **urn:ibm:security:iam:oauth:scope** as the *Identifier*.

Enter **OAuthPIP** as the *Issuer*.

Select **Subject** as the *Category* and **String** as the *Data Type*. Select the **Policy** checkbox.

Press the **Save** button to create the new attribute.

Repeat steps above to add the following attributes:

New Attribute

Name:	OAuth ClientID
Identifier:	urn:ibm:security:iam:oauth:client:id
Description:	
Issuer:	OAuthPIP
Category:	Subject
Data Type:	String
Matcher:	exact_match
Type:	<input checked="" type="checkbox"/> Policy <input type="checkbox"/> Risk
Storage Domain:	<input type="checkbox"/> Device <input type="checkbox"/> Session <input type="checkbox"/> Behavior
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Enter **OAuth ClientID** as the *Name*.

Enter **urn:ibm:security:iam:oauth:client:id** as the *Identifier*.

Enter **OAuthPIP** as the *Issuer*.

Select **Subject** as the *Category* and **String** as the *Data Type*. Select the **Policy** checkbox.

Press the **Save** button to create the new attribute.

New Attribute

Name:	OAuth username
Identifier:	urn:ibm:security:iam:oauth:username
Description:	
Issuer:	OAuthPIP
Category:	Subject
Data Type:	String
Matcher:	exact_match
Type:	<input checked="" type="checkbox"/> Policy <input type="checkbox"/> Risk
Storage Domain:	<input type="checkbox"/> Device <input type="checkbox"/> Session <input type="checkbox"/> Behavior
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Enter **OAuth username** as the *Name*.

Enter **urn:ibm:security:iam:oauth:username** as the *Identifier*.

Enter **OAuthPIP** as the *Issuer*.

Select **Subject** as the *Category* and **String** as the *Data Type*. Select the **Policy** checkbox.

Press the **Save** button to create the new attribute.

Finally, **Deploy** the changes to the runtime. This needs to be done so that the attribute will be available in the policy editor.

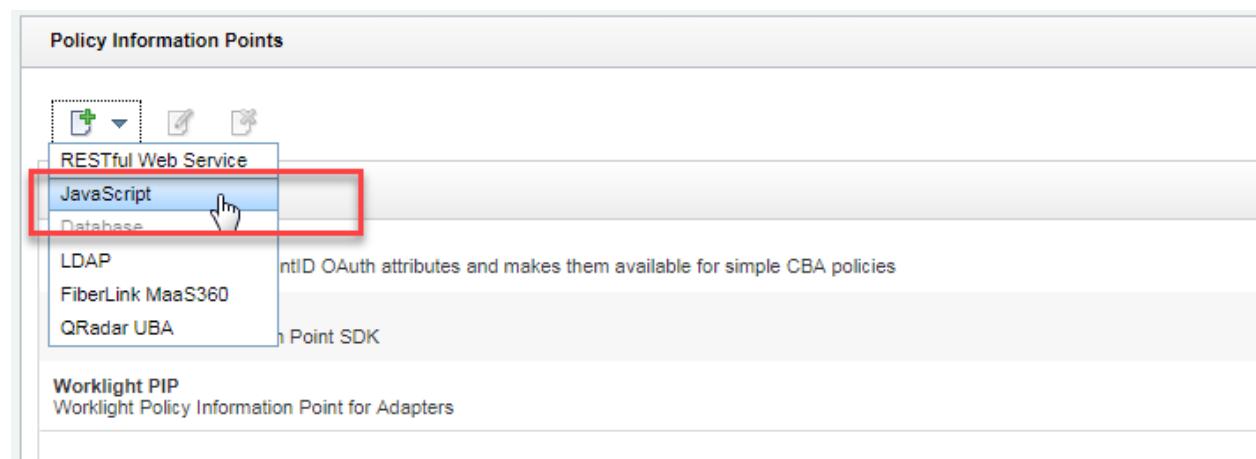
8.4 Account List Information Point

We need an extra information point to list all accounts related to a username. This PIP calls backend service **/api/nf/accounts** from API Server to retrieve this information.



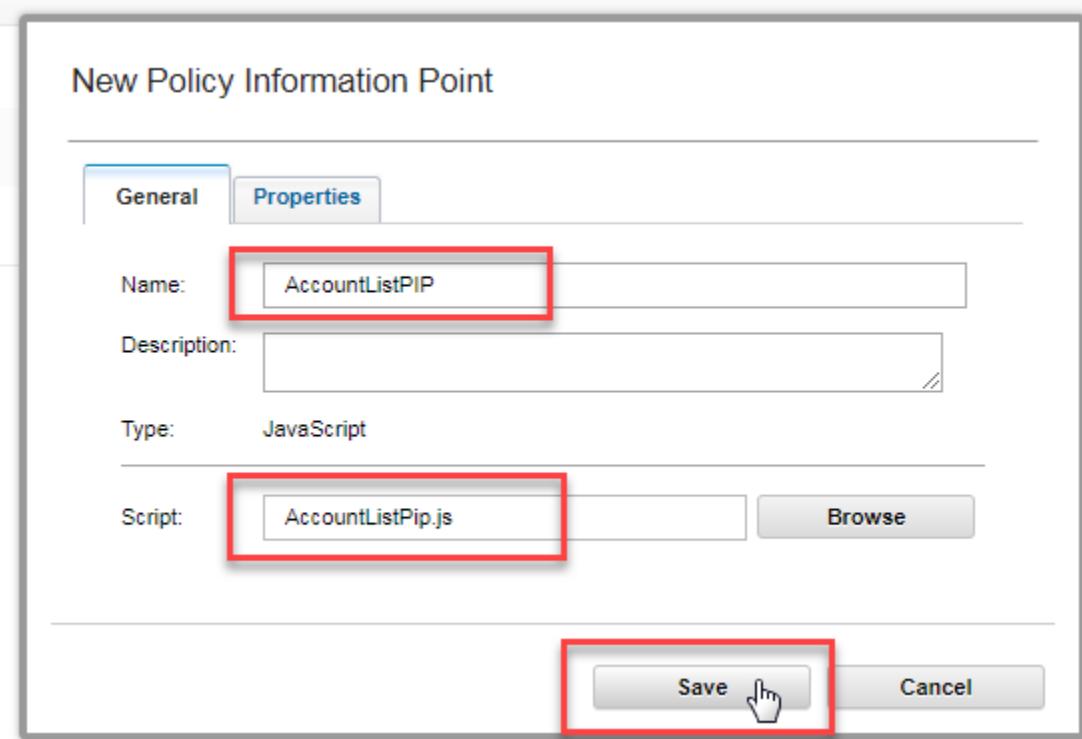
The screenshot shows the IBM Security Access Manager dashboard. At the top, there are several navigation links: Home (Appliance Dashboard), Monitor (Analysis and Diagnostics), Secure Web Settings, Secure Access Control (which is highlighted with a red box), and Secure Federated Identity. Below the navigation bar, there are three main sections: Policy, Manage, and Global Settings. The Policy section contains links for Access Control, Authentication, Risk Profiles, Attributes, Obligations, OpenID Connect and API Protection, Information Points (which is highlighted with a red box), and Extensions. The Manage section contains links for Devices, Grants, Database Maintenance, SCIM Configuration, Push Notification Providers, MMFA Configuration, Attribute Source, and Attribute Source. The Global Settings section contains links for Advanced Configuration, User Registry, Runtime Parameters, Template Files, Mapping Rules, Distributed Session Cache, Server Connections, Point of Contact, and Access Policies.

Navigate to **Secure Access Control > Policy: Information Points**.



The screenshot shows the "Policy Information Points" screen. At the top, there is a toolbar with icons for New, Edit, and Delete. Below the toolbar, there is a dropdown menu labeled "RESTful Web Service". Under this dropdown, the "JavaScript" option is highlighted with a red box and has a cursor icon pointing at it. Other options in the dropdown include "Database" and "LDAP". Below the dropdown, there is a list of other PIPs: "FiberLink MaaS360", "QRadar UBA", and "Worklight PIP". The "Worklight PIP" entry includes a note: "Worklight Policy Information Point for Adapters".

Click on **New** icon and select **JavaScript** from the drop-down menu.



Enter **AccountListPIP** as *Name* and provide a *Description*.

Click **Browse**. From **isambookdemo-infomaps** directory select **AccountListPip.js**.

Click **Save**.

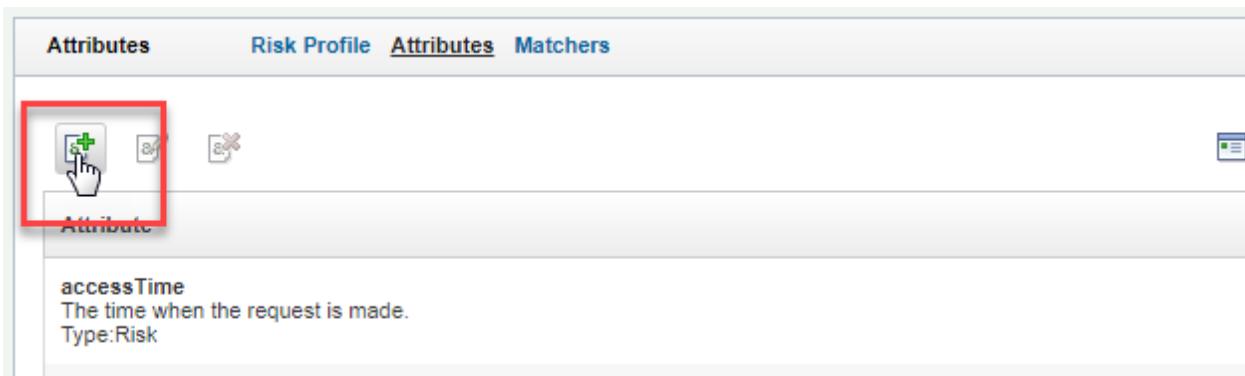
Deploy pending changes.

8.4.1 AccountList Attribute



The screenshot shows the IBM Security Access Manager dashboard. At the top, there are several navigation links: Home, Monitor, Secure Web Settings, Secure Access Control (which is highlighted with a red box), and Secure Federation. Below the navigation bar, there are three main sections: Policy, Manage, and Global Settings. The Policy section contains links for Access Control, Authentication, Risk Profiles, Attributes (which is also highlighted with a red box), Obligations, OpenID Connect and API Protection, Information Points, and Extensions. The Manage section contains links for Devices, Grants, Database Maintenance, SCIM Configuration, Push Notification Providers, MMFA Configuration, Attribute Source, and Advanced Configuration. The Global Settings section contains links for User Registry, Runtime Parameters, Template Files, Mapping Rules, Distributed Session Cache, Server Connections, Point of Contact, and Access Policies.

Navigate to **Secure Access Control > Policy: Attributes**.



The screenshot shows the 'Attributes' tab in the Secure Access Control interface. At the top, there are tabs for Attributes, Risk Profile, Attributes, and Matchers. Below the tabs, there is a toolbar with icons for creating a new attribute (highlighted with a red box), deleting an attribute, and other actions. A table lists attributes, with one row visible: 'accessTime' (Type:Risk). The 'accessTime' row includes a description: 'The time when the request is made.' and a note: 'Type:Risk'.

Select the “**New Attribute**” icon as indicated above.

New Attribute

Name:

Identifier:

Description:

Issuer:

Category:

Data Type:

Matcher:

Type: Policy Risk

Storage Domain: Device Session Behavior



Enter **AccountList** as the *Name*.

Enter **urn:authsaz:attr:accountlist** as the *Identifier*.

Enter **AccountListPIP** as the *Issuer*.

Select **Subject** as the *Category* and **String** as the *Data Type*. Select the **Policy** checkbox.

Press the **Save** button to create the new attribute.

Deploy the changes to the runtime.

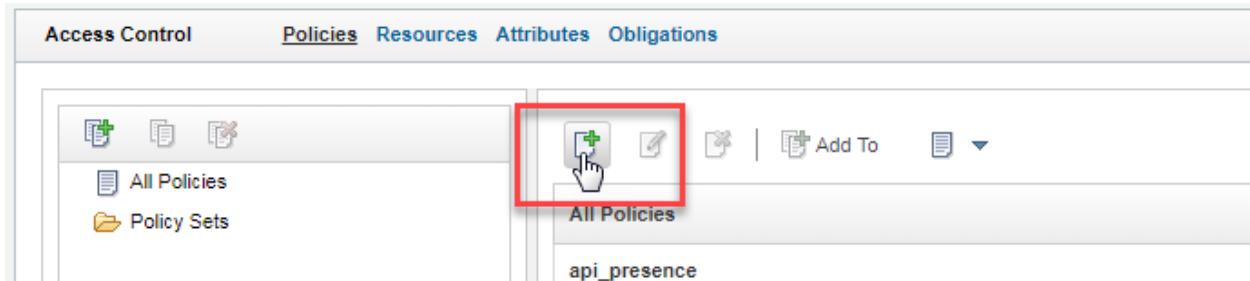
8.5 Define Context based Policies

8.5.1 Create cba_accounts policy



The screenshot shows the IBM Security Access Manager interface. The top navigation bar includes links for Home, Monitor, Secure Web Settings, **Secure Access Control** (which is highlighted with a red box), Secure Federation, and Connect IBM Cloud. Below the navigation bar is a main menu with three columns: Policy, Manage, and Global Settings. The Policy column has a sub-menu under 'Access Control' (also highlighted with a red box) which includes Authentication, Risk Profiles, Attributes, Obligations, OpenID Connect and API Protection, Information Points, and Extensions. The Manage column includes Devices, Grants, Database Maintenance, SCIM Configuration, Push Notification Providers, MMFA Configuration, and Attribute Source. The Global Settings column includes Advanced Configuration, User Registry, Runtime Parameters, Template Files, Mapping Rules, Distributed Session Cache, Server Connections, Point of Contact, and Access Policies.

Navigate to **Secure Access Control > Policy: Access Control**.



The screenshot shows the 'Policies' tab in the Secure Access Control interface. The top navigation bar includes Access Control, Policies (which is selected and highlighted in blue), Resources, Attributes, and Obligations. On the left, there's a sidebar with icons for All Policies and Policy Sets. The main area shows a list of policies with one item named 'api_presence'. At the top of the main area, there are several buttons: a plus sign icon (highlighted with a red box), a pencil icon, a magnifying glass icon, an 'Add To' button, and a dropdown arrow. The plus sign icon is the 'Add Policy' button.

Click the **Add Policy** button in the main window of the *Policies* tab.

Screenshot of the Policy Management interface showing the creation of a new policy.

The top navigation bar includes Access Control, Policies (which is selected), Resources, Attributes, and Obligations. Below the navigation are Save and Cancel buttons. The main form has fields for Name (cba_accounts) and Description, both of which are currently empty. A red box highlights the Name field.

The policy structure section shows a Subjects tree with an Add Subject button. The Rules section is expanded, showing Precedence set to Deny (with a dropdown menu showing Deny, Permit, and First), and Attributes set to Optional. The Rules list is currently empty, with an Add Rule button. A red box highlights the Precedence dropdown menu, which is open and shows the First option selected.

Enter **cba_accounts** as the *Name*. Enter a *description*.

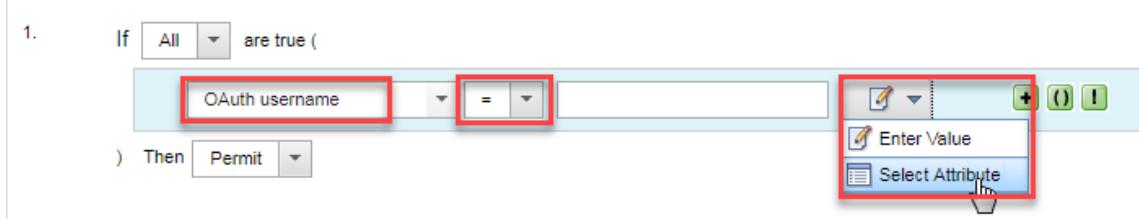
Select **First** from the *Precedence* drop-down list. This means that the rules of the policy will be evaluated in order. As soon as a rule returns a result, that result is returned.

Screenshot of the Rules configuration screen.

The top bar shows Rules (0), Precedence: First, and Attributes: Optional. The Add Rule button is highlighted with a red box. Below it, a dropdown menu shows Conditional rule selected (highlighted with a red box) and Unconditional rule.

Click **Add Rule**.

Select **Conditional rule** from the drop-down list.



In the first rule, we will check to see if the **OAuth username** attribute is equal to **input_username** attribute.

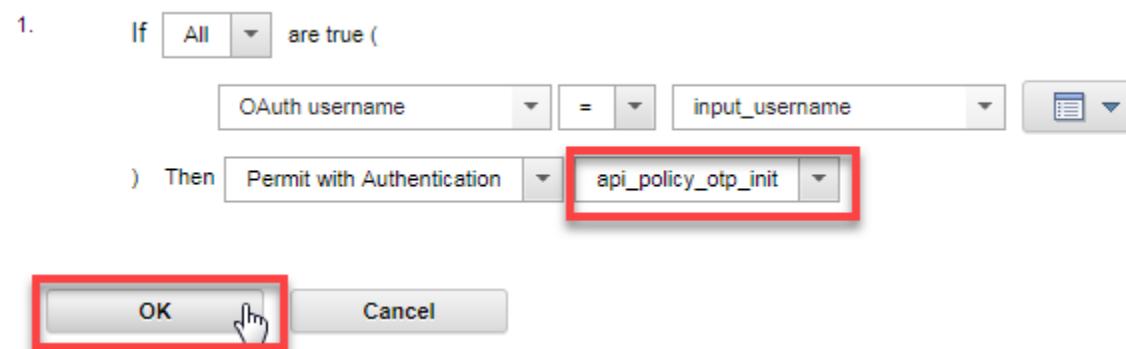
Select **OAuth username** from the drop-down list. This is available because we defined it as an attribute to be used in Policies in section 8.3.1.

Select = as the comparison operator. This is available because the type of the attribute is String.



Select **input_username** as the value for comparison.

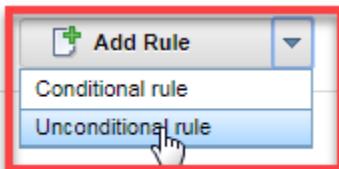
For the result, select **Permit with Authentication** from the drop-down list. This means we're going to permit access but only once an authentication policy has been successfully completed.



Select **api_policy_otp_init** from the drop-down list of authentication mechanisms.

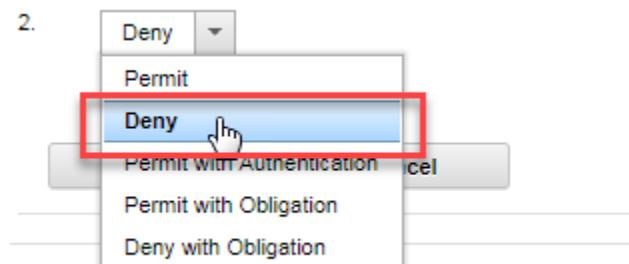
Click **OK**. The rule is closed.

1. If OAuth username = input_username
Then Permit with Authentication api_policy_otp_init



Click **Add Rule** again to add the second rule.

Select **Unconditional rule** from the drop-down list.



Select **Deny** from the drop-down list.

Click **Ok**.

A screenshot of a software interface showing a list of rules. The rules are:

1. If OAuth username = input_username
Then Permit with Authentication api_policy_otp_init
2. Deny

The 'Add Rule' button is visible at the bottom left.

Check that the policy looks as shown above.

Screenshot of the 'Access Control' interface showing the 'Policies' tab selected. A red box highlights the 'Save' button, which has a mouse cursor pointing at it. Below the button are fields for 'Name' (cba_accounts) and 'Description'. The 'Access Control' menu bar is visible at the top.

Click **Save** button at the top of the window to save the Policy. The new policy is shown in the policy list:

Screenshot of the 'Access Control' interface showing the 'Policies' tab selected. The left sidebar shows 'All Policies' and 'Policy Sets'. The main area displays a list of policies: api_presence, api_otp_finger, api_totp, and cba_accounts. The 'cba_accounts' item is highlighted with a red box and has a mouse cursor pointing at it. The 'Access Control' menu bar is visible at the top.

8.5.2 Create cba_balance policy

Screenshot of the 'Access Control' interface showing the 'Policies' tab selected. The left sidebar shows 'All Policies' and 'Policy Sets'. The main area displays a list of policies: api_presence. A red box highlights the 'Add Policy' button (a green plus icon) in the toolbar above the list. The 'Access Control' menu bar is visible at the top.

Click the **Add Policy** button in the main window of the *Policies* tab.

Name: cba_balance

Description:

Subjects

Add Subject

Rules Precedence: First Attributes: Optional

Add Rule

Conditional rule

Unconditional rule

Enter **cba_balance** as the *Name*. Enter a *description*.

Select **First** from the *Precedence* drop-down list.

Click **Add Rule**.

Select **Conditional rule** from the drop-down list.

1. If All are true (

2. has member

3. input_account

4. Add

5. Permit with Authentication

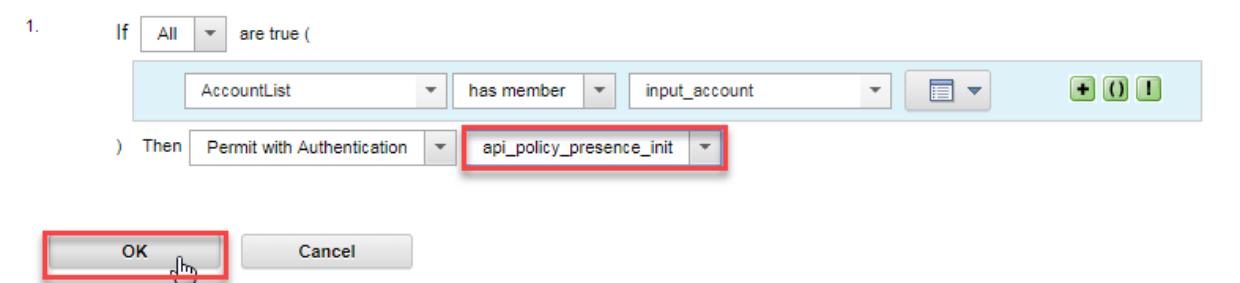
In the first rule, we will check to see if the **input_account** attribute is in **AccountList** attribute. If so, then we will permit access with further authentication.

Select **AccountList** from the drop-down list.

Select **has member** as the comparison operator.

Select **input_account** as the value for comparison.

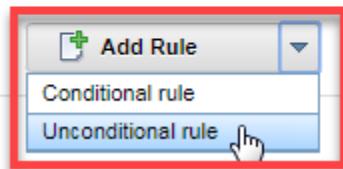
For the result, select **Permit with Authentication** from the drop-down list.



Select **api_policy_presence_init** from the drop-down list of authentication mechanisms.

Click **OK**. The rule is closed.

1. If AccountList has member input_account
Then Permit with Authentication api_policy_presence_init



Click **Add Rule** again to add the second rule.

Select **Unconditional rule** from the drop-down list.



Select **Deny** from the drop-down list.

Click **Ok**.

The screenshot shows a policy configuration interface with the following details:

- Rules (2)**: Precedence: First
- Attributes: Optional**
- Rule 1:** If AccountList has member input_account
Then Permit with Authentication api_policy_presence_init
- Rule 2:** Deny

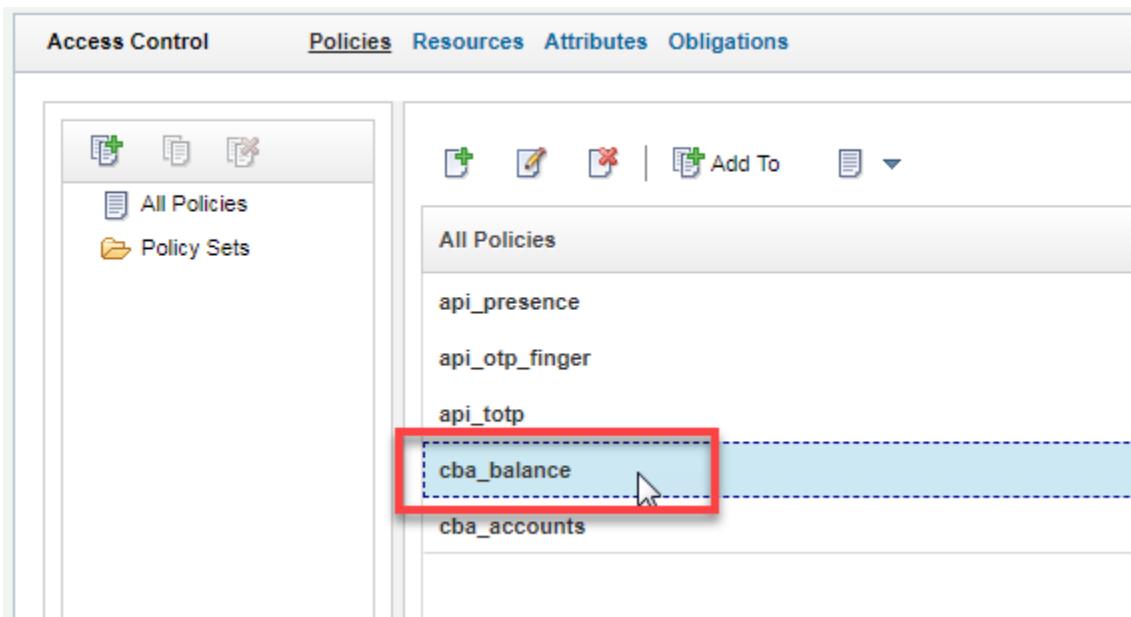
At the bottom left is a button labeled "Add Rule".

Check that the policy looks as shown above.

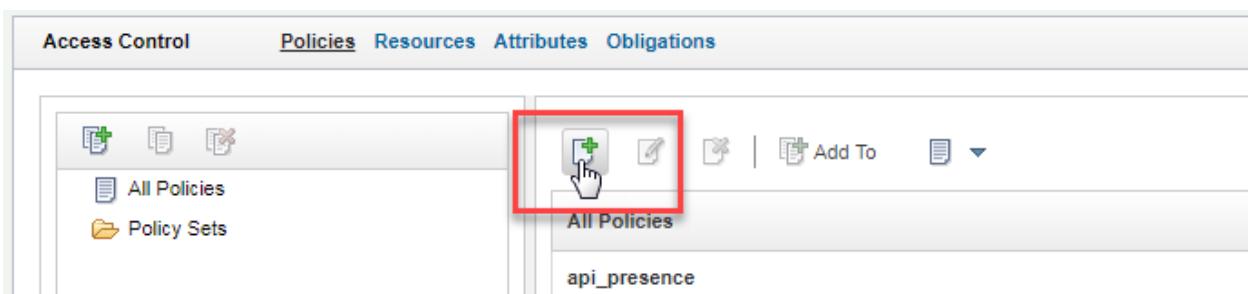
The screenshot shows a dialog box for saving a policy:

- Access Control** tab is selected.
- Policies**, **Resources**, **Attributes**, and **Obligations** tabs are also present.
- Save** button is highlighted with a red box and a cursor icon.
- Cancel** button is also visible.
- Name:** cba_balance

Click **Save** button at the top of the window to save the Policy. The new policy is shown in the policy list:



8.5.3 Create cba_transfer policy



Click the **Add Policy** button in the main window of the *Policies* tab.

