ACOS 6.0.8
Application Access Management Guide

December, 2025

Table of Contents

Getting Started

Application Access Management (AAM) optimizes Authentication, Authorization, and Accounting (AAA) for client-server traffic. It centralizes control for managing access, eliminates the need to manage individual services on multiple servers, and simplifies login through single sign-on (SSO) technology. By providing a consistent way to access services, AAM ensures a highly available and scalable means to ensure the proper authentication of client-server traffic.

The following topics are covered:

Understanding AAA

Authentication, Authorization, Accounting (AAA) is a collective mechanism used in IP-based network management and policy administration. AAA is used to manage credentials, provide profiles for what different roles can perform, and track resources.

The following topics are covered:

Authentication

Authentication is a process of confirming and asserting the identity of user upon providing login credentials. Using username/password combinations, challenge and response questions, token cards, and other methods, authentication can be established or denied.

Authorization

Authorization is a process of granting or denying a user access to network resources. After the user is authenticated, authorization services determine which resources the user can access and which operations the user is allowed to perform.

Accounting

Accounting is a process of keeping track of a user's activity while accessing the network resources. Accounting services enables the logging and recording of usage information, including the amount of time spend in the network, the services accessed while there and the amount of data transferred during the session.

Components and Features

The following topics are covered:

Logon Portal

Logon Portal – The user sign-on interface. By using a request-reply exchange or using a web-based form, ACOS obtains the user's credentials and uses a backend AAA server to verify these credentials.

Online Certificate Status Protocol (OCSP)

Online Certificate Status Protocol (OCSP) – OCSP is a service that provides certificate verification and eliminates the need to import certificate revocation list (CRL) files to the ACOS device. The CRLs are maintained on the OCSP responder (server). When a client sends its certificate as part of a request for a secured service, ACOS first sends the certificate to the OCSP responder for verification. After the certificate is verified, the client can access secured services.

Authentication Relay

Authentication Relay – When configured for authentication relay, ACOS offloads the user's AAA servers by contacting the backend AAA servers on behalf of the clients, and after a server responds, ACOS caches the reply and uses this reply for subsequent client requests.

Security Assertion Markup Language (SAML)

Security Assertion Markup Language (SAML) – The SAML protocol authenticates on the front-end. SAML is an XML-based open standard data format that allows authentication and authorization data between an identity provider (IdP) and a service provider (SP). This allows the ACOS to act as a service provider to grant resource services to clients and delegate authentication and authorization to the identity provider.

For more information about SAML, see References.

AAA Health Monitoring and Load Balancing

AAA Health Monitoring and *Load Balancing* – Load balances authentication traffic among a group of AAA servers. ACOS supports custom health checks for LDAP, RADIUS, Kerberos, and OCSP.

AAA Policies

The following topics are covered:

Understanding AAA Policies

An authentication policy uses rules, an authentication template, and an authorization policy to complete server load balancing (SLB) authentication on web and database applications. This process associates one or more authentication methods with different criteria that the user can configure (for example, internal and external IP addresses, or the URI path that is accessed by the session). The Authentication, Authorization, and Accounting (AAA) policy combines the authentication type and authorization policy to help manage user access to applications using authentication and authorization.

To define a policy, the user must configure at least one of the following rules:

- Access Control Lists (ACL), which defines the IP address from where the request originates.
- Uniform resource identifier (URI), which defines which web page the request is trying to access.
- The resulting actions, as follows:
  - Allow
  - Deny
  - Enables the authentication log

The following components are crucial in a policy:

- Authentication template, which defines how the user is authenticated.

  Authentication templates contain the authentication-server profile and logon - portal profile.
- Authorization policy, which determines whether the user is allowed to access a web page based on the list of permissions that are associated with a rule.

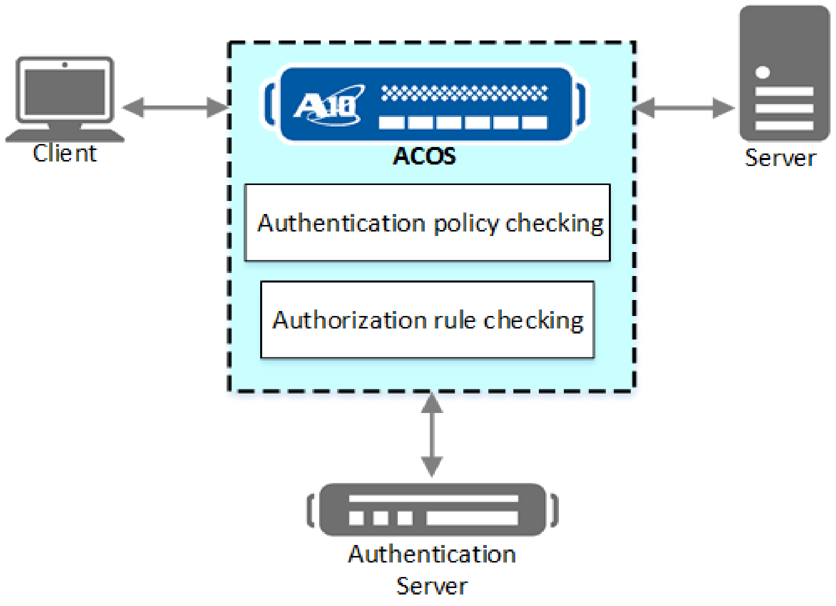
  Authorization occurs after authentication is complete and determines the privileges that are associated with the user who is being authenticated.

NOTE: The user must only need to use the rules that are relevant to the user's environment. Although authentication templates and authorization policies are not required, they are crucial to creating an authentication policy. It is recommended that the user must configure at least one template and one policy.

Understanding AAA Policy Workflow

Figure 1 illustrates the flow of the authentication and authorization policy checking in a typical scenario.

Figure 1 : Policy Checking Workflow



The following steps provide a high-level view of the process:

1. An HTTP request from the client is received by the ACOS device on which the AAA policy is configured.
2. If the HTTP request does not contain the HTTP authentication header, or the correct cookie, the ACOS device replies to the client with a 401 responses or an HTML logon form.
3. The AAA policy checking process begins:

   The process involves selecting the authentication template based on the matching criteria of the AAA policy, (for example, the session's IP, URI path, domain name, etc.).

   a. The user enters a username and password in the HTTP log-on form and submits the form through the client browser.
   b. The ACOS device parses the HTTP request to retrieve the username and password. The client's credentials and other attributes that are specific to the authentication method are forwarded on to the external authentication server.

   c. The authentication server receives the authentication information, and the result is passed to the data path thread, and the authentication process forwards the result back to the data path thread.
   d. If the matching AAA policy has an authorization policy configured, then the authorization check is completed, and the authentication is successful.

4. When authentication is successful, the client's original HTTP request is forwarded to the server.

   If authentication relay is enabled, the client's credentials are attached to the request, and the ACOS device creates a user session to track the authenticated client.

5. When the server's response is received, the ACOS device retrieves the current session, creates a secured session cookie, and stores the cookie to the session. The cookie is inserted in the HTTP response before the response is forwarded to the client.
6. The cookie is compared with later requests so that requests from an authenticated client are directly forwarded to the server.

Managing Access Control

The user can manage access to web sites and databases in one of the following ways:

- Creating an AAA Policy
- Defining AAA Policy with URIs

NOTE: To know more about the enhancement feature of the EP to access control http/https request, based on the ID Token of the OpenID connect input, in a given http header, see JWT (JSON Web Token) Authorization (AAM with JWT).

Resource Access Based On AAA Rules

Although users may pass authentication and authorization on a VIP, they do not have access to all the resources within that VIP. When a client first passes authentication and authorization, they have access to a resource configured under a specific AAA rule within an AAA policy. If the client accesses a new resource that is configured under a different AAA rule, and the new resource is configured under the same authentication template and authorization policy, the client is then granted access. If the new resource is configured under a different authentication template or under a different authorization policy, the AAM module first attempts to use the previous credentials to authenticate and authorize the client for the new resource. If that fails, then the client will be prompted to enter new credentials in order to access the resource.

If a client is authenticated and authorized for different AAA rules, they do not need to re-authenticate in order to re-access a previous AAA-rule. During the client’s session, all authentication and authorization permissions are retained.

The same process applies if the client tries to access resources on a different VIP, as long as both VIPs are configured in the same cookie-domain. For more information about resource access across VIPs, see Getting Started.

If authentication is done using a SAML deployment, the AAM module does not have access to the client’s credentials. In this case, the AAM module will ask the user to re-enter their authentication credentials, even if they are the same as the previous credentials. For more information about SAML, see Security Assertion Markup Language (SAML).

Defining AAA Policy with an Access Control List

Access control lists (ACL) are rules that are applied to ports or IP addresses. Configuring these rules allow the user to manage access to web sites. The user can use an IP standard access list, which is based on the source IP address, or an IP extended access list, which is based on both the source and destination IP addresses. For the IP standard access list, The user must specify a value between 1 and 99, and for the extended access lists filter, The user must specify a value between 100 and 199. ACLs can be configured as a rule during authentication checking. The appropriate action, depending on the result of the rule checking, is taken.

The user can apply an ACL to a wildcard VIP. The ACL controls the IP addresses and protocol ports that are allowed to access the VIP. The ACOS device can have only one wildcard VIP that does not have an ACL applied to it. The wildcard VIPs in the example deployment in this chapter all use ACLs.

- ACLs on the Outside ACOS Device's Wildcard VIPs:
    - Outbound – Permits IP traffic from IP addresses in the range 172.16.24.32-63 to any destination IP address. This is the client address range.
    - Inbound – Permits IP traffic from any source IP address to destination IP addresses in the 172.16.24.32-63 range.
- ACLs on the Inside ACOS Device's Wildcard VIPs:
    - Outbound – Denies traffic from any IP address in the range 172.16.24.32-63 to host address 172.16.242.33, which is the floating IP address used by VRRP-A in the sample deployment.
    - Permits traffic from addresses in the range 172.16.24.32-63 to any destination.
    - Inbound – Permits IP traffic from any source IP address to destination IP addresses in the 172.16.24.32-63 range. This is the client address range.

      For non-matching traffic, the ACOS device handles traffic that does not match the ACL as follows:
- If the ACOS device's configuration contains a wildcard VIP that does not use an ACL, the traffic is handled by that wildcard VIP.

  The configuration can contain one wildcard VIP that does not use an ACL. The ACOS device does not support more than one wildcard VIP without an ACL.
- If the configuration does not contain a wildcard VIP with no ACL, the traffic is routed at Layer 3.

Defining AAA Policy with URIs

The user can add more granular authorization control to an AAM deployment with authorization that is based on a URI. This additional level of control also specifies the URIs that an end-user can access.

NOTE: The current release supports this feature for RADIUS and LDAP.

The following topics are covered:

URI Access List

In this solution, the user can access the following URI:

```
ACOS-device/partition/VIP/portnum/service-type
```

For example, the user may be allowed to access only a specific service on a VIP that is load balanced by a pair of ACOS devices in partition “p1”. In this case, the AAA server might have the following entries:

```
ACOS1/p1/198.51.100.10/80/http
ACOS2/p1/198.51.100.10/80/http
```

The actual syntax for the entries depends on the AAA server type.

URI Configuration

To deploy this feature, the user must complete some configuration on the ACOS device and on the AAA server.

Configuring the ACOS Device

On the ACOS device, the user must enable the authorization-check option in the authentication-server profile for the AAA server.

When this option is enabled, ACOS requests the end-user’s URI list from the AAA server. If the end-user is authenticated, ACOS allows access only if the URI that is requested by the end-user is in the list that is provided by the AAA server.

NOTE: The user cannot configure AAM on a CGNv6 partition.

The following topics are covered:

Configuring the RADIUS Server

To configure the user’s required RADIUS server for URI-based authorization:

1. Edit the `/etc/raddb/users` file. An example of a users file:

```
user1 Cleartext-Password := "a10"
          User-Name = "user1",
          A10-AX-AUTH-URI = ACOS//vs1/80/http
user2 Cleartext-Password := "a10"
          User-Name = "user2",
          A10-AX-AUTH-URI = ACOS/l3v/vs1/80/http
```

2. Add a `dictionary.a10` file and include the following attribute:

```
A10-AX-AUTH-URI
```

3. Add the dictionary file in RADIUS dictionary file. Edit `/etc/raddb/dictionary` and add the following:

```
$INCLUDE        /etc/raddb/dictionary.a10
```

4. Add the permitted URI list to each end-user account.

```
dictionary.a10 file
```

   Here is an example of a `dictionary.a10` file that contains the attribute for URI-based authorization:

```
# A10-Networks dictionary
# Created by Software Tools of A10 Networks.
#
VENDOR A10-Networks 22610
BEGIN-VENDOR A10-Networks
ATTRIBUTE A10-AX-AUTH-URI       6    string
END-VENDOR A10-Networks
```

Configuring the LDAP Server

To configure the user's required LDAP server for URI-based authorization:

1. Copy the `a10auth.schema` file to the following directory:

   `/etc/openldap/schema`

2. To include the `a10auth.schema`, edit the following file:

   `/etc/openldap/slapd.conf`

   For example:

```
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
#
include         /etc/openldap/schema/corba.schema
include         /etc/openldap/schema/core.schema
…
include         /etc/openldap/schema/ppolicy.schema
include         /etc/openldap/schema/collective.schema
include         /etc/openldap/schema/a10.schema
include         /etc/openldap/schema/a10auth.schema
```

3. Add a new LDAP user to include an A10-AX-AUTH-URI list:

   For example:

   a. Add the following line to the ldif file:
      ```
      A10-AX-AUTH-URI: AX103//Avip20/80/http
      ```

   b. Save the `ldif` file as `user.ldif`.

   c. Enter the following command to add the new LDAP user:
      ```
      ldapadd -x -D "cn=Manager,dc=example,dc=com" -W -f "user.ldif"
      a10auth.schema
      ```

      Here is the a10auth.schema file:

      ```
      # a10auth.schema
      #
      # This is the ldif version of a10auth.schema to be used with
      cn=config.
      #
      AttributeType ( 1.3.6.1.4.1.22610.2.1.1
      NAME 'A10-AX-AUTH-URI'
      DESC 'A10-AX-AUTH-URI:ax-serial-number [a10-partition-name [a10-
      vip-name [a10-vport]]]'
      EQUALITY caseIgnoreMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
      # This is the ldif version of a10auth.schema to be used with
      cn=config.
      #
      ObjectClass ( 1.3.6.1.4.1.22610.2.2.1 NAME 'a10UserAuth'
      DESC 'a10 user authentication'
      SUP top AUXILIARY MAY A10-AX-AUTH-URI )
      ```

Authorization Policy

The following topics are covered:

Authorization Details

Authorization occurs after authentication and determines the privileges that are associated with the user who is being authenticated. Authorization determines what an authenticated user can view on the specified web site or database.

Authorization is completed in one of the following ways:

- Retrieving specific attributes from RADIUS or LDAP after authentication is complete.
- Evaluating a Boolean expression, which consists of logic operations on one or more return attributes.

  The user can configure the following types of authorization:

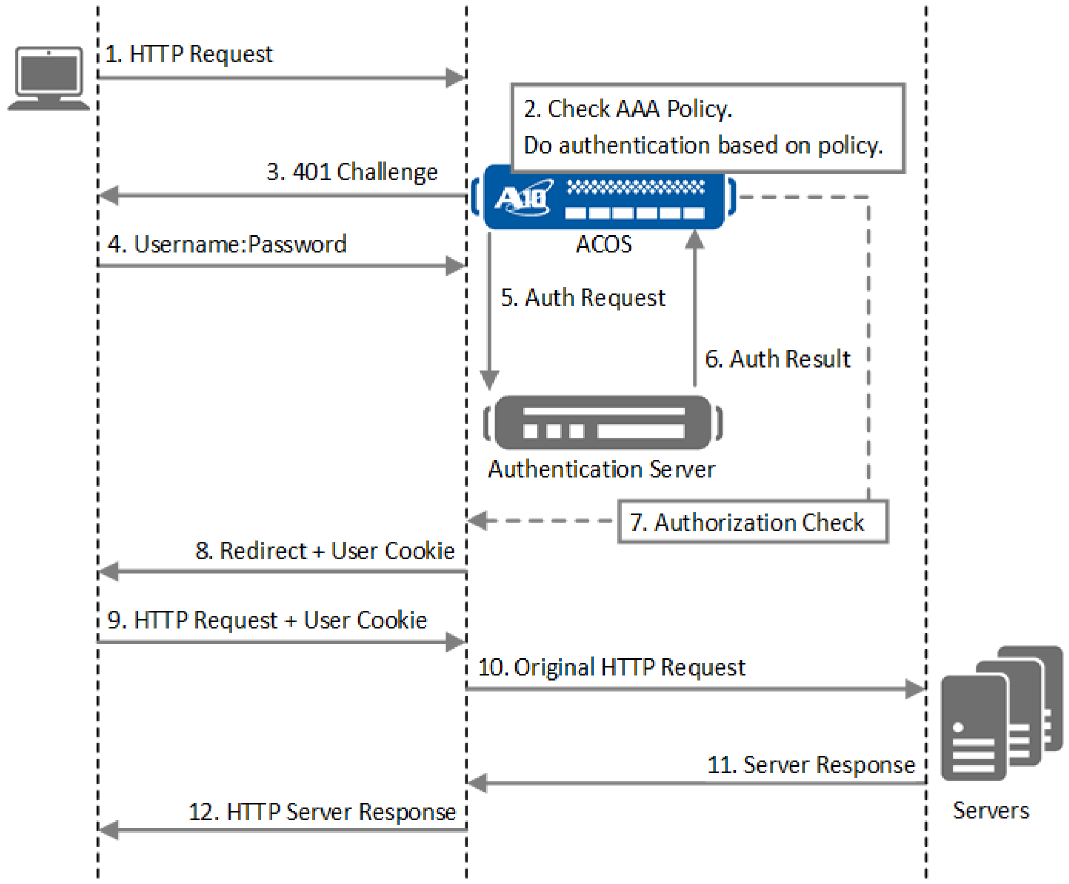
- User/Password-based authentication
  - Uses the AAA Policy to specify the return attributes to check the following:
    - Active Directory (AD) group membership
    - Multiple attributes to check in a logic statement.
- Certificate-based authentication
  - A custom LDAP search that is based on the extracted User Principal Name (UPN) from the **Subject Alternate Name** field in the client's certificate
  - LDAP object existence based on Subject Name (SN)

    Figure 2 illustrates the workflow if an AAA policy is assigned an authorization policy during configuration phase, and the authorization policy checking (step 7) is triggered after authentication succeeds. The corresponding authorization policy checking is triggered, and the request from client is authorized.

Authorization policy verifies whether the attributes that are sent from the authentication server match the criteria that are defined in the ACOS device.

NOTE: In step 10, if the user has configured authentication relay, the authorization header, with information such as the username, is added to the HTTP request before the request is sent to the external server. Even if the user does not configure authentication relay, a header is added to the HTTP request, but the authorization information is not included.

Figure 2 : Authorization Policy Workflow



After policy checking is triggered, Policy Checking Workflow illustrates the flow of this process. Authorization checking is completed in the management CPU. That means the data plane (authorization checking) needs to know which authorization attributes the user has configured.

Authorization Policy Checking

The authorization policy determines what an authenticated user can see in a web site or a database. The authorization policy is combined with an authentication template and is executed after the user has been authenticated.

The following steps provide an overview of the authorization checking process:

1. An HTTP request from the client is received by the ACOS device.
2. The authentication process is completed.
3. The authorization policy is triggered on the ACOS device.
4. The following rules, if configured, are checked for matches:
   a. URI path
   b. ACL
   c. Domain
5. One of the following occurs:
   a. There is a match, and the **Allow** action that is based on the authentication template is taken.
   b. There is no match, so one of the following tasks is completed:
      i. The policy check ends immediately.
      ii. Based on the configured AAA rules, **Deny** action is triggered, and the policy check ends.

NOTE: The **Deny** action resets the TCP connection between the client and the ACOS device, so clients cannot keep matching the next rule or access the backend server.

The user can create multiple authorization policies and apply these policies to different groups based on the following types of attributes:

- Member of
- Department
- Location

The relevant attributes must be bound to the request that is sent to the authorization server.

Decoupling Authentication from Authorization

Currently, on an ACOS device, authentication and authorization are bound together. As a result, the user cannot perform only authorization or perform authentication and authorization using different servers.

Starting from release 4.0.1, authentication has been decoupled from authorization for the following reasons:

- Authorization does not require authentication.
- Authorization can be used with any front-end, including SAML.
- Authentication and authorization can use different types of servers.

The user cannot authenticate and authorize by using a different username/password.

NOTE: If the user is using aFleX to configure this feature, the authentication and authorization credentials need not have to be the same.

Authentication Templates

Authentication templates determine how the user can authenticate the profile and help the user to verify the identity. When an HTTP request is received by the ACOS device, the ACOS devices prompts users to enter their credentials in one of the following ways:

- A browser pop-up dialog window
- A web form

The authentication check begins after the ACOS device sends the credentials to the authentication server.

An authentication template must meet the following criteria:

- Be a different policy that is based on the URI path in the same virtual IP (VIP) address, for example:
    - `http://vip/app1`
    - `http://vip/app2`
- Be associated with one or more of the following authentication methods:
    - A different authentication method, based on whether the user is internal or external
    - External user based on the IP address scheme of internal users
        - By IP addresses, IP ranges, IP subnets
        - By partial or complete domain names (for example, **.mycompany.com*)
        - Excluding IP addresses, such as all IP address except a specified IP range
- Complete authorization checks
    - Members of LDAP groups, for example, openldap or Active Directory (AD)
    - One or more custom attributes (RADIUS/LDAP) to specify the conditions
- Generate logs
    - All Failures and errors are logged and clearly indicate the specific authorization check failures, username, time, URI, and reason
    - Switch to set logging level – failures only or success and failures
    - Success logs can be used for access logs; for example, specifying who accessed what and when
    - Use the authentication logging
    - Enable logging at the policy or sub-policy level
- Work with L3V, VCS, VRRP-A, and HA

Application Category Enhancement

The user can now find new application categories with more granularity. This is based on the following points:

- These applications originally are tagged by “web”.
- The user can find the list of all web protocols and enhanced tags at: **`#RequireUrl`**.
- The compiled list is based on the protocol bundle.

This feature of ACOS provides the option to compile a list of categories according to the available protocol name.

For example: Protocol “Yahoo”, is tagged by “web”.

CLI Commands

The following command options are added to the web-service commands:

Command	Description
`gui-session-limit`	Set the max allowed GUI session.
`gui-timeout-policy`	Set the GUI time-out value in minutes.

GUI Settings for Web-Service

The GUI session limitation and GUI timeout policy parameters are added in the following page mode:

System > Settings > Web Service

This is based on the following points:

- The *GUI session limitation* specifies the number of maximum number of GUI sessions the user can run simultaneously.

  Currently, each user has a maximum session of GUI Session Limit.

- The *GUI timeout policy* is the number of minutes a GUI session can remain idle before being terminated.

  Once the GUI session is terminated, the session ID generated by the ACOS device for the session is no longer valid.

If the user sets the GUI timeout policy as **0** then the session never gets a timeout.

- The default value of the GUI sessions are as same as that of aXAPI sessions.

References

ACOS AAM supports the recommendations of RFC 2865 and RFC 2808.

- RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*
- RFC 2808, *The SecurID(r) SASL Mechanism*
- Oasis SAML Wiki at https://wiki.oasis-open.org/security/FrontPage

Authentication Client

The following topics are covered:

The authentication client is the computer that initiates an HTTP or HTTPS request.

For example, if the user is trying to access a web site on user's laptop, this laptop is considered to be the authentication client.

From the **Auth Clients** tab, the user can configure the following:

- Logon HTTP Auth, see Creating a Logon HTTP Authentication Client
- Logon Form Based, see Creating a Logon Form Based Authentication Client
- Portal, see Configuring the Appearance of the Default Portal
- SAML Service Providers, see SAML Profiles
- SAML Identity Providers, see SAML Profiles

NOTE: For an overview and configuration instructions of RSA clients, see AAM and RSA.

Logon Portal

The following topics are covered:

The logon portal provides an interface in which users can login and complete tasks.

Basic HTTP

The logon portal sends an HTTP 401 (Unauthorized) message with response code 401, which contains a WWW-Authenticate HTTP header. The client browser is expected to send a reply with the Authorization header, which contains the username and password in Base64-encoded form.

Form-based

The logon portal uses a set of web pages to collect user credentials.

In both portal types, for authentication, ACOS sends the credentials from the user to a backend AAA server.

NOTE: For LDAP, the user can change the password by using the logon portal. For more information about AAM (Application Access Management) and LDAP (Lightweight Directory Access Protocol), see AAM with LDAP.

Basic HTTP Logon

Basic HTTP logon allows ACOS to obtain a username and password by sending an HTTP 401 (Not Authorized) message with response code 4.

The following text is an example of the Authentication header that might be in the message:

```
WWW-Authenticate: Basic realm="realm-name"
```

After the user enters the credentials in the logon dialog, the client browser sends an HTTP reply that includes the following header:

```
Authorization: Basic QTEwOlRodW5kZXI=
```

This header contains the username and password in Base64-encoded form.

Form-based Logon

A form-based logon portal uses a set of web pages that contain data forms. Users can login and change their passwords by entering data in these forms. The user can either use the default logon form, or create a new logon form.

The default portal, which includes a logon page, a logon fail page, and a change password page, is provided. The user can choose to use the default portal as the logon portal and configure the look and feel of the default form pages and limit the size of HTML files and other image files.

The following topics are covered:

Form Configuration Example

There are two ways to customize the logon portal. The user can simply change the logo of the default portal, or the user can create own portal page.