The screenshot shows the Oracle Access Control interface. At the top, there are tabs for 'Access Control', 'Policies' (which is selected), 'Resources', 'Attributes', and 'Obligations'. Below the tabs are 'Save' and 'Cancel' buttons. The main area has a form with 'Name:' set to 'cba_transfer' (highlighted with a red box) and an empty 'Description:' field. A 'Subjects' section contains an 'Add Subject' button. Under 'Rules (0)', there is a 'Precedence: First' dropdown (highlighted with a red box) and an 'Add Rule' button. A dropdown menu shows 'Conditional rule' (highlighted with a red box) and 'Unconditional rule'.

Enter **cba_transfer** as the *Name*. Enter a *description*.

Select **First** from the *Precedence* drop-down list.

Click **Add Rule**.

Select **Conditional rule** from the drop-down list.

This screenshot shows the 'Conditional rule' configuration dialog. It consists of several input fields and buttons. Red numbers 1 through 7 are overlaid on the interface to indicate specific steps:

- 1: 'If' dropdown set to 'All'.
- 2: 'Then' dropdown set to 'Deny'.
- 3: 'input_account' dropdown.
- 4: 'has member' dropdown.
- 5: 'AccountList' dropdown.
- 6: 'are true (' dropdown.
- 7: 'OK' button at the bottom left.

In the first rule, we will check to see if the **input_account** attribute is not in **AccountList** attribute. If so, then we will deny access without requiring further action.

Select **AccountList** from the drop-down list.

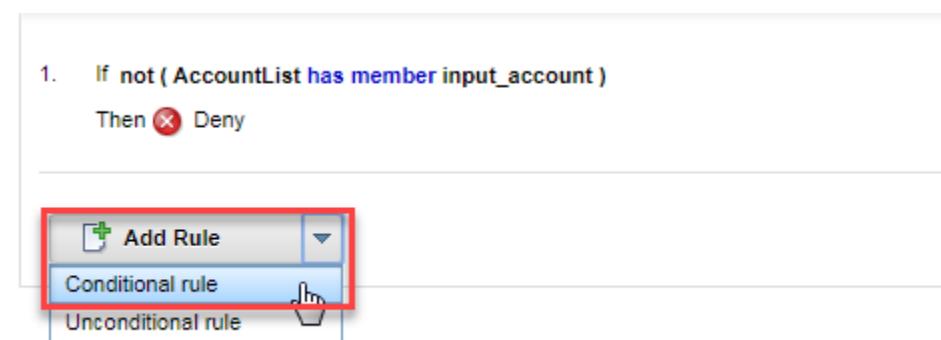
Select **has member** as the comparison operator.

Select **input_account** as the value for comparison.

Click **negate** icon in the right corner of the rule editor.

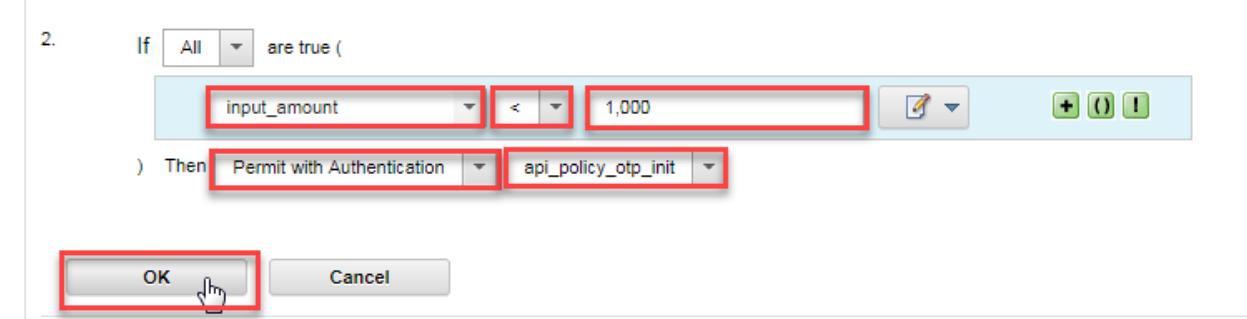
Select **deny** for the result for this rule.

Click **OK**.



Click **Add Rule** again to add the second rule.

Select **Conditional rule** from the drop-down list.



Select **input_account** from the drop-down list.

Select **<** as the comparison operator.

Enter **1000** as the value for comparison.

For the result, select **Permit with Authentication** from the drop-down list. This means we're going to permit access but only once an authentication policy has been successfully completed.

Select **api_policy_otp_init** from the drop-down list of authentication mechanisms.

Click **OK**.

1. If not (AccountList has member input_account)
Then Deny

2. If input_amount < 1,000
Then Permit with Authentication api_policy_otp_init

Click **Add Rule** again to add the third rule.

Select **Conditional rule** from the drop-down list.

3. If All are true (input_amount >= 1,000)
) Then Permit with Authentication api_policy_otp_finger_init

Select **input_account** from the drop-down list.

Select **>=** as the comparison operator.

Enter **1000** as the value for comparison.

For the result, select **Permit with Authentication** from the drop-down list.

Select **api_policy_otp_finger_init** from the drop-down list of authentication mechanisms.

Click **OK**.

Rules (3) Precedence: First Attributes: Optional

1. If not (AccountList has member input_account)
Then Deny
2. If input_amount < 1,000
Then Permit with Authentication api_policy_otp_init
3. If input_amount >= 1,000
Then Permit with Authentication api_policy_otp_finger_init

Add Rule Conditional rule Unconditional rule

Click **Add Rule** again to add the fourth rule.

Select **Unconditional rule** from the drop-down list.

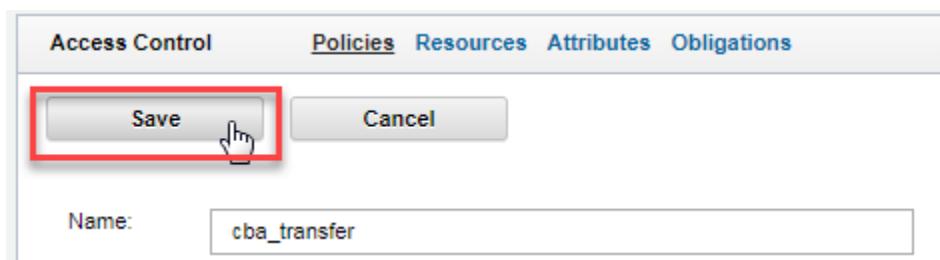


Select **Deny** from the drop-down list.

Click **Ok**.

1. If not (AccountList has member input_account)
Then Deny
2. If input_amount < 1,000
Then Permit with Authentication api_policy_otp_init
3. If input_amount >= 1,000
Then Permit with Authentication api_policy_otp_finger_init
4. Deny

Check that the policy looks as shown above.



Click **Save** button at the top of the window to save the Policy. The new policy is shown in the policy list:

The screenshot shows the Policies tab of an access control interface. On the left, there is a sidebar with icons for creating, editing, and deleting policies, and links for 'All Policies' and 'Policy Sets'. The main area displays a list of policies under 'All Policies': 'api_presence', 'api_otp_finger', 'api_totp', 'cba_balance', 'cba_transfer' (which is highlighted with a red box), and 'cba_accounts'. A cursor is pointing at 'cba_transfer'.

8.5.4 Attach Context-Based Policies to Resources

Select the **Resources** tab.

The screenshot shows the Resources tab of the access control interface. At the top, there are buttons for creating, editing, deleting, attaching, publishing, publishing all, and changing domain. The 'Resources' tab is selected. Below it, there is a tree view of resources: 'isam.authsaz.com-api-gateway' has children '/api', '/api/f/transfer', and three more items that are currently expanded. These expanded items are 'api_otp_finger', 'api_totp', and 'api_presence', each highlighted with a red box. The 'Attach' button in the toolbar is also highlighted with a red box.

Remove all previously attached policies to the resources.

The screenshot shows the 'Access Control' interface with the 'Resources' tab selected. In the toolbar, the 'Attach' button is highlighted with a red box. Below the toolbar, a tree view shows a web container 'isam.authsaz.com-api-gateway' containing several resources: '/api', '/api/f/transfer', '/api/nf/balance', and '/api/nf/get_account'. The resource '/api/f/transfer' is also highlighted with a red box.

Select the **/api/f/transfer** resource and click **Attach**.

The screenshot shows the 'Attach Policies' dialog box. At the top, it displays 'Web Container: isam.authsaz.com-api-gateway' and 'Resource: /api/f/transfer'. Below this, there are three radio buttons: 'Policies' (selected), 'Policy Sets', and 'API Protection'. A search bar contains the text 'cba_tr'. A list of policies is shown, with one policy named 'cba_transfer' selected and highlighted with a red box. At the bottom of the dialog box, the 'OK' button is highlighted with a red box.

Select the checkbox next to the **cba_transfer** and press the **OK** button to attach the specified policy to the resource.

Repeat steps above to attach **cba_balance** to **/api/nf/balance** and attach **cba_accounts** to **/api/nf/accounts**.

The final result should be something like this:

The screenshot shows the 'Access Control' interface with the 'Resources' tab selected. A red box highlights the 'Publish All' button in the top toolbar. Below the toolbar, a table lists resources with their status. The resources listed are: /api, /api/transfer, cba_transfer, /api/nf/accounts, cba_accounts, /api/nf/balance, and cba_balance. The 'cba_transfer' resource has a warning icon indicating 'Publish required'. The 'cba_balance' resource also has a warning icon indicating 'Publish required'.

Click **Publish All**.

8.5.5 Test Context-Based Access Control

Open a browser and go to <http://mechant.authsaz.com:5000>. If **Access token** value is not filled yet or token does not belong to **user1**, click **Request Access** and login as **user1** and then follow the steps to get an access token.

8.5.5.1 Test Accounts API with permitted values

Accounts

User name

user1

View Accounts

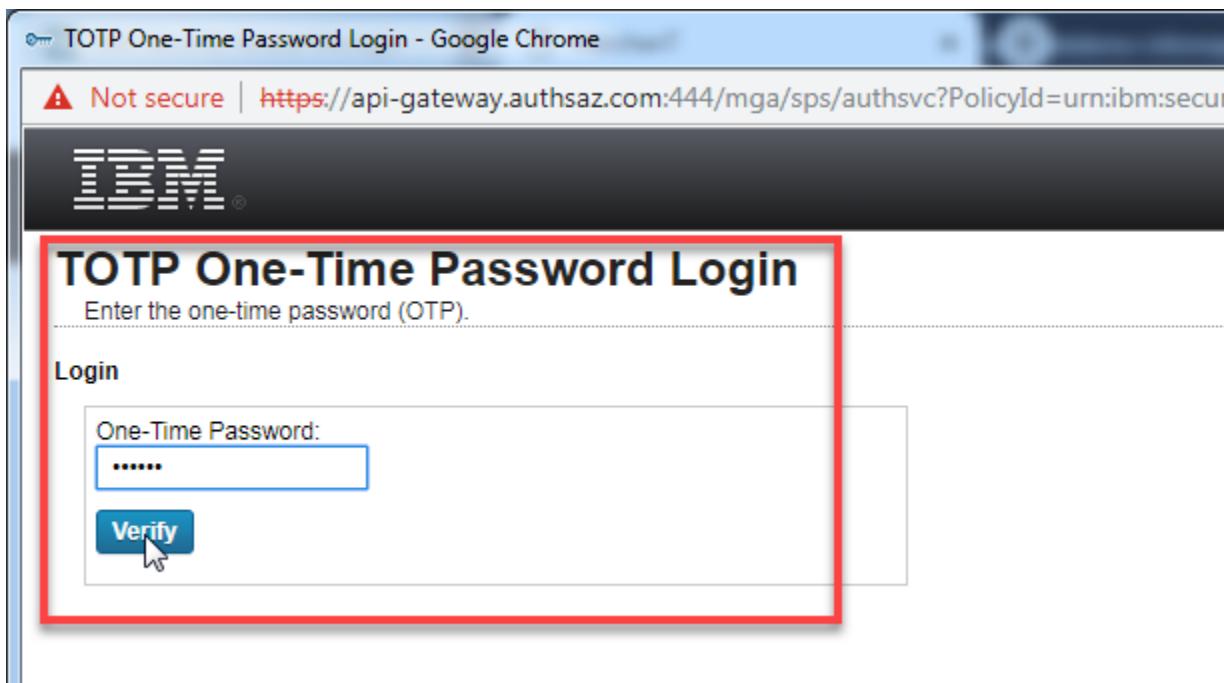
Balance

Account Id

View Balance

Enter **user1** as the *User name* in *Accounts* form.

Click **View Accounts**.



The screenshot shows a web browser window titled "TOTP One-Time Password Login - Google Chrome". The address bar indicates a "Not secure" connection and the URL <https://api-gateway.authsaz.com:444/mga/sps/authsvc?PolicyId=urn:ibm:secu>. The page features the IBM logo at the top. Below it, the title "TOTP One-Time Password Login" is displayed in large bold letters, followed by the instruction "Enter the one-time password (OTP)". A red rectangular box highlights the login form area. Inside this box, there is a label "One-Time Password:" above a text input field containing ".....". Below the input field is a blue "Verify" button with a white arrow pointing to it, indicating it is the next step to be clicked.

If everything is ok, a new window will be popped up.

Open the *IBM Verify* application on your mobile device and touch the account. You can see the current **TOTP code** here.

Enter **TOTP Code** as *One-Time Password*.

Click **Verify**.

Disabled

Hello

Access token: hnCPr...
Refresh token: V9LyA...
Expires in: 3599
Token type: bearer
Scope: financial non-financial

Revoke Access And Logout Refresh Token Query Info

Request Access

{"account":["101010","202020"],"status":"1"}

{"scope":"financial non-financial","active":true,"token_type":"bearer","exp":1543989224,"iat":1543985624,"client_id":"jSNhWHOviWPzqlB3u81","username":"user1"}

Accounts

User name

Balance

Account Id

Transfer

From Account Id

To Account Id

The message {"account": ["101010", "202020"], "status": "1"} should be returned as result.

8.5.5.2 Test Accounts API with not permitted values

The screenshot shows a web application interface. At the top, there is a dark header with the word "Disabled". Below it, a large "Hello, Cus..." message is partially visible. A modal dialog box titled "Error" is displayed, containing the message "Not Permitted." A red box highlights this error message. In the background, token information is shown: Access token: hnCFPrKBSW5j7I6ipt9Y, Refresh token: V9LyAVtHDYicNcppUsiLFs..., Expires in: 3599, Token type: bearer, Scope: financial non-financial. Below this, three buttons are visible: "Revoke Access And Logout" (red), "Refresh Token" (yellow), and "Query Info" (yellow). A "Request Access" button is also present. To the right, a JSON object is shown: {"scope":"financial non-financial","active":true,"token_type":"bearer","exp":1543989224,"iat":1543985624,"client_id":"jSNhWHOviWPzcqJB3u81","username":"user1"}. A red box highlights this JSON object. At the bottom, there are three sections: "Accounts", "Balance", and "Transfer". The "Accounts" section has a "User name" input field containing "user2", which is highlighted with a red box. A "View Accounts" button is below it, also highlighted with a red box and has a cursor icon pointing at it. The "Balance" and "Transfer" sections have empty input fields for "Account Id", "From Account Id", and "To Account Id".

Enter **user2** as the *User name* in *Accounts* form.

Click **View Accounts**.

If everything is ok, **Not Permitted** should be returned as result. The result shows that Merchant can not access **user2** accounts information by using **user1** access token.

8.5.5.3 Test Balance API with permitted values

Accounts

user name
user1

Balance

Account Id
101010

Transfer

From Account Id

To Account Id

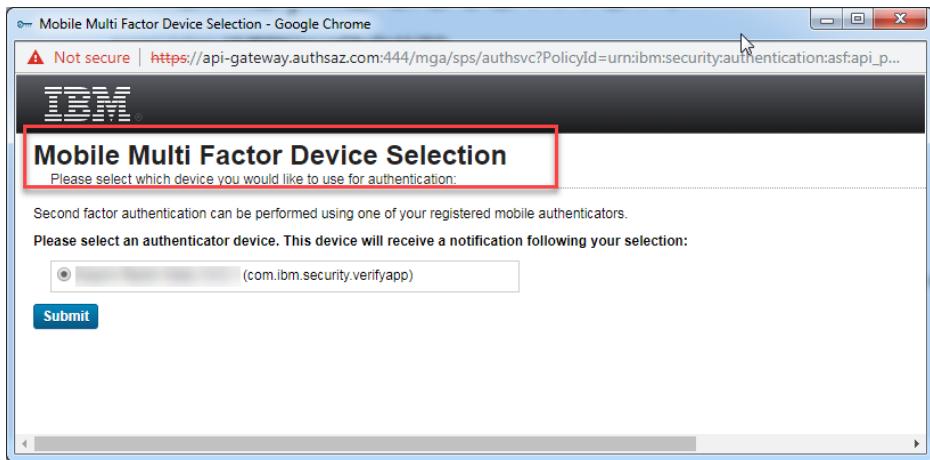
Amount

View Accounts

View Balance

Enter **101010** as *Account Id* in *Balance* form.

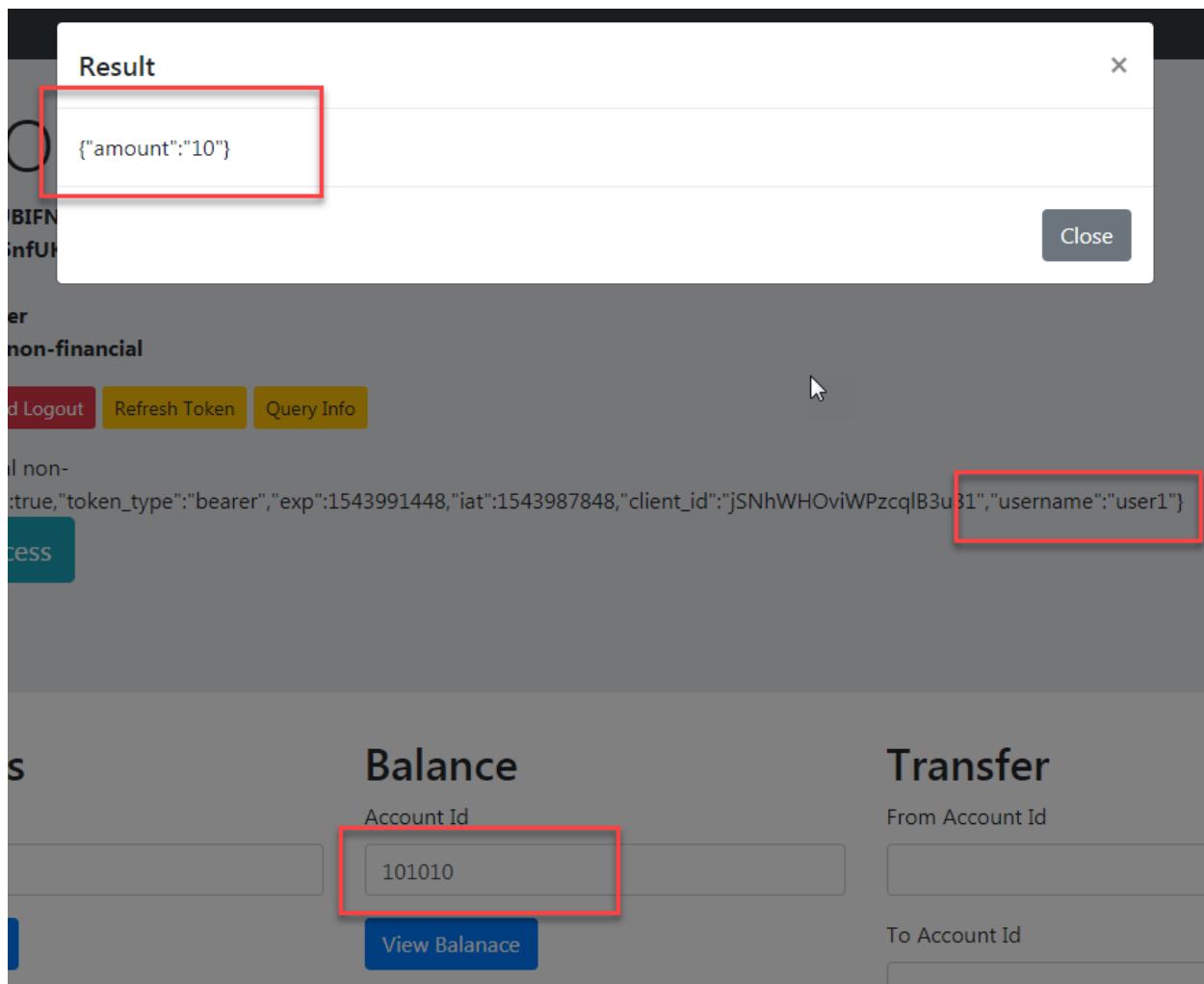
Click **View Balance**.



If everything is ok, a new window will be popped up.

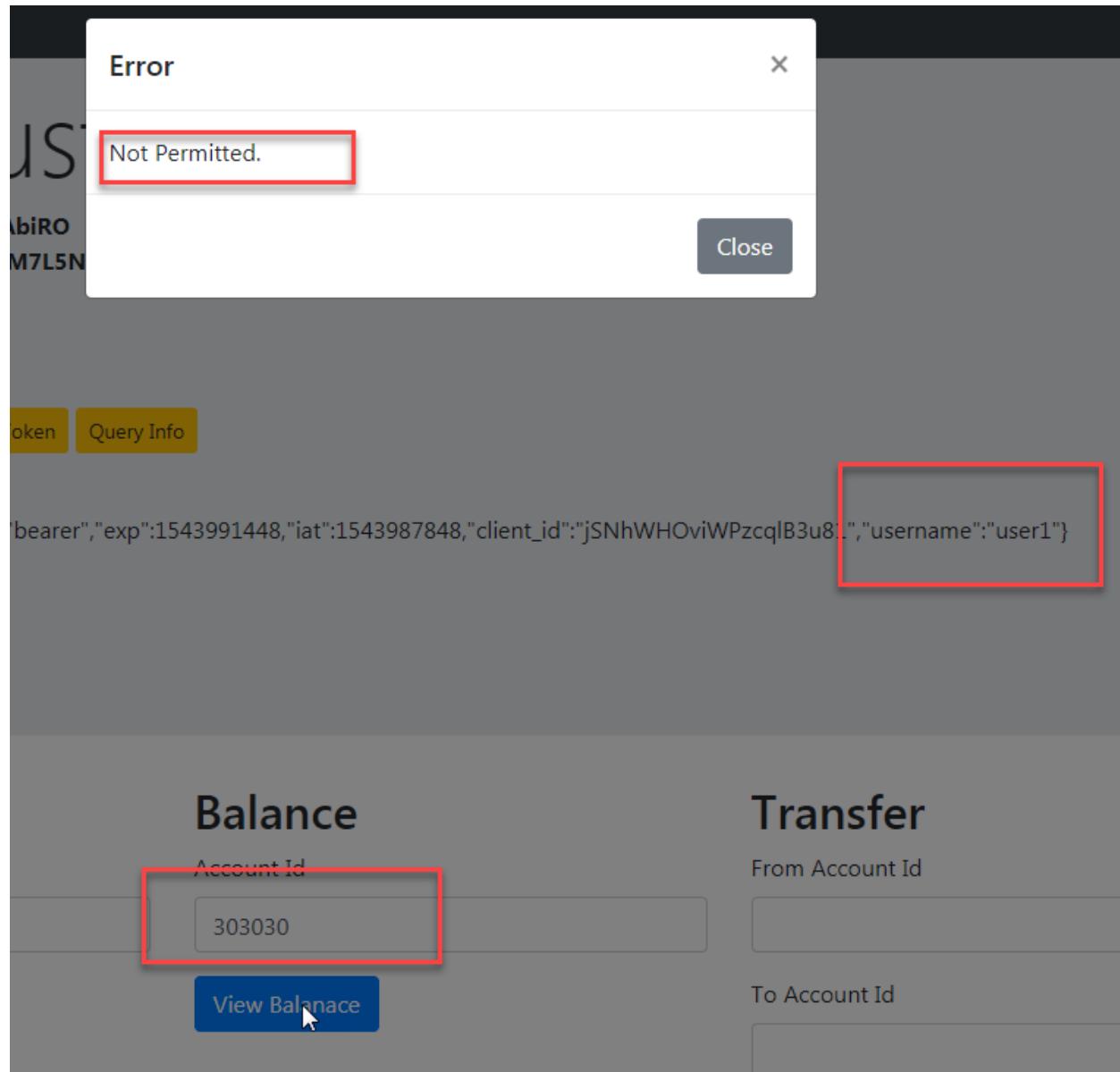
Select the radio button for your *authenticator device*.

Click **Submit** and follow the authentication mechanism in your authenticator device as we described in the previous chapter.



After the successful authentication value {"amount": "10"} will be returned.

8.5.5.4 Test Balance API with not permitted values



Repeat the steps in the previous section for *Account Id 303030*.

The result would be **Not Permitted**. That is because **303030** is not **user1 Account Id**.

8.5.5.5 Test Transfer API with Amount value lower than 1000

Transfer

From Account Id

To Account Id

Amount

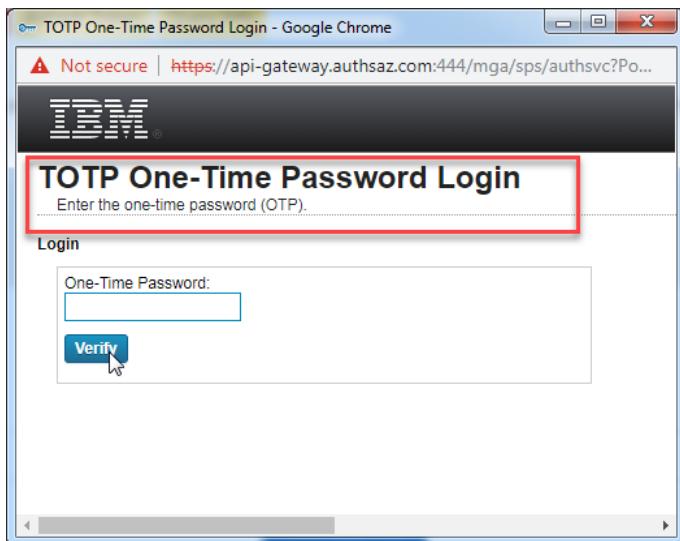
Transfer Money

Enter **101010** as *From Account Id* in *Transfer* form.

Enter **404040** as *To Account Id* in *Transfer* form.

Enter **10** as *Amount* in *Transfer* form.

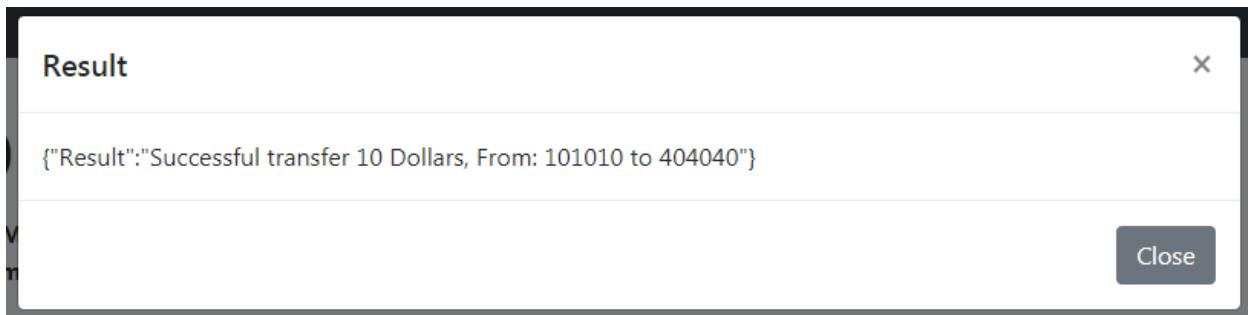
Click **Transfer Money**.



If everything is ok, a new window will be popped up.

Enter **TOTP Code** as *One-Time Password*.

Click **Verify**.



The result would be **{"Result": "Successful transfer 10 Dollars, From: 101010 to 404040"}**.

8.5.5.6 Test Transfer API with Amount value greater than 1000

Transfer

From Account Id

 101010

To Account Id

Amount

 1001

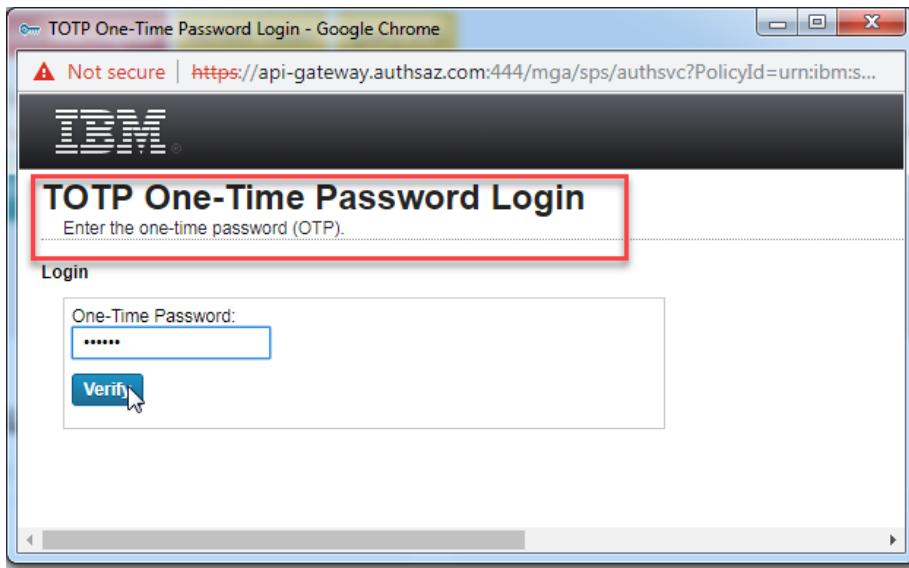
Transfer Money

Enter **101010** as *From Account Id* in *Transfer* form.

Enter **404040** as *To Account Id* in *Transfer* form.

Enter **1001** as *Amount* in *Transfer* form.

Click **Transfer Money**.



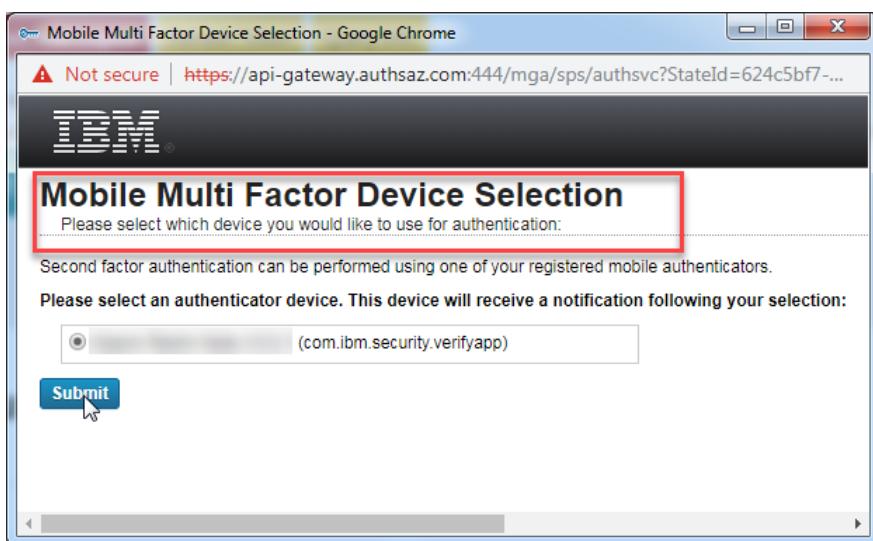
If everything is ok, a new window will be popped up.

Open the *IBM Verify* application on your mobile device and touch the account. You can see the current **TOTP code** here.

Enter **TOTP Code** as *One-Time Password*.

Click **Verify**.

To have a stronger authentication policy for transferring large amounts, the user is asked another authentication step. After successful TOTP authentication MMFA will be started.



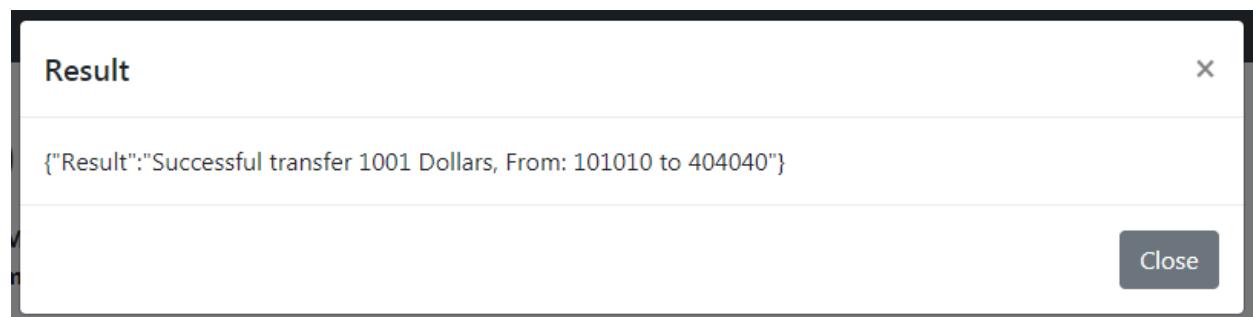
Select the radio button for your *authenticator device*.

Click **Submit**.

Open the *IBM Verify* application and touch the refresh icon in the top right corner.

Once you see the blue dot, touch the account to open it. You should immediately be prompted with the pending transaction.

Use your fingerprint to confirm the transaction.



The result would be **{"Result": "Successful transfer 1001 Dollars, From: 101010 to 404040"}**.

8.6 Using AAC Policy to express OAuth Scope

In our sample APIs implementation, we have defined two scopes: **financial** and **non-financial**. *Accounts* and *Balance* APIs are considered as non-financial and *Transfer* API is considered as financial. In our API Server, we broke down financial and non-financial APIs by assigning a different URL path to them. Financial APIs path starts with **/api/f/** and non-financial APIs path starts with **/api/nf/**.

During the consent approval in OAuth authorization code flow, the user can choose access to which scopes he wants to delegate to the client. The problem of policies that we defined in this chapter up to now is that they don't validate API scope against access token scope. Next section shows that Merchant can access an API with an access token that was not issued for access to this API.

8.6.1 Test Lack of Scope validation

Open a browser and go to <http://mechant.authsaz.com:5000>.

Token type: **bearer**

Scope: **financial non-financial**

Revoke Access And Logout



Refresh Token

Query Info

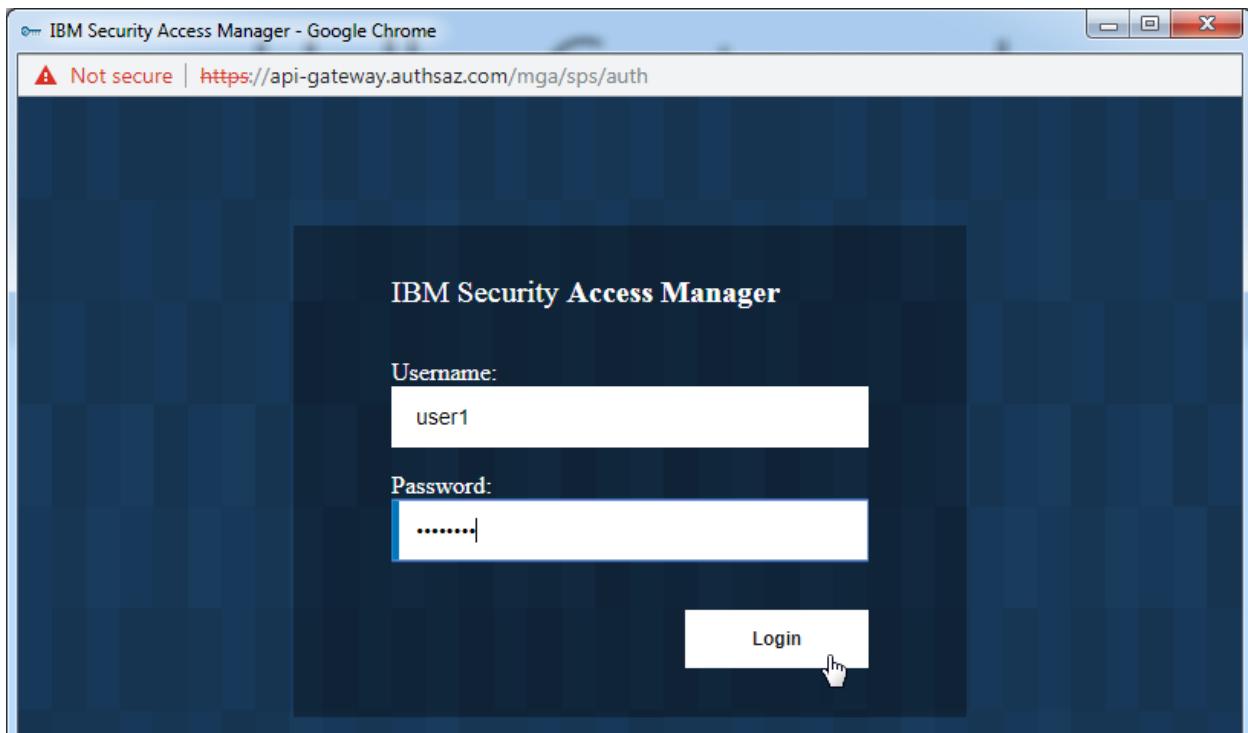
If Access Token value is not null click **Revoke Access And Logout**.

Hello, Customer!

Request Access

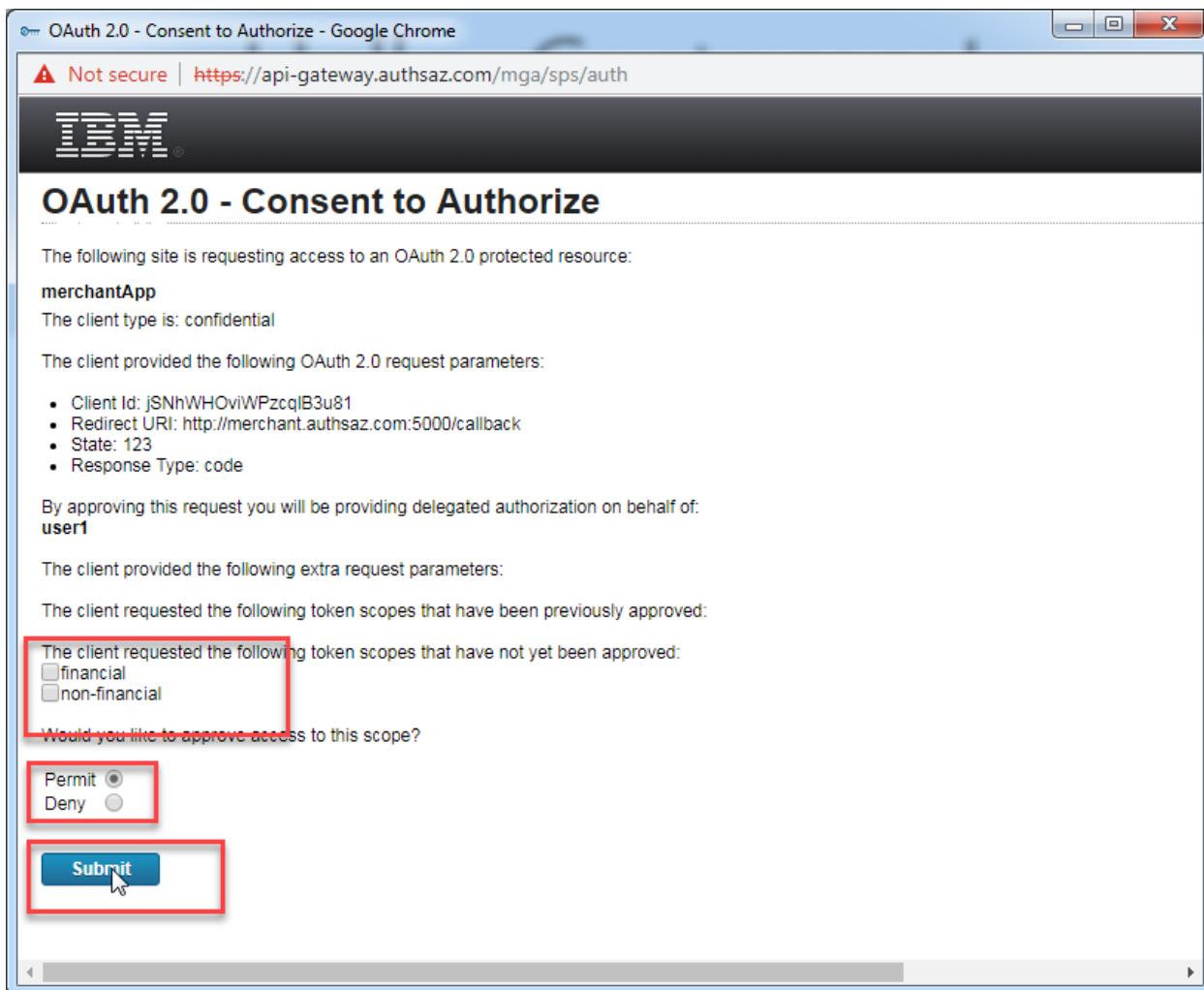


Click **Request Access**.



Enter **user1** as *Username* and **PasswOrd** as *Password*.

Click **Login**.



Deselect **financial** and **non-financial** checkboxes.

Select the radio button for **Permit**.

Click **Submit**.

Hello, Customer!

Access token: **Cpb32ZJfDXVVoBQ0VKyR**
Refresh token: **x3ABDNYFWnf90BKrrD5QoEj2hf5euFx8Y998DUUj**
Expires in: **3599**
Token type: **bearer**

Scope:

[Revoke Access And Logout](#) [Refresh Token](#) [Query Info](#)

[Request Access](#)

After successful delegation, Scope value should be null.

Accounts

User name

[View Accounts](#)

Bala

Accoun

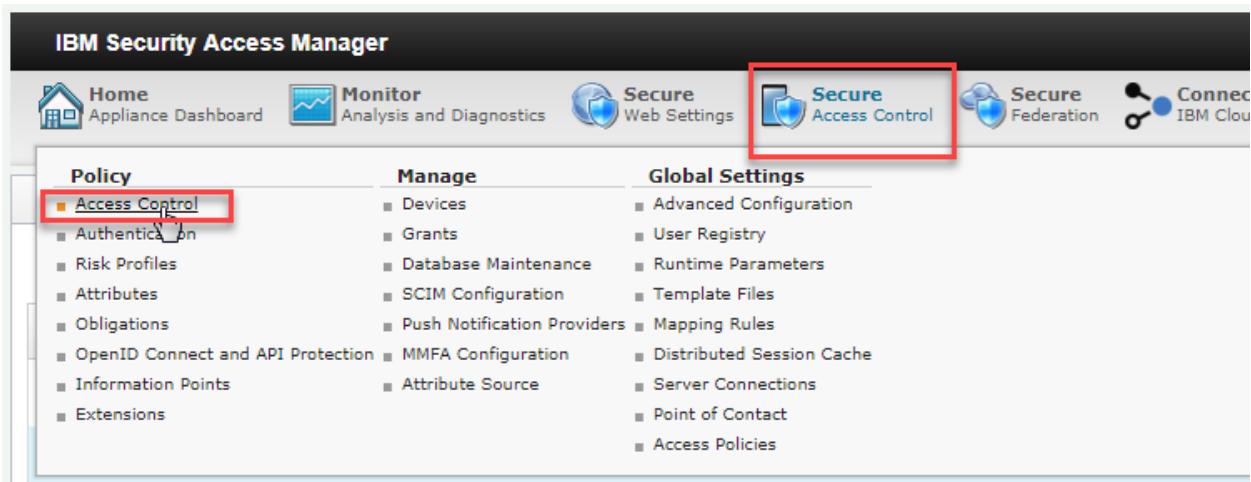
[View](#)

Enter **user1** as the *User name* in *Accounts* form.



The result would be `{"status": "1", "account": ["101010", "202020"]}`. It shows that Scope value does not affect the access decision to `/api/nf/accounts` resource.

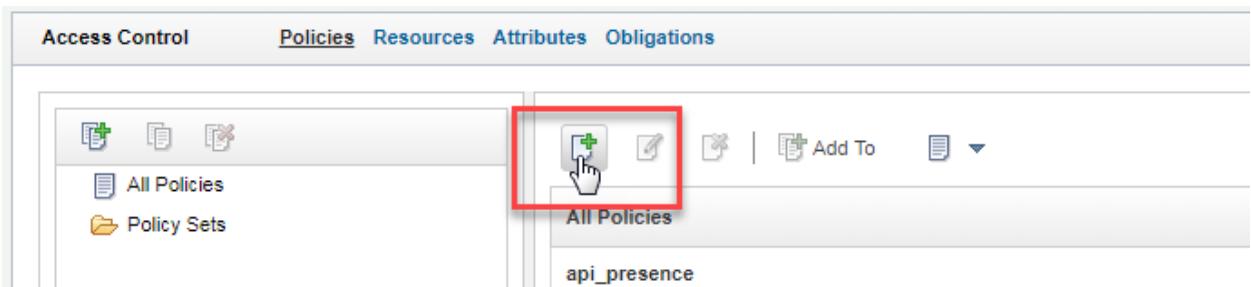
8.6.2 Define policy to validate scope



The screenshot shows the IBM Security Access Manager interface. At the top, there is a navigation bar with several tabs: Home, Monitor, Secure Web Settings, Secure Access Control (which is highlighted with a red box), Secure Federation, and Connected IBM Cloud. Below the navigation bar is a main menu area divided into three columns: Policy, Manage, and Global Settings. The Policy column contains a sub-menu with 'Access Control' highlighted by a red box. The Manage and Global Settings columns each have several sub-options listed.

Policy	Manage	Global Settings
Access Control	Devices, Grants, Database Maintenance, SCIM Configuration, Push Notification Providers, MMFA Configuration, Attribute Source	Advanced Configuration, User Registry, Runtime Parameters, Template Files, Mapping Rules, Distributed Session Cache, Server Connections, Point of Contact, Access Policies
Authentication, Risk Profiles, Attributes, Obligations, OpenID Connect and API Protection, Information Points, Extensions		

Navigate to **Secure Access Control > Policy: Access Control**.



The screenshot shows the 'Policies' tab of the Secure Access Control interface. At the top, there is a toolbar with buttons for 'Access Control', 'Policies' (which is underlined and highlighted in blue), 'Resources', 'Attributes', and 'Obligations'. Below the toolbar is a sidebar with icons for 'All Policies' and 'Policy Sets'. The main content area displays a list of policies, with one policy named 'api_presence' visible. At the top right of the main content area, there is a toolbar with several buttons, one of which is a green '+' icon labeled 'Add Policy', which is highlighted with a red box.

Click the **Add Policy** button in the main window of the *Policies* tab.

The screenshot shows the 'Access Control' interface with the 'Policies' tab selected. A new policy is being created with the following details:

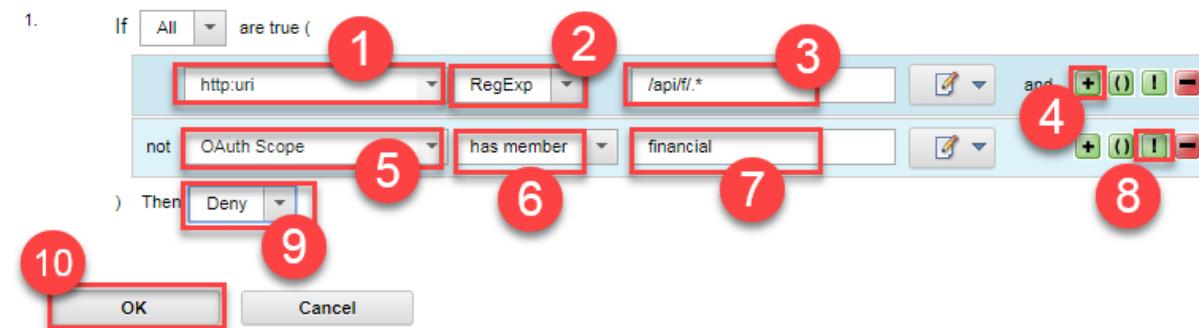
- Name:** oauth_scope (highlighted with a red box)
- Description:** (empty text area)
- Subjects:** (empty list)
- Rules:** Precedence: Deny (highlighted with a red box)
 - Add Rule (button)
 - Conditional rule (selected option, highlighted with a red box and a cursor icon)
 - Unconditional rule

Enter **oauth_scope** as the *Name*. Enter a *description*.

Select **Deny** from the *Precedence* drop-down list.

Click **Add Rule**.

Select **Conditional rule** from the drop-down list.



In the first rule, we will check to see if the **http:uri** attribute matches with **/api/f/.*** pattern. If so, then we will deny access if the **financial** attribute is not in **OAuth Scope**.

Select **http:uri** from the drop-down list.

Select **RegExp** as the comparison operator.

Enter **/api/f/.*** as the value for comparison.

Click + icon in the right corner of rule editor to add a new condition.

In newly added line click on ! icon to **negate** the condition.

Select **OAuth Scope** from the drop-down list.

Select **has member** as the comparison operator.

Enter **financial** as the value for comparison.

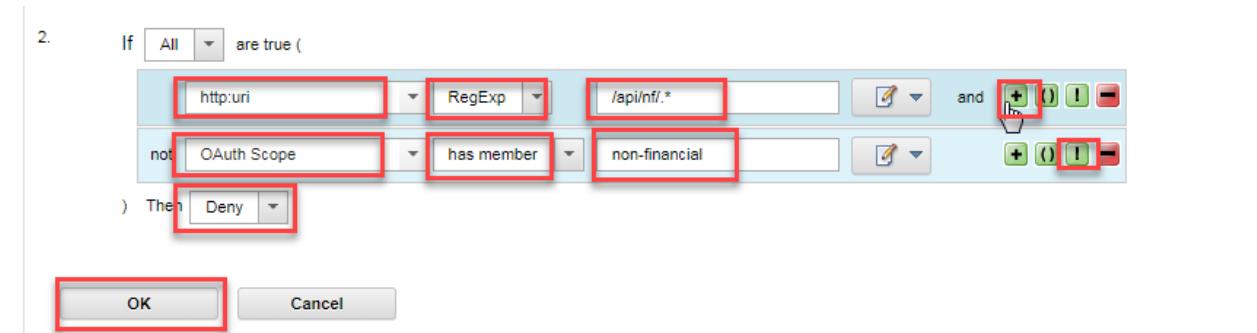
Select **deny** for the result for this rule.

Click **OK**.



Click **Add Rule** to add the second rule.

Select **Conditional rule** from the drop-down list.



In the second rule, we will check to see if the **http:uri** attribute matches with **/api/nf/.*** pattern. If so, then we will deny access if the **non-financial** attribute is not in **OAuth Scope**.

Select **http:uri** from the drop-down list.

Select **RegExp** as the comparison operator.

Enter **/api/nf/.*** as the value for comparison.

Click + icon in the right corner of rule editor to add a new condition.

In newly added line click on ! icon to negate the condition.

Select **OAuth Scope** from the drop-down list.

Select **has member** as the comparison operator.

Enter **non-financial** as the value for comparison.

Select **deny** for the result for this rule.

Click **OK**.



The screenshot shows the policy rules defined in the Access Control interface. At the top, it says 'Rules (2)' and 'Precedence: Deny'. Below that, the first rule is listed: 'If http:uri RegExp "/api/nf/.*" and not (OAuth Scope has member "financial") Then Deny'. The second rule is listed below it: 'If http:uri RegExp "/api/nf/.*" and not (OAuth Scope has member "non-financial") Then Deny'. Both rules use the 'Deny' result.

```
▼ Rules (2) ? Precedence: Deny ▼ ? Attributes: Optional ▼

1. If http:uri RegExp "/api/nf/.*" and
   not ( OAuth Scope has member "financial" )
   Then Deny

2. If http:uri RegExp "/api/nf/.*" and
   not ( OAuth Scope has member "non-financial" )
   Then Deny
```

Check that the policy looks as shown above.

Click **Save** button at the top of the window to save the Policy. The new policy is shown in the policy list:

The screenshot shows the Access Control interface with the Policies tab selected. On the left, there's a sidebar with icons for creating, viewing, and deleting policies, and buttons for 'All Policies' and 'Policy Sets'. The main area lists several policies: api_presence, api_otp_finger, api_totp, cba_balance, cba_transfer, oauth_scope (which has a dashed blue border around it), and cba_accounts.

8.6.3 Create Policy Sets

The screenshot shows the Access Control interface with the Policies tab selected. The left sidebar has a red box around the 'Create Policy Set' button (the plus sign icon). The main area lists policies: api_presence, api_otp_finger, and api_totp.

On the left-hand panel of Access Control form click on **Create Policy Set** button.

Create Policy Set

Name: pset_accounts

Description:

Policy Combining Algorithm:

Deny access if any policy in the set returns deny

Permit access if any policy in the set returns permit

Return the decision of the first policy in the set that returns either permit or deny

Save Cancel

Enter **pset_accounts** as the *Name*.

Select the radio button for **Deny access if any policy in the set returns deny**.

Click **Save**.

Repeat steps above to create two other policy sets **pset_accounts** and **pset_transfer**.