The required elements for the logon portal page, logon.html, are:

1. `<form name="logon" method="POST" action="/logon.fo">`
2. `<!-- fail-msg -->`

```
<!--
<table style="border:1px solid #FF0000;background-color:#FFE6E6;text-
align:left;vertical-align:middle;padding:10px 20px 10px;border-
radius:3px;width:100%">
<tr><td>
<font face="Arial" size="5" color="#FF0000">$a10_login_fail_
errmsg$</font>
```

```
</td></tr>
</table>
```

3. `<input type="text" name="user" maxlength="256" id="input-box">`
4. `<input type="password" name="pwd" maxlength="128" id="input-box">`
5. As long as the user has these elements, the user can surround them with other things.
6. After creating the form, zip the files and import them using the following CLI command:

```
import auth-portal my-portal use-mgmt-port scp://user@host-
ip:/path/to/portal/file/portal.zip
```

Page Types Used by Form-based Logon

The following types of pages are supported for form-based logon:

- Logon Page
- Logon Failure Page
- Change Password Page
- Configuring the Default Logon Portal

The files for the pages must be imported to the ACOS device. The user can import the files in archive files by using the `.zip`, `.tgz`, or `.tar` file formats.

By default, ACOS does not include logon portal web pages, but here are some examples of simple pages.

Logon Page

Figure 3 shows an example of a simple logon page. This page is sent by ACOS to the client in response for a request to secure services.

Figure 3 : Logon Page (example)

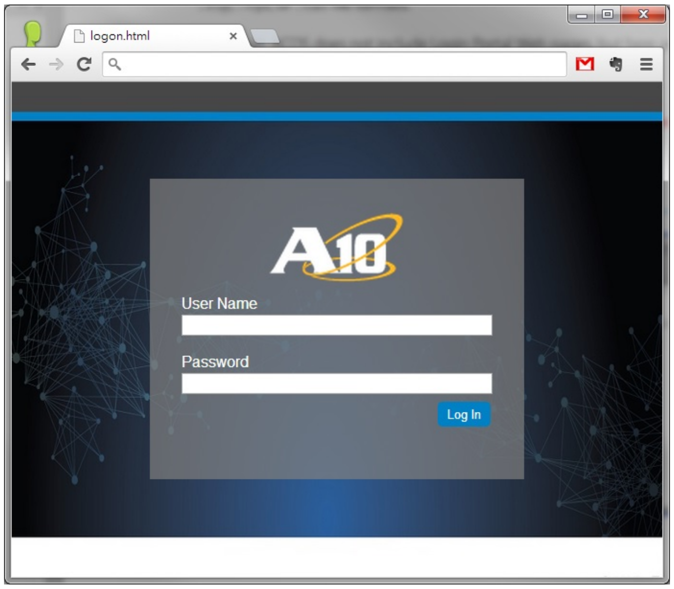
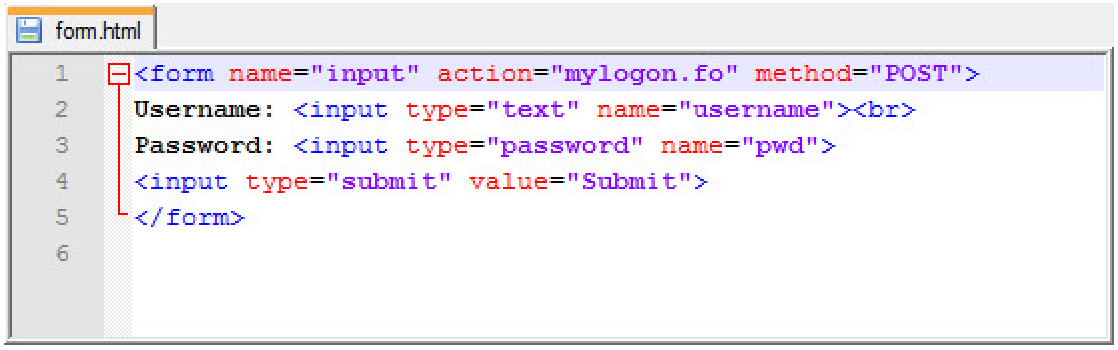


Figure 4 shows the source code for the page.

Figure 4 : Logon Page Source (example)



form.html

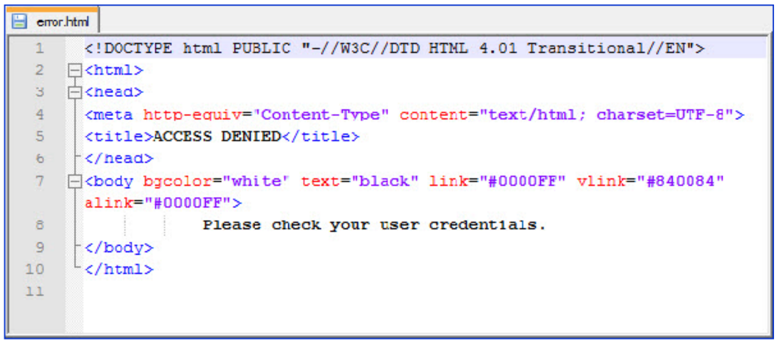
```
<form name="input" action="mylogon.fo" method="POST">
Username: <input type="text" name="username"><br>
Password: <input type="password" name="pwd">
<input type="submit" value="Submit">
</form>
```

Logon Failure Page

A Logon failure page that is sent to the client when the user enters invalid credentials. This page is sent by ACOS to a client if authentication fails.

Figure 5 shows the source code for the page.

Figure 5 : Logon Failure Page Source (example)

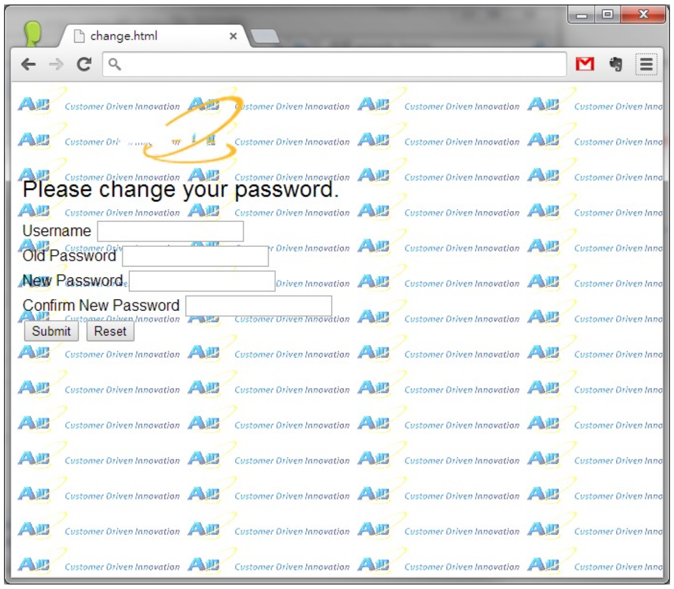

error.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv='Content-Type" content="text/html; charset=UTF-8">
<title>ACCESS DENIED</title>
</head>
<body bgcolor="white' text="black" link="#0000FF" vlink="#840084"
alink="#0000FF">
            Please check your user credentials.
</body>
</html>
```

Change Password Page

Figure 6 shows an example of Change Password page. This page is sent by ACOS to a client to allow a password change if, for example, a client password is on an LDAP server but has expire.

Figure 6 : Change Password Page (example)



Please change your password.

Username

Old Password

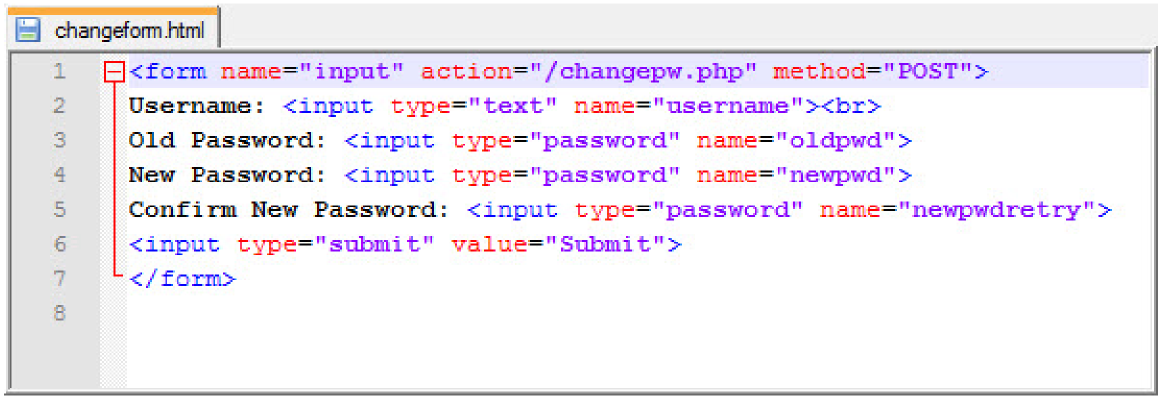
New Password

Confirm New Password

Configuring the Default Logon Portal

Figure 7 shows the source code for the page.

Figure 7 : Change Password Page Source (example)



changeform.html

```
<form name="input" action="/changepw.php" method="POST">
Username: <input type="text" name="username"><br>
Old Password: <input type="password" name="oldpwd">
New Password: <input type="password" name="newpwd">
Confirm New Password: <input type="password" name="newpwdretry">
<input type="submit" value="Submit">
</form>
```

In the default logon portal, the user can configure the logo, the **Logon** page, the **Change Password** page, and the **Logon failure** page.

Creating a Logon HTTP Authentication Client

To create a Logon HTTP authentication client:

1. Click **AAM** > **Auth Clients**.
2. On the **Logon HTTP Auth** tab, click **Create**.
3. Enter a name.
4. Optionally enter a value for the maximum number of retries and select any of the following check-boxes:

   a. **Enable SPENGO Logon**

   b. **Enable Basic Logon**

   If the user has selected the **Enable Basic Logon** check-box, then the user must complete the following steps:

   a. Select a challenge response form.

   b. Enter the new PIN page name.

   c. Enter the new PIN variable name.

   d. Enter the next token page name.

   e. Enter the next token variable page name.
   f. Enter the realm.

5. Click **Create**.

Creating a Logon Form Based Authentication Client

When the user creates a logon form, the user can use the default portal or configure the default portal.

The following topics are covered:

Using the Default Portal

To use the default portal:

1. Click **AAM** > **Auth Clients**.
2. On the **Logon Form Based** tab, click **+Create**.
3. Enter a name.
4. Enter the maximum number of fail retries.
5. To use the default portal, select the **Use Default Portal** checkbox.
6. When the user selects this check-box, the options to configure the logon page does not appear.
7. Click **Create**.

Configuring the Portal

To configure the portal:

1. Click **AAM** > **Auth Clients**.
2. On the **Logon Form Based** tab, click **Create**.
3. Enter a name for the form.
4. Enter a maximum fail retry value.
5. Leave the **Use Default Portal** check box deselected.
6. Refer to the online help for more information about these fields on this GUI page.
7. Click **Create**.

Configuring the Appearance of the Default Portal

The user can configure the default logon form by using the GUI or CLI.

The following topics are covered:

Updating the Appearance of the Logon Page

Using the GUI

To configure the appearance of the Logon Page of the default portal by using the GUI:

1. Click **AAM** > **Auth Clients** and click **Portal**.
2. To configure the **Logon** page, click **Edit** next to **logon**,
3. Configure the **General Fields**, **Username**, **Password**, **Fail Message**, and **Background** fields as needed for the user's deployment.
4. Click **Update**.

NOTE: Refer to the **Online Help** for more information about these fields on this GUI page.

Using the CLI

The following commands modify the background of the Logon page:

```
ACOS(config)# aam authentication portal default-portal
ACOS(config-portal:default-portal)# logon
```

NOTE: For a complete list of parameters available for the Logon page, see aam authentication portal default-portal.

Updating the Appearance of the Logon Failure Page

Using the GUI

To configure the appearance of the Logon Failure page of the default portal by using the GUI:

1. Click **AAM** > **Auth Clients** and click **Portal**.
2. Click **Edit** next to **logon-fail**.
3. Configure the **Title**, **Background**, and **Fail Message** fields as needed for the user's deployment.
   Refer to the on-line help for more information about these fields on this GUI page.
4. Click **Update**.

Using the CLI

The following commands modify the background of the Logon Failure page:

```
ACOS(config)# aam authentication portal default-portal
ACOS(config-portal:default-portal)# logon-fail
```

NOTE: For a complete list of parameters available for the Logon page, see aam authentication portal default-portal.

Updating the Appearance of the Change Password Page

Using the GUI

To configure the appearance of the Change Password page of the default portal by using the GUI:

1. Click **AAM** > **Auth Clients** and click **Portal**.
2. Click **Edit** next to **change-password**.
3. Configure the **General Fields**, **Title, Background**, **Username**, **New Password**, **Old Password**, and **Confirm Password** fields as needed for the user's deployment.
4. Click **Update**.

NOTE: Refer to the **Online Help** for more information about these fields on this GUI page.

Using the CLI

The following example shows how to configure the Change Password file:

```
ACOS(config)# aam authentication portal default-portal
ACOS(config-portal:default-portal)# change-password
```

NOTE: For a complete list of parameters available for the Logon page, see aam authentication portal default-portal.

Updating the Logo File

Using the GUI

To update the user's logo file:

1. Click **AAM** > **Auth Clients**.
2. On the **Portal** tab, click **Edit** next to **logo-cfg**.

3. Select the name of the logo file.
4. Enter the height and width for the new image.
5. Click **Update**.

Using the CLI

The following example shows how to configure the logo file:

```
ACOS(config)# aam authentication portal default-portal
ACOS(config-portal:default-portal)# logoname  logo_filename [height  num
| width  num]
```

NOTE: For a complete list of parameters available for the Logon page, see aam authentication portal default-portal.

Uploading a Background File to the Portal by Using the CLI

The following commands upload a background image file for the logon page:

```
ACOS(config)# aam authentication portal default-portal
ACOS(config-portal:default-portal)# logon
ACOS(config-portal:default-portal-logon)# background file file_name
```

The *file_name* is a local file name, which is in the `/a10data/auth/portal/images/` directory.

The following command displays image files:

```
ACOS# show aam authentication portal-image
FILE NAME                    FILE SIZE (byte)
--------------------------------------------------------
bg.jpg                       10000
logo.jpg                     20000
```

Deleting a File Image by Using the CLI

The following command deletes image files:

```
ACOS(config)# delete auth-portal-image local-file-name
```

Configuring HTML File Sizes by Using the CLI

The following command configures the file size limit of an HTML file:

```
ACOS(config)# system resource-usage auth-portal-html-file-size num
```

The following command configures the file size limit of an image file:

```
ACOS(config)# system resource-usage auth-portal-image-file-size num
```

Configuring Authentication Session Count using the CLI

Configure the resource usage of auth session count with maximum, minimum, and default value according to the total amount of system memory. Also this value can be set between the valid range (Min-Max) per partition.

```
ACOS(config)#system resource-usage auth-session-count ?
  <10485-104857> ------> Total auth sessions in the system
ACOS(config)#system resource-usage auth-session-count 78640
```

NOTE: Changes take effect after system reload.

Viewing HTML File Information by Using the CLI

The following command displays the values of html file size and image file size values:

```
ACOS(config)# show system resource-usage
Resource                            Current    Default    Minimum
Maximum
--------------------------------------------------------------------------
---
l4-session-count                    16777216   16777216   4194304
25165824
nat-pool-addr-count                 500        500        500        2000
class-list-ipv6-addr-count          1024000    1024000    1024000
2048000
class-list-ac-entry-count           153600     153600     153600
307200
auth-portal-html-file-size          20         20         4          120
auth-portal-image-file-size         6          6          1          80
auth-session-count                   104857     104857     10485
     104857
max-aflex-file-size                 32         32         16         256
aflex-table-entry-count             102400     102400     102400
4194304
max-aflex-authz-collection-number   512        512        256        4096
```

```
radius-table-size                    1000000     1000000     500000
1000000
monitored-entity-count               5536        5536        1360        12192
authz-policy-number                  2000        128         32          2000
```

Viewing Default Portal Configuration Information by Using the CLI

The following command displays the configuration information of the default portal:

```
ACOS(config)# show aam authentication default-portal
Authentication Default Portal
Logo File                                def_logo.png (134 x 71)
Logon Page:
Background                               color #FFFFFF
Login Failure Message                    Invalid username or password.
Please try again.
Authorization Failure Message            Authorization failed. Please
contact your system administrator.
Failure Message Attribute                font: Arial, size: 5, color:
#FF0000
Form Action URL                          /logon.fo
Username Text                            Username
Username Text Attribute                  font: Arial, size: 3, color:
#000000
Username Variable Name                   user
Password Text                            Password
Password Text Attribute                  font: Arial, size: 3, color:
#000000
Password Variable Name                   pwd
Passcode Text                            Passcode
Passcode Text Attribute                  font: Arial, size: 3, color:
#000000
Passcode Variable Name                   passcode
Submit Button Text                       Submit
Enable Passcode Field                    No

Change Password Page:
Background                               color #FFFFFF
Title                                    Please Change Your Password
Title Attribute                          font: Arial, size: 5, color:
#000000
```

```
Form Action URL                        /change.fo
Username Text                          Username
Username Text Attribute                font: Arial, size: 3, color:
#000000
Username Variable Name                 cp_usr
Old Password Text                      Old Password
Old Password Text Attribute            font: Arial, size: 3, color:
#000000
Old Password Variable Name             cp_old_pwd
New Password Text                      New Password
New Password Text Attribute            font: Arial, size: 3, color:
#000000
New Password Variable Name             cp_new_pwd
Confirm Password Text                  Confirm New Password
Confirm Password Text Attribute        font: Arial, size: 3, color:
#000000
Confirm Password Variable Name         cp_cfm_pwd
Submit Button Text                     Submit
Reset Button Text                      Reset

Logon Fail Page:
Background                             color #FFFFFF
Title                                  Try Too Many Times
Title Attribute                        font: Arial, size: 5, color:
#000000
Logon Failure Message                  Logon Failed!!
Logon Failure Message Attribute        font: Arial, size: 3, color:
#000000
Enable CAPTCHA                         Yes
CAPTCHA type                           Google reCAPTCHAv2 checkbox
reCAPTCHA site-key                     10000000-ffff-ffff-ffff-
000000000001
reCAPTCHA theme                        dark
reCAPTCHA size                         compact
reCAPTCHA badge                        bottom right
reCAPTCHA action                       A10_DEFAULT_LOGON
```

CLI Example

The following example shows how to upload an image file and configure the Logon, Logon Failure and Change Password page.

```
ACOS(config)# aam authentication portal default-portal
ACOS(config-portal:default-portal)# logo a10logo.jpg
ACOS(config-portal:default-portal)# logon
ACOS(config-portal:default-portal-logon)# background color red
ACOS(config-portal:default-portal-logon)# exit
ACOS(config-portal:default-portal)# change-password
ACOS(config-portal:default-portal-change)# background color white
ACOS(config-portal:default-portal-change)# exit
ACOS(config-portal:default-portal)# logon-fail
ACOS(config-portal:default-portal-fail)# background file bg.jpg
```

Customizing the Logon Portal

This topic describes about how the user can customize the logon portal provided by AAM.

The following topics are covered:

NOTE: The logon portal consists of multiple files for using with AAM Form-based Logon or HTTP-authenticate Logon methods.

AAM Custom Portal Files

The following topics are covered:

Files Portfolio

An AAM custom portal contains the following files:

Mandatory Files:

- Logon Page
- Logon-fail Page
- Change-password Page

Optional Files:

- Change-password-notification Page
- Challenge Page
- New-PIN Page and Next-token Page

The following topics describe the minimum requirements and the optional fields in each page.

Logon Page

Logon page is presented when a user accesses a virtual service that requires form based authentication. The Logon page consists of a combination of two mandatory fields and two optional fields.

The following procedures and options are part of this topic:

- Mandatory Fields on the Logon Page
- Optional Fields on the Logon Page

Mandatory Fields on the Logon Page

The following are mandatory fields in the Logon page:

- HTML Form with Credential Input Fields
- Logon Failure Message

HTML Form with Credential Input Fields

The logon page requires an HTML form for the end-user to enter the credentials.

As mandatory inputs, the HTML form must contain the following:

1. Form action attribute
2. Text input field for username
3. Password input field for user password

Example for the HTML Form

The following is an example of the HTML form. The highlighted fields are mandatory:

```
<form name="logon" action="logon.fo" method="POST">
    Username: <input type="text" name="user"><br>
    Password: <input type="password" name="pwd"><br>
    <input type="submit" value="Submit">
</form>
```

Logon Failure Message

When a user's authentication fails, the logon failure message is displayed. In logon page, use a special keyword and format to indicate where within the page to display the failure message.

Example for the Logon Failure Message

The following is an example of the logon failure message. The highlighted fields are mandatory:

`<!--` <p><font color="red">`$a10_login_fail_errmsg$`</font></p> -->

The following is a sequence of this synopsis:

- The user must add the special keyword "`$a10_login_fail_errmsg$`" in between the HTML comment tag.

- When the authentication fails, the Thunder Series product removes the comment tag and replaces the special keyword with the failure message.
- The failure message may be placed anywhere in the HTML page.

Optional Fields on the Logon Page

The following are optional fields in the Logon page:

- Passcode Input Field
- Change-password Page Link

NOTE: The user has the choice to select between these optional fields, which can be added in the logon page.

Passcode Input Field

In some scenarios, the authentication password is not the same as the backend server password.

For example, the authentication password is a one-time password (OTP). In this case, the user can add the passcode input filed in the HTML form.

Example for the Passcode Input Filed in the HTML Form

The following is an example of the passcode input filed in the HTML form. The highlighted fields are mandatory:

<form name="logon" action="logon.fo" method="POST">

Username: <input type="text" name="user"><br>

Password: <input type="password" name="pwd"><br>

Passcode: `<input type="password" name="otp">`<br>

<input type="submit" value="Submit">

</form>

In this example, the end user can input OTP in "`Passcode`" field and input backend password in "`Password`" field.

Feedback

NOTE: The user can configure AAM authentication relay to pass the password to backend server.

Change-password Page Link

On the logon page, the user can put a hyper-link to change-password page. If the authentication server supports the password change, then the Thunder Series product displays this link, otherwise the Thunder Series product is hidden.

The user must enter a special tag "`<!-- change-password-link -->`” and place the hyper-link inside the HTML comment tag.

Example for the Hyper-link to Change-password Page

The following is an example of the hyper-link to change-password page. The highlighted fields are mandatory:

```
<!-- change-password-link -->
```

<!--<p><a href="cp.html">Change Password</a></p>-->

NOTE: The "**change-password page link**” is displayed, only when the authentication server type is LDAP or Kerberos.

Example for the Logon Page

The following is an example for the logon page. The highlighted fields are mandatory and the bold lettered fields are optional.

```
<html>
    <body>
        <img src="logo.png"></img>
        <form name="logon" action="logon.fo" method="POST">
            <!-- <p><font size="5"
color="red">$a10_login_fail_errmsg$</font></p> -->
            Username: <input type="text" name="user"><br>
            Password: <input type="password" name="pwd"><br>
            Passcode: <input type="password" name="otp"><br>
            <input type="submit" value="Submit">
        </form>
 <!-- change-password-link -->
```

```
        <!--<p><a href="cp.html">Change Password</a></p>-->
    </body>
</html>
```

Logon-fail Page

The user must use the Logon-fail page, when the maximum limit on the logon retry is reached.

NOTE: There are no mandatory fields in this page.

Example for the Logon-fail Page

The following is an example for the logon-fail page.

```
<html>
    <body>
        <p>The account is locked due to too many logon failures.
Please try again in 30 minutes.</p>
    <body>
</html>
```

Change-password Page

The user must use the Change-password page, when there is requirement or mandatory need to change the password.

NOTE: The change-password page is accessed by user’s choice. For more information on this topic, the user must refer the “**change-password page link**” and see Logon Page.

The change-password page is also presented to the end user, whenever there is a need to change the password.

The following items are mandatory in the HTML form of the change-password page:

1. Form action attribute
2. Text input field for username
3. Password input field for old password

4. Password input field for new password
5. Password input field to confirm new password

Example for the Change-password Page

The following is an example for the Change-password page. The highlighted fields are mandatory.

```
<html>
    <body>
        <p>Change Password</p>
        <form name="cp" action="cp.fo" method="POST">
            Username: <input type="text" name="cpusr"><br>
            Old Password: <input type="password" name="cpold"><br>
            New Password: <input type="password" name="cpnew"><br>
            Confirm New Password: <input type="password" name="cpcfm"><br>
            <input type="submit" value="Submit">
        </form>
    </body>
</html>
```

Change-password-notification Page

The Change-password-notification page is used, when there is a need to remind the end user about the soon to be expired password, with a cap on the notification and days.

The following items are the required fields for the Change-password-notification page:

- HTML Form
- JavaScript for Different Action URLs
- Change-password-notification Message
- Example for the Change-password-notification Page

HTML Form

The following are the various fields and details for the HTML form in the Change-password-notification page:

1. The text input field for the username.
2. Three password input fields:
   - First, for the old password
   - Second, for the new password
   - Third, for confirming the new password
3. Two action URLs:
   - First, for changing the password
   - Second, for skipping the change

JavaScript for Different Action URLs

For the ease of multiple action URLs, the user must use the JavaScript to achieve the result.

NOTE: For more information, see Example for the Change-password-notification Page.

Change-password-notification Message

There will be a message displayed in the page to indicate the end user her password is about to expire. In change-password-notification page, use a special keyword and format to indicate where in the page to display the failure message.

Example for the Change-password-notification Message

The following is an example for the Change-password-notification message. The highlighted fields are mandatory.

```
<!-- <p><font color="red">$ a10_notify_chpw_msg$</font></p> -->
```

The following is a sequence of this synopsis:

- The user must add the special keyword "`$a10_notify_chpw_msg$`”, in between the HTML comment tag.
- The Thunder Series product removes the comment tag and replaces the special

keyword with the notification message.

- The notification message may be placed anywhere in the HTML page.

Example for the Change-password-notification Page

The following is an example for the Change-password-notification page. The highlighted fields are mandatory.