Access Control Policies Resources Attributes Obligations

All Policies

Policy Sets

pset_balance

pset_transfer

pset_accounts

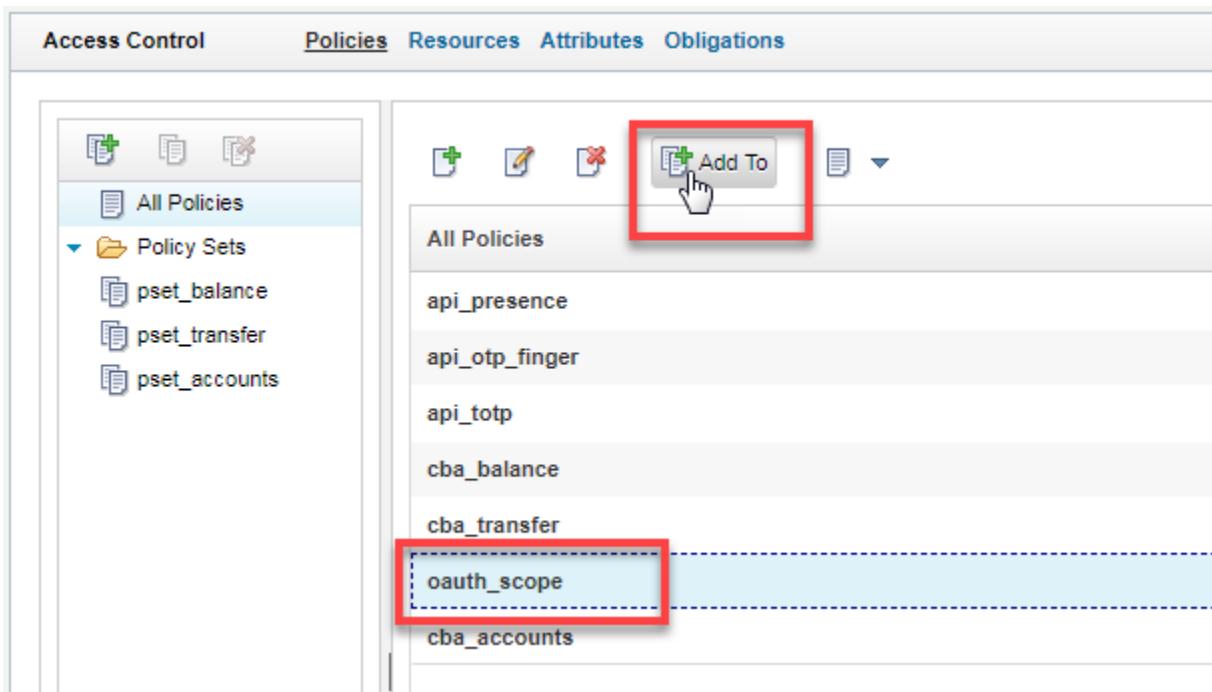
All Policies

api_presence

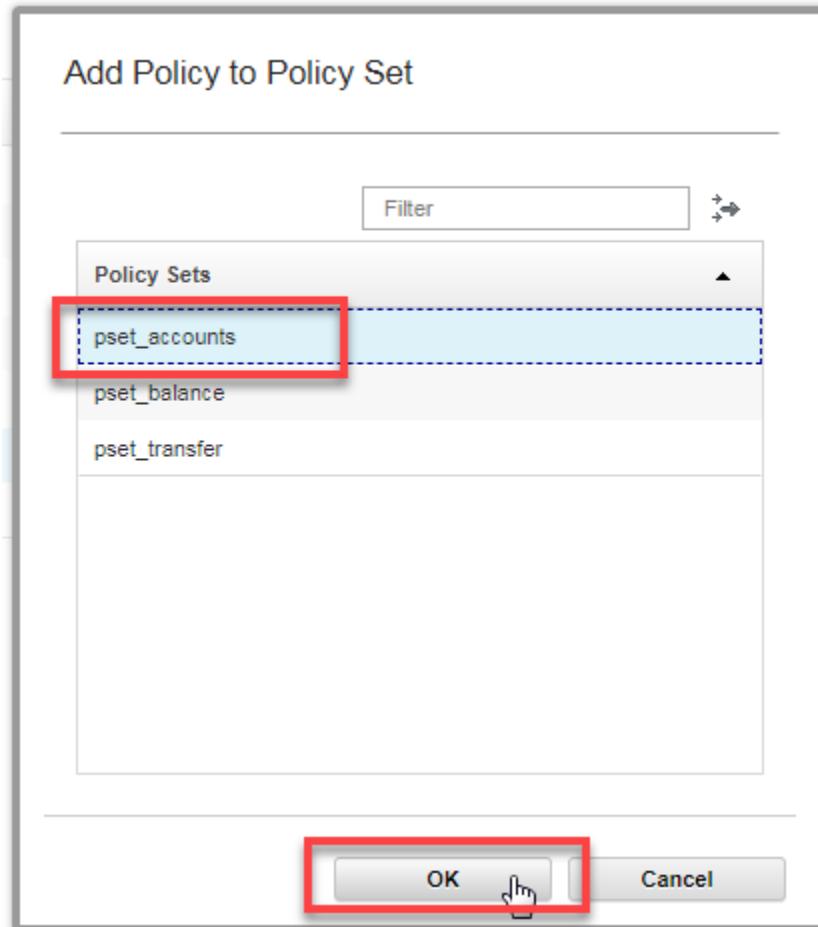
api_otp_finger

api_totp

After successfully creating policy sets, the left-hand panel should be something like the picture above.

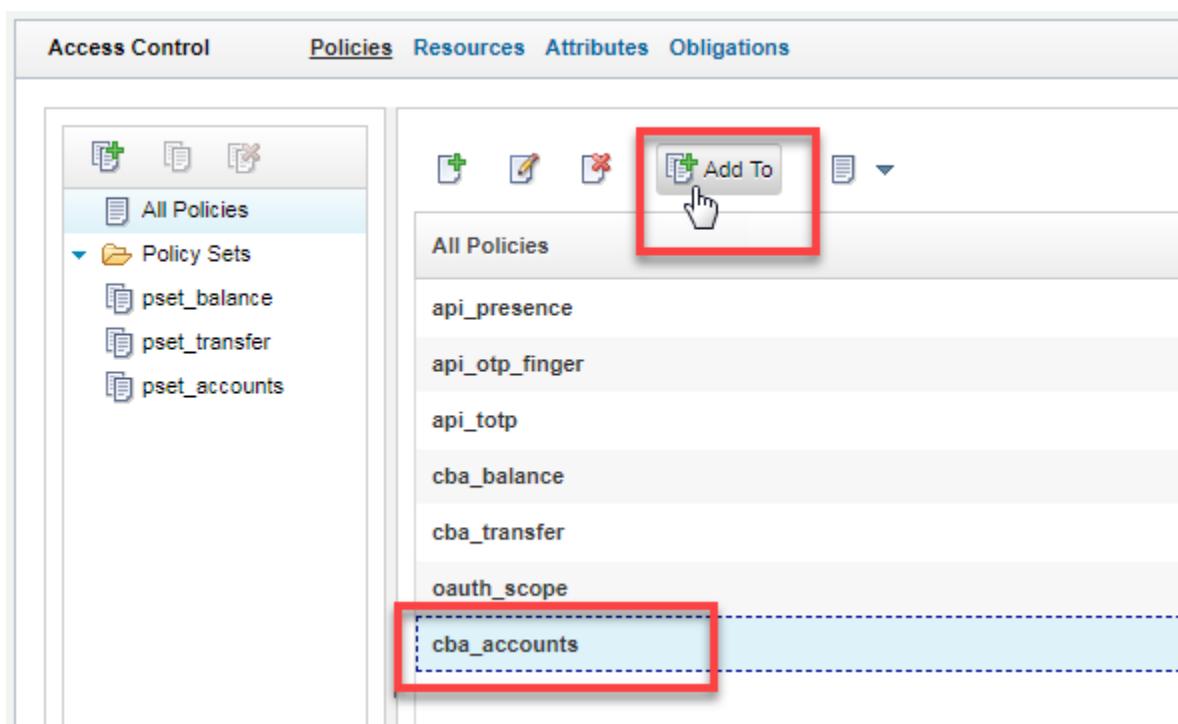


On the right-hand panel of *Access Control* form, select the **oauth_scope** policy and then click **Add To** button.

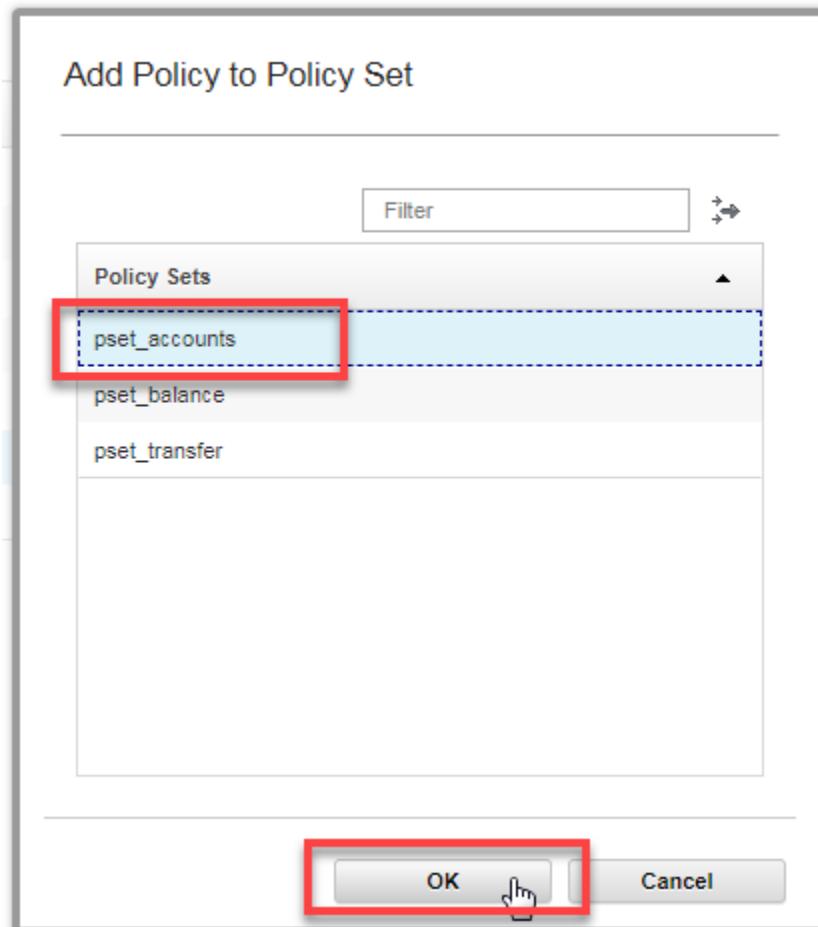


From *Add Policy to Policy Set* window select **pset_accounts** from *Policy Sets* list.

Click **Ok**.



On the right-hand panel of *Access Control* form, select the **cba_accounts** policy and then click **Add To** button.



From Add Policy to Policy Set window select **pset_accounts** from Policy Sets list.

Click **Ok**.

The screenshot shows the "Access Control" interface with the "Policies" tab selected. The left sidebar has icons for "All Policies", "Policy Sets" (which is expanded), "pset_balance", "pset_transfer", and "pset_accounts" (which is highlighted with a red rectangular border). The main panel displays a list titled "Policies for 'pset_accounts'" containing "oauth_scope" and "cba_accounts", both of which are also highlighted with a red rectangular border. There are "Add To" and "Remove" buttons above the list, along with up and down arrows for reordering.

By clicking on **pset_accounts** policy set you should see **oauth_scope** and **cba_accounts** as its members.

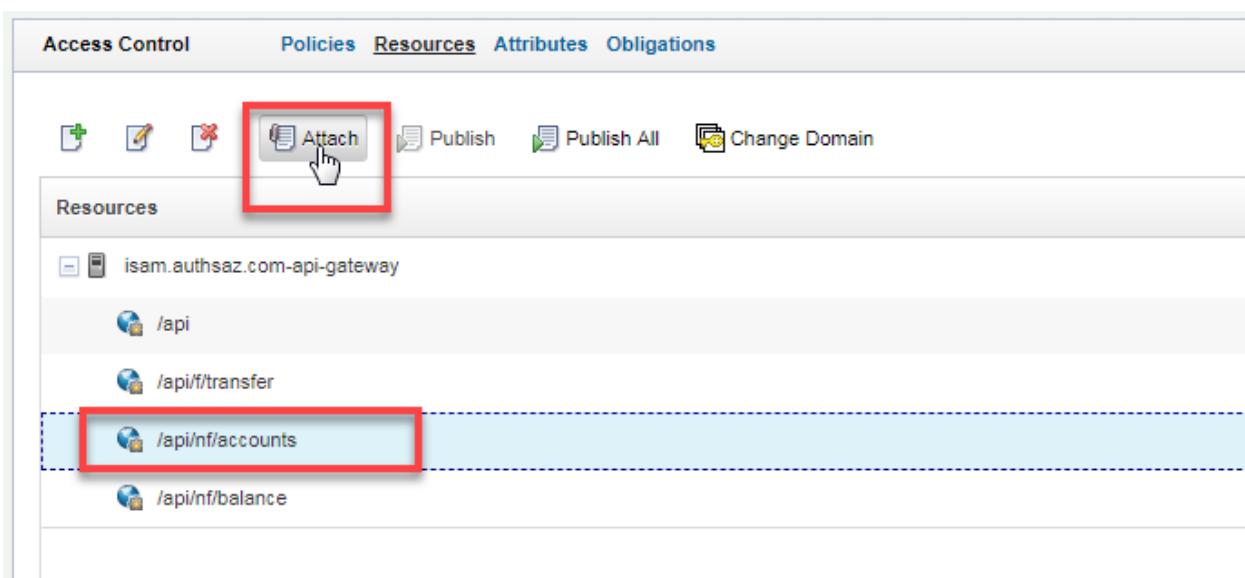
Repeat steps above to add members **oauth_scope** and **cba_balance** to **pset_balance** and add members **oauth_scope** and **cba_transfer** to **pset_transfer**.

8.6.4 Attach Policy Sets to Resources

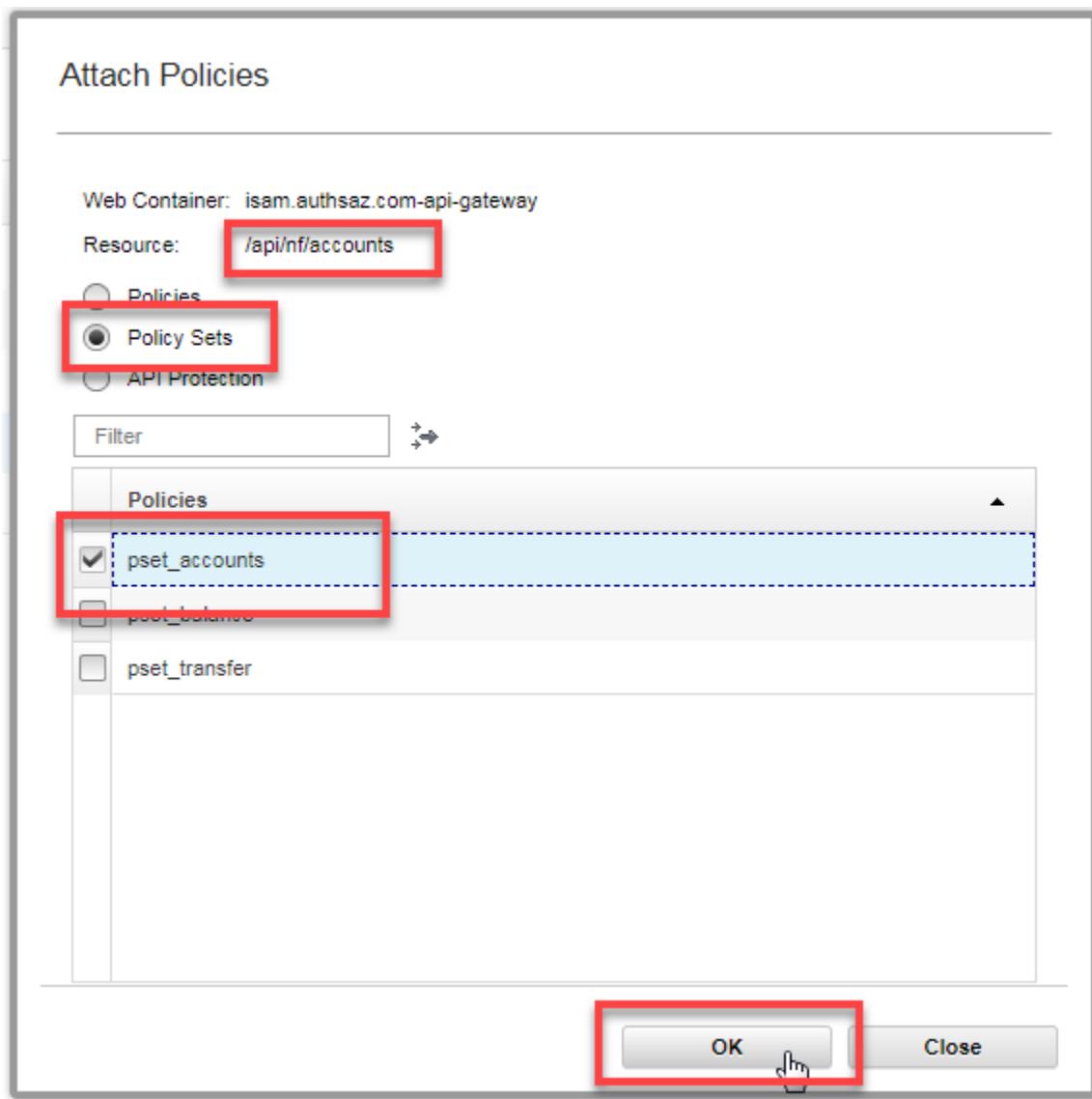
Navigate to the **Resources** tab.

The screenshot shows the Oracle Access Control interface with the 'Resources' tab selected. The toolbar includes buttons for creating (+), deleting (trash), attaching (highlighted with a red box), publishing, publishing all, and changing domain. The 'Resources' section lists several resources: 'isam.authsaz.com-api-gateway', '/api', '/api/f/transfer', '/api/nf/accounts', '/api/nf/balance'. Under '/api/f/transfer', the 'cba_transfer' policy set is listed. Under '/api/nf/accounts', the 'cba_accounts' policy set is listed. Under '/api/nf/balance', the 'cba_balance' policy set is listed. All three policy sets are highlighted with red boxes.

Remove all previously attached policy to **/api/f/transfer**, **/api/nf/accounts** and **/api/nf/balance** resources.



Select the **/api/nf/accounts** resource and click **Attach**.



Select radio button for **Policy Sets**.

Select the checkbox next to the **pset_accounts** and press the **OK** button to attach the specified policy to the resource.

Repeat steps above to attach **pset_balance** to **/api/nf/balance** and attach **pset_transfer** to **/api/f/transfer**.

The final result should be something like this:

The screenshot shows the 'Access Control' interface with the 'Resources' tab selected. At the top, there are several buttons: 'Attach', 'Publish', 'Publish All' (which is highlighted with a red box), and 'Change Domain'. Below these, a table lists resources under the 'isam.authsaz.com-api-gateway' domain. The table has two columns: 'Resources' and 'Status'. Under 'Resources', there are four entries: '/api', '/api/transfer', '/api/nf/accounts', and '/api/nf/balance'. Under '/api/transfer', three sub-resources are listed: 'pset_transfer', 'pset_accounts', and 'pset_balance'. Each resource entry includes a small icon and a status indicator: a warning icon with 'Publish required' next to it.

Click **Publish All**.

8.6.5 Test pset_accounts policy for Accounts API

Repeat steps in section 8.6.1 to create an access token with Scope value equal to null.

The screenshot shows a web application interface. In the background, there is a dark-themed dashboard with sections for 'Accounts', 'Balance', and 'Transfer'. The 'Accounts' section has a form with a 'User name' field containing 'user1' and a 'View Accounts' button. The 'Balance' section has a 'View Balance' button. The 'Transfer' section has fields for 'From Account Id' and 'To Account Id'. In the foreground, a modal dialog box titled 'Error' is displayed, with the message 'Not Permitted.' highlighted by a red box. Below the modal, the user's access token information is shown: 'Access token: Cpb32ZJfDXVVoBQ0VKyR', 'Refresh token: x3ABDNYFWnf90BKrrD5Q', 'Expires in: 3599', 'Token type: bearer', and a 'Scope:' field which is empty and highlighted by a red box. At the bottom of the modal are 'Close' and 'Request Access' buttons, with the 'Request Access' button also highlighted by a red box. To the right of the modal, a JSON object is partially visible: '{"scope":"","active":true,"token_type":"bearer","exp":1543993118,"iat":1543989518,"client_id":"jSNhWHOviWPzcqlB3u81","username":"user1"}'.

Enter **user1** as the *User name* in *Accounts* form.

The result would be **Not Permitted**. It is because the *Scope* value for this *Access Token* is null and the access request will be rejected by **oauth_scope** policy.

8.6.6 Test pset_balance policy for Balance API

The screenshot shows a user interface for managing accounts and balances. At the top, there is a dark header with the word "Disabled". Below it, a large "Hello, Customer!" greeting is displayed. A modal dialog box titled "Error" contains the message "Not Permitted." A red box highlights this message. In the background, there are three main sections: "Accounts", "Balance", and "Transfer". The "Accounts" section has a "User name" input field containing "user1" and a "View Accounts" button. The "Balance" section has an "Account Id" input field containing "101010" and a "View Balance" button, which is currently being clicked. A red box highlights the "Account Id" field. The "Transfer" section has "From Account Id" and "To Account Id" input fields, both currently empty. A red box highlights the "From Account Id" field. At the bottom of the page, there is a footer with the number "228".

Enter **101010** as *Account Id* in *Balance* form.

The result would be **Not Permitted**. It is because the *Scope* value for this *Access Token* is null and the access request will be rejected by **oauth_scope** policy.

8.6.7 Test pset_transfer policy for Transfer API

The screenshot shows a web application interface with several components:

- Accounts:** A section with a "User name" input field containing "user1" and a "View Accounts" button.
- Balance:** A section with an "Account Id" input field containing "101010" and a "View Balance" button.
- Error Dialog:** A modal window titled "Error" with the message "Not Permitted." This dialog has a red border around its content area.
- Transfer Form:** A section with three input fields:
 - "From Account Id" input field containing "101010" (highlighted with a red border).
 - "To Account Id" input field containing "303030".
 - "Amount" input field containing "1".A "Transfer Money" button is located below these fields.
- Token Information:** A sidebar on the left displays token details:
 - Access token: Cpb32ZJfDXVVbQ0VKyR...
 - Refresh token: x3ABDNYFWnf90BKrrD5Q...
 - Expires in: 3599
 - Token type: bearer
 - Scope: (empty field)Below this, a JSON object is shown: {"scope": "", "active": true, "token_type": "bearer", "exp": 1543993118, "iat": 1543989518, "client_id": "jSNhWHOviWPzcqIB3i81", "username": "user1"}

Enter **101010** as *From Account Id* in *Transfer* form.

Enter **303030** as *To Account Id* in *Transfer* form.

Enter **1** as *Amount* in *Transfer* form.

The result would be **Not Permitted**. It is because the *Scope* value for this *Access Token* is null and the access request will be rejected by **oauth_scope** policy.

9 A deep dive into our api_policy mapping rules

In chapter 7 we imported some mapping rules without discussing them more. In this chapter, we go through these mapping rules and describe the ideas behind each one of these files.

Some scenarios for using MMFA has been discussed in detail in “**IBM Verify Cookbook Mobile Multi-Factor Authentication with IBM SAM**”. There is a big difference between scenarios we considered for Protecting APIs and scenarios that have been discussed in that book. In their scenarios, user authentication state and AAC policies stay within the same HTTP session. In our scenarios, Merchant calls APIs in its HTTP session and user authenticate against AAC policies in a different HTTP session. Merchant calls APIs on behalf of the user by presenting the user’s OAuth token and depending on the access policy defined for APIs it may require the user to do more authentication steps. So to access an API, two different authentication flows should be carried out, one by OAuth access token on the merchant side and the other on the user side.

The problem is how we can branch authentication policies between Merchant and user in a way that after a successful authentication on user side we could tell the Merchant that the user side authentication has been carried out successfully. We came up with four ideas to tackle this issue:

1. Branching Authentication Policy Using cache mechanism
2. Branching Authentication Policy Using an external service
3. Branching Authentication Policy Using a custom encryption mechanism
4. Branching Authentication Policy Using STS

For implementing the ideas mentioned above, we have developed some infomaps. There is a brief description for each of these infomaps in the table below(We imported these infomaps in chapter 7).

api_policy_init.js	This infomap is used as the first step of the authentication flow on Merchant side and is responsible for creating a challenge and validating the proof corresponding to the challenge based on the method that has been configured in the api_policy_config.js file.
api_policy_resp_init.js	This infomap is used as the first step of the authentication flow on the user side and is responsible for extracting the user information from the given token and initiating next authentication steps.

api_policy_resp_finish.js	This infomap is used as the last step of the authentication flow on the user side and is responsible for creating a proof of completion corresponding to the given challenge. The proof will be sent back to the callback URL provided by the Merchant.
api_policy_config.js	This infomap is used as a general configuration file for other infomaps. Challenge creation/proof validation method and JWT signing secret will be configured in this infomap.
api_policy_cache.js	This infomap is used to create a challenge and validate corresponding proof using ISAM cache mechanism.
api_policy_WebService.js	This infomap is used to create a challenge and validate corresponding proof using an external service.
api_policy_JWT.js	This infomap is used to create a challenge and validate corresponding proof using a custom encryption mechanism.
jrsasign.js	This infomap implemented JWT library for javascript. It has been included in api_policy_JWT.js
api_policy_STS.js	This infomap is used to create a challenge and validate corresponding proof using ISAM STS module.

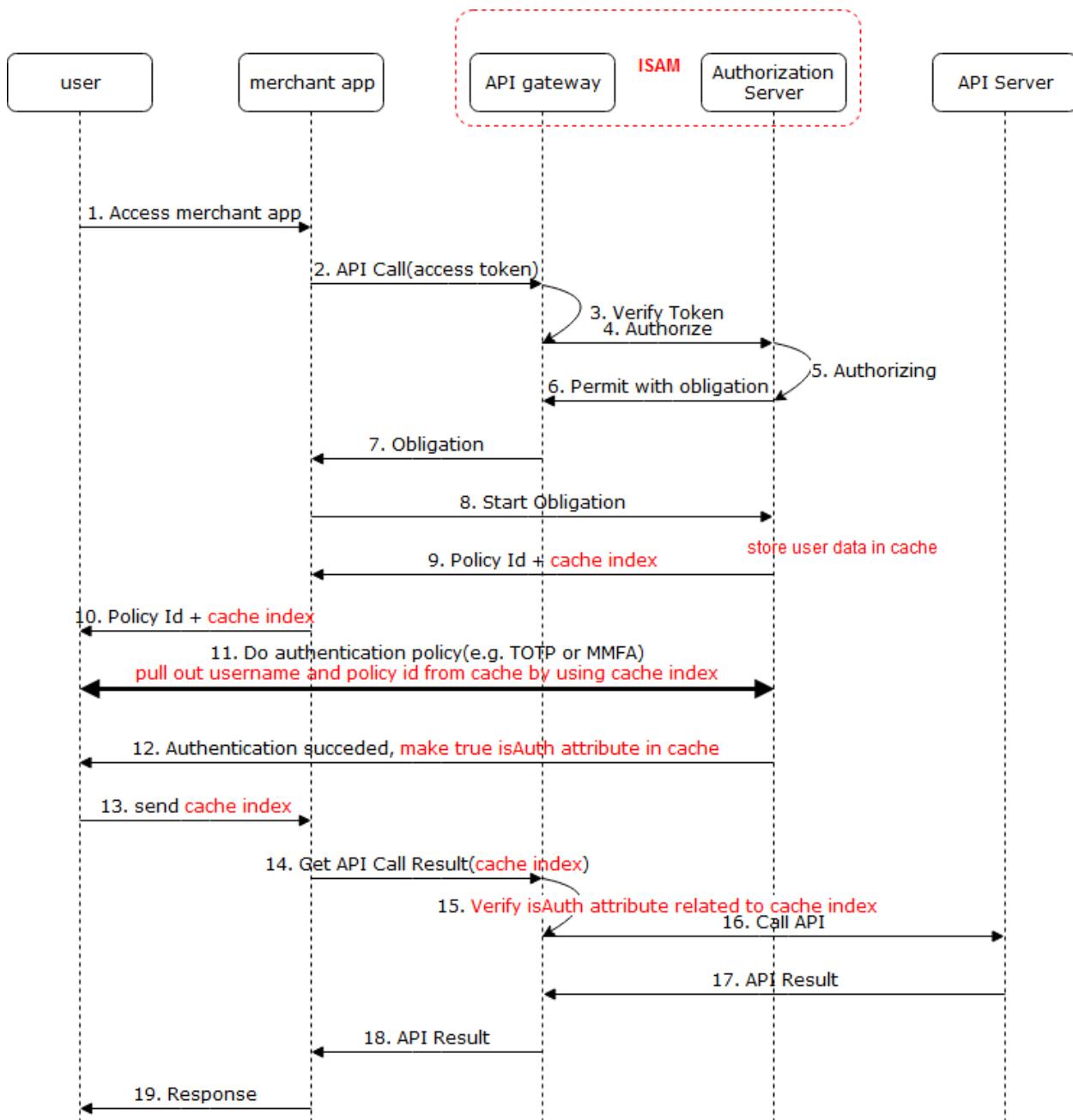
9.1 Branching Authentication Policy Using Cache mechanism

In “**Branching Authentication Policy in ISAM Advanced Access Control**”¹¹ Shane Weeden has discussed a technique to branching authentication policy between two different HTTP sessions. The challenge that has been discussed in this post was about managing state information between the different AAC policies, since starting a new policy will not preserve the “context” object that was being populated from the first policy. He mentioned that this can easily be carried out with the help of a Javascript InfoMap to programmatically store the context you want to share using one of these methods:

- `IDMappingExtUtils.setSPSSessionData(key, value)`
- `IDMappingExtUtils.getIDMappingExtCache().put(key, value, timeout)`

We used this idea to implement our API Protection approach. The following steps shown by red color should be carried out to store a shared context between user and merchant:

¹¹ <https://www.ibm.com/blogs/sweeden/branching-authentication-policy-isam-advanced-access-control/>



- The **api_policy_init** mechanism in the obligation policy (in step 8) stores an object comprising these attributes in ISAM cache and waits for the proof:
 - username
 - User side authentication Policy Id
 - timestamp
 - A random string as the cache index
 - isAuth boolean attribute

- Policy id along with cache index will be sent back to Merchant(step 9)
- Merchant forward the policy id and cache index to the user(step 10)
- The user starts an authentication flow regarding the given Policy Id. **api_policy_resp_init** pull out the object corresponding to the cache index from ISAM cache and move on to next authentication steps(step 11)
- After successfully completion of authentication steps, **api_policy_resp_finish** update isAuthenticated attribute to true in ISAM cache(step 12).
- User forward cache index to Merchant(step 13)
- Merchant use this cache index to call the API(step 14)
- **api_policy_init** uses given cache index to verify isAuthenticated attribute as the proof of successful completion of user side authentication(step 15).