```
<html>
 <script type="text/javascript">
        function submitForm(action)
        {
            document.getElementById('cpn').action = action;
            document.getElementById('cpn').submit();
        }
    </script>
    <body>
        <p>Change Password Notification</p>
        <form name="cpn" id="cpn" action="" method="POST">
<!-- <p><font size="5"
color="red">$a10_notify_chpw_msg$</font></p> -->
            Username: <input type="text" name="cpusr"><br>
            Old Password: <input type="password" name="cpold"><br>
            New Password: <input type="password" name="cpnew"><br>
            Confirm New Password: <input type="password" name="cpcfm"><br>
<input type="button" onclick="submitForm('cont.fo')" value="Continue">
            <input type="button" onclick="submitForm('cpn.fo')"
value="Change">
        </form>
    </body>
</html>
```

NOTE:

- The change-password-notification page is only supported when the authentication server is Active Directory, authentication type is LDAP, and the option "`prompt-pw-change-before-exp`" in LDAP authentication server must be set.
- For more information and for example reference of the configuration of LDAP authentication server, see Example for the LDAP Authentication Server Configuration.
- The number of days before expiration is displayed in the change-password-notification message.
- Challenge Page

The user is expected to use the Challenge page for the execution of the 2-factor authentication process, such as *RSA SecurID* or *Entrust Identity Guard*.

The following items are the required fields for the Challenge page:

1. A special keyword for authentication server reply message.
   - In 2-factor authentication, the end user inputs the first factor on the logon page.
   - Once the authentication server confirms the first factor, it requests the second factor, with a reply message, which is to inform the end user, what the second factor is.
   - Thus, the user has a special keyword "`$replymsg$`" to display or enter this reply message.
2. HTML form for second factor:
   - The user must also need a form for the second factor inputs.
   - There is no need for an action attribute, as the user reuses the action URL on the logon page.

Example for the Challenge Page

The following is an example for the Challenge page. The highlighted fields are mandatory:

```
<html>
```

```
    <body>
        <form name="chall" method="POST">
 $replymsg$: <input type="text" name="chalv">
            <input type="submit" value="Submit">
        </form>
    </body>
</html>
```

New-PIN Page and Next-token Page

The user is expected to use the New-PIN page and the Next-token page during the 2-factor authentication process, specifically during the usage of the *RSA SecurID 6.0*.

The user is expected to use the challenge page for the *RSA SecurID 8.0* or better.

The New-PIN page is presented when the *RSA SecurID* authentication server requires the end user to change or set a new PIN.

The Next-token page is used when the *RSA SecurID* authentication server challenges the end user to enter the next token code.

In each of these pages require an HTML form for the end user to input new PIN or next token.

Example for the New-PIN Page

The following is an example for the New-PIN page. The highlighted fields are mandatory:

```
<html>
    <body>
        <form name="np" method="POST">
            New PIN: <input type="text" name="npv">
            <input type="submit" value="Submit">
        </form>
    </body>
</html>
```

Example for the Next-token Page

The following is an example for the Next-token page. The highlighted fields are mandatory:

```
<html>
    <body>
        <form name="nv" method="POST">
            Next token: <input type="text" name="ntv">
            <input type="submit" value="Submit">
        </form>
    </body>
</html>
```

Creating and Importing the Custom Portal File

To create the custom portal files, the user must import the custom portal into the Thunder Series product, and, the portal files must be zipped as flat files without any directory structure.

Example for Creating and Importing the Custom Portal File

The following is an example for creating and importing the custom portal file. The highlighted fields are mandatory:

```
cchen@dev-tp5:~/portal/sample-custom-portal$ ls
chall.html  cp.html  cpn.html  fail.html  logon.html  logon-otp.html
logo.png  np.html  nt.html
cchen@dev-tp5:~/portal/sample-custom-portal$ zip sample-custom-portal *
  adding: chall.html (deflated 40%)
  adding: cp.html (deflated 59%)
  adding: cpn.html (deflated 62%)
  adding: fail.html (deflated 42%)
  adding: logon.html (deflated 48%)
  adding: logon-otp.html (deflated 50%)
  adding: logo.png (stored 0%)
  adding: np.html (deflated 41%)
  adding: nt.html (deflated 41%)
cchen@dev-tp5:~/portal/sample-custom-portal$ ls
chall.html  cp.html  cpn.html  fail.html  logon.html  logon-otp.html
logo.png  np.html  nt.html  sample-custom-portal.zip
```

```
cchen@dev-tp5:~/portal/sample-custom-portal$
```

NOTE: The zip file which is identified and created is used in the following steps or sections, wherever necessary. For more information, see Configuring the AAM Custom Portal in Form-based Logon.

After creating the custom portal file, the user can import it onto the Thunder Series product with "`import auth-portal`" command.

```
ACOS(config)#import auth-portal sample-custom-portal use-mgmt-port
tftp://192.168.90.132/sample-custom-portal.zip
Done.
ACOS#show aam authentication portal sample-custom-portal
FILE NAME                                                      FILE SIZE
(byte)
---------------------------------------------------------------------------
-------
fail.html                                                      66
chall.html                                                     202
nt.html                                                        197
logon-otp.html                                                 440
logo.png                                                       3289
cpn.html                                                       912
logon.html                                                     514
cp.html                                                        468
np.html                                                        194
ACOS#
```

Configuring the AAM Custom Portal in Form-based Logon

Before the configuration, the user must import the custom portal.

NOTE: For more information, refer and see Creating and Importing the Custom Portal File.

The following topics are covered:

Configuring the Custom Portal with a Minimal Requirement

The following is an example for the procedure to configure the custom portal with a minimal requirement. The highlighted fields are mandatory:

```
ACOS#show run aam authentication logon form-based custom_portal_basic
!Section configuration: 439 bytes
!
aam authentication logon form-based custom_portal_basic
  portal sample-custom-portal logon logon.html failpage fail.html
changepasswordpage cp.html
  action-url logon.fo
  username-variable user
  password-variable pwd
  changepassword-url cp.fo
  changepassword-username-variable cpusr
  changepassword-old-password-variable cpold
  changepassword-new-password-variable cpnew
  changepassword-password-confirm-variable cpcfm
!
ACOS#
```

Configuring the Custom Portal with the OTP Field

The following is an example for the procedure to configure the custom portal with OTP field. The highlighted fields are mandatory:

```
ACOS#show run aam authentication logon form-based custom_portal_otp
!Section configuration: 466 bytes
!
aam authentication logon form-based custom_portal_otp
  portal sample-custom-portal logon logon-otp.html failpage fail.html
changepasswordpage cp.html
  action-url logon.fo
  username-variable user
  password-variable pwd
  passcode-variable otp
  changepassword-url cp.fo
  changepassword-username-variable cpusr
  changepassword-old-password-variable cpold
  changepassword-new-password-variable cpnew
  changepassword-password-confirm-variable cpcfm
!
ACOS#
```

Configuring the Custom Portal with the change-password-notification Page

The following is an example for the procedure to configure the custom portal with change-password-notification page. The highlighted fields are mandatory:

```
ACOS#show run aam authentication logon form-based custom_portal_cpn
!Section configuration: 561 bytes
!
aam authentication logon form-based custom_portal_cpn
  portal sample-custom-portal logon logon.html failpage fail.html
changepasswordpage cp.html notifychangepasswordpage cpn.html
  action-url logon.fo
  username-variable user
  password-variable pwd
  changepassword-url cp.fo
  changepassword-username-variable cpusr
  changepassword-old-password-variable cpold
  changepassword-new-password-variable cpnew
  changepassword-password-confirm-variable cpcfm
  notifychangepassword-change-url cpn.fo
  notifychangepassword-continue-url cont.fo
!
ACOS#
```

Example for the LDAP Authentication Server Configuration

The following is an example for the procedure to configure the LDAP authentication server. The highlighted fields are mandatory:

```
ACOS#show run aam authentication server ldap instance ad-base
```

```
!Section configuration: 301 bytes
!
aam authentication server ldap ad-base
  host 192.168.90.136
  base cn=Users,dc=a10-tplab,dc=com
  admin-dn cn=Administrator,cn=Users,dc=a10-tplab,dc=com
  admin-secret encrypted
rdb9WoR9H3y25a5C93YqpDwQjLjV2wDnPBCMuNXbAOc8EIy41dsA5zwQjLjV2wDn
  auth-type ad
  prompt-pw-change-before-exp 10
!
ACOS#
```

Configuring the Custom Portal with the Challenge Page

The following is an example for the procedure to configure the custom portal with challenge page. The highlighted fields are mandatory:

```
ACOS#show run aam authentication logon form-based custom_portal_chall
!Section configuration: 493 bytes
!
aam authentication logon form-based custom_portal_chall
  portal sample-custom-portal logon logon.html failpage fail.html
changepasswordpage cp.html challenge-page chall.html
  action-url logon.fo
  username-variable user
  password-variable pwd
  changepassword-url cp.fo
  changepassword-username-variable cpusr
  changepassword-old-password-variable cpold
  changepassword-new-password-variable cpnew
  changepassword-password-confirm-variable cpcfm
  challenge-variable chalv
!
ACOS#
```

Configuring the Custom Portal with the New-PIN Page and the Next-token Page

The following is an example for the procedure to configure the custom portal with the New-PIN Page and the Next-token page. The highlighted fields are mandatory:

```
ACOS#show run aam authentication logon form-based custom_portal_rsa
!Section configuration: 533 bytes
!
aam authentication logon form-based custom_portal_chall
  portal sample-custom-portal logon logon.html failpage fail.html
changepasswordpage cp.html new-pin-page nt.html next-token-page nt.html
  action-url logon.fo
  username-variable user
  password-variable pwd
  changepassword-url cp.fo
  changepassword-username-variable cpusr
  changepassword-old-password-variable cpold
  changepassword-new-password-variable cpnew
  changepassword-password-confirm-variable cpcfm
  new-pin-variable npv
  next-token-variable ntv
!
ACOS#
```

Configuring the Custom Portal Without Displaying the Change-password Page Link on the Logon Page

The following is an example for the procedure to configure the custom portal without displaying the Change-password page link on the Logon page. The highlighted fields are mandatory:

```
ACOS#show run aam authentication logon form-based custom_portal_no_cplink
!Section configuration: 475 bytes
!
aam authentication logon form-based custom_portal_no_cplink
  portal sample-custom-portal logon logon.html failpage fail.html
changepasswordpage cp.html
  action-url logon.fo
  username-variable user
  password-variable pwd
disable-change-password-link
  changepassword-url cp.fo
  changepassword-username-variable cpusr
  changepassword-old-password-variable cpold
  changepassword-new-password-variable cpnew
  changepassword-password-confirm-variable cpcfm
```

```
!
ACOS#
```

Configuring the AAM Custom Portal in HTTP-authenticate Logon

Before the configuration, the user must import the custom portal.

NOTE: For more information, refer and see Creating and Importing the Custom Portal File.

The following topics are covered:

Configuring the Custom Portal with the Challenge Page

The following is an example for the procedure to configure the custom portal with the challenge page. The highlighted fields are mandatory:

```
ACOS#show run aam authentication logon http-authenticate instance custom_
portal_chal
!Section configuration: 210 bytes
!
aam authentication logon http-authenticate custom_portal_chal
  auth-method basic challenge-response-form sample-custom-portal
challenge-page chall.html challenge-variable chalv
  auth-method basic enable
!
ACOS#
```

Configuring the Custom Portal with the New-PIN Page and the Next-token Page

The following is an example for the procedure to configure the custom portal with the New-PIN page and the Next-token page. The highlighted fields are mandatory:

```
ACOS#show run aam authentication logon http-authenticate instance custom_
portal_npnt
!Section configuration: 249 bytes
!
```

```
aam authentication logon http-authenticate custom_portal_npnt
  auth-method basic challenge-response-form sample-custom-portal new-pin-
page np.html next-token-page nt.html new-pin-variable npv next-token-
variable ntv
  auth-method basic enable
!
ACOS#
```

Integrating the AAM Features with the New HTTP Proxy

The following topics are covered:

Overview

This feature explains the integration and support for the AAM feature with the new HTTP/2 proxy. The following is a list of its salient features:

- The user requirement needs to be implemented for AAM with the new data structure and APIs.
- The expected behavior of the AAM in this new proxy is the same as in the original proxy, with an additional feature of handling the HTTP/2 traffic.
- The AAM can get, modify, or remove the plain text header and the body by the new APIs easily and gets cleared by a few common codes with other modules, such as the usage of aFleX flags for queuing packet.

Limitations or Known Issues

This feature has the following limitations or known issues:

- The following is a list, not supported by HTTP/2 proxy:
  - OCSP-stapling
- Only the following HTTP protocol combinations are supported:
  - Both the client and the server using HTTP/1.1
  - Both the client and the server are using HTTP/2
  - The client using the HTTP/2 and the server using HTTP/1.1
- Currently, there is no support for the combination of the client using the HTTP/1.1 and the server using the HTTP/2.

Requirements

The requirement for this feature is to make the AAM work with HTTP/2 proxy. The user can use the HTTP 1.x and HTTP 2.0 scenarios with the same AAM configuration. The following is a list of required settings:

- The same behavior as in the old proxy, with the same;
  - AAM configuration
  - Counters and stats
  - Syslog
- Both IPv4 and IPv6 are supported.
- Working with HTTP 2.0 messages.
  - Client and server both use HTTP 1.x (same as old proxy).
  - Client uses HTTP 2.0 and server uses HTTP 1.x or HTTP 2.0.

Scenario

The sequence of scenarios, or the user stories derived, or the interfaces considered for this feature are as the following:

- HTTP-basic logon + RADIUS authentication server + Kerberos relay
- form-based logon + LDAP authentication server + form-based relay
- The unchanged AAM configuration with the following components:
  - **Virtual server:** AAA-policy needs to be bound in a vPort.
  - **Client-SSL template:** AAM supports OCSP, OCSP-stapling, and LDAP certificate authorization in client-ssl template.
- The unchanged AAM algorithm and logic as the following requirements:
  - Client trying to access the resources behind the ACOS device.
  - The AAM returning a form or a pop-up or a dialog to verify the user credential.
  - The clients submitting the user credential.
  - The AAM verifying the user credential.
  - The authenticated user can access the resource; otherwise, the user credential process repeats again.
  - The AAM relaying the user credential to the backend server, if configured.

CLI Configuration Commands

For this feature, there is no new CLI command. However, the user must consider the following:

- To enable the AAM feature with new HTTP/2 proxy, the user must set "**support-http2**" and "**aaa-policy**" under vPort, as listed in the following example:

```
slb virtual server vs92 192.168.92.135
  port 80 http
    support-http2
    source-nat auto
    service-group l3vsg80
    aaa-policy aaa
!
```

- There is no new show command. All the AAM related show commands are started with "**show aam**".

- There are no new counters and stats.
- The AAM counters can be displayed in "**`show counters aam XXX`**" commands.
- The AAM stats can be displayed in "**`show aam authentication statistics XXX`**" commands.

aFleX

The following topics are covered:

For this feature, the same aFlex events and commands are supported in the new HTTP/2 proxy.

AAM aFlex Events

```
AAM_AUTHENTICATION_INIT
AAM_AUTHORIZATION_CHECK
AAM_RELAY_INIT
```

AAM aFlex Commands

```
AAM::attribute
AAM::attribute_collection
AAM::authentication
AAM::relay
AAM::session
AAM::saml
AAM::client
AAM::authorization
```

System Information

The following topics are covered:

Syslog

For this feature, the information regarding the Syslog is not changed, it can be checked by the following:

- show log (locally)
- acos-events (remotely)
- logging host (remotely)

Debug log

For this feature, the information regarding the Debug log is changed after this feature, because of the change in the internal flow. The Debug log can be checked by the following:

- **`debug auth`**: AAM related debug logs
- **`debug http-proxy`**: HTTP flow related logs
- **`debug l7session`**: detailed function call and packet flow
- **`debug http2`**: HTTP/2 flow related logs

Important

For this feature, the following are the important points to consider:

- There is no configuration change in the AAM.
- There must not be any impact on the existing licensing support.
- There must not be any compatibility issue.
- The AAM must work under the VRRP-A.
- The L3V is supported as well as the old proxy.
- Currently, the service-partition is not supporting the AAM.
- The AAM info in the show tech is not changed.

Configuring Security Features

ACOS provides additional security features to mitigate clickjacking attacks on the AAM Authentication Logon Form-Based page using X-Frame Options (XFO) feature. Clickjacking (or click hijacking) is an interface-based cyberattack that involves tricking the user into clicking on actionable content on a hidden website element (iframe layer).

To enhance security for the **AAM authentication logon form-based** page, configure the following HTTP security features or policies:

- **Content-Security-Policy (CSP) response header (frame ancestors):** This allows an administrator to specify whether a page should be rendered in the <frame> or <iframe>. For more information, see Configuring Content Security Policy.
- **Strict-Transport-Security (HSTS) response header:** This allows an administrator to set the timeout value for the page and specifies whether the page (or sub-domains) should only be viewed through HTTPS until the timeout expires. For more information, see Configuring HTTP Strict Transport Security (HSTS) Policy.

Known Limitations

- The Clickjacking feature is supported only when the HTTP/2 protocol support is enabled.

NOTE: To insert the new supported HTTP headers (X-Frame-Options, Content-Security-Policy, and Strict-Transport-Security) the client traffic must be HTTP/2.

- AAM does not support the service partition.

Configuring Content Security Policy

You can configure the CSP feature using the `csp-support` command. This command instructs the browser or page to insert HTTP Content-Security-Policy and X-Frame-Options headers.

NOTE: This feature supports only for the form-based authentication logon.

CLI Configuration

To configure CSP, use the following commands:

```
ACOS(config)#aam authentication logon form-based <profile-name>
ACOS(config-form-based auth logon:<profile-name>)#csp-support
<none|self|specificURI>
```

- To deny the browser from rendering or embedding a page inside a <FRAME> or <IFRAME> tag, use the following command:

  ```
  ACOS(config-form-based auth logon:<profile-name>)#csp-support none
  ```

- To allow the current page to be displayed in a frame on another page, but only within the current domain, use the following command:

  ```
  ACOS(config-form-based auth logon:<profile-name>)#csp-support self
  ```

- To allow the current page to be displayed in a frame but only in a specific URI, use the following command:

  ```
  ACOS(config-form-based auth logon:<profile-name>)#csp-support
  specificURI https://a.example.com
  ```

  You can set a maximum of two URIs.

NOTE: `csp-support none` is mutually exclusive to `self` and `specificURI`. However, you can configure `self` and `specificURI` simultaneously.

Configuring HTTP Strict Transport Security (HSTS) Policy

You can configure the HSTS policy using the `hsts-support timeout` command. This command instructs the browser to always connect the page (or subdomains) using HTTPS and retain the HSTS policy for a certain period. After the specified timeout, the page connects using HTTP.

CLI Configuration

The `hsts-support timeout` command works on form-based authentication logon, basic logon, and http-authenticate.

To configure the HSTS policy on authentication logon form-based, use the following commands:

```
ACOS(config)#aam authentication logon form-based <profile-name>
ACOS(config-form-based auth logon:<profile-name>)#hsts-support timeout
<seconds>
```

To configure the HSTS policy on authentication logon http-authenticate, use the following commands:

```
ACOS(config)#aam authentication logon http-authenticate <profile-name>
ACOS(config-http-authenticate auth logon:<profile-name>)#hsts-support
timeout <seconds>
```

The timeout value can be between 0-315360000 seconds. Set the timeout value to 0, to disable the HSTS policy on a client browser.

Show Commands

To view the information about CSP and HSTS policies, use the following show command:

```
ACOS#show aam authentication logon form-based
Form-based Authentication Logon
Name                                    recaptcha
Type                                    form-based
Portal file                             sample-custom-portal-recaptcha
Logon page name                         logon.html
Logon fail page name                    fail.html
Change password page name               cp.html
.
.
.
Csp support none                        Enabled
HSTS timeout                            XXXsec
Disable change password link            No
```

To view the information about HSTS policy, use the following show command:

```
ACOS#show aam authentication logon http-authenticate
HTTP-Authenticate Authentication Logon
Name                                    basic
Type                                    http-authenticate
Mode                                    Basic
Basic-Realm
Retry count                             3
```

```
Account lock                            Disabled
.
.
.
HSTS timeout                            XXXsec
Disable change password link            No
```

SAML Service Providers

For information on how to configure a SAML service provider, see Configuring by Using the GUI.

SAML Identity Providers

For information on how to configure a SAML identity provider, see Creating a SAML Identity Provider.

Authentication Relays

Authentication relay offloads the user's AAA servers by providing backend logon services on behalf of authenticated clients.

ACOS contacts the backend AAA servers on behalf of the clients. After a server responds, ACOS caches the reply and uses the cached reply for subsequent client requests.

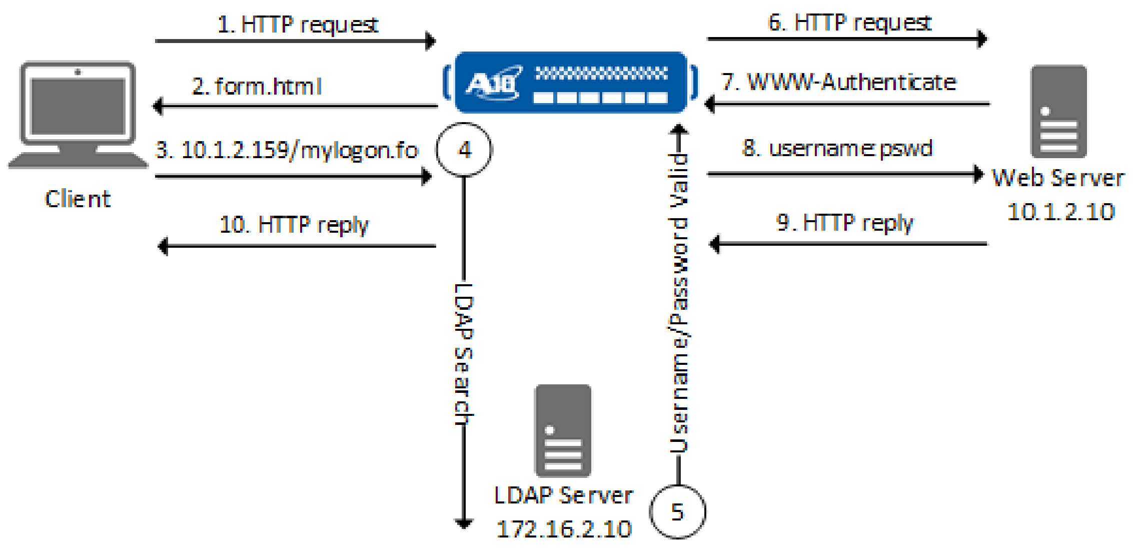
ACOS uses backend AAA servers to authenticate a client's initial request. If authentication is successful, ACOS logs into load-balanced services for which the client is authorized, when those services are requested by the client. The end-user does not need to enter the credentials again.

The AAM solution uses a backend LDAP server to verify an end-user's credentials during the first logon, then reuses those credentials to logon to load-balanced content servers on behalf of that end-user.

Figure 8 shows an example of HTTP Basic authentication relay.

Figure 8 : Form-based Logon with LDAP and HTTP Basic Authentication Relay

From the **Auth Relays** tab, the user can configure the following:



The following topics are covered:

Authentication Relay in other Chapters

- AAM and Kerberos
- AAM and SAML

HTTP Basic Authentication Relay

In the *HTTP basic authentication relay* deployment, the content servers do not need to understand LDAP. Instead, Basic-HTTP authentication is used between ACOS and the servers.

The following topics are covered:

Traffic Walkthrough of a HTTP Basic Authentication Relay (Username and Password)

The following steps correspond to the steps in Form-based Logon with LDAP and HTTP Basic Authentication Relay.

1. Client sends initial HTTP request to VIP.
2. ACOS replies with a form containing input fields for the end-user credentials (username and password).

3. The end-user enters a username and password, which the client sends back to ACOS.
4. ACOS extracts the username and password from the form, and sends them to the LDAP server in a Search request.
5. The LDAP server replies. In this example, the username and password are present in the LDAP server's user database.
6. ACOS uses SLB to select a server, then sends the client's HTTP request to the server.
7. The server replies with an HTTP 401 (Not Authorized) message with response code 4, containing an Authentication header such as the following:

   ```
   WWW-Authenticate: Basic realm="realm-name"
   ```

8. ACOS sends an HTTP reply that includes the username and password, in Base64-encoded form, in an Authorization header:

   ```
   Authorization: Basic QTEwOlRodW5kZXI=
   ```

9. If the username and password are valid, the server replies with the requested content.
10. ACOS caches the credential verification from the LDAP server, and forwards the server reply to the client.

Configuring an HTTP Basic Authentication Relay

The following topics are covered:

Using the GUI

To configure HTTP Basic authentication relay:

1. Click **AAM** > **Auth Relays**.
2. On the **HTTP Basic** tab, click **Create**.

3. Enter the relay name, domain, and the domain format.
4. Click **Create**.

Using the CLI

The following command creates a new HTTP Basic authentication relay type with the domain format that appends the domain with a user principal name format (for example, user@domain):

```
ACOS(config)# aam authentication relay http-basic 23
ACOS(config-http basic auth relay:23)# domain-format user-principal-name
```

NOTE:

- For more information on this command, see aam authentication relay http-basic.
- AAM relay is recommended in most cases where the backend server requires authentication. If AAM Relay is not configured and the backend real server requires authentication, HTTP headers such as "Authorization" may be removed from the proxied HTTP requests when forwarded to the backend server, which can result in authentication failure with the backend server. To address this issue, use the following aFlex to reinsert required HTTP headers, such as the "Authorization" header.