9.1.1 Pros and cons

`setSPSSessionData` is faster and more efficient than `getIDMappingExtCache` as it is in memory, but as the state information needed to persist across user and Merchant sessions we must use `getIDMappingExtCache`. Using `getIDMappingExtCache` will downgrade performance and in case of clustering multiple ISAMs, it requires to use a single Cache for all ISAMs.

9.1.2 Configure infomaps to Activate Cache Mechanism

The infomap related to Cache Mechanism is **api_policy_cache.js**. We have imported this infomap in chapter 7.

The screenshot shows the IBM Security Access Manager web interface. At the top, there is a navigation bar with several tabs: Home, Monitor, Secure Web Settings, Secure Access Control (which is highlighted with a red box), Secure Federation, and Configuration. Below the navigation bar, there is a main content area with three columns. The left column is labeled 'Policy' and contains links to Access Control, Authentication, Risk Profiles, Attributes, Obligations, OpenID Connect and API Protection, Information Points, and Extensions. The middle column is labeled 'Manage' and contains links to Devices, Grants, Database Maintenance, SCIM Configuration, Push Notification Providers, MMFA Configuration, Attribute Source, and Attribute Source. The right column is labeled 'Global Settings' and contains links to Advanced Configuration, User Registry, Runtime Parameters, Template Files, Mapping Rules (which is highlighted with a red box), Session Cache, Server Connections, Point of Contact, and Access Policies. The 'Mapping Rules' link is specifically highlighted with a red box.

Navigate to **Secure Access Control > Global Settings: Mapping Rules**.

Mapping Rules	
Add	Import
Edit	Delete
Export	Replace
Category: InfoMap	
USC_PasswordReset_Success	Category: InfoMap
api_policy_JWT	Category: InfoMap
api_policy_STS	Category: InfoMap
api_policy_WebService	Category: InfoMap
api_policy_cache	Category: InfoMap
api_policy_config	Category: InfoMap
api_policy_init	Category: InfoMap
api_policy_resp_finish	Category: InfoMap
api_policy_resp_init	Category: InfoMap
jrsasign	Category: InfoMap

Select **api_policy_config** infomap from *Mapping Rules* table.

Click **Edit**.

```

/*
var implementation = 1;

switch(implementation){
    case 1: // cache
        importMappingRule("api_policy_cache");
        conf = {};
        break;
    case 2: // WebService
        importMappingRule("api_policy_WebService");
        conf = {
            webservice: "http://apps.authsaz.com:7001/internal/token"
        };
        break;
    case 3: // JWT
        importMappingRule("api_policy_JWT");
        conf = {secret: secret};
        break;
    case 4: // STS
        importMappingRule("api_policy_STS");
        conf = {};
        break;
    default:
        throw "no implementation is selected.";
}

```

Name:

Category:

Save **Close**

Modify **api_policy_config** mapping rule by assigning **1** to variable **implementation**.

Click **Save**.

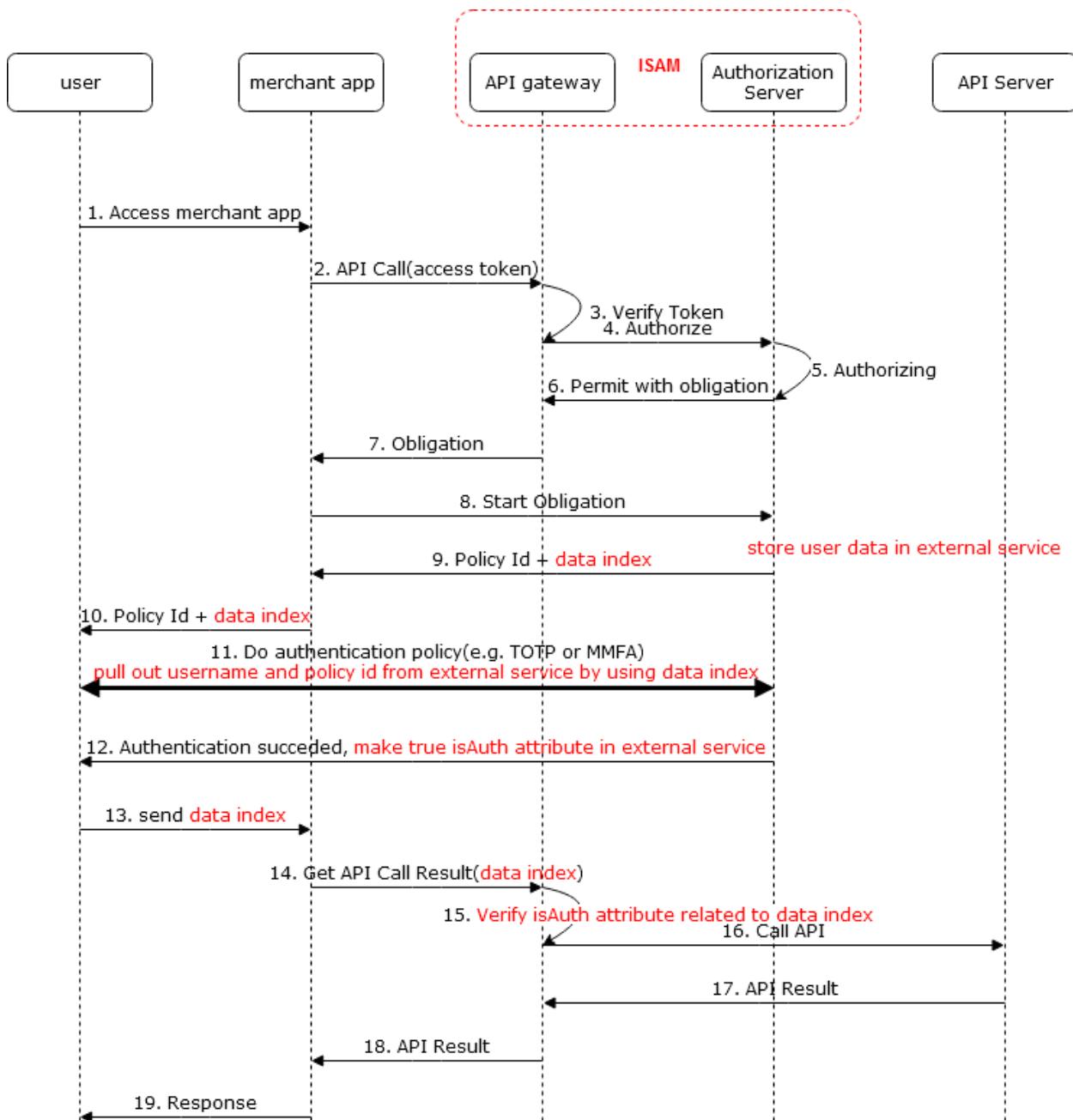
Deploy the changes using the link in the yellow warning message.

Repeat section 7.9 (Test API policies) tests to see the changes in the way of exchanging challenge and proof. The challenge for Cache Mechanism would be a random UUID like this:

https://api-gateway.authsaz.com:444/mga/sps/authsvc?PolicyId=urn:ibm:security:authentication:asf:api_policy_otp_resp&**challenge=8d1cf56a-9f74-436a-b4fe-51702bf4b093**&callback=http://merchant.authsaz.com:5000/apicallback

9.2 Branching Authentication Policy Using an external service

In this approach, we use an external service to create shared storage between Merchant and user. The following steps shown by red color should be carried out to use an external service as shared storage:



- The **api_policy_init** mechanism in the obligation policy (in step 8) stores an object comprising these attributes in external service and waits for the proof:
 - username
 - User side authentication Policy Id
 - timestamp
 - A random string as data index
 - isAuth boolean attribute

- Policy id along with data index will be sent back to Merchant(step 9)
- Merchant forward the policy id and data index to the user(step 10)
- The user starts an authentication flow regarding the given Policy Id. **api_policy_resp_init** pull out the object corresponding to the data index from external service and move on to the next authentication steps(step 11)
- After successfully completion of authentication steps, **api_policy_resp_finish** update isAuth attribute to true in external service(step 12).
- User forward data index to Merchant(step 13)
- Merchant use this data index to call the API(step 14)
- **api_policy_init** uses given data index to verify isAuth attribute as the proof of successful completion of user side authentication(step 15).

9.2.1 Pros and cons

Calling an external service for every request may significantly downgrade performance. Furthermore, you should responsible for the availability of the external service as well as the security of it.

9.2.2 Configure the external service

We developed a simple service to store and retrieve data. This service is part of API Server implementation:

<https://github.com/authsaz/isambookdemo-api/blob/master/server.py>

```

38     @app.route('/internal/token', methods=['POST'])
39     def internal_token():
40         content = request.json
41         res = {'status':'1'}
42         if(content['action'] == "save"):
43             tokens[content['uuid']] = content['data']
44         if(content['action'] == "load"):
45             res['data'] = tokens[content['uuid']]
46         return Response(json.dumps(res), content_type='application/json')
47

```

Although the code for this service is part of API Server implementation, it can not be accessed through API Server reverse proxy, since we haven't defined any junction for exposing it. Its address in backend API Server is <http://apps.authsaz.com:7001/internal/token>. This service accepts two actions, **save** to store a data and **load** to retrieve it. A **uuid** is used to refer to a stored data object.

POST ▾ http://apps.authsaz.com:7001/internal/token

Send

200 OK TIME 0 ms SIZE 15 B

JSON Auth Query Header 1 Docs

```

1+ {
2 "action": "save",
3 "uuid": "3ed5b2b0-1722-11e9-b56e-0800200c9a66",
4+ "data": {
5   "text": "hello"
6 }
7 }
```

Preview ▾ Header 4 Cookie Timeline

```

1+ {
2 "status": "1"
3 }
```

On successful save action, a JSON object with the status value equal to 1 will be returned.

POST ▾ http://apps.authsaz.com:7001/internal/token

Send

200 OK TIME 16 ms SIZE 42 B

JSON Auth Query Header 1 Docs

```

1+ {
2 "action": "load",
3 "uuid": "3ed5b2b0-1722-11e9-b56e-0800200c9a66"
4 }
```

Preview ▾ Header 4 Cookie Timeline

```

1+ {
2 "status": "1",
3 "data": {
4   "text": "hello"
5 }
6 }
```

On successful load action, a JSON object with the status value equal to 1 along with a data object will be returned.

9.2.3 Configure infomaps to Activate external service mechanism

The infomap related to External Service Mechanism is **api_policy_WebService.js**. We have imported this infomap in chapter 7.

IBM Security Access Manager

Home Appliance Dashboard Monitor Analysis and Diagnostics Secure Web Settings **Secure Access Control** Secure Federation Configuration

Policy

- Access Control
- Authentication
- Risk Profiles
- Attributes
- Obligations
- OpenID Connect and API Protection
- Information Points
- Extensions

Manage

- Devices
- Grants
- Database Maintenance
- SCIM Configuration
- Push Notification Providers
- MMFA Configuration
- Attribute Source

Global Settings

- Advanced Configuration
- User Registry
- Runtime Parameters
- Template Files
- Mapping Rules**
- Distributed Session Cache
- Server Connections
- Point of Contact
- Access Policies

Navigate to **Secure Access Control > Global Settings: Mapping Rules**.

Mapping Rules	
Add	Import
Edit	Delete
Export	Replace
Category: InfoMap	
USC_PasswordReset_Success	Category: InfoMap
api_policy_JWT	Category: InfoMap
api_policy_STS	Category: InfoMap
api_policy_WebService	Category: InfoMap
api_policy_cache	Category: InfoMap
api_policy_config	Category: InfoMap
api_policy_init	Category: InfoMap
api_policy_resp_finish	Category: InfoMap
api_policy_resp_init	Category: InfoMap
jrsasign	Category: InfoMap

Select **api_policy_config** infomap from *Mapping Rules* table.

Click **Edit**.

```
/*
var implementation = 2;

switch(implementation){
    case 1: // cache
        importMappingRule("api_policy_cache");
        conf = {};
        break;
    case 2: // WebService
        importMappingRule("api_policy_WebService");
        conf = {
            webservice: "http://apps.authsaz.com:7001/internal/token"
        };
        break;
    case 3: // JWT
        importMappingRule("api_policy_JWT");
        conf = {secret: secret };
        break;
    case 4: // STS
        importMappingRule("api_policy_STS");
        conf = {};
        break;
    default:
        throw "no implementation is selected.";
}
```

Name:

Category:

Save  **Close**

Modify **api_policy_config** mapping rule by assigning **2** to variable **implementation** and “**http://apps.authsaz.com:7001/internal/token**” to variable **webservice**.

Click **Save**.

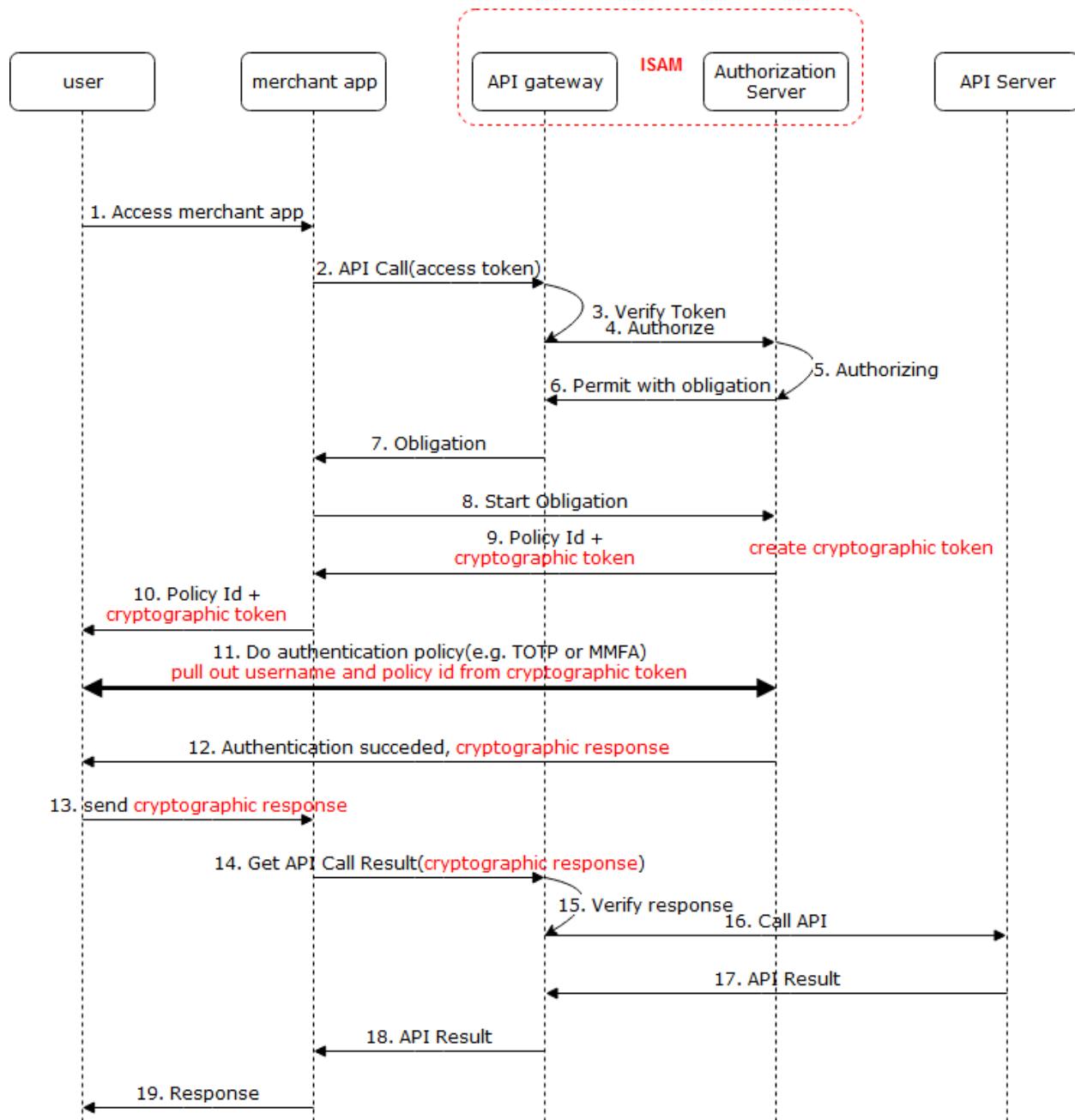
Deploy the changes using the link in the yellow warning message.

Repeat section 7.9 (Test API policies) tests to see the changes in the way of exchanging challenge and proof. The challenge for External Service Mechanism would be a random UUID like this:

https://api-gateway.authsaz.com:444/mga/sps/authsvc?PolicyId=urn:ibm:security:authentication:asf:api_policy_otp_resp&challenge=3805b093-f9c1-4956-ab3f-52bbdc7159a3&callback=http://merchant.authsaz.com:5000/apicallback

9.3 Branching Authentication Policy Using a custom encryption mechanism

In this approach, we used some cryptographic techniques to create a trust relationship between Merchant and user. The following steps shown by red color should be carried out to create a trust relationship:



- The **api_policy_init** mechanism in the obligation policy (in step 8) stores an object comprising these attributes in a JWT token as a cryptographic object and waits for the proof:
 - a. username
 - b. User side authentication Policy Id
 - c. timestamp
 - d. A random string as a nonce
 - e. isAuthenticated boolean attribute
- Policy id along with JWT token will be sent back to Merchant(step 9)
- Merchant forward the policy id and JWT token to the user(step 10)
- The user starts an authentication flow regarding the given Policy Id. **api_policy_resp_init** verify given JWT token and extract its claims and then move on to next authentication steps(step 11)
- After successful completion of authentication steps, **api_policy_resp_finish** issue a new JWT token with the same claims in the first JWT token except for the value of the isAuthenticated attribute is made true(step 12).
- User forward second JWT to Merchant(step 13)
- Merchant use second JWT token to call the API(step 14)
- **api_policy_init** verify given JWT and check whether the isAuthenticated attribute in this token is true or not(step 15).

9.3.1 Pros and cons

For implementing this approach it's not required to have a federation license and in terms of performance, this approach may be a good one but it brings some security concerns. As there is no facility in ISAM to store custom confidential data, cryptographic keys should be hardcoded in infomaps.

9.3.2 Configure infomaps to Activate custom encryption mechanism

The infomap related to Custom Encryption Mechanism is **api_policy_JWT.js**. We have imported this infomap in chapter 7.

The screenshot shows the IBM Security Access Manager dashboard. At the top, there are several navigation links: Home (Appliance Dashboard), Monitor (Analysis and Diagnostics), Secure Web Settings, Secure Access Control (which is highlighted with a red box), Secure Federation, and Connect. Below the navigation bar is a main menu with three columns: Policy, Manage, and Global Settings. The Global Settings column contains a link to 'Mapping Rules' which is also highlighted with a red box.

Navigate to **Secure Access Control > Global Settings: Mapping Rules**.

The screenshot shows the 'Mapping Rules' table. At the top, there are buttons for Add, Import, Edit (which is highlighted with a red box), Delete, Export, and Replace. The table lists various mapping rules, each with a 'Category' field. One row, 'api_policy_config', is highlighted with a dashed blue box. The 'Edit' button for this row is also highlighted with a red box.

Category	Rule Name
InfoMap	USC_PasswordReset_Success
InfoMap	api_policy_JWT
InfoMap	api_policy_STS
InfoMap	api_policy_WebService
InfoMap	api_policy_cache
InfoMap	api_policy_config
InfoMap	api_policy_init
InfoMap	api_policy_resp_finish
InfoMap	api_policy_resp_init
InfoMap	jsrsasign

Select **api_policy_config** infomap from *Mapping Rules* table.

Click **Edit**.

```

Mapping Rules - api_policy_config

var secret = "CHANGE_ME";
var expiresInSeconds = 60;

/*
implementation:
1   cache
2   WebService
3   JWT
4   STS
*/

var implementation = 3;

switch(implementation){
    case 1: // cache
        importMappingRule("api_policy_cache");
        conf = {};
        break;
    case 2: // WebService
        importMappingRule("api_policy_WebService");
        conf = {
            webservice: "http://apps.authsaz.com:7001/internal/token"
        };
        break;
    case 3: // JWT
        importMappingRule("api_policy_JWT");
        conf = {secret: secret };
        break;
    case 4: // STS
}

Name: api_policy_config
Category: InfoMap

```

Save **Close**

Modify **api_policy_config** mapping rule by assigning **3** to variable **implementation** and a random string to the variable **secret**.

Click **Save**.

Deploy the changes using the link in the yellow warning message.

Repeat section 7.9 (Test API policies) tests to see the changes in the way of exchanging challenge and proof. The challenge for Custom Encryption Mechanism would be a JWT token like this:

```

https://api-
gateway.authsaz.com:444/mga/sps/authsvc?PolicyId=urn:ibm:security:authentication:asf:api_
policy_otp_resp&challenge=eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9.eyJpYXQiOjE1NDczODEz
MjEslmV4cCI6MTU0NzM4MTM4MSwic3ViljoidXNlcjEiLCJwb2xpY3kiOiJ1cm46aWJtOnNIY
3VyaXR5OmF1dGhlbnRpY2F0aW9uOmFzZjphcGlfcG9saWN5X290cF9yZXNwliwiaXNBdXR
oljmpmYWxzZSwidXVpZCI6ijczODUyNjljLTc0MTctNGQ4Ni05NTFILWM1NTc0MDExNDE5NiJ
9.HCMkDjvZ7LO7FGBguWx8BW2ClelHiKGMA4wz_Y4B1c&callback=http://merchant.auths

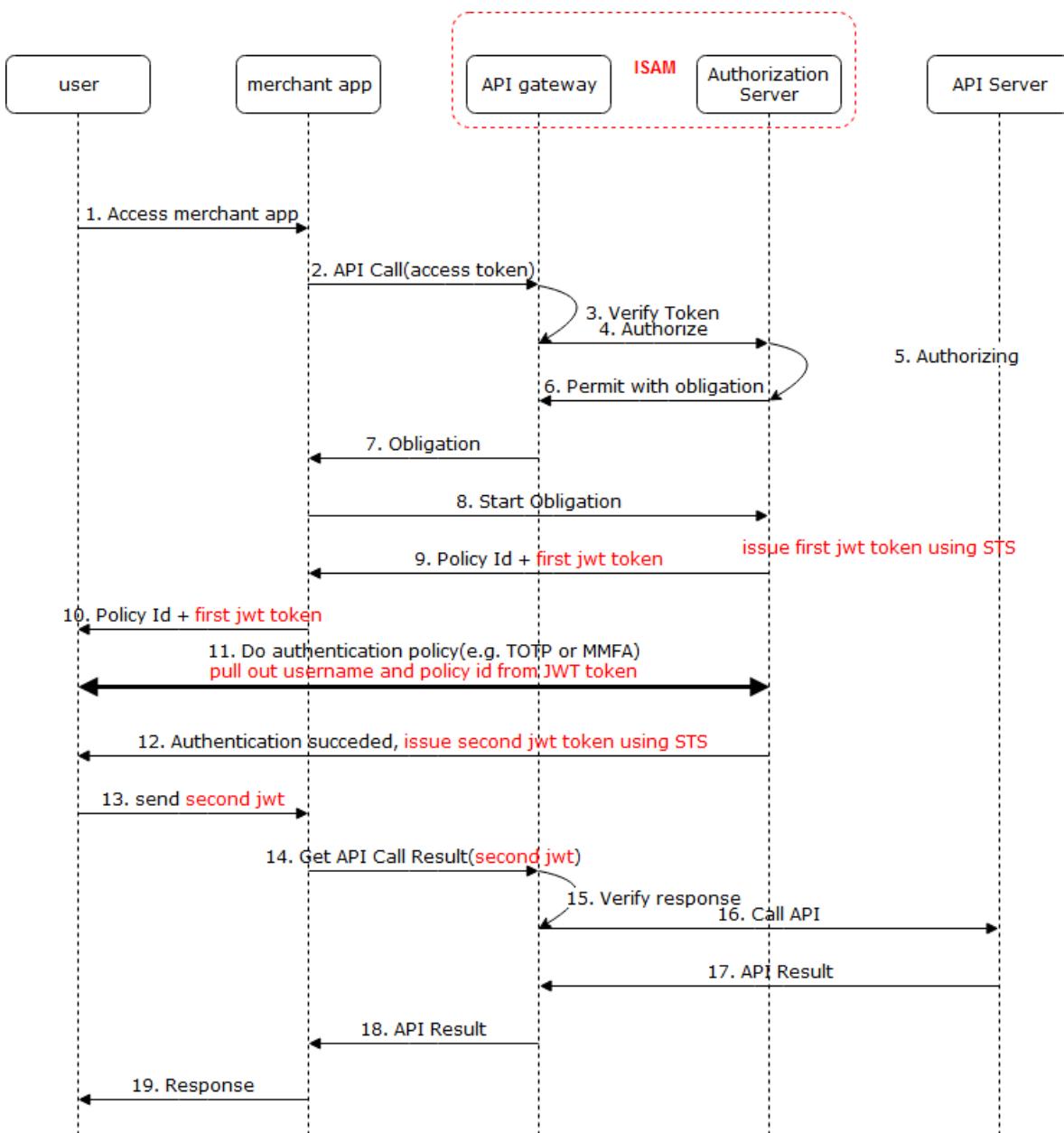
```

az.com:5000/apicallback

9.4 Branching Authentication Policy Using STS

The Security Token Service(STS) is a component of the federation runtime that accepts WS-Trust XML/SOAP requests for the validation and exchange of one security token type for another. It is used by the federation runtime, and can also be used by standalone WS-Trust clients, or by the ISAM reverse proxy for “TFIM-SSO” junctions.

In this section, we use STS to create a trust relationship between Merchant and user. The following steps shown by red color should be carried out to create a trust relationship:



- The **api_policy_init** mechanism in the obligation policy (in step 8) ask ISAM STS to issue a JWT token with these attributes and waits for the proof:
 - a. username
 - b. User side authentication Policy Id
 - c. timestamp
 - d. A random string as a nonce
 - e. isAuthenticated boolean attribute
- Policy id along with JWT token will be sent back to Merchant(step 9)

- Merchant forward the policy id and JWT token to the user(step 10)
- The user starts an authentication flow regarding the given Policy Id. **api_policy_resp_init** verify given JWT token using ISAM STS and extract its claims and then move on to next authentication steps(step 11)
- After successful completion of authentication steps, **api_policy_resp_finish** asks ISAM STS to issue a new JWT token with the same claims in the first JWT token except for the value of the isAuthenticated attribute is made true(step 12).
- User forward second JWT to Merchant(step 13)
- Merchant use second JWT token to call the API(step 14)
- **api_policy_init** verify given JWT using ISAM STS and check whether the isAuthenticated attribute in this token is true or not(step 15).

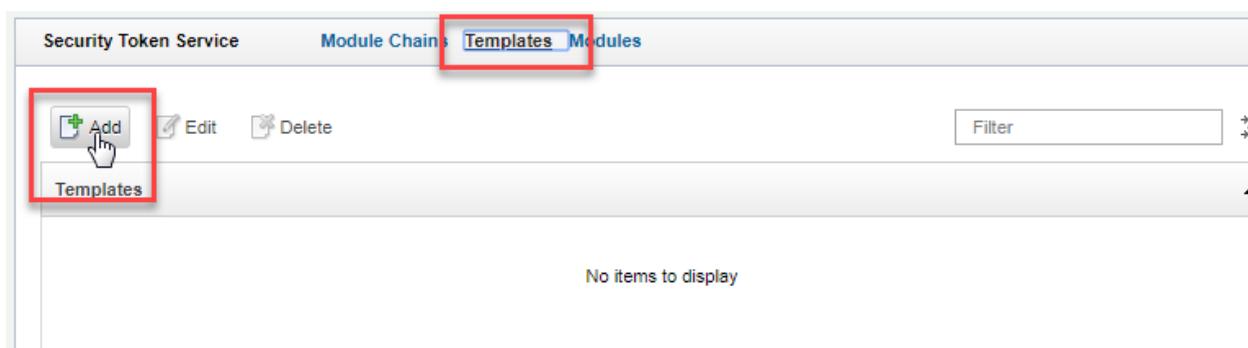
9.4.1 Configuring Chain Templates

First, we create an STS Chain Template. This defines an ordered list of Modules that will make up any chain built on this template. This section has been developed with the help of Leo Farrell's blog post and scripts titled "**OAuth: JWT as an Access Token on ISAM**"¹².

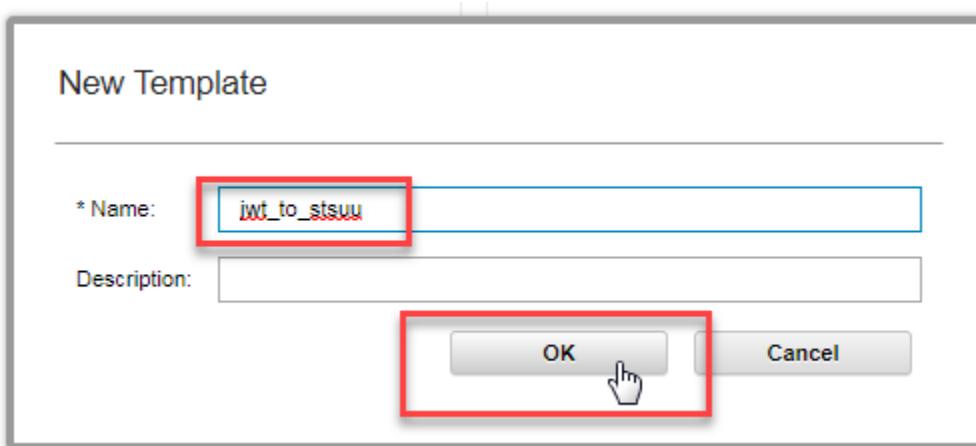
The screenshot shows the IBM Security Access Manager web interface. At the top, there is a navigation bar with links for Home, Monitor, Secure Web Settings, Secure Access Control, and Secure Federation. The Secure Federation link is highlighted with a red box. Below the navigation bar, there is a main menu with sections for Manage, Global Settings, and Global Keys. The Manage section is expanded, showing sub-options like Federations, Grants, and OpenID Connect and API Protection. Under Federations, the "Security Token Service" option is selected and highlighted with a red box. To the right of the Manage section, there are lists for Global Settings and Global Keys, each with several sub-options. The overall interface is a standard web-based administration tool with a clean, modern design.

Navigate to **Secure Federation > Manage: Security Token Service**.

¹² <https://www.ibm.com/blogs/security-identity-access/oauth-jwt-access-token/>

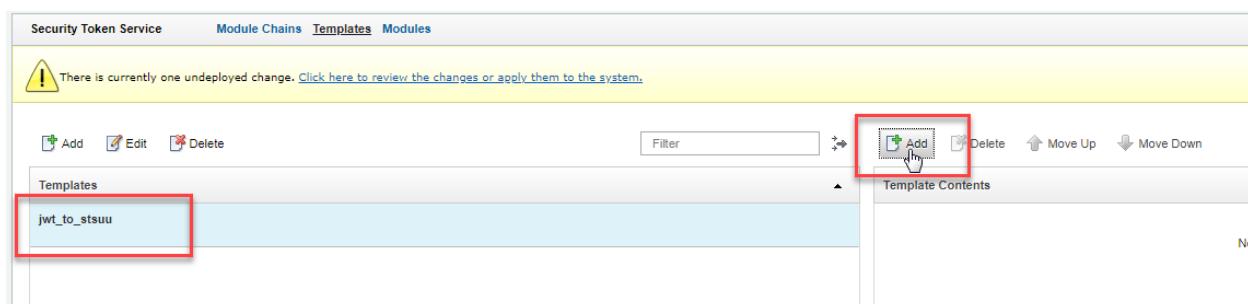


Click on the **Templates** menu and then click the **Add** button to create a new template.

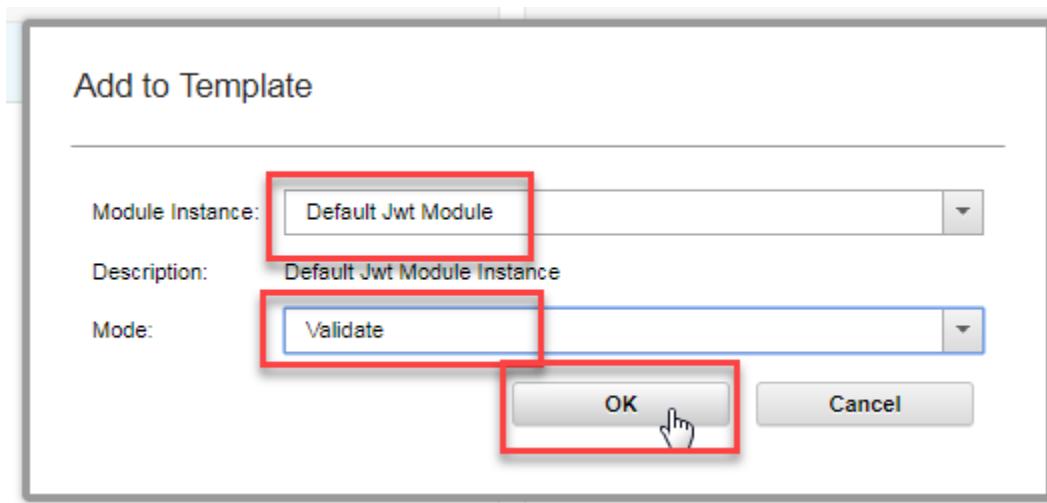


Enter **jwt_to_stsuum** as Name. Click **OK**.

The template has been created. Now we populate it by adding modules to the Template Contents.

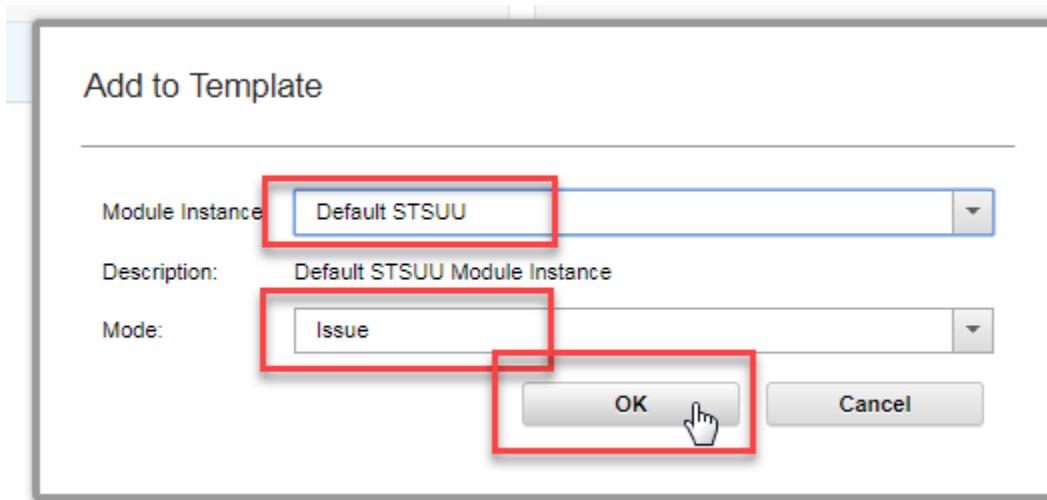


Select the newly added template and click **Add** on the right-hand panel to add a Module Instance to the template contents.



First, we're going to add a **Default Jwt Module** Instance in **Validate** mode. Enter these values and click **OK**.

Click **Add** again to add a second Module Instance.



Now we add a **Default STSUU** in **Issue** mode. Click **OK**.

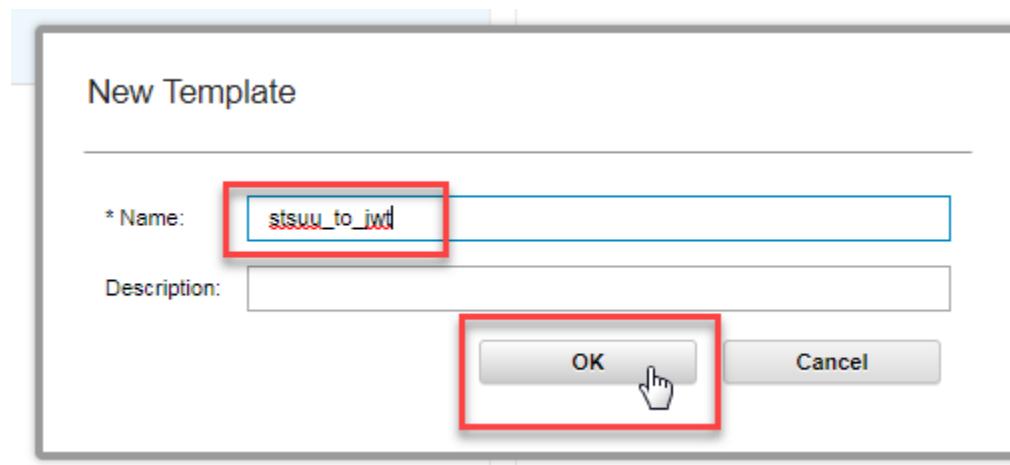
The template contents should look like this:

A screenshot of a software interface titled 'Templates'. On the left, there's a list of templates with 'jwt_to_stsuum' selected. On the right, the 'Template Contents' pane displays two entries: 'Default Jwt Module' with 'Mode: Validate' and 'Default STSUU' with 'Mode: Issue'.

Now, we need a second Template to issue JWT tokens.

A screenshot of the 'Templates' screen. The 'Add' button in the top-left toolbar is highlighted with a red box. The 'Template Contents' pane on the right shows the same two modules as the previous screenshot: 'Default Jwt Module' (Mode: Validate) and 'Default STSUU' (Mode: Issue).

Click on the **Templates** menu again and then click the **Add** button to create another template.

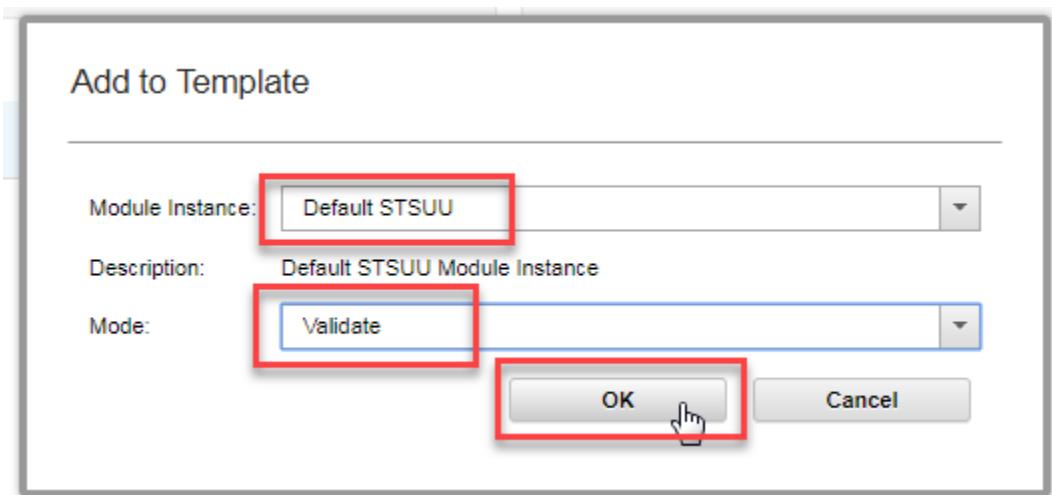


Enter **stsuum_to_jwt** as Name. Click **OK**.

Now the template has been created. We need to populate it by adding modules to the Template Contents.

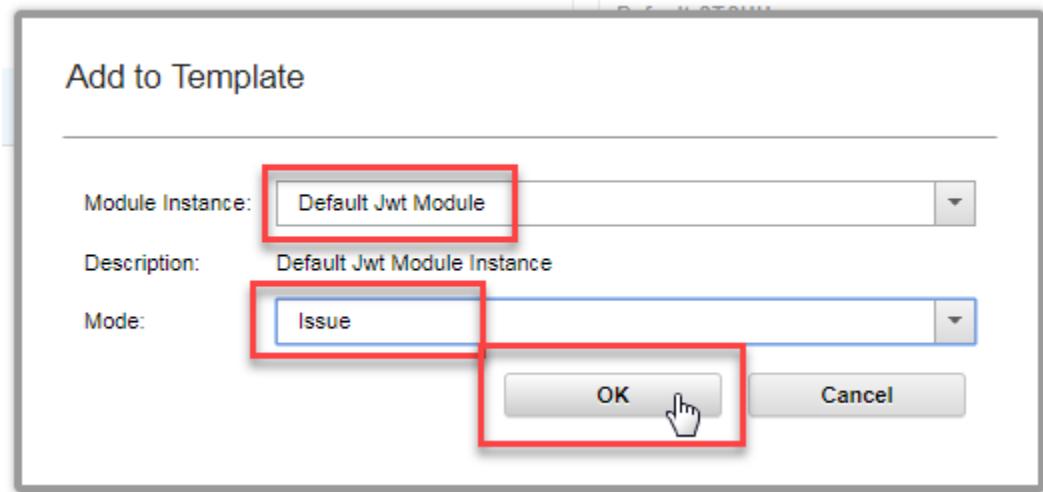
The screenshot shows the 'Templates' section of the Security Token Service. A template named 'stsuu_to_jwt' is selected and highlighted with a red box. On the right, the 'Template Contents' panel shows an 'Add' button, also highlighted with a red box.

Select the newly added template and click **Add** on the right-hand panel to add a Module Instance to the template contents.



First, we're going to add a **Default STSUU** Instance in **Validate** mode. Enter these values and click **OK**.

Click **Add** again to add a second Module Instance.



Now we add a **Default Jwt Module** in **Issue** mode. Enter these values and click **OK**.

Deploy the pending changes.

9.4.2 Configure Module Chains

We will now create a Module Chain from the new template.

As we've seen, the template determines the modules in the chain, the mode they will operate in, and the order in which they will run. The rest of the configuration is specified at the Module Chain level.

Still, in the Security Token Service screen, click on the **Module Chains** tab and then click **Add** to add a new Module chain.

New Module Chain

Overview **Lookup** Security Properties

* Name: **jwt_to_stsuu**

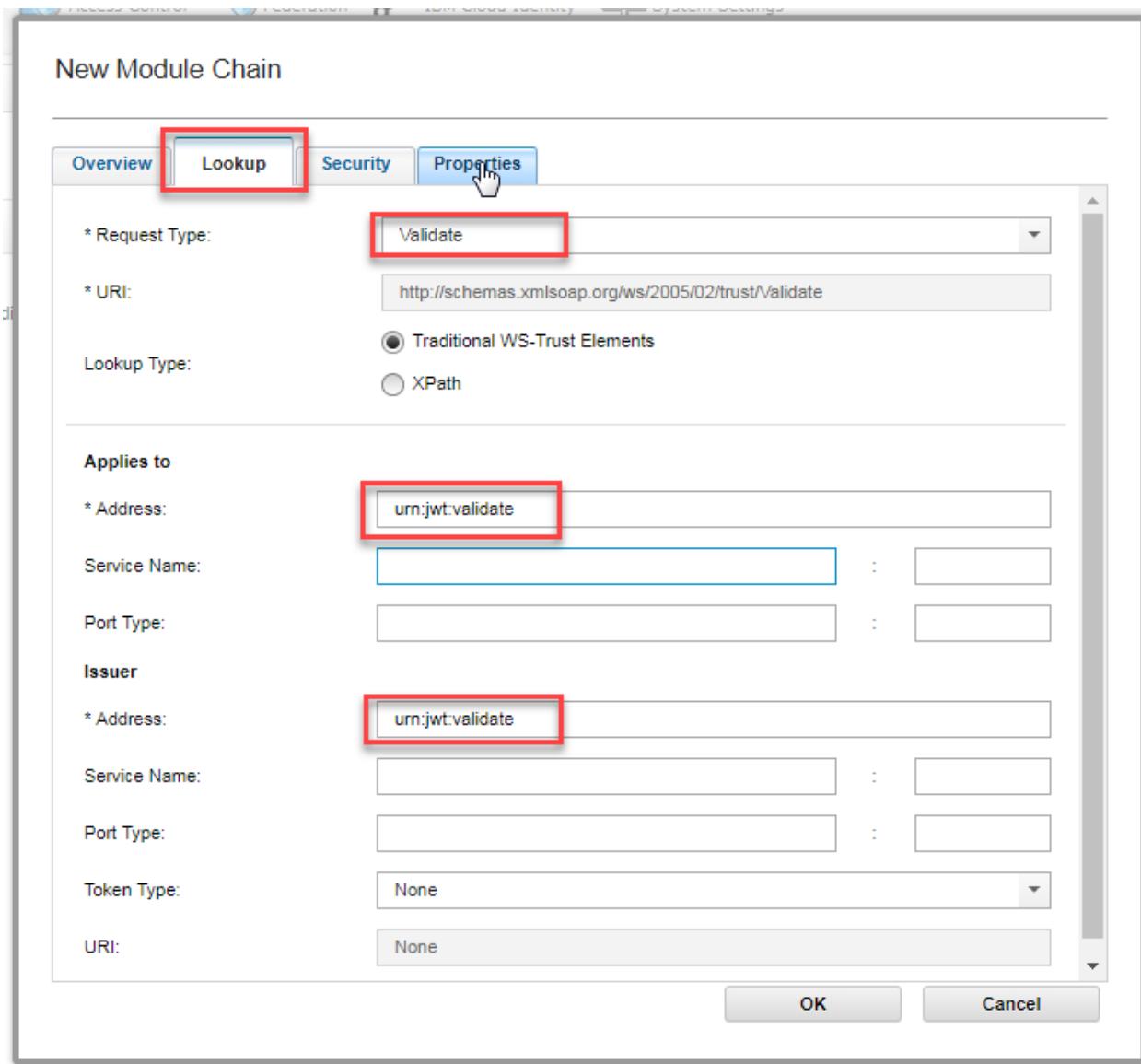
Description:

* Template: **jwt_to_stsuu**

Description:

Enter **jwt_to_stsuu** as the Name for the chain and provide a description. Select the **jwt_to_stsuu** Template for the chain.

Click on the **Lookup** tab.

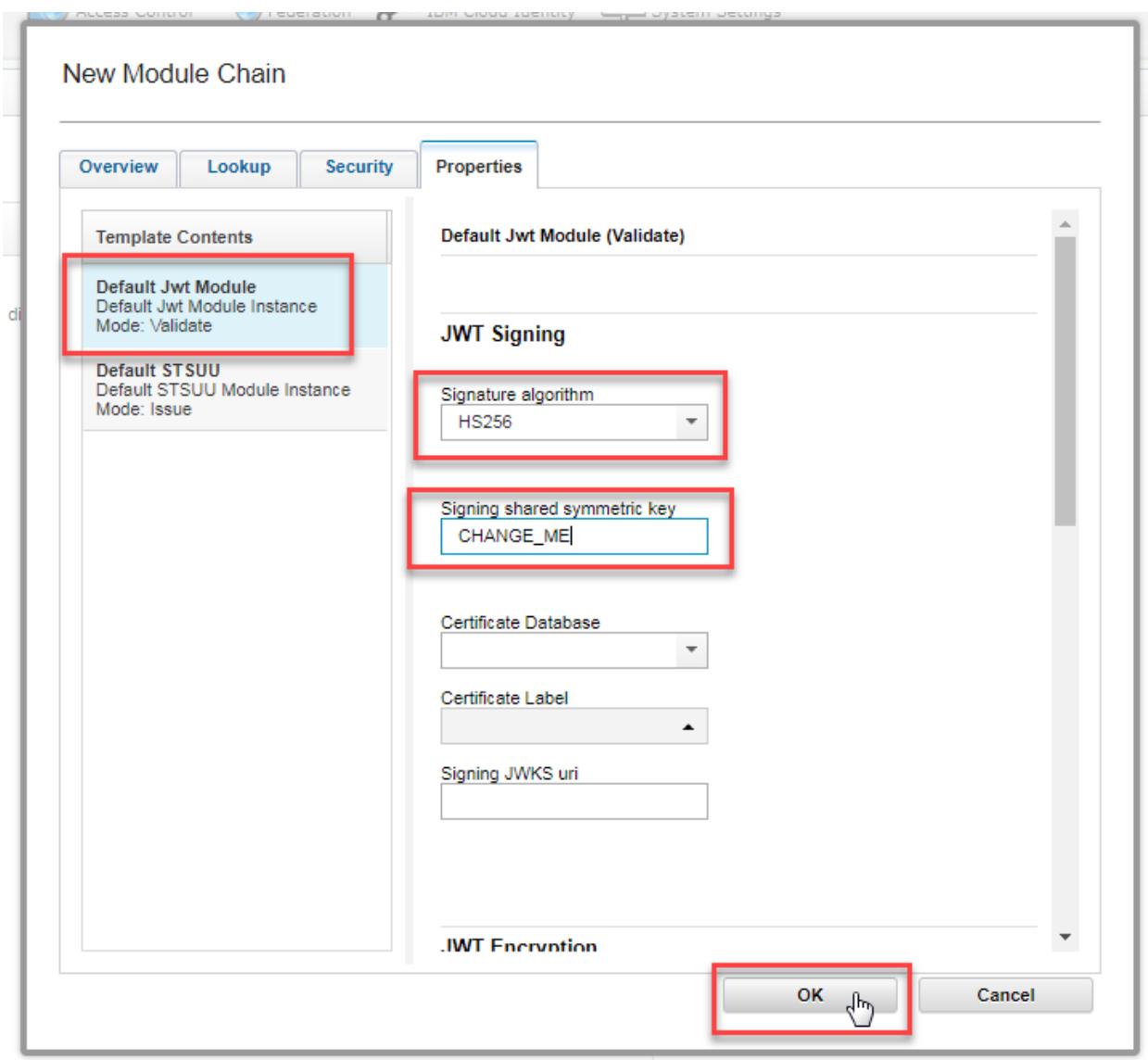


When a WS-Trust request arrives at the STS, the information provided in the **Lookup** tab of each defined Module Chain is used to determine which one should process the request. Only if all specified Lookup fields match the incoming WS-Trust request it is considered a match.

Select **Validate** from the *Request Type* drop-down list.

Enter **urn:jwt:validate** in the *Address box* under *Applies to*.

Enter **urn:jwt:validate** in the *Address box* under *Issuer*.



Click on the **Properties** tab.

In the **Properties** tab, we configure the chain-specific properties for each module in the chain.

Select the **Default Jwt Module** from the list of modules on the left-hand side. This opens the properties panel for that module.

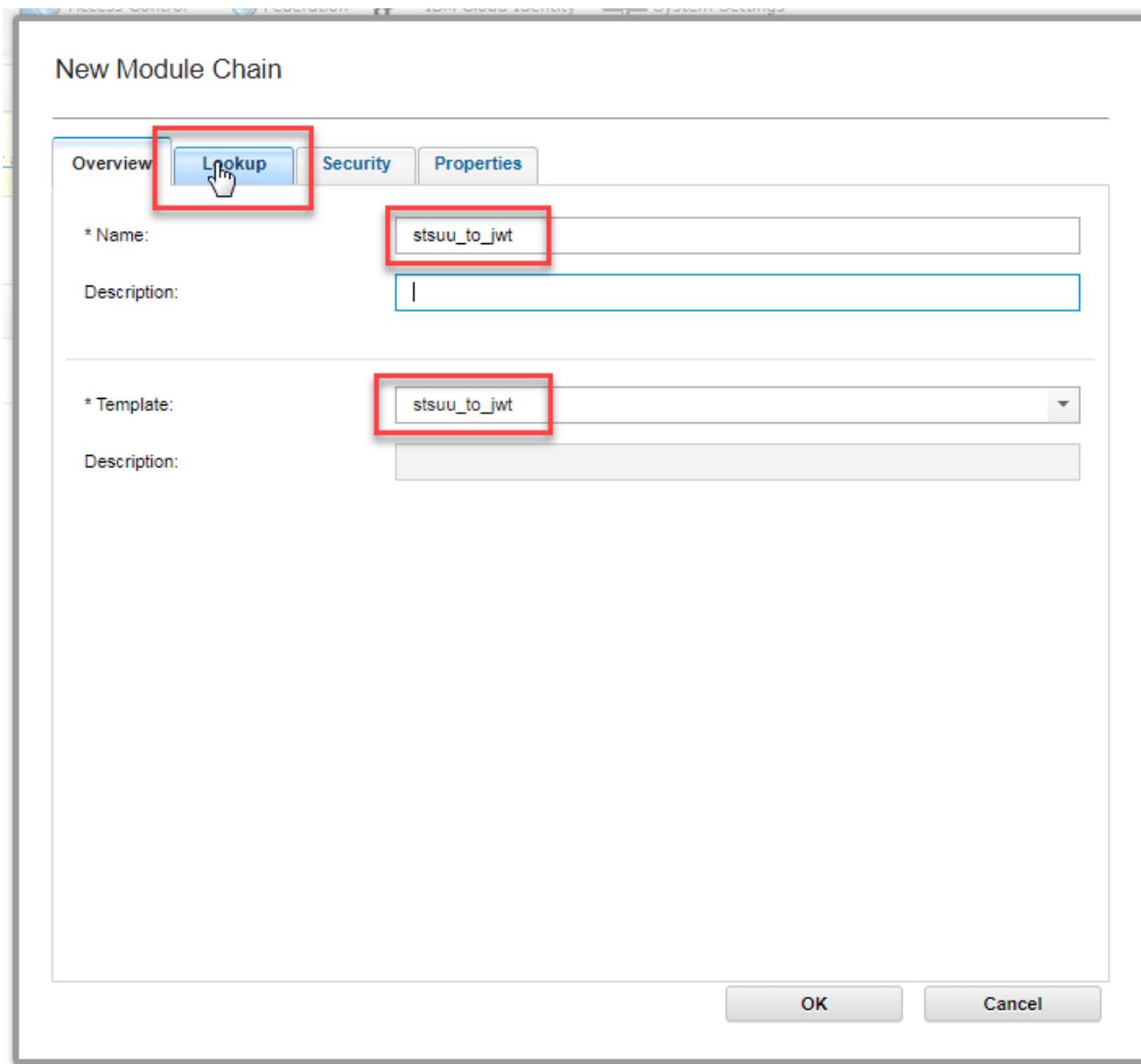
On the right-hand side, select **HS256** as *Signature algorithm* and enter a random string as *Signing shared symmetric key*.