```
when HTTP_REQUEST {
 set auth 0
 if { [HTTP::header exists Authorization] } {
 set auth 1
 set authorizationHeader [HTTP::header "Authorization"]
}
}
when HTTP_REQUEST_SEND {
 if { $auth == 1 } {
 HTTP::header insert Authorization "$authorizationHeader"
}
}
```

For more information see aFleX Scripting Language Reference Guide.

Form-based Relay

The following topics are covered:

Details on the Form-based Relay

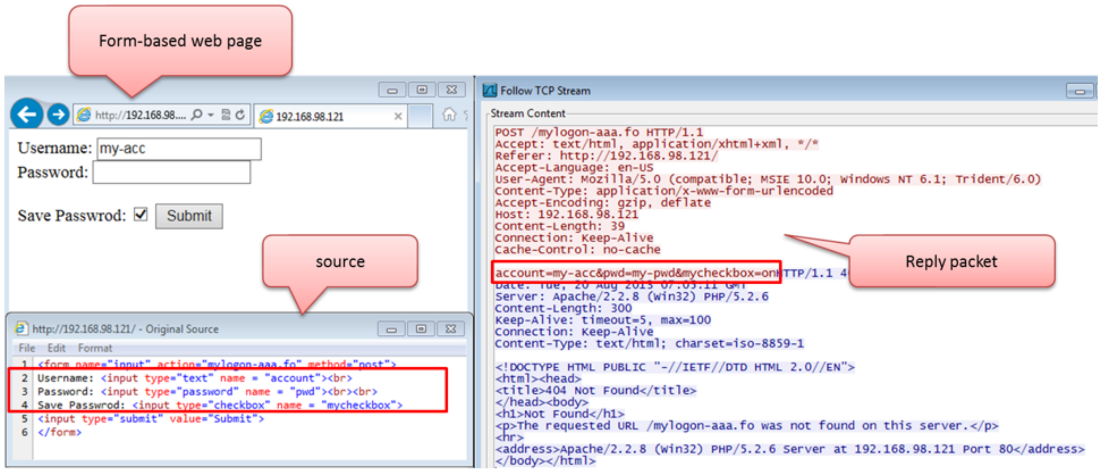
In addition to the HTTP Basic, and Kerberos types of authentication relay, there is also Form-based relay. Here, when the application server replies to the client with a form-based web page for authentication, the end-user does not need to enter the credentials again.

When the user configures form-based relay, the name of variables are set. If the application server replies with a form-based web page for authentication (ACOS knows the page is for authentication by checking some keywords), ACOS uses the information in *Auth-relay* and *auth_info* to reply to the application server and complete the authentication.

The user can also use the multiple authentication web pages in one VIP. In this case, the user can configure multiple settings for different web pages. Every setting has a different **request-url parameter** as its index, and each setting is responsible for its corresponding web page.

The form-based web page contains a form with several input elements. The client must enter the appropriate information and submit the completed form to the server. Figure 9 illustrates how the form-based relay form is structured.

Figure 9 : Form-based Relay



```
<form name="input" action="mylogon-aaa.fo" method="post">
Username: <input type="text" name = "account"><br>
Password: <input type="password" name = "pwd"><br><br>
Save Passwrod: <input type="checkbox" name = "mycheckbox">
<input type="submit" value="Submit">
</form>
```

```
POST /mylogon-aaa.fo HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http://192.168.98.121/
Accept-Language: en-US
User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/6.0)
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: 192.168.98.121
Content-Length: 39
Connection: Keep-Alive
Cache-Control: no-cache

account=my-acc&pwd=my-pwd&mycheckbox=onHTTP/1.1 4
Date: Tue, 20 Aug 2013 07:03:11 GMT
Server: Apache/2.2.8 (Win32) PHP/5.2.6
Content-Length: 300
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /mylogon-aaa.fo was not found on this server.</p>
<hr>
<address>Apache/2.2.8 (Win32) PHP/5.2.6 Server at 192.168.98.121 Port 80</address>
</body></html>
```

The reply format is as follows:

- “**`name-1=value-1&name-2=value-2&name-3=value-3…`**”

  In the example, the POST packet replies:

  “**`account=my-acc&pwd=my-pwd&mycheckbox=on`**”

  `my-acc` and **`my-pwd`** are client’s username and password. They can be found in **`auth_info`** structure.

The following parameters can be used to configure form-based relay:

NOTE: Only the *request-uri* parameter is required. The other parameters are optional.

- The Username-variable is the name of the username variable.

  If the web page contains a username field, the following text is the parameter:

  `Username:` *`input type="text" name = "account"`* <br>

  In this example, the username parameter is defined as **`account`**.

- The Password-variable is the name of the password variable.
- If the web page contains a password field, the following text is the parameter:

`Password:` *`input type="password" name = "pwd"`*<br>

In this example, the password parameter is defined as **`pwd`**.

- The Other-variable is used for other input elements on the web page. The user can designate the parameter name and value here.

  For example, the user can configure **mycheckbox=on&mycheck2=on** to pretend that the check box is clicked.

- The **Domain** is the name of the domain variable.
- The **request-URI** is the URI of the web page for authentication.
- The **action-URI** is the action URI that is required in POST or GET requests. If the user does not configure an action-URI, ACOS tries to access the URI from the web page's action field.

One form-based *`auth-relay`* configuration can contain multiple settings for multiple authentication pages. Each setting contains the six parameters, and the **`request-URI`** parameter is used as an index. This means that no two identical **`request-URIs`** exist in an *`auth-relay`*. If the user configures a new **request-URI**, a new setting is created. If the user enters an existing request-URI, an old setting is restored that the user can modify.

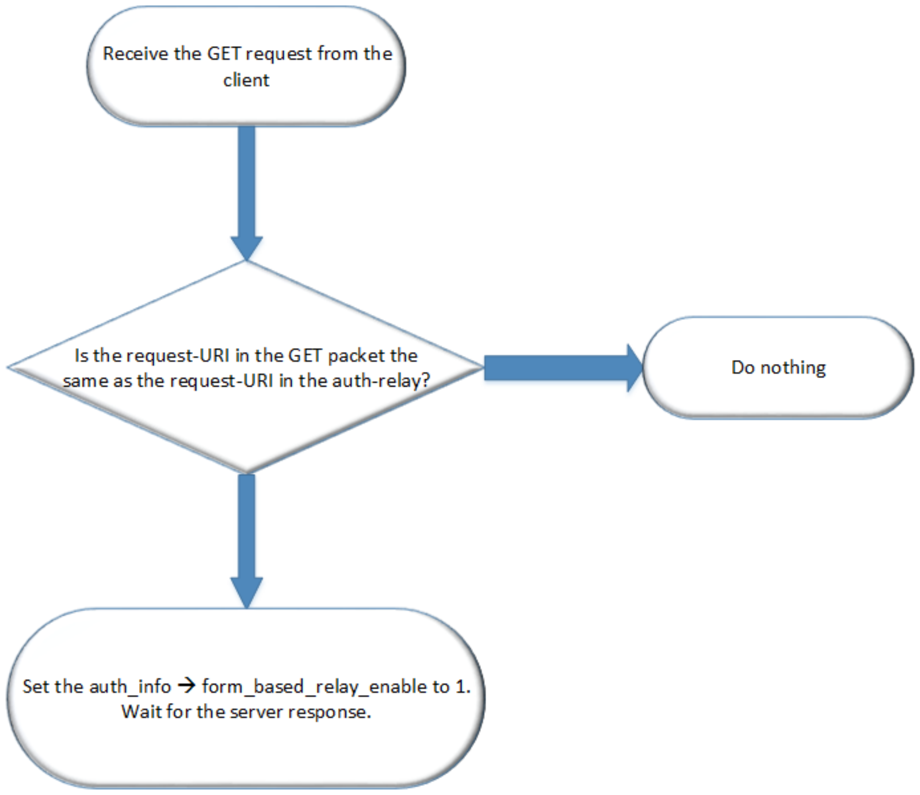
NOTE: Each form-based authentication relay can have a maximum 32 settings.

Workflow of the Form-based Relay

The form-based relay completes in two phases.

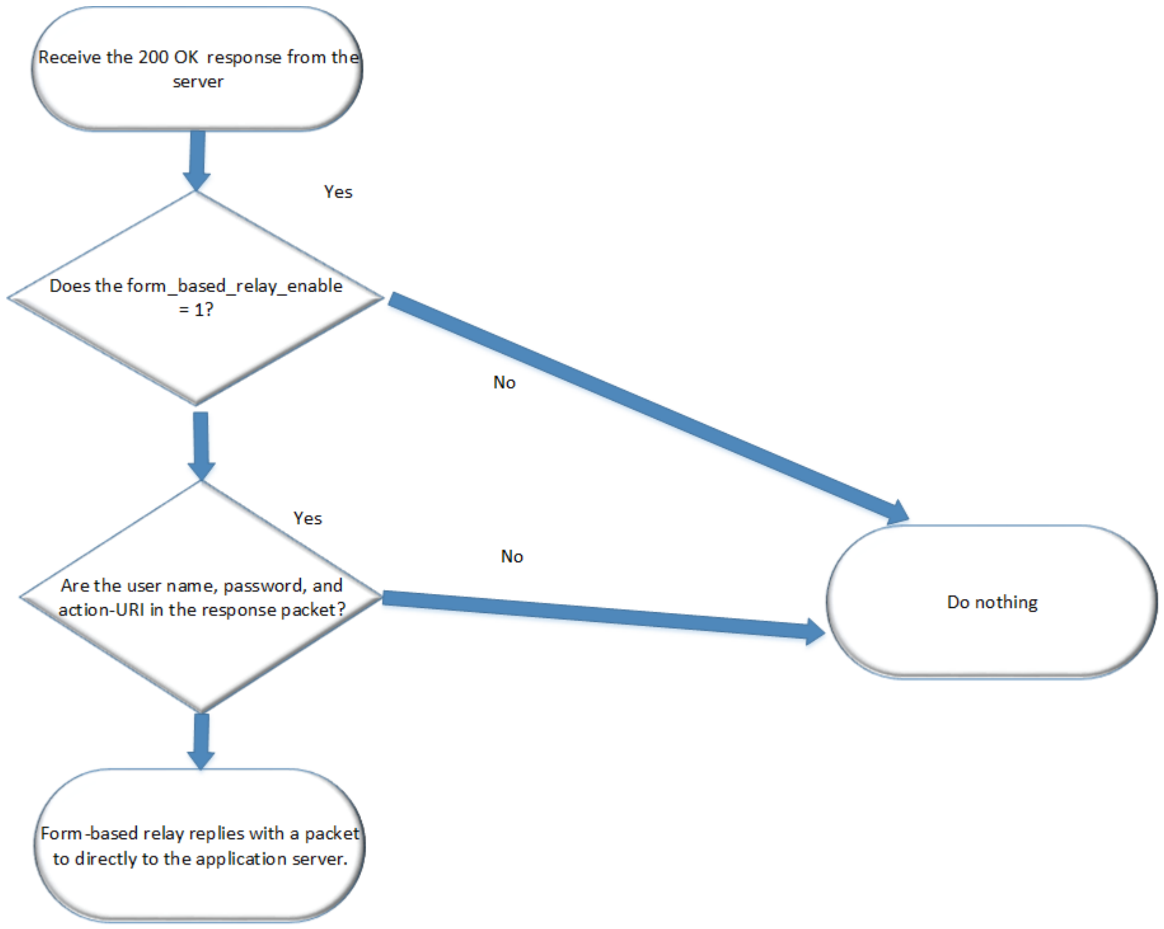
The first phase, shown in Figure 10, is when ACOS receives a client request in which the request-URI is the same as the setting in form-based *`auth-relay`*, it sets the *`form_based_relay_enable`* to 1 and awaits the server response.

Figure 10 : Form-based Relay (Phase 1)


Receive the GET request from the client
Is the request-URI in the GET packet the same as the request-URI in the auth-relay?
Do nothing
Set the auth_info → form_based_relay_enable to 1. Wait for the server response.

The second phase, shown in Figure 11, is if ACOS receives a 200 OK packet and the *form_based_relay_enable* parameter equals 1, ACOS checks the reply packet by using the parameters in the Auth-Relay. If the server's reply packet contains all of the keywords, it is treated as a web page that wants to authenticate with the client. The form-based relay replies directly to the packet instead of forwarding it to client.

Figure 11 : Form-based Relay (Phase 2)


Receive the 200 OK response from the server
Yes
Does the form_based_relay_enable = 1?
No
Yes
No
Are the user name, password, and action-URI in the response packet?
Do nothing
Form-based relay replies with a packet to directly to the application server.

Configuring a Form-based Authentication Relay

The following topics are covered:

Using the GUI

To create form-based relay:

1. Click **AAM** > **Auth Relays**.
2. On the **Form Based** tab, click **Create**.
3. Enter the relay name.
4. In the **URI** section, click **Create**.
5. Enter a value for the remaining fields.
6. Click **Create**.
7. In the **Update Authentication Relay - Form Based** page, click **Update**.

Using the CLI

The following command adds a new Form Based authentication relay type:

```
ACOS(config)# aam authentication relay form-based name
```

The following CLI example configures an authentication relay with a username, password, and relay URI.

```
ACOS(config)# aam authentication relay form-based relay1
ACOS(config-form-based auth relay:relay1)# request-uri contains
example.com
ACOS(config-form-based auth relay:relay1-setti...)# user1-variable  user_
account_1
ACOS(config-form-based auth relay:relay1-setti...)# password-variable
user_pw
ACOS(config-form-based auth relay:relay1-setti...)# other-variable  my-
check-box=on
ACOS(config-form-based auth relay:relay1-setti...)# action-uri  mylogon-
aaa.fo
ACOS(config-form-based auth relay:relay1-setti...)# max-packet-collect-
size 10000
```

NOTE: For more information about the `aam authentication relay form-based` command, see aam authentication relay form-based

max-packet-collect-size Command

The `max-packet-collect-size` command specifies the number of bytes reserved for response packet collection in bytes.

When a response from the application server is greater than the size specified by this command, ACOS outputs the following warning log, where `<size>` is the actual size of the response and and `<max>` is the size set by the command.

```
The content length of reply packet [<size>] is larger than max expected
size [<max>]. The packet collection may not succeed. You can change it by
CLI cmd max-collect-size.
```

Components of an Authentication Relay Deployment

The deployment requires the following resources:

- Zip archive of the web portal files required for end-user logon
- Logon-portal profile
- Authentication-server profile for the backend LDAP server
- Health monitor, server configuration, and service group for the backend LDAP server
- Authentication-logon profile for form-based collection of end-user credentials
- Authentication-relay profile for sending end-user credentials to content servers
- Authentication template containing the authentication-server profile and authentication-logon profile
- Server configuration and service group for the application server
- VIP configuration

Configuring the Realm Option in Basic Relay

The following topics are covered:

Details on the Realm Option in Basic Relay

The user can use this feature to optionally not require a domain to be entered when users enter their credentials in an authentication server.

The following are some of the sample scenarios:

- The back-end server is a resource in company B, and a user in company A wants to access this resource. Before the user can access the resource, the user must be authenticated. The authentication server is in company A, and it might be a RADIUS, LDAP or a Windows authentication server. Because the user is authenticating at company A, the domain/realm information may not necessary for authentication. For example, the RADIUS server does not have the domain information, and LDAP can configure the default domain for authentication.
- The user from company A is authenticated and wants to access the back-end server resource in company B. Company B might have a trust relationship with company A, which includes allowing the user from company A to access the resource. Despite the trust relationship, company B still needs to know that the user is coming from company A. This feature can add a default domain when relaying user credentials to the back-end server.
- Company A uses an authentication server that does not require users to enter their domain when authenticating. However, the back-end real server requires that a domain be included in the user’s credential when basic relay is running. In this scenario, a realm in basic relay is applicable.

Prerequisites

Ensure that the following objects have been created. If any of these objects do not exist, then the user must first configure them:

- AAM authentication logon
- AAM authentication server
- AAM authentication relay http-basic
- AAM authentication template

Figure 12 : Basic Relay Path with Realm Option Enabled

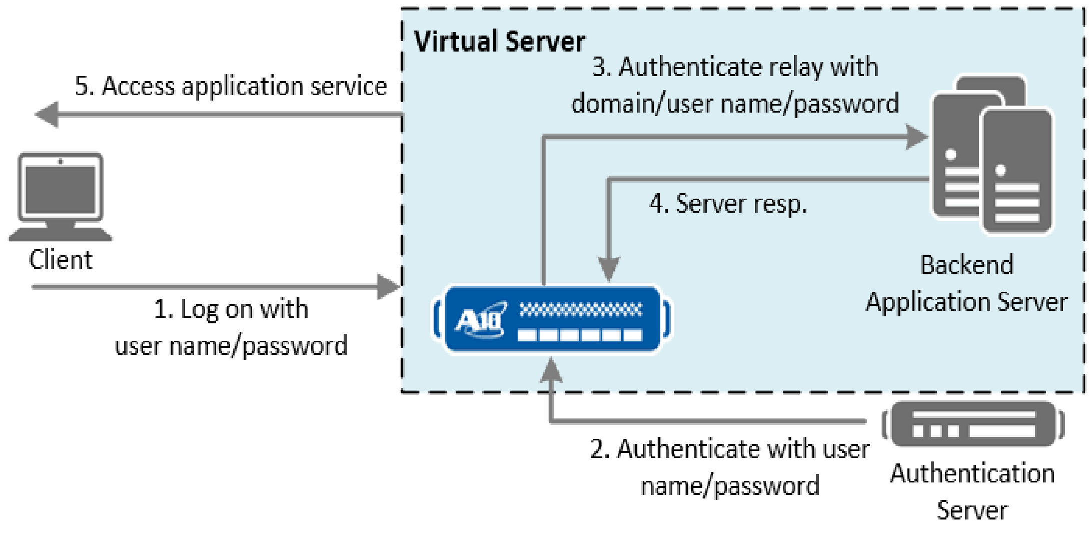

Virtual Server
5. Access application service
3. Authenticate relay with domain/user name/password
4. Server resp.
Client
Backend Application Server
1. Log on with user name/password
2. Authenticate with user name/password
Authentication Server

Figure 12 illustrates the workflow.

1. The user enters their username and password into ACOS by using form-based or http-basic logon. The credentials are saved in ACOS for authentication and authorization relay.
2. ACOS uses the credentials to authenticate the user with the authentication server.
3. If the user is successfully authenticated, ACOS relays the credentials with the domain to the back-end application server.

   ACOS determines whether to relay the domain and/or the request based on the following condition:

   a. If the domain/realm option is configured, and the user did not enter the domain when logging on, ACOS inserts domain/realm in the AUTHORIZATION header in the HTTP request.
   The user can enter one of the following header formats:

   - *domain\username:password*
   - *user@domain:password*

NOTE: The *domain\user:password* format is base64 encoded.

   b. If domain/realm option is configured, and user entered the domain while logging in, ACOS does nothing and just relays the request.

4. The server responds successfully, and ACOS forwards the server response to user.

Using the GUI

To configure relay support for the realm option in basic relay in the GUI :

1. Click **AAM** > **Auth Relays**.
2. On the HTTP Basic tab, click **Create** or **Edit**.
3. Select an option in the **Domain-Format** drop-down menu.

Using the CLI

The example below configures the “corp” domain and a domain format of user-principal-name, which uses the *user@domain* format:

```
ACOS(config)# aam authentication relay http-basic br1
ACOS(config-http basic auth relay:br1)# domain corp
ACOS(config-http basic auth relay:br1)# domain-format user-principal-name
```

CLI Example

The following commands import the logon-portal files onto the ACOS device:

```
ACOS(config)# import auth-portal portal.zip use-mgmt-port sftp:
Address or name of remote host []?fileserver1
Username []?admin1
Password []?********
File name [/]?portal.zip
...
```

The following commands configure the logon-portal profile:

```
ACOS(config)# aam authentication logon form-based f1
ACOS(config-form-based auth logon:f1)# portal portal.zip logon form.html
failpage error.html changepasswordpage changeform.html
ACOS(config-form-based auth logon:f1)# action-url /mylogon.fo
ACOS(config-form-based auth logon:f1)# username-variable username
ACOS(config-form-based auth logon:f1)# password-variable pwd
```

The following commands create an authentication-server profile for the LDAP Active Directory:

```
ACOS(config)# aam authentication server ldap l1
ACOS(config-ldap auth server:l1)# host 192.0.2.30
ACOS(config-ldap auth server:l1)# base cn=Users,dc=umin,dc=com
ACOS(config-ldap auth server:l1)# auth-type ad
```

The following commands create the SLB configuration for the LDAP server:

```
ACOS(config)# health monitor ldap-sr
ACOS(config-health:monitor)# method ldap run-search BaseDN
dc=a10networks,dc=com query (objectclass=*) AcceptNotFound
ACOS(config-health:monitor)# exit
ACOS(config)# slb server ldap-sr 192.0.2.30
ACOS(config-real server)# port 389 tcp
ACOS(config-real server-node port)# health-check ldap-sr
ACOS(config-real server-node port)# aam authentication server l1
ACOS(config-real server-node port)# aam authentication service-group sg
tcp
ACOS(config-real server-node port)# exit
ACOS(config-real server)# exit
ACOS(config)# slb service-group ldap-sg tcp
ACOS(config-slb svc group)# member ldap-sr 389
```

The following commands configure the logon-portal profile:

```
ACOS(config)# aam authentication logon form-based f1
ACOS(config-form-based auth logon:f1)# portal portal.zip logon form.html
failpage error.html changepasswordpage changeform.html
ACOS(config-form-based auth logon:f1)# action-url /mylogon.fo
ACOS(config-form-based auth logon:f1)# username-variable username
ACOS(config-form-based auth logon:f1)# password-variable pwd
```

The following commands create an HTTP-basic logon profile, and specify the AAA realm name:

```
ACOS(config-form-based auth logon:f1)# aam authentication-relay http-basic
r1
ACOS(config-http basic auth logon:f1)# realm example-realm
```

The following commands configure the authentication template:

```
ACOS(config)# aam authentication template t1
ACOS(config-auth template:t1)# logon f1
ACOS(config-auth template:t1)# service-group sg
ACOS(config-auth template:t1)# relay r1
```

The following commands add the SLB configuration. This is the same configuration used in the other AAM examples in this topic.

```
ACOS(config)# slb server rs_http 203.0.113.10
ACOS(config-real server)# port 80 tcp
ACOS(config-real server-node port)# no health-check
ACOS(config-real server-node port)# exit
ACOS(config-real server)# exit
ACOS(config)# slb service-group http_g_1 tcp
ACOS(config-slb svc group)# member rs_http 80
ACOS(config-slb svc group-member:80)# exit
ACOS(config-slb svc group)# exit
ACOS(config)# slb virtual-server vip_auth 203.0.113.139
ACOS(config-slb vserver)# port 80 http
ACOS(config-slb vserver-vport)# service-group http_g_1
ACOS(config-slb vserver-vport)# template authentication t1
```

Displaying Authentication Statistics

The following topics are covered:

The user can display the statistics for the authentication server and for authentication relay by entering one of the following commands:

```
ACOS(config)# show aam authentication server
ACOS(config)# show aam authentication relay
```

Displaying an Authentication Session

To display an authentication session, enter the following command:

```
ACOS(config)# show aam authentication session [partition | source-addr
{ipv4 | ipv6} addr | username username | vip name]
```

CLI Example

```
ACOS# show aam authentication session
TTL = Session Idle timeout (Sec), Lifetime = SAML Token/Assertion Lifetime
(Sec)
Total Sessions: 1
---------------------------------------------------------------------------
-----
ID Type VIP User Client IP Created Time VPort Domain Domain-group TTL
Lifetime
---------------------------------------------------------------------------
------
36 SAML vip4 a@aamtest.com 172.128.9.171 04-03-15 10:08:03 443
.aamtest.com 31 290 3531
3 OPENID-IP vs1  N/A      192.168.98.103  01-09-20 20:29:51 443   0  294
3593
2 OPENID    vs1  N/A      192.168.98.103  01-09-20 20:29:0 443    0  248
3547
1 OAUTH     vs1   N/A     192.168.98.103  01-09-20 20:28:11 443   0  195
34
```

Clearing an Authentication Session

To clear an authentication session, enter the following command:

```
ACOS(config)# clear aam authentication session {all | source-addr {ipv4 |
ipv6} addr | username username | vip name}
```

For more information about these commands, see Config Commands: Application Access Management.

Accounts

The user can configure a Kerberos SPN authentication account by using the GUI or CLI.

The following topics are covered:

Using the GUI

The user can configure a Kerberos SPN authentication account by using the GUI mode as explained in the following steps:

1. Click **AAM** > **Accounts**.
2. Click **Create**.
3. Enter an account name.
4. Enter the SPN domain account name.
5. Enter a Kerberos realm.
6. Enter a service principal name.
7. Enter a secret string.
8. Click **Create**.

Using the CLI

To configure a a Kerberos SPN authentication account, enter the following command at the global configuration level:

```
ACOS(config)# aam authentication account kerberos-spn account-name
```

This command changes the CLI to the authentication account configuration level.

For more information on these commands, see aam authentication account kerberos-spn.

CLI Example

The following example configures an AD domain account that is associated with a SPN named **k1** and a service principal named `sp1`:

```
ACOS(config)# aam authentication account kerberos-spn k1
ACOS(config-kerberos-spn:12)# service-principal-name sp1
```

Authentication Servers

The authentication server provides the authentication services for the AAA policy. In the user’s deployment, the user can include one or multiple authentication servers.

For more information about these servers, see the following topics in this guide:

- *AAM and Kerberos*
- Windows Integrated Authentication
- AAM with LDAP
- AAM with RADIUS

This chapter provides information on different deployment scenarios of single or multiple authentication server(s), configuration of authentication servers, and health monitoring of authentication servers.

The following topics are covered:

Feedback

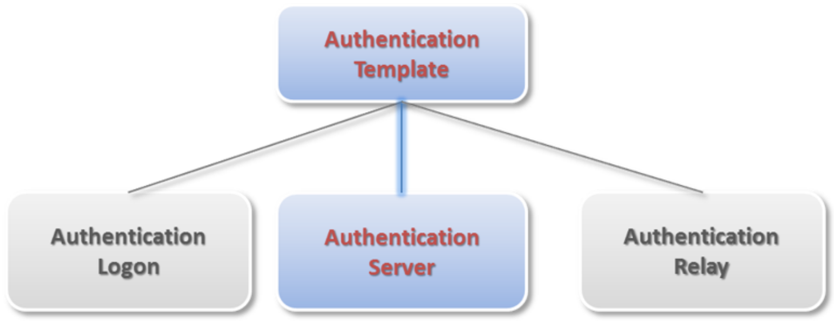
Overview

The following topics are covered:

One Authentication Server

An authentication server is bound to an authentication template with the authentication logon and authentication relay. For example, if the user is using a Kerberos KDC, the user must create a Kerberos authentication server and assign the server to an authentication template.

Figure 13 : One Authentication Server



Multiple Load-Balanced Authentication Servers

The user can create as many authentication servers as the user needs. As illustrated in Figure 14, the user must create the same number of real servers as the number of authentication servers and bind each authentication server to a real server. For example, if the user created five authentication servers, then the user must also create five real servers and bind each authentication server to a real server. The real servers are assigned to a service group, and the service group is assigned to an authentication template.

The user can use one of the following authentication relay types:

- HTTP Basic type authentication relay
- Authentication relay with one KDC
- Authentication relay with multiple KDCs

The user can create an authentication-relay instance for the first two types. For multiple KDCs, the user must create one service group and assign the instance to the authentication relay to represent multiple KDCs. One KDC from the service group is selected based on load balance algorithm and authentication relay is identical to the single KDC scenario.

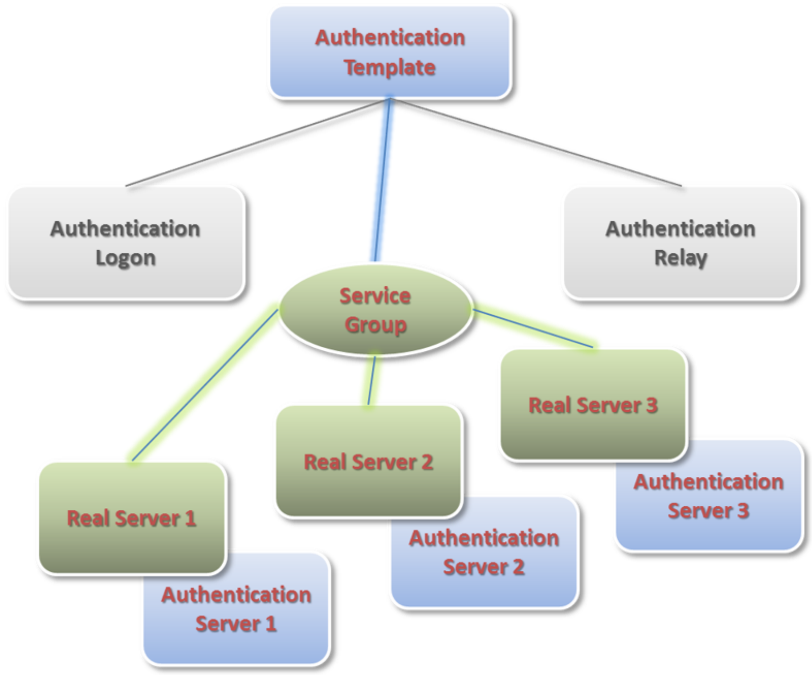
When the user selects a KDC, the user must remember the following goals:

- Choose a KDC only when ACOS needs to communicate with the KDC.

  For example, if a ticket exists, ACOS does not need to select a KDC.

- For every connection, the user only needs to select a KDC once.

Figure 14 : Multiple Authentication Servers



For more information about configuring multiple authentication servers, see the *AAM and Kerberos* topic.

Reuse Real Server as Authentication Server and SLB Server

After the real servers are created, the user can reuse the same real server IP for both the authentication server and the SLB server. This feature allows the SLB real server to perform all the actions as that of the authentication server. The real server's traffic is reused between the authentication servers (LDAP, RADIUS, Kerberos, OCSP) and SLB server.

For example, when the traffic from the client browser is sent to the virtual ports. The client browser then sends a client certificate, and ACOS verifies it and then performs an OCSP check. The OCSP and SLB server statistics are increased simultaneously since they are reusing the same real server.

Known Limitations

- The real server is reused between the authentication server and the SLB server.
- The real server shared the statistics between the authentication server and the SLB server. However, the service-group does not share the statistics between the authentication server and the SLB server.
- The real server related health-check must be applied to the authentication server only, reused by the SLB server.
- The users can still configure other options like weight, service-principal-name, etc. in the SLB server. However, these may affect the behavior of the authentication server.

CLI Configuration Guidelines

- Configure the authentication server using **`aam authentication server`** command.
- Bind the authentication server with the service-group to create the real server using **`aam authentication service-group`** command.
- Configure the health monitoring in authentication server using **`health monitor`** command.
- Configure slb server by reusing the configured authentication server using **`slb server <authentication server IP> use-aam-server`** command.

Creating Authentication Servers

The user can create authentication servers by using the GUI or the CLI.

The following topics are covered:

Using the GUI

To create an authentication server:

1. Click **AAM** > **Auth Servers**.
2. Select the **RADIUS** tab.
3. Click **Create**.
4. Configure the settings of the authentication server as needed. Refer to the online help for more information about the fields on this GUI page.
5. Click **Create**.

Using the CLI

The following example configures a RADIUS server server22:

```
ACOS(config)# aam authentication server radius 22
```

Active Directory Support for DBLB

There are two parts to this feature:

- Front-end Authentication
- Back-end Authentication

Additionally the following topics are also covered:

- Configuring Active Directory Support for DBLB

Front-end Authentication

By using database load balancing (DBLB), the user can configure database queries to be load-balanced across multiple servers. Based on the database system (MySQL or MS-SQL) and query type, the user can specify which server processes a set of queries. Database requests are secured by applying SSL and TLS protocols on the front- and back-end servers to mask sensitive user credentials and validate client credentials against username-password pairs. In addition, ACOS can screen requests against aFleX scripts to parse SQL queries and intelligently direct queries among servers.

In addition to using class-lists for username and password authentication, the user can also use the user's AAA policy for authentication. When the AAA policy is applied, ACOS supports a variety of authentication servers.

The following authentication methods exist for MS-SQL:

- SQL authentication
- Windows authentication

NOTE: DBLB only supports SQL authentication on the front-end and back-end, so only SQL authentication is supported in this release.

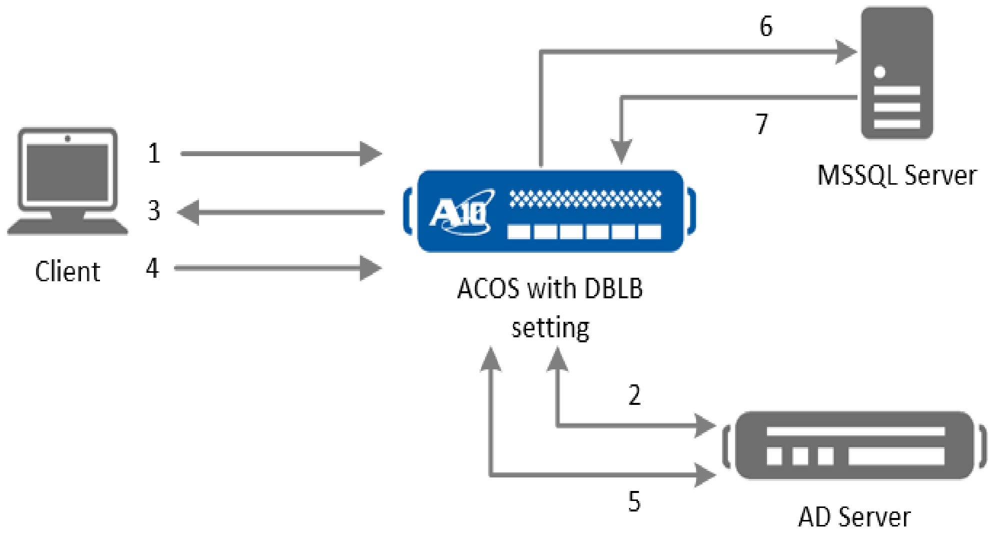
After the user enters the username and password in a class-list, these credentials are used to verify the information in the Logon packet that is sent by the client. The MS-SQL proxy uses the AAA policy to verify the credential first. If this authentication fails, the credentials are verified in the class-list.

NOTE: SQL relay can only handle different requests inside the same session. If requests come from the same sessions and are load balanced to different SQL backend servers, ACOS can help relay the authentication credential. However, if requests are sent to the ACOS device in a new session, because SQL does not have cookies, the user must enter the credentials again.

Back-end Authentication

Back-end authentication supports Windows authentication.

Figure 15 : Backend Windows Authentication



The following procedure provides a high-level overview of the backend authentication in Windows:

1. The clients send SQL authentication logon packets to ACOS.
2. ACOS takes the account/password in logon packet and uses it to get a TGT.
3. After TGT is accessed, the authentication is successful.
4. A loginack message is sent to client.
5. The client sends an SQL request.
6. Before establishing a connection to MSSQL server, the service ticket is accessed from AD.
7. A connection to the MSSQL server is created by using Windows authentication with the service ticket that was accessed in step 6.
8. MSSQL accepts the logon request and returns a loginack packet.
9. The connection is established.

Feedback

NOTE: Steps 2 and 5 use a Kerberos protocol to communicate with the AD server. Step 6 uses GSS-API/Kerberos in the TDS packet.

Configuring Active Directory Support for DBLB

The user can configure Active Directory support for DBLB by using the GUI or CLI.

The following topics are covered:

Creating a DBLB Template by using the GUI

To create a DBLB template:

1. Click **ADC** > **Templates**.
2. On the **Application** tab, click **Create** and select **DBLB**.
3. Enter a name.
4. Select the server version.
5. Select a class list.
6. Click **OK**.

Deleting a DBLB Template by Using the GUI

To update a DBLB template:

1. Click **ADC** > **Templates**.
2. On the **Application** tab, select the DBLB template that the user wants to update and click **Delete**.

Updating a DBLB Template by Using the GUI

To update a DBLB template:

1. Click **ADC** > **Templates**.
2. On the **Application** tab, select the DBLB template that the user wants to update and click **Edit**.
3. Update the server version.
4. Update the class list.
5. Click **Update**.

Configuring Active Directory Support for DBLB by Using the CLI

1. The following command creates an authentication server:
```
ACOS(config)# aam authentication server ldap  server-name
 host ip-addr
 secret secret-phrase
```
2. The following command configures an authentication template:
```
ACOS(config)# aam authentication template template-name
 server auth-server-name
```
3. The following command binds the authentication template to the policy:
```
ACOS(config)# aam aaa-policy aaa-policy-name
aaa-rule num
authentication-template auth-template-name
```
4. The following command configures the SQL server:
```
ACOS(config)# slb server sql-server-name ip-addr
port num {tcp | udp}
```
5. The following command adds the server to a service group:
```
ACOS(config)# slb service-group sg-name {tcp | udp}
member sql-server-name port-num
```
6. The following command configures the servers:
```
ACOS(config)# slb virtual-server sql-vs-name ip-addr

port port-num mssql

source-nat auto

service-group sg-name

aaa-policy aaa-policy-name
```

Health Monitoring

ACOS supports load balancing of client-server AAA traffic among a group of AAA servers. The user can configure load balancing for the following types of AAA servers:

- LDAP
- RADIUS
- Kerberos

NOTE: The user also can configure load balancing of OCSP servers (responders). However, OCSP is used by ACOS to verify whether certificates presented by clients are still valid. OCSP is not used to verify end-user credentials (that is, username and password).

The following topics are covered:

LDAP Health Monitoring

ACOS sends an LDAP request to the LDAP port, but the request can also be directed to a specific Distinguished Name (DN).

NOTE: SSL can be enabled for the health check.

If the server requires a password, ACOS must also send a valid password. This process is successful when the server sends a reply that contains the result code 0. If a DN and a password are sent in the health check, these values must match the values that are configured in the LDAP server.

NOTE: The user need not have to install a certificate on the ACOS device, as the device always accepts the server certificate presented by the server.

LDAP health monitors offer the following options:

- STARTTLS – Uses encryption and requires an encrypted reply from the LDAP server. When this option is enabled, ACOS begins the health check with a STARTTLS request. The STARTTLS request is defined in terms of an ExtendedRequest. The requestName is 1.3.6.1.4.1.1466.20037. ACOS will not send any LDAP PDUs at this LDAP message layer until ACOS receives a STARTTLS Extended response and, in the case of a successful response, completes TLS negotiations.
- Support for searchRequest and searchResponse – Include Distinguished Name (DN) queries and database queries.
- Longer DN strings – Up to 127 symbols are supported.

The user can use these health checks in AAA SLB deployments and AAM deployments.

The following topics are covered:

Using the GUI

To configure an LDAP health monitor:

1. Click **ADC** > **Health Monitors**.
2. Click **Create**.
3. Enter a monitor name.
4. In **Method type**, select **LDAP**.
5. Enter a desired value for the remaining fields.
6. In the **LDAP** section, refer to the online help for information on the LDAP fields.
7. Click **Create Monitor**.

Using the CLI

1. To configure an LDAP health monitor, enter the following commands:

```
ACOS(config)# health monitorname
ACOS(config)# health monitormethodldaprun-search BaseDN string
```

2. To configure a method to check accessibility and allow the KDC to obtain a TGT,

enter the following command:

```
ACOS(config)# health monitormethod ldap [STARTTLS] [binddndnstring
password] [overssl] [portport-num] [run-searchoptions]
```

More information about the CLI commands can be found in the **Health Monitor** topic of the *Command Line Interface Reference Guide*.

Kerberos Health Monitoring

Depending on the method options that the user opts, a Kerberos health monitor can check accessibility to each of the following components:

- Key Distribution Center (KDC) Ticket Granting Ticket (TGT) service (Kinit) – Tests ability to access the Kerberos server as a new client requesting a TGT by trying to obtain a ticket.
- Kerberos administrative system (Kadmin) – Tests ability to access the Kerberos server as a user account administrator by logging in as an administrator.
- Kerberos password change system (Kpasswd)– Tests ability to access the Kerberos server as a user attempting to change a password by logging in as a client and sending a request to change the password..

NOTE: The user must configure a valid administrator and end-user accounts on the target server.

These processes are successful when the Kerberos server responds and allows access.

When the Kerberos health monitor is designed for one application server, it needs the KDC address. When the user configures multiple KDCs, the IP address is stored in the real server structure, so the KDC address is optional. If the user does not enter the KDC address, the health monitor can get the KDC address from the real server. The **kpasswd** and **kadmin** services can be on the same server (hostname or IP address) or different servers.

The user can use these health checks in AAA SLB deployments and in Authentication Proxy deployments.

NOTE: Before the user configures the Kerberos health monitoring, the user must create a Kerberos server.

For more information about creating and configuring a Kerberos server, see the *AAM and Kerberos* topic.

The following topics are covered:

Kerberos Health Monitoring Workflow

The following procedure provides a high-level overview of Kerberos health monitoring:

1. The ACOS device sends an AS(TGT) request to the KDC authentication server.
2. The KDC authentication server sends the TGT tickets back to the ACOS device.
3. The ACOS device sends a TGS(http service) request to the KDC TGS.
4. The KDC TGS sends the TGS tickets back to the ACOS device.
5. The ACOS device sends a GET / HTTP/1.1 message to the HTTP server.
6. The client is authorized.
7. The HTTP server sends a 200 OK message to the ACOS device.

Using the GUI

To create and configure Kerberos Health Monitoring:

1. Click **ADC** > **Health Monitors**.
2. Click **Create**.
3. Enter a monitor name.
4. In **Method type**, select **Kerberos KDC**.
5. Expand the **Kerberos KDC** section.
6. Select either **Kerberos KDC**, **Kerberos Admin**, or **Kerberos Change Password**.

7. **Complete the appropriate steps. Refer to the online help for more information about the fields on this GUI page.**
8. Click **Create Monitor**.

Using the CLI

To configure a Kerberos health monitor:

1. The following command configures a Kerberos health monitor:

```
ACOS(config)# health monitor monitor-name
```

2. This command changes the CLI to the configuration level for the monitor, where the following commands configure Kerberos health-check methods for the monitor:

```
ACOS(config)# health monitormethod kerberos-kdc kinit
principal password
{kdc-hostname | kdc-ipaddr} [portport-num]
[tcp-only]
```

3. The following command configures a method to check accessibility of the Kerberos server for user account administration:

```
ACOS(config)# health monitor method kerberos-kdc kadmin
realm-name principal password
{kdc-hostname | kdc-ipaddr} [portport-num]
{admin-hostname | admin-ipaddr} [portport-num]
```

4. The following command configures a method to check the accessibility of the Kerberos server for user password change

```
ACOS(config)# health monitor method kerberos-kdc kpasswd
principal password
{kdc-hostname | kdc-ipaddr} [portport-num]
{pwd-hostname | pwd-ipaddr} [portport-num]
```

NOTE: Health checks for access to the Kerberos administrative system are not supported with Windows Server Domain Controllers (DCs).

For more information about AAM and Kerberos, see the *AAM and Kerberos* topic.

RADIUS Health Monitoring

RADIUS sends a Password Authentication Protocol (PAP) request to authenticate the username and password that are specified in the health check configuration. The RADIUS server sends an Access Accepted message (reply code 2).

The following topics are covered:

Remember the following issues:

- The requested username and password must be configured in the server's user database.
- The shared secret that is sent in the health check must be valid on the server.

Using the GUI

To configure an RADIUS health monitor in the GUI:

1. Click **ADC** > **Health Monitors**.
2. Click **Create**.
3. Enter a name for the monitor.
4. In **Method type**, select **Radius**.
5. Enter a desired value for the remaining fields.
6. Expand the **Radius** section and complete **the appropriate steps. Refer to the online help for more information about the fields on this GUI page.**
7. Click **Create Monitor**.

Using the CLI

To configure health monitoring for RADIUS:

1. To access the user's RADIUS server, enter the following commands:

```
ACOS(config)# health monitor radius1
```

2. To configure health monitoring, enter the following command:

```
ACOS(config-health:monitor)# methodradius username
namepasswordpasswordsecretcrypto-key [expect response-codecode]
[portport-num]
```

NOTE: More information about the CLI commands can be found in the "Health Monitor" topic of the *Command Line Interface Reference Guide*.

CLI Example

The following text is an example for configuring RADIUS health monitoring:

```
ACOS(config)# health monitor radius_hm
ACOS(config-health:monitor)# method radius username example_username
password ex_pw secret ex_secret_string expect response-code 12 port 123
```

NOTE: The user must include the `secret` variable to configure the password.

Port Health Checks

The user can run or disable port health checks on the user's authentication server by using the GUI or the CLI.

NOTE: If the **health-check** or **health-check-disable** commands are not set, the default health monitor is used.

The following topics are covered:

For more information about health monitors, see the following:

- The *Online Help*
- The *Application Delivery Server Load Balancing Guide*

Running a Port Health Check on a RADIUS Server by Using the CLI

The following command runs a port health check on a RADIUS server at the RADIUS authentication server configuration level:

```
ACOS(config)# aam authentication server radius port1
ACOS(config-radius auth server:port1)# health-check monitor1
```

The following command disables a port health check on a RADIUS server at the RADIUS authentication server configuration level:

```
ACOS(config)# aam authentication server radius port1
ACOS(config-radius auth server:port1)# health-check-disable
```

The default RADIUS server's authentication port is port 1812.

Running a Port Health Check on an LDAP Server by Using the CLI

The following command runs a port health check on an LDAP server at the LDAP authentication server configuration level:

```
ACOS(config)# aam authentication server ldap port1
ACOS(config-ldap auth server:port1)# health-check monitor1
```

The following command disables a port health check on an LDAP server at the LDAP authentication server configuration level:

```
ACOS(config)# aam authentication server ldap port1
ACOS(config-ldap auth server:port1)# health-check-disable
```

The default LDAP server's authentication port is 389.

Running a Port Health Check on an OCSP Server by Using the GUI

To run a port health check on a OCSP server:

1. Click **AAM** > **Auth Servers**.
2. On the **OCSP** tab, select an OCSP server, and click **Edit**.
3. In the **Server's Health Check** drop-down list, select one of the following options:
   - **None** (default)
   - **Disable Health Check**

   For more information about health monitors, see the following:
   - The *Online Help*
   - The *Application Delivery Server Load Balancing Guide*
4. Click **Update**.

Running a Port Health Check on an OCSP Server by Using the CLI

The following command runs a port health check on an OCSP server at the OCSP authentication server configuration level:

```
ACOS(config)# aam authentication server ocsp port1
ACOS(config-ocsp auth server:port1)# health-check monitor1
```

The following command disables a port health check on an OCSP server at the OCSP authentication server configuration level:

```
ACOS(config)# aam authentication server ocsp port1
ACOS(config-ocsp auth server:port1)# health-check-disable
```

Running a Port Health Check on a Windows Server by Using the CLI

The following command runs a port health check on a Windows server for Kerberos at the Windows authentication server configuration level:

```
ACOS(config)# aam authentication server windows server1
ACOS(config-windows auth server:1)# auth-protocol kerberos-port port1
health-check name1
```

The following command disables a port health check on a Windows server for Kerberos at the Windows authentication server configuration level:

```
ACOS(config)# aam authentication server windows server1
ACOS(config-windows auth server:1)# auth-protocol kerberos-port port1
health-check-disable
```

The default Kerberos port is port 88.

Enabling Authentication for HTTP or HTTPS Traffic

For more information about enabling authentication for HTTP or HTTPS traffic.

The following topics are covered:

Configuration Details

This feature cannot be configured by using the CLI.

To enable the authentication for HTTP or HTTPS, enter the following commands:

```
ACOS(config-health:monitor)# method {http | https}
  [username name password password
    [Kerberos-auth realm realm_name kdc ip_addr | ipv6_addr [port num]]]
```

NOTE: More information about the CLI command can be found in the “Health Monitor” topic of the *Command Line Interface Reference Guide*.

CLI Example

```
ACOS(config)# health monitor Kerberos-http
ACOS(config-health:monitor)# method http username example_username
password example_pw Kerberos-auth realm EXAMPLE.COM kdc 192.0.2.10  port
80
```

Online Certificate Status Protocol Health Monitoring

There is a health monitor that checks the health of Online Certificate Status Protocol (OCSP) servers. This health monitor supplements the other AAA-related health monitors, for RADIUS and LDAP.

Service Groups

A service group, which is a set of real servers, manages the authentication services that are managed by the ACOS device.

The following topics are covered:

Creating a Service Group

The following topics are covered:

Using the GUI

To create an authentication service group:

1. Click **AAM** > **Service Groups**.
2. Click **Create**.
3. Enter a service group name.
4. Select a protocol.
5. Select a load balancing method.
6. Click **Create**.

Using the CLI

The following command creates an authentication service-group **group1** with TCP AAM service and health-check disabled:

```
ACOS(config)# aam authentication service-group group1 tcp
ACOS(config-aam svc group:123)# health-check-disable
```

Updating a Service Group

The following topics are covered:

Using the GUI

To update an authentication service group:

1. Click **AAM** > **Service Groups**.
2. Select the service group that you want to update, and click **Edit**.
3. On the **Update Authentication Service Group** page, complete the following tasks:
4. Select another load balancing method.
5. Click **Create** to create or update service group members.

Using the CLI

To create a service group:

The following command updates the authentication service-group **group1** with UDP AAM service and health-check disabled:

```
ACOS(config)# aam authentication service-group group1 udp
ACOS(config-aam svc group:123)# health-check-disable
```

AAA and AAA Policies

The following topics are covered:

Creating an AAA Policy

The user must complete the following tasks to create an AAA policy:

1. Create an authentication template.
2. Create authentication logon, server, and relay objects based on the user requirement.
3. Bind the corresponding authentication objects that the user created earlier to the authentication template.
4. Optionally, create an authorization policy. The user do not have to create an authorization policy in an AAA policy.
5. Create the AAA policy.
6. Configure the rules in the AAA policy.

NOTE: The user only need to create the components and objects that are relevant for the user’s environment. None of the components or objects are required.

The following topics are covered:

Using the GUI

The user can create an AAA policy by referring the following GUI steps:

1. Click **AAM** > **Policies and Templates**.
2. On the **AAA Policies** tab, click **Create**.
3. Enter a policy name.
4. Select a virtual port binding, and click **Add**. Alternatively, click **New VP** to create a new virtual port.

   a. If the user created a new virtual port, either select an existing virtual server or configure a new virtual server.

5. Expand the **General Field** section and configure the appropriate options.
6. Expand the **Templates** section.
7. For each template type that the user want to use, complete one of the following tasks:
   a. Select an existing template.
   b. Click **Add+** to create a template.
8. Click **Create**.
9. On the **Update AAA Policy** page, in the **AAA Rules** section, click **Create** and enter the desired value for each parameter.
10. On the **Update AAA Policy** page, click **Update**.

Using the CLI

The following example creates an AAA policy `policy1` with `aaa-rule1` specifying `URI` path for aaa-rule1:

```
ACOS(config)# aam aaa-policy policy1
ACOS(config-aaa policy:1)# aaa-rule1
ACOS(config-aaa policy:1-aaa rule:1)# uri
```

NOTE: For information on the rules of AAA policy, see AAM AAA Rule Configuration Commands.

The following command specifies the order in which rule checking is completed:

```
aaa-rule move from index_start index_end
```

The user can configure the ACOS device to use a specific authentication policy during the policy checking.

- The following command specifies the IP address of the virtual server the user wants to use during authentication checking:

```
ACOS(config)# slb virtual-server server_nameserver_IP_address
```

- The following command specifies the type of traffic the user wants to check:

```
port port_num port_type
```

- The following command specifies the AAA policy that the user wants to use for authentication checking:

```
template AAA-policyAAA-policy_name
```

Creating an Authorization Policy

The following topics are covered:

Using the GUI

The user can create an authorization policy by referring the following GUI steps:

1. Click **AAM** > **Policies and Templates**.
2. On the **Authorization Policies** tab, click **Create**.
    a. Enter a name string and attribute rule.
    b. Click **Create**.
3. In the **Update Auth Policy** page, complete the following steps to enter an attribute rule:
    a. In the **AAA Authorization Attributes** section, click **Create.**
    b. Enter an attribute ID.
    c. Select one of the following options:
        i. A specific attribute name type.
        ii. A10 Dynamic Defined

To specify an attribute name type, complete the following steps:

a. Select an attribute type.

b. Enter an attribute name.

c. Select the integer matching scheme.

d. Enter an attribute integer value.

e. Click **Create**.

4. In the **Update Authorization Policy** page, click **Update**.

Using the CLI

The following command specifies the authorization policy that the user wants to use in the AAA policy.