The STSUU module does not have any chain-specific properties so we don't need to worry about that.

Module Chain configuration is now complete. Click **OK** to save the new Module Chain.

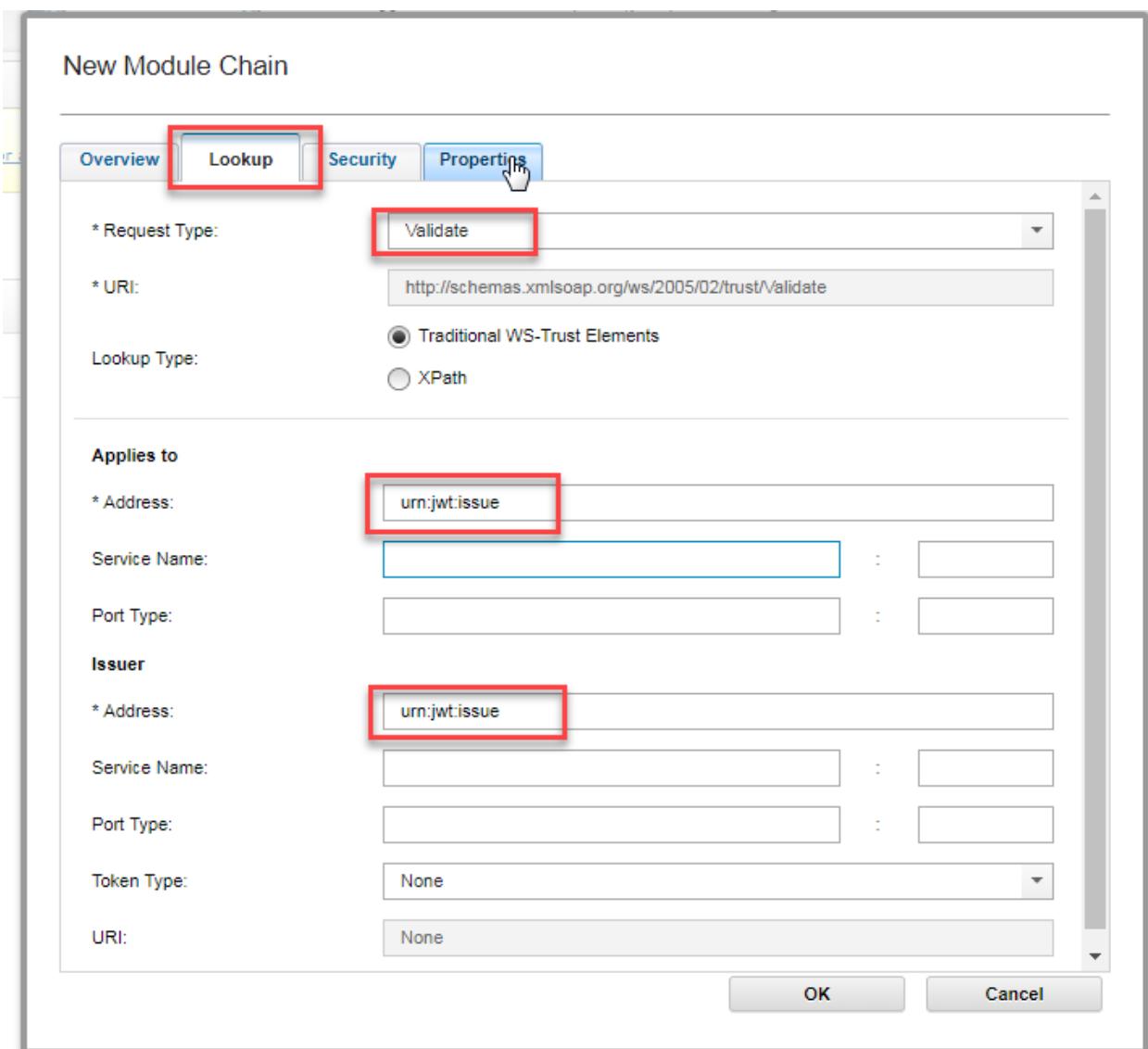
The screenshot shows the 'Security Token Service' interface with the 'Module Chains' tab selected. A yellow warning bar at the top states: 'There is currently one undeployed change. [Click here to review the changes or apply them to the system.](#)' Below the bar, there are buttons for 'Edit' and 'Delete'. A red box highlights the 'Add' button, which has a hand cursor icon over it. To the right of the buttons is a 'Filter' input field and a refresh icon. The main area displays a table titled 'Module Chains' with one row: 'jwt_to_stsuum'. The table has a header row with a small triangle icon.

Click on the **Module Chains** tab again and then click **Add** to add a new Module chain.



Enter **stsuu_to_jwt** as the Name for the chain and provide a description. Select the **stsuu_to_jwt** Template for the chain.

Click on the **Lookup** tab.

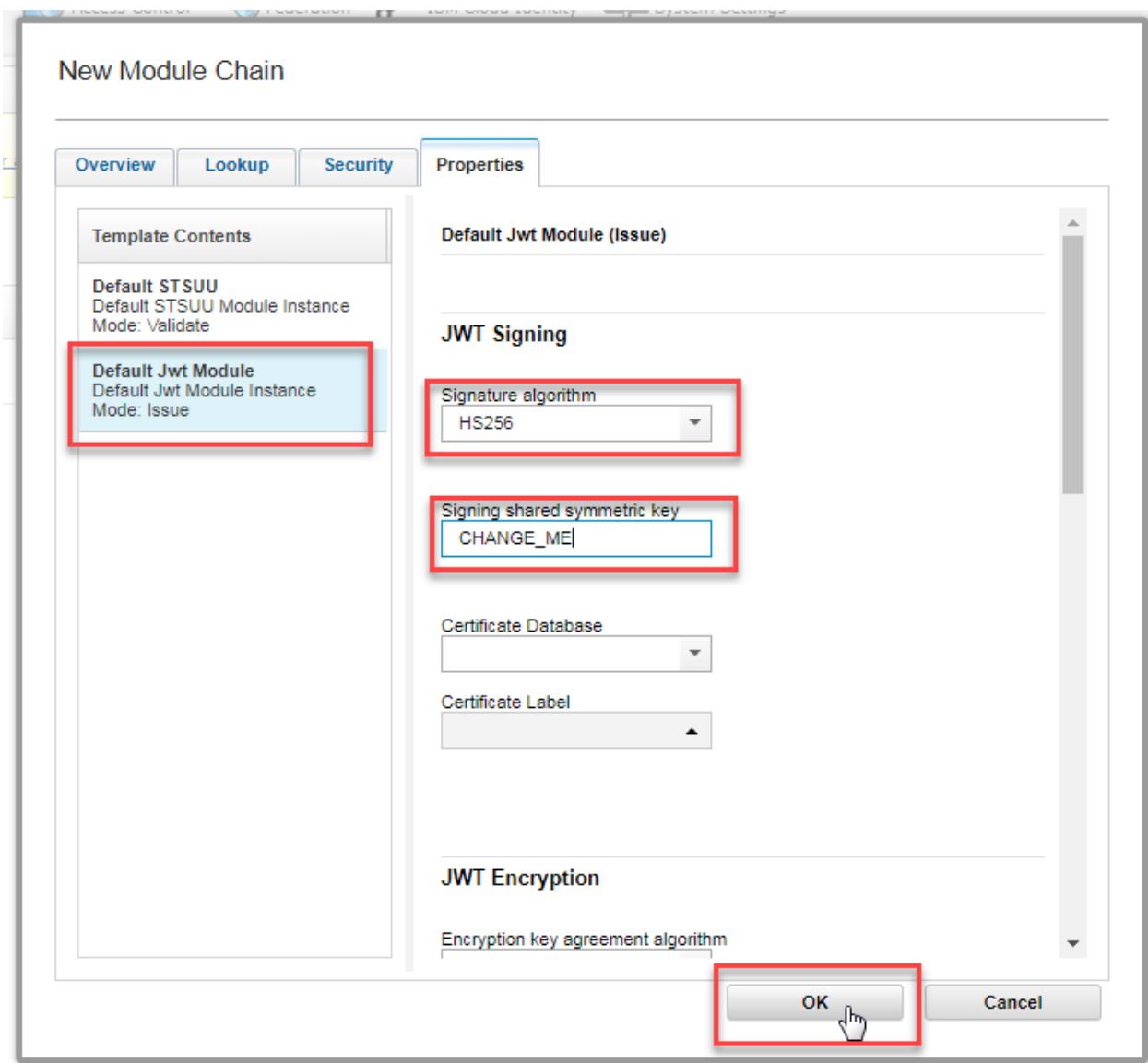


Select **Validate** from the *Request Type* drop-down list.

Enter **urn:jwt:issue** in the *Address* box under *Applies to*.

Enter **urn:jwt:issue** in the *Address* box under *Issuer*.

Click on the **Properties** tab.



In the **Properties** tab, we configure the chain-specific properties for each module in the chain.

The STSUU module does not have any chain-specific properties so we don't need to worry about it.

Select the **Default Jwt Module** from the list of modules on the left-hand side. This opens the properties panel for that module.

On the right-hand side, select **HS256** as *Signature algorithm* and enter the same value we used for *Signing shared symmetric key* in **jwt_to_stsuu** module chain as *Signing shared symmetric key*.

Module Chain configuration is now complete. Click **OK** to save the new Module Chain.

The screenshot shows the 'Security Token Service' interface with the 'Module Chains' tab selected. A yellow warning bar at the top states: 'There is currently one undeployed change. [Click here to review the changes or apply them to the system.](#)'

The new chain is shown in the list of Module Chains. **Deploy** pending changes.

9.4.3 Configure infomaps to Activate STS mechanism

The infomap related to STS Mechanism is **api_policy_STS.js**. We have imported this infomap in chapter 7.

The screenshot shows the 'IBM Security Access Manager' interface. The navigation bar includes links for Home, Monitor, Secure Web Settings, Secure Access Control (which is highlighted with a red box), Secure Federation, and Configuration. The main content area has three columns: Policy (Access Control, Authentication, Risk Profiles, Attributes, Obligations, OpenID Connect and API Protection, Information Points, Extensions), Manage (Devices, Grants, Database Maintenance, SCIM Configuration, Push Notification Providers, MMFA Configuration, Attribute Source), and Global Settings (Advanced Configuration, User Registry, Runtime Parameters, Template Files, Mapping Rules, Distribution Cache, Session Cache, Server Connections, Point of Contact, Access Policies). The 'Mapping Rules' link under Global Settings is also highlighted with a red box.

Navigate to **Secure Access Control > Global Settings: Mapping Rules**.

Mapping Rules	
Add	Import
Edit	Delete
Export	Replace
Category: InfoMap	
USC_PasswordReset_Success	Category: InfoMap
api_policy_JWT	Category: InfoMap
api_policy_STS	Category: InfoMap
api_policy_WebService	Category: InfoMap
api_policy_cache	Category: InfoMap
api_policy_config	Category: InfoMap
api_policy_init	Category: InfoMap
api_policy_resp_finish	Category: InfoMap
api_policy_resp_init	Category: InfoMap
jrsasign	Category: InfoMap

Select **api_policy_config** infomap from *Mapping Rules* table.

Click **Edit**.

```

/*
var implementation = 4;

switch(implementation){
    case 1: // cache
        importMappingRule("api_policy_cache");
        conf = {};
        break;
    case 2: // WebService
        importMappingRule("api_policy_WebService");
        conf = {
            webservice: "http://apps.authsaz.com:7001/internal/token"
        };
        break;
    case 3: // JWT
        importMappingRule("api_policy_JWT");
        conf = {secret: secret};
        break;
    case 4: // STS
        importMappingRule("api_policy_STS");
        conf = {};
        break;
    default:
        throw "no implementation is selected.";
}

```

Name:

Category:

Save **Close**

Modify **api_policy_config** mapping rule by assigning **4** to variable **implementation**.

Click **Save**.

Deploy the changes using the link in the yellow warning message.

Repeat section 7.9 (Test API policies) tests to see the changes in the way of exchanging challenge and proof. The challenge for STS Mechanism would be a JWT token like this:

https://api-gateway.authsaz.com:444/mga/sps/authsvc?PolicyId=urn:ibm:security:authentication:asf:api_policy_otp_resp&**challenge=eyJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE1NDczODEzODMsImV4cCI6MTU0NzM4MTQ0Mywic3ViljoidXNIcjEiLCJwb2xpY3kiOiJ1cm46aWJtOnNIY3VyaXR5OmF1dGhlbnRpY2F0aW9uOmFzZjphcGlfcG9saWN5X290cF9yZXNwliwiaXNBdXRoljpmYWxzZSwidXVpZCI6ljlhZDBhZmMOLTUwNzctNDk3ZC05MmM4LTcxM2U2NjcyMGYyYj9.Ym05qVeZiuvJfxX9D30xad4Gjn10yPVj3bYSiCxJl3Q&callback=http://merchant.authsaz.com:5000/apicallback**

10 Appendix I: API Developer Guide

This section provides a list of some important ISAM APIs along with a short description and sample requests and responses for each API.

10.1 OAuth APIs

Endpoint Name	Description
Authorization endpoint	An authorization URL where the resource owner grants authorization to the OAuth client to access the protected resource
	Sample URI: /mga/sps/oauth/oauth20/authorize
	Sample Input: https://github.com/authsaz/isambookdemo-httpproxy/blob/master/OAuth/GET_authorize.req.txt https://github.com/authsaz/isambookdemo-httpproxy/blob/master/OAuth/POST_authorize.req.txt
Token endpoint	Sample Output: https://github.com/authsaz/isambookdemo-httpproxy/blob/master/OAuth/GET_authorize.res.txt https://github.com/authsaz/isambookdemo-httpproxy/blob/master/OAuth/POST_authorize.res.txt
	A token request URL where the OAuth client exchanges an authorization grant for an access token and an optional refresh token.
	Sample URI: /mga/sps/oauth/oauth20/token
Introspect endpoint	Sample Input: https://github.com/authsaz/isambookdemo-httpproxy/blob/master/OAuth/POST_token_CODE.req.txt https://github.com/authsaz/isambookdemo-httpproxy/blob/master/OAuth/POST_token_REFRESHTOKEN.req.txt
	Sample Output: https://github.com/authsaz/isambookdemo-httpproxy/blob/master/OAuth/POST_token_CODE.res.txt https://github.com/authsaz/isambookdemo-httpproxy/blob/master/OAuth/POST_token_REFRESHTOKEN.res.txt
	A URL where an access_token can be inspected by an oauth_client.
	Sample URI: /mga/sps/oauth/oauth20/introspect

	<p>Sample Input: https://github.com/authsaz/isambookdemo-httpprograms/blob/master/OAuth/POST_introspect.req.txt</p>
	<p>Sample Output: https://github.com/authsaz/isambookdemo-httpprograms/blob/master/OAuth/POST_introspect.res.txt</p>
Logout endpoint	<p>A URL where you can end a session by revoking an access_token. The token must be provided in the Authorization header or a session cookie must be used.</p>
	<p>Sample URI: /mga/sps/oauth/oauth20/logout</p>
	<p>Sample Input: https://github.com/authsaz/isambookdemo-httpprograms/blob/master/OAuth/POST_logout.req.txt</p>
	<p>Sample Output: https://github.com/authsaz/isambookdemo-httpprograms/blob/master/OAuth/POST_logout.res.txt</p>

10.2 Mobile APIs

Endpoint Name	Description
MMFA details endpoint	<p>Retrieve further details required for registration, such as the token endpoint</p>
	<p>Sample URI: /mga/sps/mmfa/user/mgmt/details?client_id={client}</p>
	<p>Sample Input: https://github.com/authsaz/isambookdemo-httpprograms/blob/master/IBM_Verify/GET_details.req.txt</p>
	<p>Sample Output: https://github.com/authsaz/isambookdemo-httpprograms/blob/master/IBM_Verify/GET_details.res.txt</p>
MMFA token endpoint	<p>When the value of OAuth Scope equals to “mmfaAuthn”, some other parameters like “device_type”, “platform_type”, “os_version”, ... is needed to be sent to ISAM to register the mobile as a MMFA device.</p>
	<p>Sample URI: /mga/sps/oauth/oauth20/token</p>
	<p>Sample Input: https://github.com/authsaz/isambookdemo-httpprograms/blob/master/IBM_Verify/POST_token.req.txt</p>
	<p>Sample Output: https://github.com/authsaz/isambookdemo-httpprograms/blob/master/IBM_Verify/POST_token.res.txt</p>

Authenticator Endpoints	MMFA authentication mechanism management endpoint
Sample URI: /scim/Me?attributes=urn:ietf:params:scim:schemas:extension:isam:1.0:MMFA:Authenticator:userPresenceMethods	
Sample Input: https://github.com/authsaz/isambookdemo-httpproxy/blob/master/IBM_Verify/PATCH_scim_USERPRESENCE.req.txt https://github.com/authsaz/isambookdemo-httpproxy/blob/master/IBM_Verify/PATCH_scim_FINGERPRINT.req.txt https://github.com/authsaz/isambookdemo-httpproxy/blob/master/IBM_Verify/PATCH_scim_AUTHENTICATORS.req.txt https://github.com/authsaz/isambookdemo-httpproxy/blob/master/IBM_Verify/GET_totp.req.txt	
Sample Output: https://github.com/authsaz/isambookdemo-httpproxy/blob/master/IBM_Verify/PATCH_scim_USERPRESENCE.res.txt https://github.com/authsaz/isambookdemo-httpproxy/blob/master/IBM_Verify/PATCH_scim_FINGERPRINT.res.txt https://github.com/authsaz/isambookdemo-httpproxy/blob/master/IBM_Verify/PATCH_scim_AUTHENTICATORS.res.txt https://github.com/authsaz/isambookdemo-httpproxy/blob/master/IBM_Verify/GET_totp.res.txt	
Transaction pendings Endpoint	<p>This is a filter of the /scim/Me endpoint, requesting that only the transactionsPending and attributePending schemas be returned</p> <p>Sample URI: /scim/Me?attributes=urn:ietf:params:scim:schemas:extension:isam:1.0:MMFA:Transaction:transactionsPending,urn:ietf:params:scim:schemas:extension:isam:1.0:MMFA:Transaction:attributesPending </p> <p>Sample Input: https://github.com/authsaz/isambookdemo-httpproxy/blob/master/IBM_Verify/GET_scim_PENDING_1.req.txt https://github.com/authsaz/isambookdemo-httpproxy/blob/master/IBM_Verify/GET_scim_PENDING_2.req.txt</p>

	<p>Sample Output: https://github.com/authsaz/isambookdemo-httpprojects/blob/master/IBM_Verify/GET_scim_PENDING_1.res.txt https://github.com/authsaz/isambookdemo-httpprojects/blob/master/IBM_Verify/GET_scim_PENDING_2.res.txt</p>
Start MMFA Transaction	<p>Start MMFA Transaction</p> <p>Sample URI: /mga/sps/apiauthsvc?MmfaTransactionId={transactionId}</p> <p>Sample Input: https://github.com/authsaz/isambookdemo-httpprojects/blob/master/IBM_Verify/POST_MmfaTransactionId.req.txt</p> <p>Sample Output: https://github.com/authsaz/isambookdemo-httpprojects/blob/master/IBM_Verify/POST_MmfaTransactionId.res.txt</p>
Signed challenge Endpoint	<p>Submit Signed challenges</p> <p>Sample URI: /mga/sps/apiauthsvc?StatId={statId}</p> <p>Sample Input: https://github.com/authsaz/isambookdemo-httpprojects/blob/master/IBM_Verify/PUT_statId_SIGNEDCHALLENGE_1.req.txt https://github.com/authsaz/isambookdemo-httpprojects/blob/master/IBM_Verify/PUT_statId_SIGNEDCHALLENGE_2.req.txt</p> <p>Sample Output: https://github.com/authsaz/isambookdemo-httpprojects/blob/master/IBM_Verify/PUT_statId_SIGNEDCHALLENGE_1.res.txt https://github.com/authsaz/isambookdemo-httpprojects/blob/master/IBM_Verify/PUT_statId_SIGNEDCHALLENGE_2.res.txt</p>

10.3 Dashboard APIs

Endpoint Name	Description
grant	<p>View a list of your OAuth 2.0 authorization grants.</p> <p>Details of your OAuth 2.0 authorization grant.</p> <p>Update your OAuth 2.0 authorization grant.</p> <p>Delete your OAuth 2.0 authorization grant.</p>

	<p>Sample URI: /mga/sps/mga/user/mgmt/grant</p> <p>Sample Input: https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/GET_grant.req.txt https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/GET_grant_UUID.req.txt https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/PUT_grant_UUID.req.txt https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/DELETE_grant_UUID.req.txt</p> <p>Sample Output: https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/GET_grant.res.txt https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/GET_grant_UUID.res.txt https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/PUT_grant_UUID.res.txt https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/DELETE_grant_UUID.res.txt</p>
authenticators	<p>View a list of your authenticators.</p> <p>Details of your authenticator.</p> <p>Delete your authenticator.</p> <p>Delete your authenticator, authentication mechanisms.</p> <p>Sample URI: /mga/sps/mmfa/user/mgmt/authenticators</p> <p>Sample Input: https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/GET_authenticators.req.txt https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/GET_authenticators_UUID.req.txt https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/DELETE_authenticators_UUID.req.txt https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/DELETE_auth_methods_UUID.req.txt</p> <p>Sample Output: https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/GET_authenticators.res.txt https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/GET_authenticators_UUID.res.txt https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/DELETE_authenticators_UUID.res.txt https://github.com/authsaz/isambookdemohttprequests/blob/master/Dashboard/DELETE_auth_methods_UUID.res.txt</p>

QR	Generate a QR code containing authorization code and client id
	<p>Sample URI: /mga/sps/mmfa/user/mgmt/qr_code?code={code}&client_id={client}</p> <p>Sample Input: https://github.com/authsaz/isambookdemo- httprequests/blob/master/Dashboard/GET_qr_code_json.req.txt https://github.com/authsaz/isambookdemo- httprequests/blob/master/Dashboard/GET_qr_code_gif.req.txt</p> <p>Sample Output: https://github.com/authsaz/isambookdemo- httprequests/blob/master/Dashboard/GET_qr_code_json.res.txt https://github.com/authsaz/isambookdemo- httprequests/blob/master/Dashboard/GET_qr_code_gif.res.txt</p>
clients	<p>Register dynamic clients View a list of registered dynamic clients Details of your dynamic client Delete a dynamic client</p>
	<p>Sample URI: /mga/sps/mga/user/mgmt/clients /mga/sps/oauth/oauth20/register/{definition}?client_id={client}</p> <p>Sample Input: https://github.com/authsaz/isambookdemo- httprequests/blob/master/Dashboard/POST_register.req.txt https://github.com/authsaz/isambookdemo- httprequests/blob/master/Dashboard/GET_clients.req.txt https://github.com/authsaz/isambookdemo- httprequests/blob/master/Dashboard/GET_register_CLIENTID.req.txt https://github.com/authsaz/isambookdemo- httprequests/blob/master/Dashboard/DELETE_register_CLIENTID.req.txt</p> <p>Sample Output: https://github.com/authsaz/isambookdemo- httprequests/blob/master/Dashboard/POST_register.res.txt https://github.com/authsaz/isambookdemo- httprequests/blob/master/Dashboard/GET_clients.res.txt https://github.com/authsaz/isambookdemo- httprequests/blob/master/Dashboard/GET_register_CLIENTID.res.txt https://github.com/authsaz/isambookdemo- httprequests/blob/master/Dashboard/DELETE_register_CLIENTID.res.txt</p>

11 Appendix II: IBM Verify connectivity to ISAM by forwarding ports

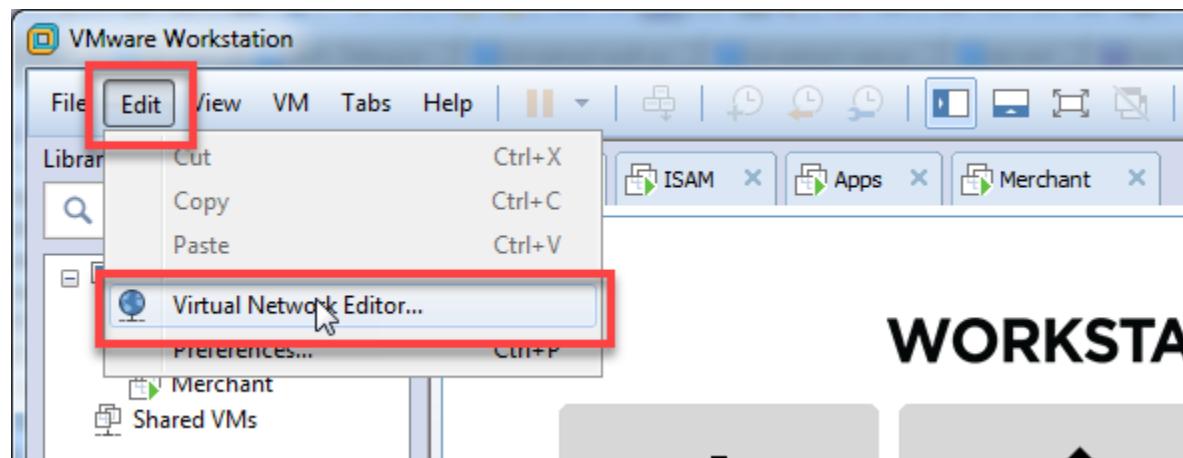
If there is a route from the mobile device to the Host machine, we can provide access for the mobile device to ISAM Reverse Proxy by configuring NAT port forwarding under VMware.

In case there is no route from the mobile device to Host IP address, We can use remote port forwarding. A simple tool for creating a remote port forward link is *plink*.

The following sections will describe both NAT port forwarding under VMware and remote port forwarding under *plink*.