```
ACOS(config)# aam authorization-policy authorization-policy_name
```

Updating an Authorization Policy

The user can update an authorization policy by referring the following GUI steps:

1. Click **AAM** > **Policies and Templates**.
2. On the **Authorization Policies** tab, select a policy, and click **Edit**.
3. In the **AAA Authorization Attribute** section, select an attribute and click **Edit**.
4. Modify the relevant attributes.
5. Click **Update**.

Using Configuration and Server Combinations

The following topics are covered:

Details

Table 1 displays the possible server combinations, which are described in more detail below as Cases 1-5:

Table 1 : Basic Server Combinations

Authentication Server	Authorization Server	Remark
All types	-	Case 1
-	LADP / RADIUS	Case 3 **NOTE:** The user must still enter the username and password to login.
LDAP	LDAP	Case 2 / 5
LDAP	RADIUS	Case 4
RADIUS	LDAP	Case 4
RADIUS	RADIUS	Case 2 / 5
SecurID / Kerberos	LDAP / RADIUS	Case 4
Client Cert	LDAP / RADIUS	Case 4 **NOTE:** The user must still enter the username and password to login.
SAML	LDAP / RADIUS	Case 4

The user can use this feature to configure one of the following combinations:

- Authentication only: Authenticate and authorize to the same authentication server and is a single-request query.
- Authorization only: Authenticate and authorize by using different types of authentication servers and is a multi-request query.
- Authentication and authorization by using same type of authentication server with a different server instance. This is a multi-request query.

Configuration Scenarios

This topic describes the possible configuration scenarios.

In all scenarios, an AAM LDAP authentication server and an SLB virtual-server VIP are pre-configured. The following example shows the configuration for the AAM authentication server and the SLB virtual-server VIP:

```
ACOS(config)# aam authentication server ldap auth-server-one
ACOS(config-ldap auth server)# host 192.0.2.100
ACOS(config-ldap auth server)# base ou=People,dc=examplelab,dc=com
ACOS(config-ldap auth server)# dn-attribute uid

ACOS(config)# slb virtual-server vip-one 203.0.113.10
ACOS(config-slb vserver)# port 80 http
ACOS(config-slb vserver-vport)# source-nat auto
ACOS(config-slb vserver-vport)# service-group http_sg
ACOS(config-slb vserver-vport)# aaa-policy aaa-policy-one
```

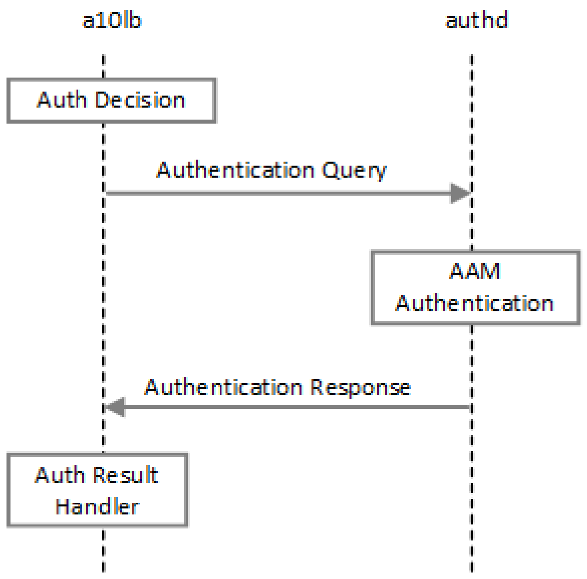
The following topics are covered:

- Case 1. Authentication Only
- Configuration Example 1: Authentication Only
- Case 2. Authentication and Authorization to the Same Authentication Server
- Configuration Example 2: Authentication and Authorization to the Same Authentication Server
- Case 3: Authorization Only
- Configuration Example 3A: Authorization Only with a Server
- Configuration Example 3B: Authorization with a Service Group
- Case 4. Authentication and Authorization by Using Different Types of Servers
- Configuration Example 4: Authentication and Authorization by Using Different Types of Servers
- Case 5. Authentication and Authorization Using Same Server Type, Different Server Instances
- Configuration Example 5: Authentication and Authorization Using Same Server Type, Different Server Instances

Case 1. Authentication Only

In this scenario, the authorization policy is not configured, and the AOCS device only completes authentication. Figure 16 illustrates the workflow.

Figure 16 : Case 1: Authentication Only



Configuration Example 1: Authentication Only

The following text is an example of how to configure this scenario:

```
ACOS(config)# aam authentication template template-one
ACOS(config-auth template:template-one)# logon logon-template-one
ACOS(config-auth template:template-one)# server auth-server-one

ACOS(config)# aam aaa-policy aaa-policy-one
ACOS(config-aaa policy:aaa-policy-one)# aaa-rule 1
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# action allow
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# authentication-template
template-one
```

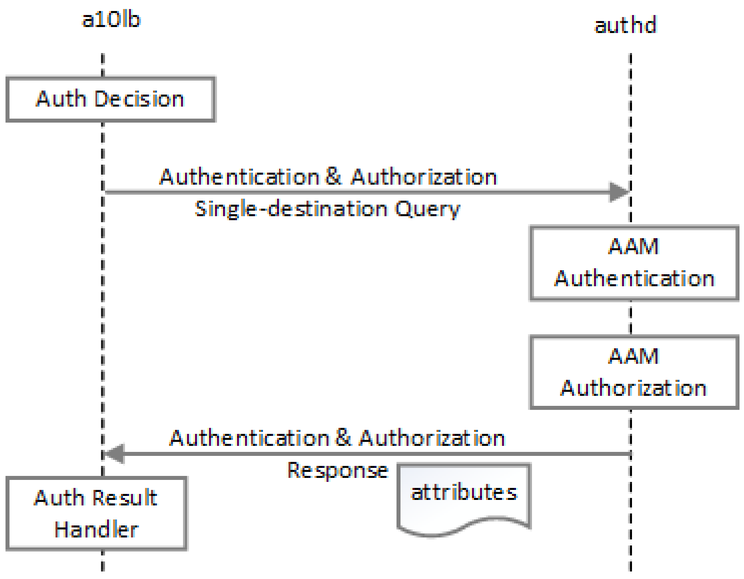
Case 2. Authentication and Authorization to the Same Authentication Server

In this scenario, the authentication server has been defined, and authorization policy has been created, but no authorization server has been configured. The ACOS device considers the authentication server and/or service-group and the authorization server and/or service-group as the same, when the following conditions are met:

- The authentication server is set, and there is no authorization server in the authorization policy.
- The authentication server is set, there is no authorization server, and both servers are the same instance.

Figure 17 illustrates the workflow.

Figure 17 : Case 2: Authentication and Authorization to the Same Authentication Server



a10lb sends a single-request query to authd, which contains only one piece of information about the server for authentication and authorization.

Configuration Example 2: Authentication and Authorization to the Same Authentication Server

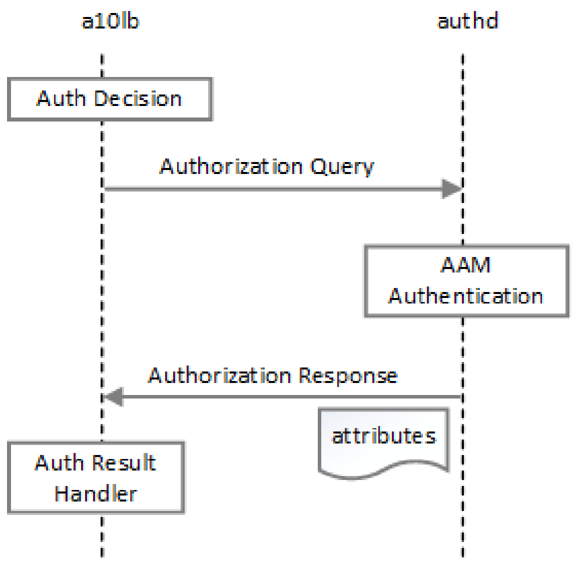
The following text is an example of how to configure this scenario:

```
ACOS(config)# aam authorization policy auth-policy-one
ACOS(config-authorization policy:auth-policy-one)# attribute 1 A10-AX-
AUTH-URI attr-
       type string match a10-dynamic-defined
ACOS(config-authorization policy:auth-policy-one)# attribute-rule 1
ACOS(config-authorization policy:auth-policy-one)# server auth-server-one
ACOS(config)# aam authentication template template-one
ACOS(config-auth template:template-one)# logon logon-template-one

ACOS(config)# aam aaa-policy aaa-policy-one
ACOS(config-aaa policy:aaa-policy-one)# aaa-rule 1
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# action allow
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# authentication-template
template-one
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# authorization-policy
auth-policy-one
```

Case 3: Authorization Only

In this case, the authorization policy is configured with the **`authorization-server`** option and the authentication template is not configured with an authentication server. Figure 18 illustrates the workflow.

Figure 18 : Case 3: Authorization Only



Configuration Example 3A: Authorization Only with a Server

The following text is an example of how to configure this scenario:

```
ACOS(config)# aam authorization policy auth-policy-one
ACOS(config-authorization policy:auth-policy-one)# attribute 1 A10-AX-
AUTH-URI attr-
        type string match a10-dynamic-defined
ACOS(config-authorization policy:auth-policy-one)# attribute-rule 1
ACOS(config-authorization policy:auth-policy-one)# server auth-server-one

ACOS(config)# aam authentication template template-one
ACOS(config-auth template:template-one)# logon logon-template-one
ACOS(config-auth template:template-one)# relay ldap_relay_template

ACOS(config)# aam aaa-policy aaa-policy-one
ACOS(config-aaa policy:aaa-policy-one)# aaa-rule 1
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# action allow
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# access-list 1
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# authentication-template
template-one
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# authorization-policy
auth-policy-one
```

Configuration Example 3B: Authorization with a Service Group

The following text is an example of how to configure this scenario:

```
ACOS(config)# aam authentication server ldap auth-server-two
ACOS(config-ldap auth server)# host 192.0.2.101
ACOS(config-ldap auth server)# base ou=People,dc=examplelab,dc=com
ACOS(config-ldap auth server)# dn-attribute uid

ACOS(config)# aam authentication service-group auth-sg tcp
ACOS(config-aam svc group:auth-sg)# member auth-server-one 389
ACOS(config-aam svc group:auth-sg)# member auth-server-two 389

ACOS(config)# aam authorization policy auth-policy-one
ACOS(config-authorization policy:auth-policy-one)# attribute 1 A10-AX-
AUTH-URI attr-
        type string match a10-dynamic-defined
ACOS(config-authorization policy:auth-policy-one)# attribute-rule 1
ACOS(config-authorization policy:auth-policy-one)# server service-group
auth-sg
```

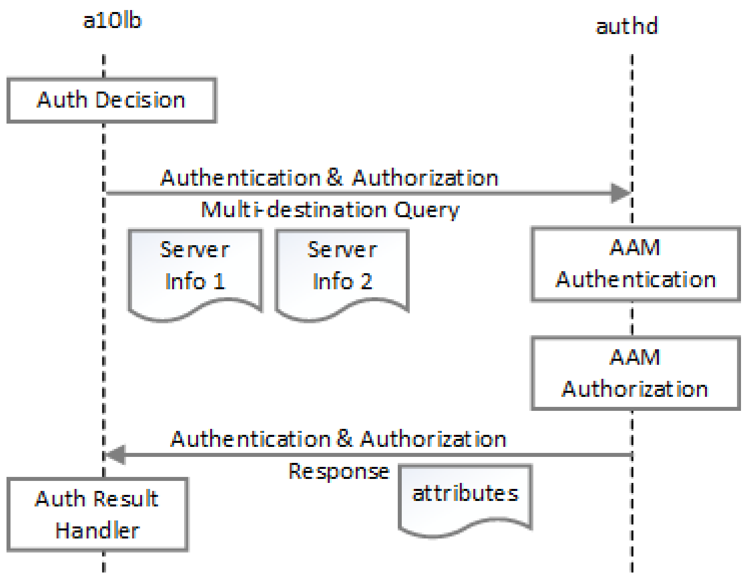
NOTE: The user can only use an LDAP or RADIUS authentication server.

```
ACOS(config)# aam authentication template template-one
ACOS(config-auth template:template-one)# logon logon-template-one

ACOS(config)# aam aaa-policy aaa-policy-one
ACOS(config-aaa policy:aaa-policy-one)# aaa-rule 1
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# action allow
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# access-list 1
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# authentication-template
template-one
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# authorization-policy
auth-policy-one
```

Case 4. Authentication and Authorization by Using Different Types of Servers

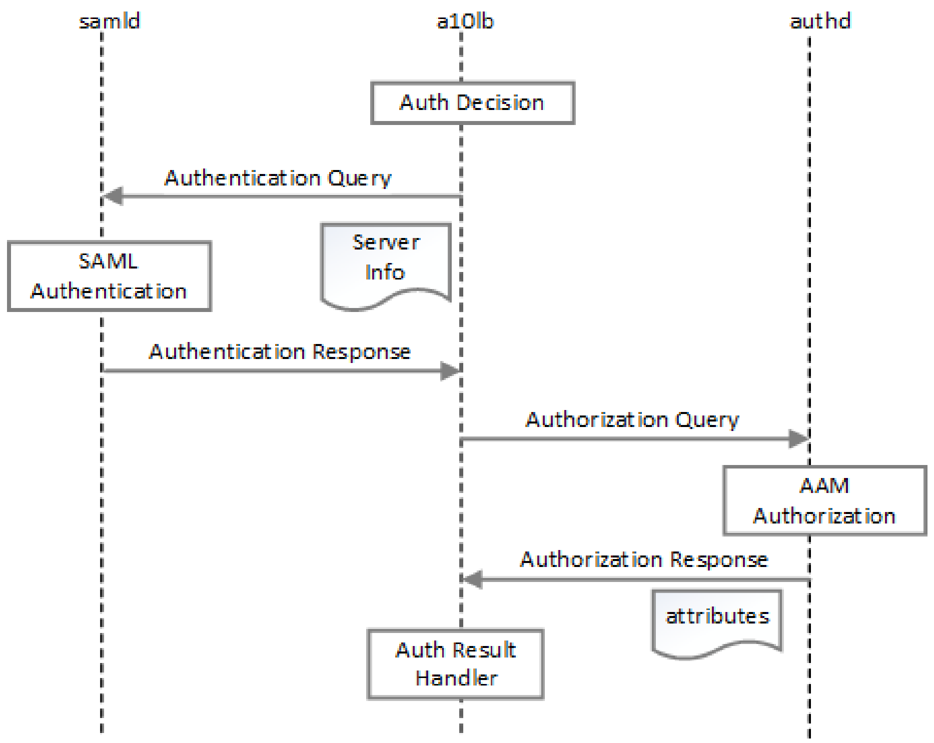
The authorization policy is configured with the `server` option and the authentication template has an authentication server. After the client has been successfully authenticated, the ACOS device completes the authorization. Figure 19 illustrates the workflow. a10lb sends a multi-request query to authd. The query contains two pieces of information for the server, one piece of information for authentication, and two pieces of information for authorization.

Figure 19 : Case 4:Authentication and Authorization by Using Different Types of Servers



If the combination of authentication and authorization is SAML + LDAP/RADIUS, the work flow is illustrated in Figure 20.

Figure 20 : Workflow for SAML and RADIUS Authentication and Authorization



In Figure 20, a10lb sends an authentication query to samd and then an authorization query to authd.

Configuration Example 4: Authentication and Authorization by Using Different Types of Servers

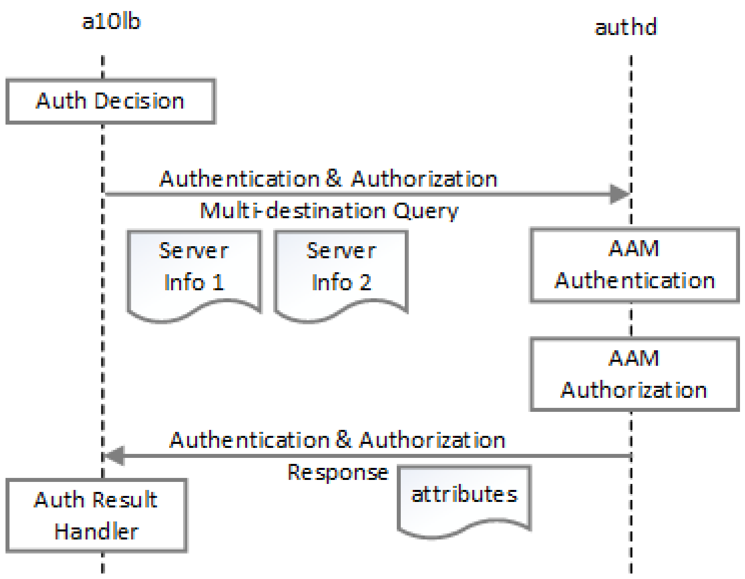
The following text is an example of how to configure this scenario:

```
ACOS(config)# aam authentication server radius auth-radius-server
ACOS(config-radius auth server:auth-radius-server)# host 192.0.2.10
ACOS(config-radius auth server:auth-radius-server)# secret passcode-string
ACOS(config-radius auth server:auth-radius-server)# port 1812 health-
check-disable

ACOS(config)# aam authorization policy auth-policy-one
ACOS(config-authorization policy:auth-policy-one)# attribute 1 A10-AX-
AUTH-URI attr-
        type string match a10-dynamic-defined
ACOS(config-authorization policy:auth-policy-one)# attribute-rule 1
ACOS(config-authorization policy:auth-policy-one)# server auth-radius-
server

ACOS(config)# aam authentication template template-one
ACOS(config-auth template:template-one)# logon logon-template-one
ACOS(config-auth template:template-one)# server auth-server-one

ACOS(config)# aam aaa-policy aaa-policy-one
ACOS(config-aaa policy:aaa-policy-one)# aaa-rule 1
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# action allow
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# access-list 1
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# authentication-template
template-one
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# authorization-policy
auth-policy-one
```

Case 5. Authentication and Authorization Using Same Server Type, Different Server Instances

In this case, a10lb sends a multi-request query to authd. Figure 21 illustrates this workflow.

Figure 21 : Case 5: Authentication and Authorization Using the Same Types of Servers, Different Server Instances



In Figure 21, a10lb sends a multi-request query to authd. The multi-destination query contains two pieces of server information, one piece of information for authentication, and two pieces of information for authorization.

Configuration Example 5: Authentication and Authorization Using Same Server Type, Different Server Instances

The following text is an example of how to configure this scenario.

```
ACOS(config)# aam authentication server ldap auth-server-two
ACOS(config-ldap auth server)# host 192.0.2.101
ACOS(config-ldap auth server)# base ou=People,dc=examplelab,dc=com
ACOS(config-ldap auth server)# dn-attribute uid

ACOS(config)# aam authorization policy auth-policy-one
ACOS(config-authorization policy:auth-policy-one)# attribute 1 A10-AX-
AUTH-URI attr-
        type string match a10-dynamic-defined
ACOS(config-authorization policy:auth-policy-one)# attribute-rule 1
ACOS(config-authorization policy:auth-policy-one)# server auth-server-one

ACOS(config)# aam authentication template template-one
ACOS(config-auth template:template-one)# logon logon-template-one
ACOS(config-auth template:template-one)# server auth-server-two

ACOS(config)# aam aaa-policy aaa-policy-one
ACOS(config-aaa policy:aaa-policy-one)# aaa-rule 1
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# action allow
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# access-list 1
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# authentication-template
template-one
ACOS(config-aaa policy:aaa-policy-one-aaa rule:1)# authorization-policy
auth-policy-one
```

Configuring by Using the CLI or GUI

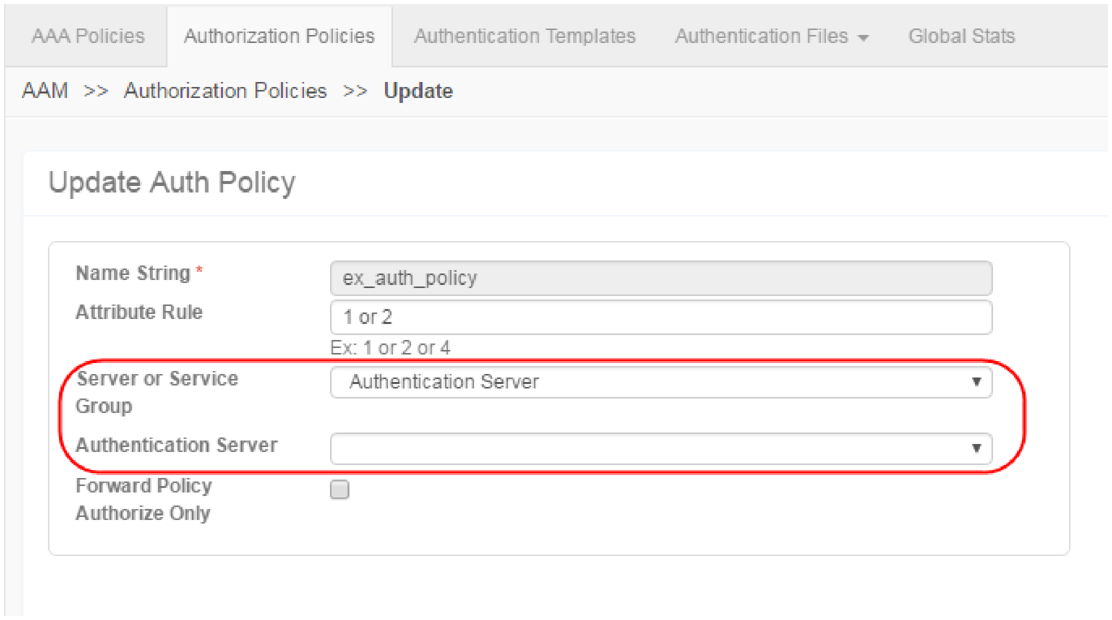
The user can configure this feature by using the GUI or the CLI.

In authentication and authorization scenarios, after the user sets the server or service group in the authorization policy, bind the authorization policy. In authorization-only scenarios, bind the authorization template with a server or a service group on an AAA policy or an AAA rule.

Using the GUI

1. On the **Update Authorization Policy** page, select the server in the **Authorization Server** drop-down list, then click **Update**.

Figure 22 : Using the GUI to Set Authentication and Authorization



Update Auth Policy

Name String *	ex_auth_policy
Attribute Rule	1 or 2
	Ex: 1 or 2 or 4
Server or Service Group	Authentication Server
Authentication Server	
Forward Policy Authorize Only	☐

2. In the **Authorization Server** drop-down menu, the user can select **LDAP** or **RADIUS** server object or a service group object that contains LDAP or RADIUS as the assigned value.

Using the CLI

The user can add the authorization server when the user configures the authorization policy.

The following text is an example of this command:

```
aam authorization policy example_auth_policy
   attribute 1 A10-AX-AUTH-URI attr-type string match a10-dynamic-defined
   attribute-rule 1
   server example_auth_server
```

NOTE:

A CLI error occurs if the user tries to perform one of the following:

- Sets OCSP or Kerberos as the authorization server or service group.
- Binds an authorization policy that has no server or a service group with an authentication template, and the authentication server is OSCP, Kerberos, or is empty.

Creating an Authentication Template

The following topics are covered:

Using the GUI

To create an authentication template using the GUI:

1. Click **AAM** > **Policies and Templates**.
2. On the **Authentication Templates** tab, click **Create**.
3. Enter a name.
4. Select one of the following template types:
   - **Standard**
   - **SAML**

Using a Standard Template Type

1. Enter a logout URL.
2. Select an existing authentication logon type or click **New Authentication Logon.**

3. In the **Update Authentication Template** page, select an authentication logon.
4. Select an existing authentication relay or click **New Authentication Relay**.
5. Select Chain checkbox, then Server or Service Group table is visible
6. Specify either Server or Service Group and the priority.
7. In **Server or Service Group**, select an authentication server or a service group.
   - If the user selected **Authentication Server**, then the user must either select an existing authentication server or click **New Authentication Server**.
   - If the user selected **Service Group**, then the user either select an existing service group or click **New Service Group**.
8. In **Account**, either select an existing account or click **New Account**.
9. (Optional) In Authentication Session Mode, choose a method for tracking sessions from the drop down list.
   - Cookie Based - Select this to check and track sessions through an authentication key cookie (auth-key)
   - IP Based - Select this to check sessions by client IP.
10. Note: For more information, seeTracking Sessions.
11. Select an option for the log.
12. Enter a cookie domain and click **Add**.
13. Enter a cookie domain group ID and click **Add**.

Using a SAML Template Type

1. Select a SAML service provider.
2. Select a SAML identity provider.
3. Select an existing authentication relay or click **New Authentication Relay**.
4. Select an option for the log.
5. Enter a cookie domain and click **Add**.
6. Enter a cookie domain group ID and click **Add**.
7. Click **Create**.

Updating an Authentication Template Using the GUI

To update an authentication template:

1. Click **AAM** > **Policies and Templates**.
2. On the **Authentication Templates** tab, select the authentication template that the user wants to modify.
3. Click **Edit**.
4. Modify the relevant values and options for the template.
5. Click **Update**.

Using the CLI

The following command creates an authentication template to be used in the AAA policy:

```
ACOS(config)# aam authentication template authentication_template_name
```

The following command creates an authentication object that the user can bind to the authentication template:

```
ACOS(config)# aam authentication {logon | relay | server} object_name
```

The user must create at least one object in an AAA policy.

NOTE: To specify a logging level, enter the **log** {**success** | **failure**} command.

Multiple servers/service-groups are added in the authentication template using option `chain` The server in the chain can be an authentication server or authentication service group. Use the following command too create authentication server or authentication service group and assign priority.

```
ACOS(config)# aam authentication template <template-name>
ACOS(config-auth template:template-name)# logon <logon-name>
ACOS(config-auth template:template-name)# chain server <server-name>
priority <1-5>
ACOS(config-auth template:template-name)# chain service-group <sg-name>
priority <1-5>
```

NOTE:

- Priority number defines the priority. 5 being highest and 1 is lowest priority.
- Duplicate priority is not permitted.
- Option `chain`, it is mutually exclusive for `server` and `service-group`.
- Maximum five server/service-group chain entries are allowed.

The `auth-failure-byapss` forwards the client request to real server even if authentication is failed. The option `auth-failure-byapss` is configured under aaa rule. When this feature is configured it works in conjunction with the server chain and authenticates the ACOS first and then all servers in the chain, if the user is successfully authenticated with any server in chain, `auth-failure-bypass` is not executed.