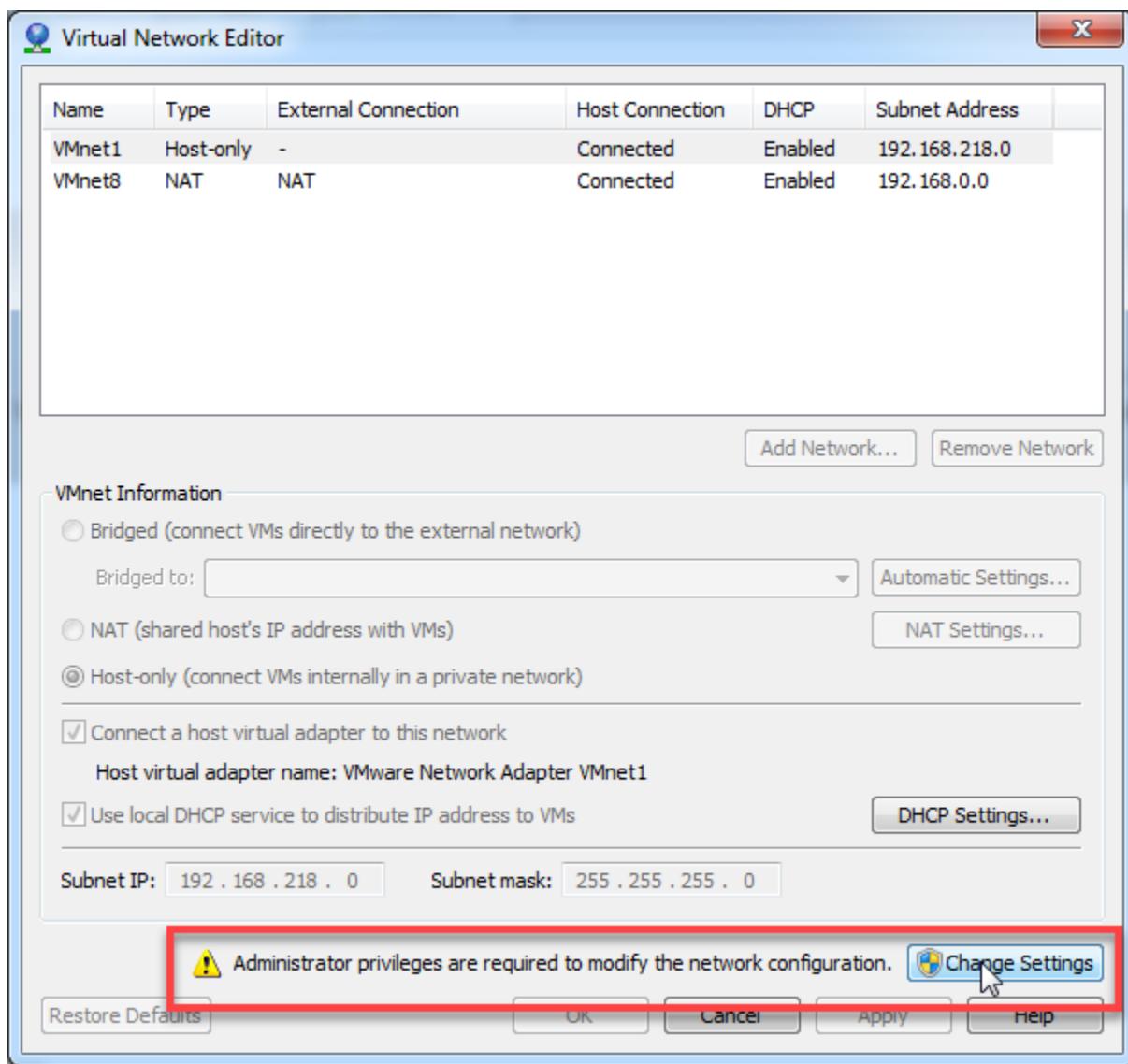
11.1 VMware NAT port forwarding

This section provides a step by step guide in order to forward Host port 444 to ISAM port 444.

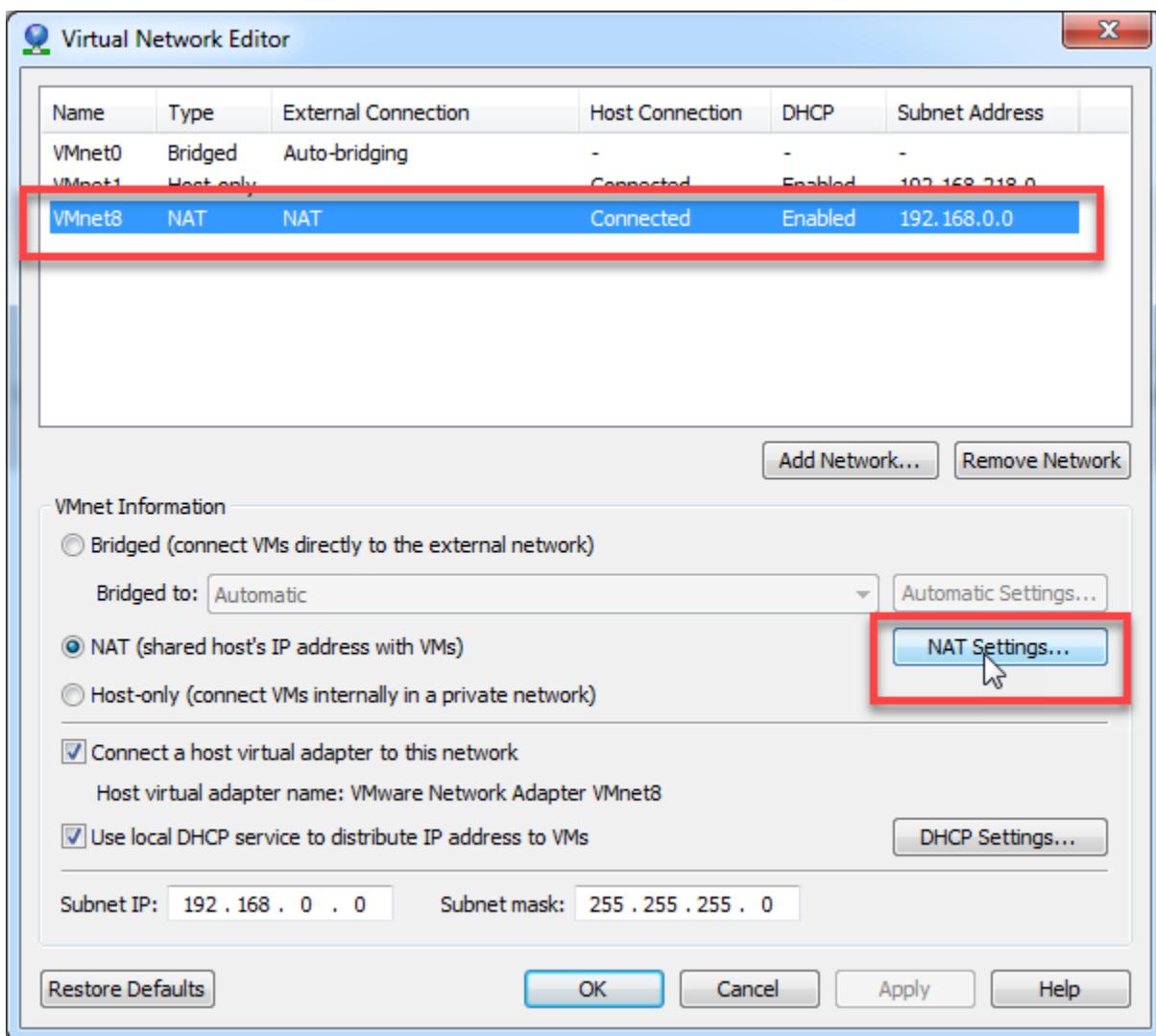


Run VMware Workstation.

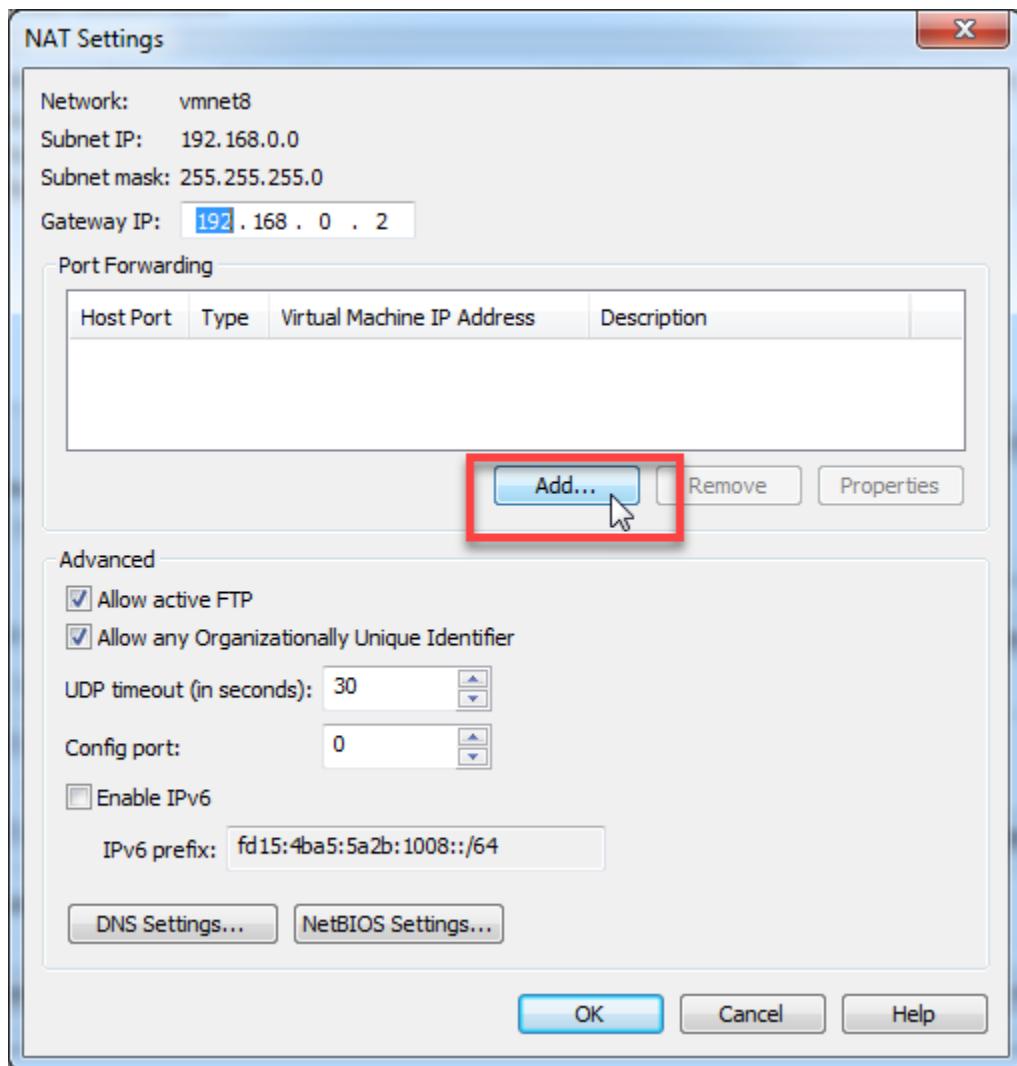
Navigate to **Edit > Virtual Network Editor...**



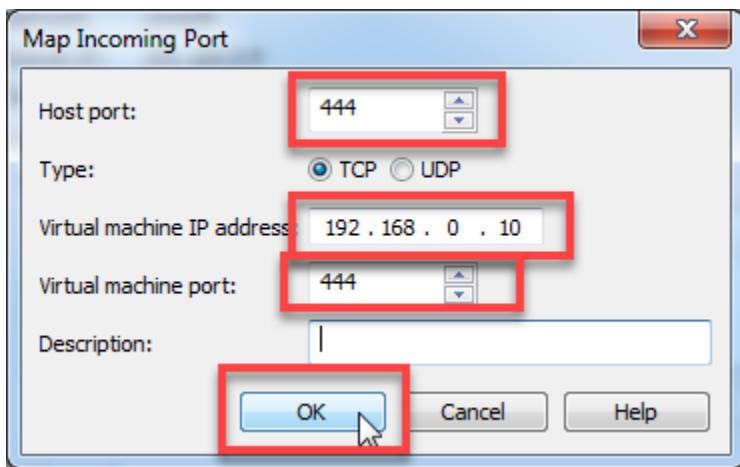
Click **Change Settings** button.



Click **NAT Settings...** button.



Click **Add...** Button.



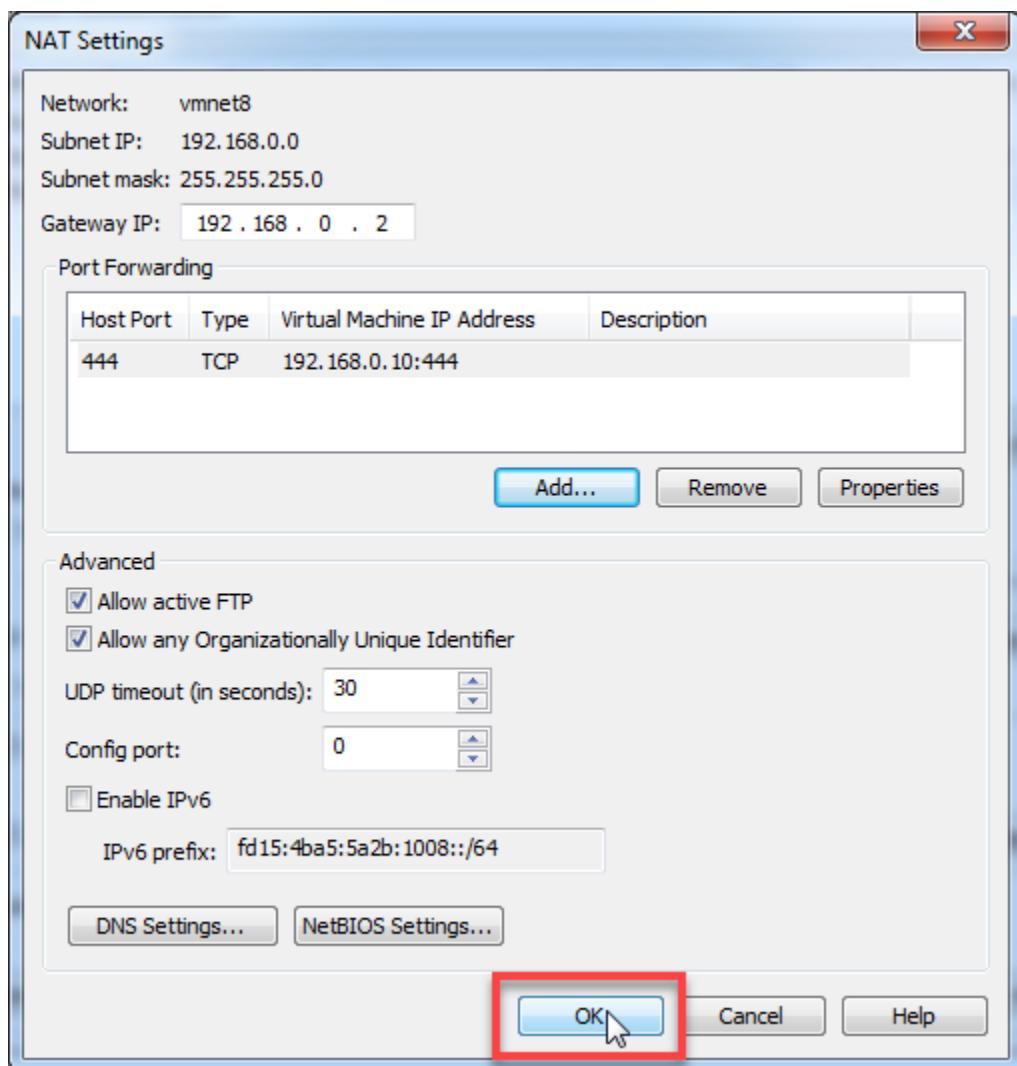
Enter **444** as the *Host port*.

Select the radio button for **TCP**.

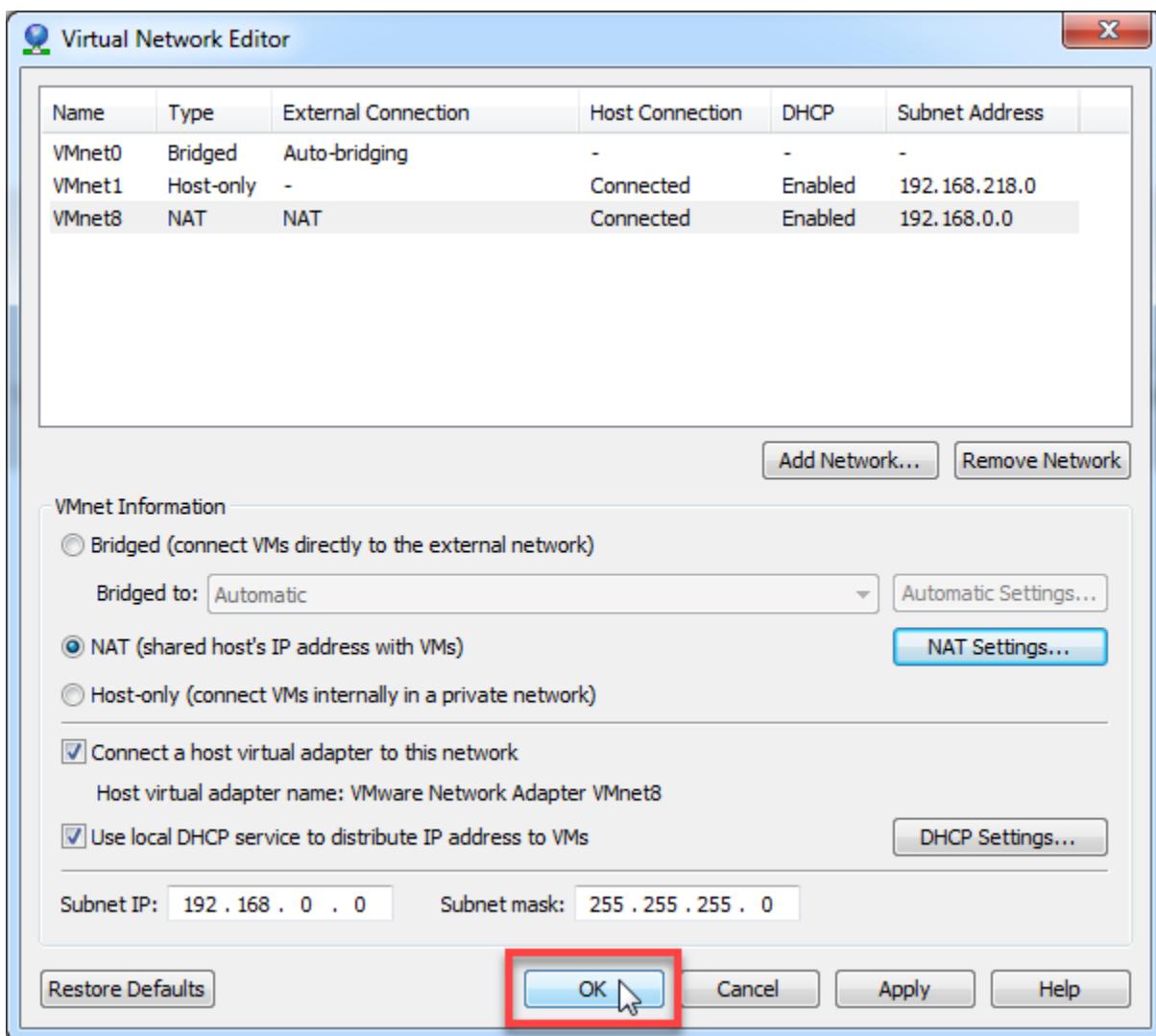
Enter **192.168.0.10** as *Virtual machine IP address*.

Enter **444** as *Virtual machine port*.

Click **Ok**.



In NAT Settings form Click **Ok**.



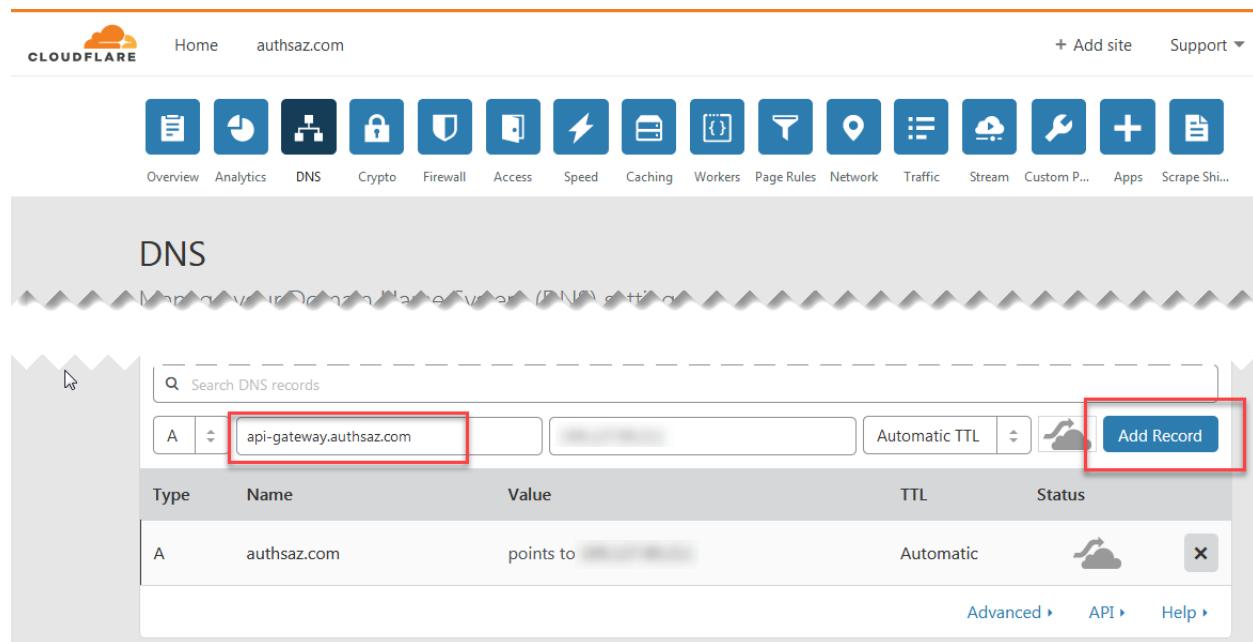
In Virtual Network Editor form click **Ok**.

11.2 PLINK remote port forwarding

To create a remote port forwarding link you must have a server in which is accessible by both the mobile device and Host operating system. Using *plink* to create remote port forwarding require an SSH server to be run on this server.

In this section, we describe a scenario in which both the Host system and mobile device are connected to the internet and we have a server with a valid IP address. So both Host and mobile device can connect to this server.

As all connections from the mobile device to ISAM Reverse proxies has been made by domain name, it is required to register corresponding domain names for the valid IP address of our server.



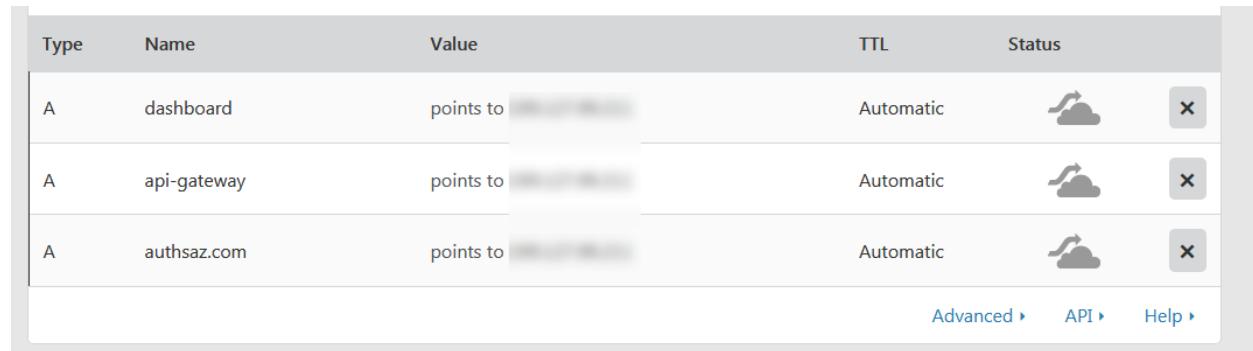
The screenshot shows the Cloudflare DNS management interface. At the top, there's a navigation bar with the Cloudflare logo, 'Home', 'authsaz.com', '+ Add site', and 'Support'. Below the navigation is a row of icons representing various services: Overview, Analytics, DNS, Crypto, Firewall, Access, Speed, Caching, Workers, Page Rules, Network, Traffic, Stream, Custom P..., Apps, and Scrape Shi... The 'DNS' icon is highlighted. The main area is titled 'DNS' and contains a search bar with 'Search DNS records'. Below the search bar, there's a dropdown menu set to 'A' and a text input field containing 'api-gateway.authsaz.com'. To the right of this input field is a red box highlighting the 'Add Record' button, which is also highlighted with a red box. Below this, a table lists existing DNS records:

Type	Name	Value	TTL	Status
A	authsaz.com	points to [REDACTED]	Automatic	[Cloudflare icon]

At the bottom of the table are links for 'Advanced', 'API', and 'Help'.

We used Cloudflare DNS service to register DNS records for the following domains:

- api-gateway.authsaz.com
- dashboard.authsaz.com
- authsaz.com



This screenshot shows a list of DNS records for the domain 'authsaz.com'. The table has columns for Type, Name, Value, TTL, and Status. There are three entries:

Type	Name	Value	TTL	Status
A	dashboard	points to [REDACTED]	Automatic	[Cloudflare icon]
A	api-gateway	points to [REDACTED]	Automatic	[Cloudflare icon]
A	authsaz.com	points to [REDACTED]	Automatic	[Cloudflare icon]

At the bottom of the table are links for 'Advanced', 'API', and 'Help'.

It is worth mentioning that the IP address that will be resolved for these domain names is different in Host machine and mobile device. It is because that in Host machine we added these records in the local host file and the IP corresponding to each of these domains would be fetched

from local host file, not DNS server, but for the mobile device, their IP address would be resolved from the configured DNS server



Now download 32/64 bit version of *plink* from following links to the Host machine:

- **32 bit:** <https://the.earth.li/~sgtatham/putty/latest/w32/plink.exe>
- **64 bit:** <https://the.earth.li/~sgtatham/putty/latest/w64/plink.exe>

Run this command from the Host operating system and on prompt enter root password for your server:

```
> plink.exe -l root YOUR_SERVER_IP_ADDRESS -R  
YOUR_SERVER_IP_ADDRESS:444:192.168.0.10:444 -R  
YOUR_SERVER_IP_ADDRESS:443:192.168.0.10:443
```

YOUR_SERVER_IP_ADDRESS is your server IP address. By running this command, all TCP packets received at port **443** on the server will be forwarded to VM IP **192.168.0.10** port **443** and all TCP packets received at port **444** on the server will be forwarded to VM IP **192.168.0.10** port **444**.

12 Notices

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to Authsaz Group, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. Authsaz Group, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to Authsaz Group for the purposes of developing, using, marketing.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© **Authsaz Group 2019.** Portions of this code are derived from Authsaz Group. Sample Programs.

© Copyright Authsaz Group 2019. All rights reserved.

If you are viewing this information in softcopy form, the photographs and color illustrations might not be displayed.