```
ACOS(config)# aam aaa-policy aaa-policy-name
ACOS(config-aaa policy:policyname)# aaa-rule num
ACOS(config-aaa policy:policyname-aaa rule:...)# authenticationtemplate
auth-template-name
ACOS(config-aaa policy:policyname-aaa rule:...)# auth-failure-bypass
```

Tracking Sessions

An AAA policy can be configured for explicit proxy through the use of `auth-sess-mode` from the CLI in authentication template sub-configuration, or by selecting one of the available tracking methods from the drop-down list in Authentication Session Mode using the GUI (See (Optional) In Authentication Session Mode, choose a method for tracking sessions from the drop down list.Cookie Based - Select this to check and track sessions through an authentication key cookie (auth-key)IP Based - Select this to check sessions by client IP. in Creating an Authentication Template). This enables ACOS to check for authenticated sessions to determine whether AAM is needed for client requests to an explicit proxy or take action to get the necessary proxy information for authentication and authorization and track for future requests in one of two ways.

- Cookie-based

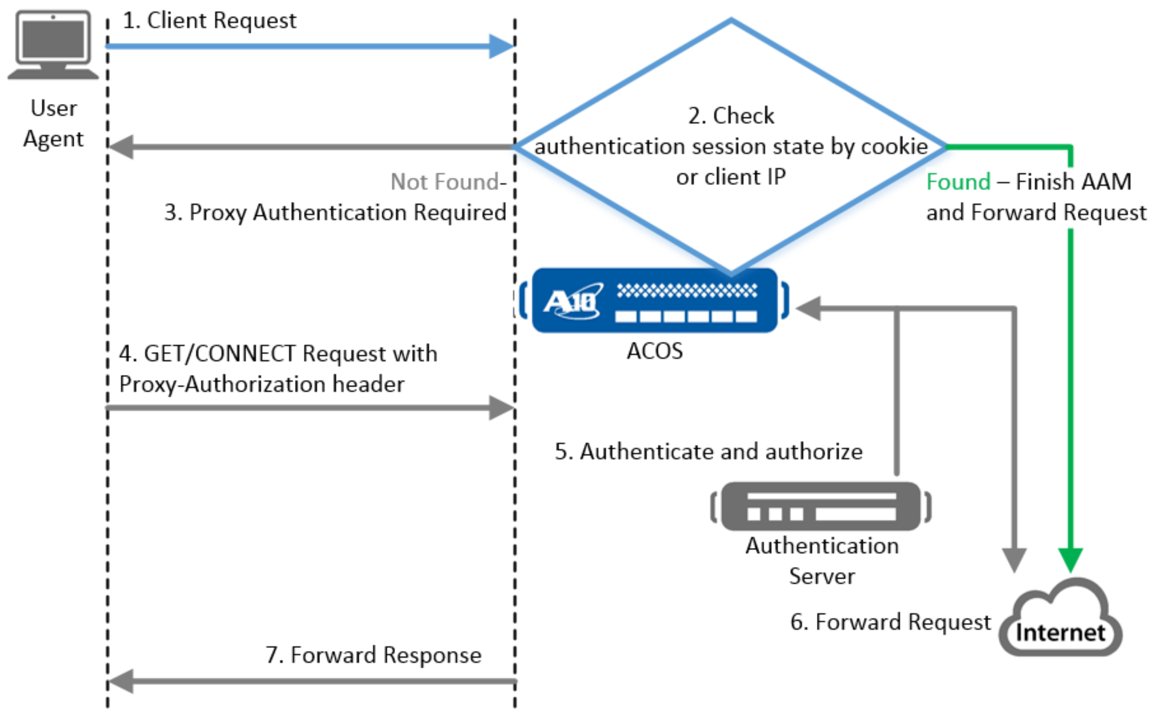
  ACOS checks the Auth-Key cookies to determine whether session has been authenticated.

- IP-based

  ACOS checks the client IP to determine whether session has been authenticated. This cannot be used if clients share the same IP.

NOTE: See Explicit Proxy Permission with AAM Policy in the *Application Delivery Controller and Server Load Balancing Guide* for a configuration example using tracking sessions.

Figure 23 : General Flow for HTTP/HTTPS Logon (No Form Logon)


1. Client Request
User Agent
2. Check authentication session state by cookie or client IP
Not Found-
3. Proxy Authentication Required
Found – Finish AAM and Forward Request
ACOS
4. GET/CONNECT Request with Proxy-Authorization header
5. Authenticate and authorize
Authentication Server
6. Forward Request
Internet
7. Forward Response

Limitations

This feature has the following limitations:

- For form-based logon of HTTPS, the SSLi module must be enabled to ensure the ACOS logon request is respected.
- If the action `forward-to-internet` is configured for an explicit proxy, authentication relay is not supported.
- If SSLi-bypass is configured, form-based logon will not work for HTTPS connections.
- It only takes effect if the virtual port is part of an explicit proxy or transparent proxy configuration.
- If a form-based logon is encountered in a cookie-based configuration, AAM will automatically switch to ip-based mode to ensure form content is correctly displayed.

JWT (JSON Web Token) Authorization (AAM with JWT)

This is a feature to enhance the EP to access control http/https request, based on the ID Token of the OpenID Connect, in a given http header. This can be referred as the SLB policy template support along with the JWT authorization.

The following topics are covered:

Overview

JSON Web Token (JWT) is an open standard (RFC 7519) that creates a compact, self-contained, and URL-safe access tokens to provide security information between parties. The token is digitally signed and encrypted, so that it can be verified and trusted. JWT is useful for authorization and information exchange between parties.

Thunder AAM is capable to authorize requests containing JWTs, generate JWTs, and transform SAML assertions to JWT.

The following topics are covered:

JWT Authorization

JWTs are used for controlling access to resources. When the user successfully logs in with the assigned credentials, a JWT is returned to the user-agent (which is a web browser). Then, when the user wants to access to a protected resource, the user agent sends the JWT with the access request. Typically, the JWT is in the HTTP Authorization header, using the Bearer schema, and application servers, which receives the JWT, and it can verify the JWT, and authorize the request through the claims contained in the JWT.

The JWT contains the topics such as header, payload, and signature as three parts. Each part is a `base64url` encoded JSON-based message and is separated by '.' character.

The following topics are covered:

- Header
- Payload
- Signature

Header

Typically the ***Header*** contains the token type and the signature algorithm.

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

Payload

Typically the ***Payload*** contains claims to the authorization for information exchange.

```
{
  "iss": "http://auth.a10-tplab.com",
  "sub": "z3gE4i_uQe0M",
  "aud": "john@a10-tplab.com",
  "exp": 1546214399,
  "iat": 1540361069,
  "name": "john",
  "email": "john@a10-tplab.com",
  "groups": [
    "Engineer",
    "TP"
  ]
}
```

The payload is signed using a private key corresponding to the chosen algorithm. ACOS supports the following signature algorithms for token creation and verification process:

- RS256 (RSA Signature using SHA-256): This widely used algorithm is based on the RSA algorithm and uses asymmetric key pairs for signing and verification.
- ES256 (Elliptic Curve Digital Signature Algorithm (ECDSA) using P-256 and SHA-256): This algorithm is based on elliptic curve cryptography and offers strong security with smaller key sizes compared to the RSA-based algorithms. It provides efficient and robust authentication and data integrity verification.

You can choose the algorithm based on security requirements, performance considerations, and compatibility with existing systems.

Signature

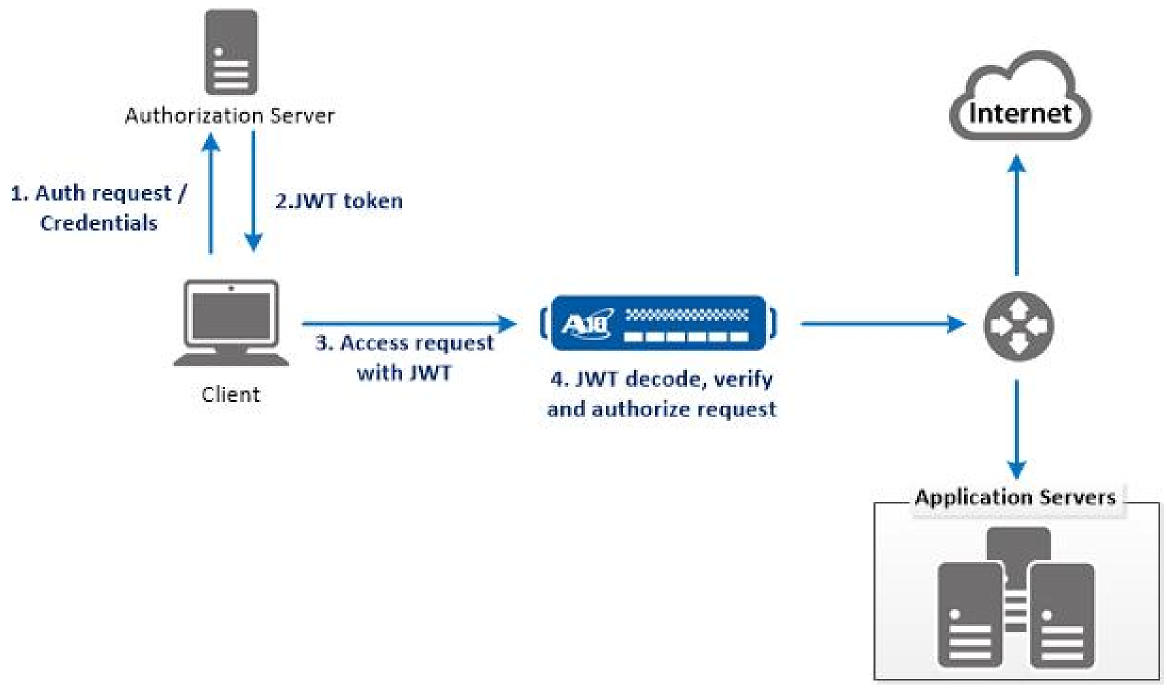
The ***signature*** part is used to verify the token and the signing algorithm is specified in JWT header by “**alg**” claim (in this case **HS256**). The signature is calculated by signing the message of “**base64url encoded header**” **+** '.' **+** “**base64url encoded payload**”.

```
HMAC-SHA256(
encodeBase64Url(header) + '.' +
encodeBase64Url(payload),
)
```

The JWT for this example looks like the following:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
.eyJpc3MiOiJodHRwOi8vYXV0aC5hMTAtdHBsYWIuY29tIiwic3ViIjoiejNnRTRpX3VRZTBNI
iwiYXVkIjoiam9obkBhMTAtdHBsYWIuY29tIiwiZXhwIjoxNTQ2MjE0Mzk5LCJpYXQiOjE1NDA
zNjEwNjksIm5hbWUiOiJqb2huIiwiZW1haWwiOiJqb2huQGExMC10cGxhYi5jb20iLCJncm91c
HMiOlsiRW5naW5lZXIiLCJUUCJdfQ._LiWg2iHFRIpDf6xxdI5-vhifGPvtZmjJe0Ntdhfpxc
```

Traffic Walkthrough (Example Use Case)


Authorization Server
1. Auth request / Credentials
2.JWT token
Internet
Client
3. Access request with JWT
4. JWT decode, verify and authorize request
Application Servers

The example use case and traffic walkthrough are as the following:

1. Client authenticates with the authorization server.
2. If authentication succeeds, the authorization server provides a JWT to the client.
3. When the client accesses the protected resource, the user agent (which is the browser) sends requests with a JWT in HTTP Authorization header, using Bearer schema.

4. Thunder receives the request with the JWT and performs the following:
    a. Verifies the JWT signature ensuring the token is issued by a trusted party.
    b. Decodes the JWT and checks the validity, such as mandatory claims check, token expiration check, and so on.
    c. Authorizes the request using the claims in JWT and configures the authorization policy.
5. Allows or rejects the request according to the result of ***step 4***.

Configuring JWT Authorization Template

This configuration is performed for the Thunder JWT verification and authorization.

The verification is configured in JWT authorization template, and the rule for JWT claims authorization is configured in AAM authorization template (which is described in the Configuring Authorization Policy with JWT Template topic).

In the JWT authorization template, the user can specify, how it can be verified by using the JWT signature, token expiration behavior, token caching for performance, and JWT forwarding behavior.

The following topics are covered:

Verifying JWT Signature by Secret

The **`verification-secret`** option verifies the JWT signature using a shared secret.

Configure a JWT authorization template **test** which uses a secret **password** to verify the incoming JWTs:

```
ACOS(config)#aam jwt-authorization test
ACOS(config-jwt-authz:test)# verification-secret password
```

Verifying JWT Signature by Certificate

The **verification-cert** option allows you to verify JWT signature using the specified certificate.

Configure a JWT authorization template **test** which uses an imported certificate **pubkey.pem** to verify the JWT signature.

```
ACOS(config)# aam jwt-authorization test
ACOS(config-jwt-authz:test)# show pki cert
Name: pubkey.pem  Type: certificate  Expiration: Aug  8 01:24:59 2024 GMT
[Unexpired, Unbound]
ACOS(config-jwt-authz:test)# verification-cert pubkey.pem
```

Verifying JWT Signature by JWKS

The **verification-jwks** option allows you to specify a JWK file to verify the JWT signature.

First, you need to import a JWT file (ES256 or RS256 based) which contains the key information to verify the signature.

```
ACOS(config)# import auth-jwks jwk scp://john@192.168.92.54:/jwk
Password []?
Done
```

After importing the file, you need to configure a JWT authorization template (**test** in this case) to use the JWK file **jwk** for verifying the signatures of the incoming JWTs.

```
ACOS(config)# aam jwt-authorization test
ACOS(config-jwt-authz:test)# verification-jwks jwk
```

JWT Signature Expiration Behavior

The **exp-claim-required** option allows you to configure the behavior for JWT signature expiration.

By default, the **exp** claim is used to identify the expiration time of the token. It is not mandatory, but, if it is present in JWT claims, then the Thunder denies the expired JWTs.

The ACOS provides an option called **exp-claim-required**, which requires **exp** claim in the client JWT. The JWTs without **exp** claims (or expired) are denied.

```
ACOS(config)# aam jwt-authorization test
ACOS(config-jwt-authz:test)# exp-claim-required
```

JWT Caching Mechanism

By default, ACOS performs the JWT verification and authorization for every request with JWT. But only after a client is authenticated to the authorization server and gets the JWT. It then uses this JWT to send many access requests. For verified and authorized JWTs, it is not necessary to verify again and authorize it every time. Therefore, ACOS provides a token caching mechanism to cache verified JWTs and prevent repeated verification of the same JWT every time.

ACOS stores the previously verified JWTs. When a matching JWT is found in the cache, ACOS skips redundant verification and authorization checks.

When JWT caching is enabled:

- Tokens are cached until their exp claim expires.
- If a token does not include an exp claim, ACOS uses a default cache duration of 1800 seconds (configurable).
- If a token includes an exp claim but its TTL exceeds 2592000 seconds, ACOS automatically limits the TTL to this maximum allowed value.

CLI Configuration

To enable JWT cache to be enabled, use the following command:

```
ACOS(config)#aam jwt-authorization test
ACOS(config-jwt-authz:test)#jwt-cache-enable
```

To configure the default token cache lifetime, use the following command:

```
ACOS(config)#aam jwt-authorization test
ACOS(config-jwt-authz:test)#jwt-exp-default <1-2592000>
```

Show Command

To view cached JWTs to trace and verify which tokens are currently stored in the cache:

```
ACOS#show aam authorization jwt cache
Token cached: 1
Token cache hit: 1
Max token cache: 102400
---------------------------------------------------------------------------
------
ID     Issuer                                        TTL
       Subject                Audience               Client IP
---------------------------------------------------------------------------
------
4      A10-TPLAB                                     2591968
       jwt-test@10-tplab.com  www.a10-tplab.com      1092:1068:10::10
```

JWT Forwarding Behavior

For the JWT authorization, client needs to provide JWT in HTTP Authorization header to Thunder. But, after the request is accepted, the request is then forwarded to the application server or the internet. Then the JWT is no longer necessary in the request. Therefore, by default AAM removes the JWT, after the authorization. In this scenario, the ACOS provides an option to keep the JWT after the authorization.

```
ACOS(config)# aam jwt-authorization test
ACOS(config-jwt-authz:test)# jwt-forwarding
```

JWT Authorization Logging

The ACOS provides syslog for the JWT authorization, in which the level for logging is configured by log-level option in the JWT authorization template.

Configure the following command to enable the logging for all events:

```
ACOS(config)#aam jwt-authorization test
ACOS(config-jwt-authz:test)#log-level ?
  0  log disable
  1  only log authorization fail (default)
  2  only log authorization success
  3  log all
```

```
ACOS(config-jwt-authz:test)#log-level 3
```

Configure JWT Cache Capacity at Global Level

If you want to increase the global or system level capacity for a high-traffic environment, you configure the JWT cache capacity up to 1024000 entries, depending on platform memory and system-level cache limits.

Follow the steps below to configure and verify the system-level JWT cache capacity.

1. Execute the following command to verify your system's available memory to determine the supported JWT cache capacity range.

```
ACOS(config)#show hardware
Thunder Series Unified Application Service Gateway TH3350
      Serial No   : TH330E5024100034
      CPU         : Intel(R) Xeon(R) D-2146NT CPU @ 2.30GHz
                    16 cores
                    4 stepping
      Storage     : Single 232G drive, Free storage is 160G
      Memory      : Total System Memory 32589 Mbytes, Free Memory 20481
Mbytes
      SSL Cards   : 2 device(s) present
                    2 QAT SSL device(s)

      L2/3 ASIC   : 0 device(s) present

      Ports       : 20
      Flags       : CF
      SMBIOS      : Build  5.14
                     08/25/2023
      MCPLD Type : 1
                    Date: 11/12/2019
```

2. Execute the following command to verify your system's maximum JWT cache capacity range.

```
ACOSconfig)#show system resource-usage
Resource                            Current    Default    Minimum    Maximum
----------------------------------------------------------------------------
l4-session-count                    67108864   67108864   16777216   134217728
nat-pool-addr-count                 2000       2000       500        10000
class-list-ipv6-addr-count          4096000    4096000    4096000    8192000
class-list-ac-entry-count           3072000    3072000    3072000    6144000
auth-portal-html-file-size          20         20         4          120
auth-portal-image-file-size         6          6          1          80
max-aflex-file-size                 32         32         16         1024
aflex-table-entry-count             102400     102400     102400     20971520
max-aflex-authz-collection-number   512        512        256        4096
radius-table-size                   8000000    8000000    4000000    8000000
monitored-entity-count              131584     131584     3840       162816
authz-policy-number                 128        128        32         2000
ram-cache-memory-limit              12288      12288      3072       12288
ipsec-sa-number                     10000      10000      40         10000
auth-session-count                  419424     419424     41942      419424
class-list-entry-count              8192000    8192000    8192000    16384000
ngwaf-cache-entry                   4800000    4800000    480000     4800000
jwt-cache-entry                     102400     102400     102400     819200
```

NOTE:

- By default, the **Current** field JWT cache value matches the **Default** field value (that is, 102400).
- The **Current** field value updates only when a custom cache size is configured.
- The **Maximum** field shows the highest supported cache value for the platform.

3. Configure the JWT cache capacity by entering a value between 102400 and the system-supported maximum value.

```
ACOS(config)# system resource-usage jwt-cache-entry 819100
```

NOTE: The system rejects JWT cache values that are higher than the supported memory capacity.

4. Execute the following command again to verify the configured JWT cache capacity.

```
ACOS(config)#show system resource-usage
Resource                           Current    Default    Minimum    Maximum
----------------------------------------------------------------------------
l4-session-count                   67108864   67108864   16777216   134217728
nat-pool-addr-count                2000       2000       500        10000
class-list-ipv6-addr-count         4096000    4096000    4096000    8192000
class-list-ac-entry-count          3072000    3072000    3072000    6144000
auth-portal-html-file-size         20         20         4          120
auth-portal-image-file-size        6          6          1          80
max-aflex-file-size                32         32         16         1024
aflex-table-entry-count            102400     102400     102400     20971520
max-aflex-authz-collection-number  512        512        256        4096
radius-table-size                  8000000    8000000    4000000    8000000
monitored-entity-count             131584     131584     3840       162816
authz-policy-number                128        128        32         2000
ram-cache-memory-limit             12288      12288      3072       12288
ipsec-sa-number                    10000      10000      40         10000
auth-session-count                 419424     419424     41942      419424
class-list-entry-count             8192000    8192000    8192000    16384000
ngwaf-cache-entry                  4800000    4800000    480000     4800000
jwt-cache-entry                    819100     102400     102400
819200
```

NOTE:

- The **Current** value should match your configured value.
- To view JWT cache details, such as the number of tokens cached and cache hits, refer to the Show Command topic.

Configuring the JWT Response Code

When JWT (JSON Web Token) validation fails, you can customize or configure the HTTP response code between 400 and 499 that is returned to the client under the

JWT Authorization template. This applies to scenarios when the token is expired, invalid, or missing.

This configuration allows the server to inform the clients regarding authentication and authorization failures.

By default, the system returns 401 (Unauthorized) for AAM authentication flows and 407 (Proxy Authentication Required) for Explicit Proxy or Proxy Chaining flows.

To configure the JWT response code, use the following command:

```
ACOS(config)# aam jwt-authorization j1
ACOS(config-jwt-authz:j1)# client-error-resp-code <400-499>
```

Configuring Authorization Policy with JWT Template

The following topics are covered:

AAM Authorization Policy with JWT

For token attributes (or claims) authorization, ACOS AAM provides authorization policy for the user to configure attributes and rules to authorize client tokens.

The **`AAM authorization policy`** works on JWTs, the user needs to bind the JWT authorization template to the authorization policy. And then the user must configure the attributes (which are the claims that need to be authorized) and the attribute rules for it.

The following is an example to create an authorization policy “**`p1`**” and to bind the JWT authorization template “**`test`**” to it.

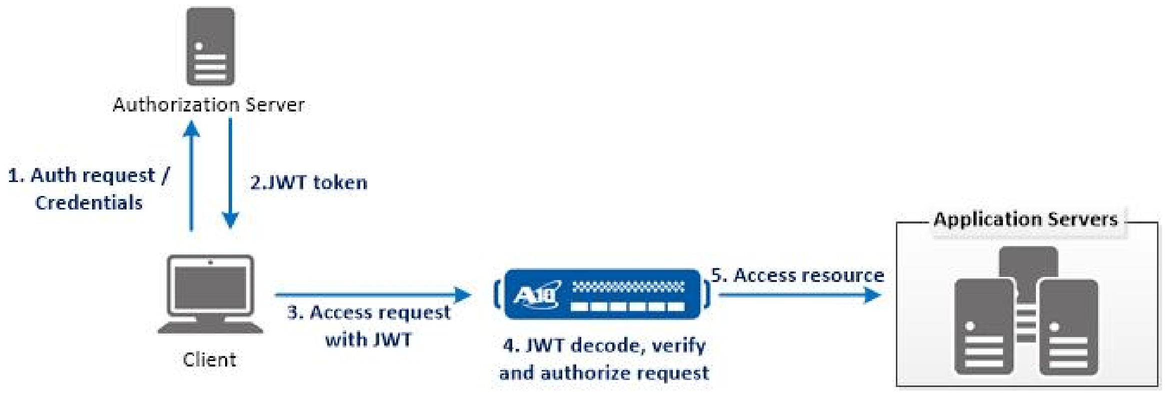
```
ACOS(config)#aam authorization policy p1
ACOS(config-authorization policy:p1)#jwt-authorization test
```

Further, the matching tokens by “**`groups`**” claim with “**`Engineer`**” or “**`TP`**” value.

```
ACOS(config-authorization policy:p1)#attribute 1 groups attr-type string
match Engineer
```

```
ACOS(config-authorization policy:p1)#attribute 2 groups attr-type string
match TP
ACOS(config-authorization policy:p1)#attribute-rule 1 or 2
```

AAM JWT Authorization Example Configuration


Authorization Server
1. Auth request / Credentials
2.JWT token
Client
3. Access request with JWT
4. JWT decode, verify and authorize request
5. Access resource
Application Servers

This example configures a virtual server which is required for the client to provide JWTs with the correct "**groups**" claim before accessing the application server resource, as in the following example for the JWT with expected claims:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
{
  "iss": "http://auth.a10-tplab.com",
  "sub": "z3gE4i_uQe0M",
  "aud": "john@a10-tplab.com",
  "exp": 1546214399,
  "iat": 1540361069,
  "name": "john",
  "email": "john@a10-tplab.com",
  "groups": [
    "Engineer",
    "TP"
  ]
}
```

For this scenario, Thunder needs a virtual server with **`aaa-policy`** binding on it. And the **`aaa-policy`** contains authorization policy, which binds with a JWT authorization template.

```
aam jwt-authorization test
  verification-secret password
  jwt-cache-enable
  log-level 3
  exp-claim-required
!
aam authorization policy p1
  attribute-rule 1 or 2
  jwt-authorization test
  attribute 1 groups attr-type string match Engineer
  attribute 2 groups attr-type string match TP
!
aam aaa-policy aaa-policy-vip1-http
  aaa-rule 1
    action allow
    authorize-policy p1
!
slb server web 192.168.90.132
  health-check-disable
  port 80 tcp
    health-check-disable
!
slb service-group http-sg tcp
  member web 80
!
slb virtual-server vip1 192.168.91.56
  port 80 http
    source-nat auto
    service-group http-sg
    aaa-policy aaa-policy-vip1-http
!
```

In this configuration, the user configures a JWT authorization template "**`test`**", which uses **`HS256`** with a secret "**`password`**" to verify the JWT signature. It also performs the following ***Optional*** tasks:

- JWT caching mechanism is enabled (optional)
- Syslog log level is log_all (optional)
- Token expiration claim 'exp' is required in the JWT (optional)

Further to this, the authorization policy "**p1**" is binded with this JWT authorization template, which then only allows the client requests with "**Engineer**" or "**TP**" groups claim in their JWT.

In addition, the authorization policy "**p1**" is binding to aaa-rule 1 of aaa-policy "**aa-policy-vip1-http**" with action "**allow**". Then it binds this aaa-policy "**aaa-policy-vip1-http**" to the target virtual server (which is the **vip1**) ports.

Authorizing Forward-Policy with JWT

The following topics are covered:

Forward-Policy JWT Authorization

For SSLi explicit and transparent proxy, **AAM authorization policy** can also be configured as the **forward-policy** source matching criteria. Therefore, the ACOS can provide JWT authorization feature for **forward-policy**.

NOTE: For more on **forward-policy**, see *SSL Insight (SSLi) Configuration Guide* and refer the *Explicit and Transparent Proxy* topic.

As a primary step, the user must configure JWT authorization template "**test**" and authorization policy "**p1**" for **forward-policy** JWT authorization.

```
ACOS(config)#aam jwt-authorization test
ACOS(config-jwt-authz:test)#verification-secret password
ACOS(config-jwt-authz:test)#jwt-cache-enable
ACOS(config-jwt-authz:test)#exp-claim-required
ACOS(config-jwt-authz:test)#aam authorization policy p1
ACOS(config-authorization policy:p1)#jwt-authorization test
ACOS(config-authorization policy:p1)#attribute 1 groups attr-type string
match Engineer
```

```
ACOS(config-authorization policy:p1)#attribute 2 groups attr-type string
match TP
ACOS(config-authorization policy:p1)#attribute-rule 1 or 2
```

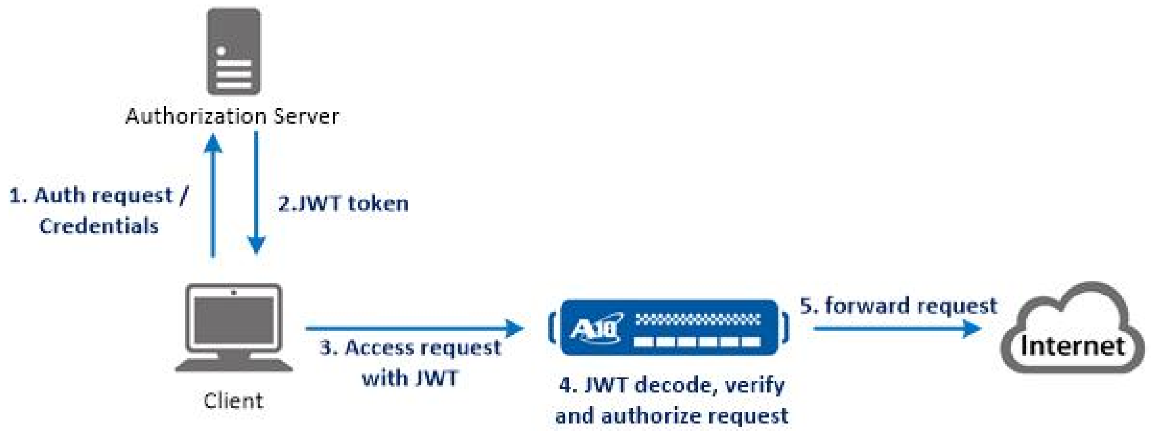
Then, as the next step, the user must configure **`forward-policy`** “**`HTTP-policy`**” with the authorization policy “**`p1`**” as the source matching criteria in the source rule “**`S1`**”.

```
ACOS(config)#slb template policy HTTP-policy
ACOS(config-policy)#forward-policy
ACOS(config-policy-forward-policy)#action A1
ACOS(config-policy-forward-policy-action)forward-to-internet http-sg snat
NAT
ACOS(config-policy-forward-policy-action)#source S1
ACOS(config-policy-forward-policy-source)#match-authorize-policy p1
ACOS(config-policy-forward-policy-source)#destination any action A1
```

With this configuration, the client requests with “**`Engineer`**” or “**`TP`**” in groups claim is matched to S1 source rule. This is followed by **`forward-policy`** which can handle the request depending on the configured action on which configured in source rule is set.

Forward-Policy JWT Authorization Example Configuration

This example configures a transparent proxy which only allows the requests with correct claims in JWT. The following is an representation of this topology:


Authorization Server
1. Auth request / Credentials
2.JWT token
Client
3. Access request with JWT
4. JWT decode, verify and authorize request
5. forward request
Internet

Before the client can access the internet, it needs to be authenticated with the authorization server and to get the JWTs. After this, it sends request with JWT for accessing the internet. Further, the Thunder verifies the token and only allows client with correct claims in the JWT to access the internet.

The following is an example of the JWT with the expected claims:

```
{
  "alg": "HS256",
  "typ": "JWT"
}
{
  "iss": "http://auth.a10-tplab.com",
  "sub": "z3gE4i_uQe0M",
  "aud": "john@a10-tplab.com",
  "exp": 1546214399,
  "iat": 1540361069,
  "name": "john",
  "email": "john@a10-tplab.com",
  "groups": [
    "Engineer",
    "TP"
  ]
}
```

For this scenario, the Thunder needs a transparent proxy virtual server with **`slb template policy`** binding on it. And the **`slb template policy`** configures **`forward-policy`** which contains the authorization policy as the source matching criteria in the source rule. And the authorization policy is binding with a JWT authorization template. The following is a representation of the same:

```
aam jwt-authorization test
  verification-secret password
  jwt-cache-enable
  exp-claim-required
!
aam authorization policy p1
  attribute-rule 1 or 2
  jwt-authorization test
  attribute 1 groups attr-type string match Engineer
  attribute 2 groups attr-type string match TP
!
slb template policy HTTP-policy
  forward-policy
```

```
    action A1
      forward-to-internet http-sg snat NAT
    action A2
      drop
    source S_ANY
      match-any
      priority 100
      destination any action A2
    source S1
      match-authorize-policy p1
      priority 105
      destination any action A1
!
slb virtual-server tp1 0.0.0.0
  port 80 http
    template policy HTTP-policy
    service-group http-sg
    aaa-policy aaa-policy-vip1-http
!
```

In this configuration, the user configures a JWT authorization template "**`test`**", which use **`HS256`** with secret "**`password`**" to verify the JWT signature. It also performs the following ***Optional*** tasks:

- JWT caching mechanism is enabled (optional)
- Token expiration claim 'exp' is required in the JWT (optional)

Further to this, the authorization policy "**`p1`**" is binding with this JWT authorization template, which authorizes the client requests with "**`Engineer`**" or "**`TP`**" groups claim in their JWT.

Also, the authorization policy "**`p1`**" is binding to **`forward-policy`** in **`slb template policy`** "**`HTTP-policy`**" as source matching criteria by **`match-authorize-policy`** option.

Further, when the client request matches to this source rule, it allows accessing the internet by action A1. Otherwise the request matches to **`S_ANY`** source rule which request is dropped by action A2. Then the **`slb policy template`** is binded to a transparent proxy virtual server.

In addition to this, the action is initiated for aaa-rule 1 of aaa-policy “**aa-policy-vip1-http**” with action “**allow**”. Then it initiates the bind process for this aaa-policy “**aaa-policy-vip1-http**” to target virtual server (which is the **vip1**) ports.

Authorizing Forward-Policy with JWT by aFleX

The JWT authorization support for **forward-policy** can be also configured by aFleX script. With aFleX solution, the user can quickly and dynamically deploy the **forward-policy** with JWT authorization.

The following topics are covered:

aFleX Forward-Policy JWT Authorization (Verifying Signature by Certificate)

In this example, with an assumption that the user is already having the **forward-policy** deployed in the virtual server, and the need is to verify the Thunder also to verify JWT, and to use the token claims to match the **forward-policy** source rules.

The already deployed configuration example is as the following:

```
slb template policy TEST
  forward-policy
    action A1
      forward-to-internet http_sg snat NAT
      log
    action sg
      forward-to-service-group http_8888_sg snat NAT
      log
    source S1
      priority 1024
      destination class-list proxy action proxy host priority 1
      destination class-list ce-ncu action A1 host priority 5
      destination class-list sg action sg host priority 10
      destination class-list drop action drop host priority 1023
      destination any action A1
    source S2
      priority 1023
      destination any action sg
!
slb template dynamic-service dns
  dns server 168.95.1.1
!
slb virtual-server vip 192.168.92.30
  port 1234 http
    template policy TEST
    aflex jwt-grp-authz
    source-nat auto
    service-group http_sg
    template dynamic-service dns
!
```

The aFleX script example is as the following:

```
when HTTP_REQUEST {
    set jwt [HTTP::header Bearer]
    if {($jwt equals "")} {
        log "Missing necessary information. Skip JWT authorization."
        return
    }

    set header [getfield $jwt . 1]
    set payload [getfield $jwt . 2]
    set signature [getfield $jwt . 3]

    set header_b64urldecoded [b64urldecode $header]
    set payload_b64urldecoded [b64urldecode $payload]
    set signature_b64urldecoded [b64urldecode $signature]

    # Verify token signature
    set jwt_msg $header.$payload
    if { not [rsha256 verify $jwt_msg cert test.pem $signature_
b64urldecoded] } {
        log  "Incorrect signing message. Reject this request."
        reject
        return
    }

    set lines [split $payload_b64urldecoded ","]
    set grp_idx 1
    set prasing_grp 0
    set flag_exp 0
    set flag_group 0
    set flag_iss 0
    set flag_aud 0

    foreach line $lines {
        if { $line contains "\"exp\":" } {
            # Get the value of exp string
            set exp [getfield $line ":" 2]
            set current [TIME::clock seconds]
            if { $current > $exp } {
                log "This message is expired!"
                reject
                return
            }
            set flag_exp 1
        } elseif { $line contains "\"groups\":" } {
            set grp$grp_idx [getfield $line "\[" 2]
            set prasing_grp 1
            set flag_group 1
            incr grp_idx
        } elseif { $prasing_grp } {
            set grp$grp_idx $line
            if { $line contains "\]" } {
                set prasing_grp 0
            }
            incr grp_idx
        } elseif { $line contains "\"iss\"" } {
            set iss [getfield $line "\":" 2]
            if { not [string equal $iss "\"https://auth.a10-tplab.com\""]
} {
                log "ISS mismatch. "
                reject
                return
            }
            set flag_iss 1
} elseif { $line contains "\"aud\"" } {
            set aud [getfield $line ":" 2]
            if { not [string equal $aud "\"john@a10-tplab.com\""] } {
                log "AUD mismatch. "
                reject
                return
            }
            set flag_aud 1
        }
    }

    if { not $flag_iss } {
        log  "Missing iss info. Message is incomplete. Reject this
request."
        reject
        return
    }
    if { not $flag_aud } {
        log  "Missing aud info. Message is incomplete. Reject this
request."
        reject
        return
    }
    if { not $flag_exp } {
        log "Missing expiration time. Message is incomplete. Reject this
request."
        reject
        return
    }
    if { not $flag_group } {
        log  "Missing group info. Message is incomplete. Reject this
request."
        reject
        return
    }

    # Set source rule according to group matching result
    if { ($grp1 contains "Engineer") or ($grp2 contains "TP") } {
            log "Groups are matched. Set EP source rule as \"S1\""
            POLICY::source_rule set S1
    } elseif { ($grp1 contains "bbb") } {
            log "Groups are matched. Set EP source rule as \"S2\""
            POLICY::source_rule set S2
    } else {
            log "No groups matched. Skip"
    }
}

when HTTP_REQUEST_SEND {
    # remove bearer from header
    if { [HTTP::header exists "Bearer"] } {
        HTTP::header remove "Bearer"
    }
}
```

In this aFleX script, the user can perform the following:

1. Getting the JWT from HTTP Bearer header from client request.
2. Then, getting the JWT header, payload, and signature parts.

   As it is **`base64url`** encoded, the script also gets decoded.
3. Verifying the token signature through **`RS256`** algorithm by certificate file **`test.pem`**.
4. Checking “**`iss`**” claim, it must be “https://auth.a10-tplab.com”.
5. Checking "**`aud`**" claim, it must be "john@a10-tplab.com".
6. Verifying the expiration time.
7. Getting group claim information.
8. Selecting the source rule based on the group authorization result.
9. Removing the Bearer header after **`forward-policy`** forward the request.

aFleX Forward-Policy JWT Authorization (Verifying Signature by Public Key)

In this example, with an assumption that the user is already having the **`forward-policy`** deployed in the virtual server, and the need is to verify the Thunder also to verify JWT, and to use the token claims to match the **`forward-policy`** source rules.

The already deployed configuration example is as the following:

```
slb template policy TEST
  forward-policy
    action A1
```

```
      forward-to-internet http_sg snat NAT
      log
    action sg
      forward-to-service-group http_8888_sg snat NAT
      log
    source S1
      priority 1024
      destination class-list proxy action proxy host priority 1
      destination class-list ce-ncu action A1 host priority 5
      destination class-list sg action sg host priority 10
      destination class-list drop action drop host priority 1023
      destination any action A1
    source S2
      priority 1023
      destination any action sg
!
slb template dynamic-service dns
  dns server 168.95.1.1
!
slb virtual-server vip 192.168.92.30
  port 1234 http
    template policy TEST
    aflex jwt-grp-authz
    source-nat auto
    service-group http_sg
    template dynamic-service dns
!
The aFleX script example is as the following:
when HTTP_REQUEST {
    set jwt [HTTP::header Bearer]
    if {($jwt equals "")} {
        log "Missing necessary information. Skip JWT authorization."
        return
    }

    set header [getfield $jwt . 1]
    set payload [getfield $jwt . 2]
    set signature [getfield $jwt . 3]

    set header_b64urldecoded [b64urldecode $header]
    set payload_b64urldecoded [b64urldecode $payload]
    set signature_b64urldecoded [b64urldecode $signature]

    # Verify token signature
    # e , n is public key.
    set jwt_msg $header.$payload

    set n_b64url "ALsmiIFNcbzhSedzFUW1dn2yiurXgnPZF17PeL_
zPDVkHQirealmWDIf6Rmt0lvz0EamdFrwHLtKqAdrgg9w7fq1Ws_
lK0zFefMXsVI7OA4TXYhQYADnOe0wkUEKCngj7dfejFZ-06iu6Hrza7U1Nb-
CSqM42zDCwvdvUY2K6Vxx"
    set e_b64url "AQAB"

    set n [b64urldecode $n_b64url]
    set e [b64urldecode $e_b64url]

    if { not [rsha256 verify $jwt_msg public-key $n $e $signature_
b64urldecoded] } {
        log  "Incorrect signing message. Reject this request."
        reject
        return
    }

    set lines [split $payload_b64urldecoded ","]
    set grp_idx 1
    set prasing_grp 0
    set flag_exp 0
    set flag_group 0
    set flag_iss 0
    set flag_aud 0

    foreach line $lines {
        if { $line contains "\"exp\":" } {
            # Get the value of exp string
            set exp [getfield $line ":" 2]
            set current [TIME::clock seconds]
            if { $current > $exp } {
                log "This message is expired!"
                reject
                return
            }
            set flag_exp 1
        } elseif { $line contains "\"groups\":" } {
            set grp$grp_idx [getfield $line "\[" 2]
            set prasing_grp 1
            set flag_group 1
            incr grp_idx
        } elseif { $prasing_grp } {
            set grp$grp_idx $line
            if { $line contains "\]" } {
                set prasing_grp 0
            }
            incr grp_idx
        } elseif { $line contains "\"iss\"" } {
            set iss [getfield $line "\":" 2]
            if { not [string equal $iss "\"https://auth.a10-tplab.com\""]
} {
                log "ISS mismatch. "
                reject
                return
            }
            set flag_iss 1
} elseif { $line contains "\"aud\"" } {
            set aud [getfield $line ":" 2]
            if { not [string equal $aud "\"john@a10-tplab.com\""] } {
                log "AUD mismatch. "
                reject
                return
            }
            set flag_aud 1
        }
    }

    if { not $flag_iss } {
        log  "Missing iss info. Message is incomplete. Reject this
request."
        reject
        return
    }
    if { not $flag_aud } {
        log  "Missing aud info. Message is incomplete. Reject this
request."
        reject
        return
    }
    if { not $flag_exp } {
        log "Missing expiration time. Message is incomplete. Reject this
request."
        reject
        return
    }
    if { not $flag_group } {
        log  "Missing group info. Message is incomplete. Reject this
request."
        reject
        return
    }

    # Set source rule according to group matching result
    if { ($grp1 contains "Engineer") or ($grp2 contains "TP") } {
            log "Groups are matched. Set EP source rule as \"S1\""
            POLICY::source_rule set S1
    } elseif { ($grp1 contains "bbb") } {
            log "Groups are matched. Set EP source rule as \"S2\""
            POLICY::source_rule set S2
    } else {
            log "No groups matched. Skip"
    }
}
```

In this aFleX script, the user can perform the following:

1. Getting the JWT from HTTP Bearer header from client request.
2. Then, getting the JWT header, payload, and signature parts.

   As it is `base64url` encoded, the script also gets decoded.
3. Verifying the token signature through `RS256` algorithm by certificate file `test.pem`.
4. Checking “`iss`” claim, it must be “https://auth.a10-tplab.com”.

5. Checking "**aud**" claim, it must be "john@a10-tplab.com".
6. Verifying the expiration time.
7. Getting group claim information.
8. Selecting the source rule based on the group authorization result.
9. Removing the Bearer header after **forward-policy** forward the request.

```
aFleX Forward-Policy JWT Authorization (verifying Signature by JWKS)
In this example, with an assumption that the user is already having the
forward-policy deployed in the virtual server, and the need is to verify
the Thunder also to verify JWT, and to use the token claims to match the
forward-policy source rules.
The already deployed configuration example is as the following:
slb template policy TEST
  forward-policy
    action A1
      forward-to-internet http_sg snat NAT
      log
    action sg
      forward-to-service-group http_8888_sg snat NAT
      log
    source S1
      priority 1024
      destination class-list proxy action proxy host priority 1
      destination class-list ce-ncu action A1 host priority 5
      destination class-list sg action sg host priority 10
      destination class-list drop action drop host priority 1023
      destination any action A1
    source S2
      priority 1023
      destination any action sg
!
slb template dynamic-service dns
  dns server 168.95.1.1
!
slb virtual-server vip 192.168.92.30
  port 1234 http
    template policy TEST
    aflex jwt-grp-authz
    source-nat auto
    service-group http_sg
    template dynamic-service dns
!
```

The aFleX script example is as the following:

```
when HTTP_REQUEST {
    set jwt [HTTP::header Bearer]
    if {($jwt equals "")} {
        log "Missing necessary information. Skip JWT authorization."
        return
    }
    set header [getfield $jwt . 1]
    set payload [getfield $jwt . 2]
    set signature [getfield $jwt . 3]
    set header_b64urldecoded [b64urldecode $header]
    set payload_b64urldecoded [b64urldecode $payload]
    set signature_b64urldecoded [b64urldecode $signature]
    # Verify token signature
    set jwt_msg $header.$payload
    if { not [rsha256 verify $jwt_msg jwk test.jwk $signature_
b64urldecoded] } {
        log  "Incorrect signing message. Reject this request."
        reject
        return
    }
    set lines [split $payload_b64urldecoded ","]
    set grp_idx 1
    set prasing_grp 0
    set flag_exp 0
    set flag_group 0
    set flag_iss 0
    set flag_aud 0
    foreach line $lines {
        if { $line contains "\"exp\":" } {
            # Get the value of exp string
            set exp [getfield $line ":" 2]
            set current [TIME::clock seconds]
            if { $current > $exp } {
                log "This message is expired!"
                reject
                return
            }
            set flag_exp 1
        } elseif { $line contains "\"groups\":" } {
            set grp$grp_idx [getfield $line "\[" 2]
            set prasing_grp 1
            set flag_group 1
            incr grp_idx
        } elseif { $prasing_grp } {
            set grp$grp_idx $line
            if { $line contains "\]" } {
                set prasing_grp 0
            }
            incr grp_idx
        } elseif { $line contains "\"iss\"" } {
            set iss [getfield $line "\":" 2]
            if { not [string equal $iss "\"https://auth.a10-tplab.com\""]
} {
                log "ISS mismatch. "
                reject
                return
            }
            set flag_iss 1
} elseif { $line contains "\"aud\"" } {
            set aud [getfield $line ":" 2]
            if { not [string equal $aud "\"john@a10-tplab.com\""] } {
                log "AUD mismatch. "
                reject
                return
            }
            set flag_aud 1
        }
    }
    if { not $flag_iss } {
        log  "Missing iss info. Message is incomplete. Reject this
request."
        reject
        return
    }
    if { not $flag_aud } {
        log  "Missing aud info. Message is incomplete. Reject this
request."
        reject
        return
    }
    if { not $flag_exp } {
        log "Missing expiration time. Message is incomplete. Reject this
request."
        reject
        return
    }
    if { not $flag_group } {
        log  "Missing group info. Message is incomplete. Reject this
request."
        reject
        return
    }
    # Set source rule according to group matching result
    if { ($grp1 contains "Engineer") or ($grp2 contains "TP") } {
            log "Groups are matched. Set EP source rule as \"S1\""
            POLICY::source_rule set S1
    } elseif { ($grp1 contains "bbb") } {
            log "Groups are matched. Set EP source rule as \"S2\""
            POLICY::source_rule set S2
    } else {
            log "No groups matched. Skip"
    }
}
when HTTP_REQUEST_SEND {
    # remove bearer from header
    if { [HTTP::header exists "Bearer"] } {
        HTTP::header remove "Bearer"
    }
}
```

In this aFleX script, the user can perform the following:

1. Getting the JWT from HTTP Bearer header from client request.
2. Then, getting the JWT header, payload, and signature parts.

As it is **base64url** encoded, the script also gets decoded.

3. Verifying the token signature through **RS256** algorithm by certificate file **test.pem**.
4. Checking "**iss**" claim, it must be "https://auth.a10-tplab.com".
5. Checking "**aud**" claim, it must be "john@a10-tplab.com".
6. Verifying the expiration time.
7. Getting group claim information.
8. Selecting the source rule based on the group authorization result.
9. Removing the Bearer header after **forward-policy** forward the request.

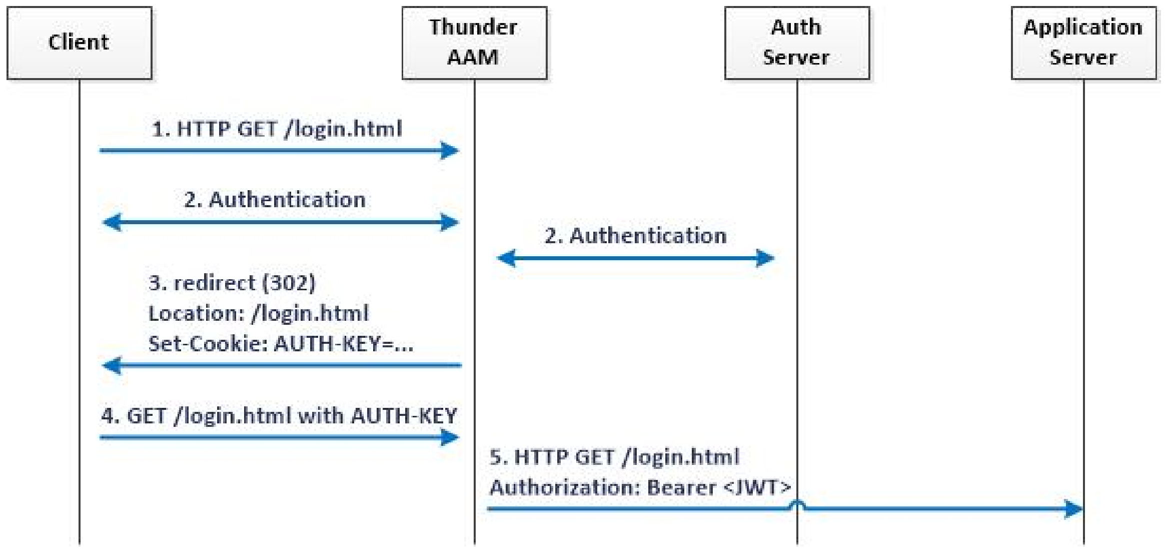
JWT Relay

The following topics are covered:

Overview

JWT is JSON-based, compact token, which provides easy and secure way to exchange information between parties. For the deployment of AAM, Thunder can do the client request authentication, and authorization for backend application servers, but sometime application server may want to have some information about the client. As a need, this feature provides the ability for AAM to issue JWTs for authenticated client, and then application server can have client information in the request.

The Traffic Walkthrough



The following is an example of the traffic flow:

1. First, the client tries to access the application resource through Thunder AAM virtual server.
2. The AAM then reciprocates to the client to perform the authentication first.
3. Once the client is successfully authenticated with the AAM, the later provides the **AUTH-KEY** cookie to the client, then the client can use this cookie to access the resource again.
4. The client sends the access request with the **AUTH-KEY** cookie.
5. The AAM forwards the client request for JWT inserted to HTTP Authorization header with Bearer schema.

Configuring JWT Template

The issuance and relay behavior for the JWT is configured in the JWT template.

The following is a series of examples to configure the JWT template for the JWT issuance.

1. Creating a JWT template called “test”, and the action for this template is “relay”:

```
ACOS(config)#aam authentication jwt test
ACOS(config-jwt:test)#action relay
```

2. Configuring the issuer claim (which is the JWT '**iss**' claim) for JWT is “aam.a10-

tplab.com”:

```
ACOS(config-jwt:test)#issuer http://aam.a10-tplab.com
```

3. Configuring the secret to sign the JWT signature:

```
ACOS(config-jwt:test)#signature-secret password
```

4. Configuring JWT **`life-time`** (which is JWT '**`exp`**' claim):

```
ACOS(config-jwt:test)#token-lifetime 600
```

5. The JWT, which is created by this example looks like the following:

```
{
  "iss" : "http://aam.a10-tplab.com",
  "sub" : "<aam username>",
  "exp" :  1475874757,
  "iat" :  1475874457,
  "jti" : "<unique-id>"
}
```

The “**`iss`**” and “**`exp`**” claim is generate according the issuer and token-lifetime configuration.

The “**`iat`**” claim is the time of token generation. “**`sub`**” is the username client used to authenticate with AAM. The “**`jti`**” is a unit identifier among JWTs.

Configuring Attribute to Claim Mapping

For JWT generation, application servers may also need some client information in the JWT claims to provide service. Thus, the Thunder also provides a mechanism to generate claims which comes from authentication attributes. The authorization attribute to JWT claim mapping is configured in the **`aam authorization policy`**.

The following is a configuration example for the AAM authorization attribute map to JWT claim:

```
aam authorization policy p2
  attribute 1 User-Name any
  attribute 2 Class any
  attribute 3 Service-Type any
  attribute 4 Valid any
  jwt-claim-map 1 claim nid
  jwt-claim-map 2 claim grp type string value ATEN
  jwt-claim-map 3 claim svc type number value 3
  jwt-claim-map 4 claim flg type boolean value true
!
```

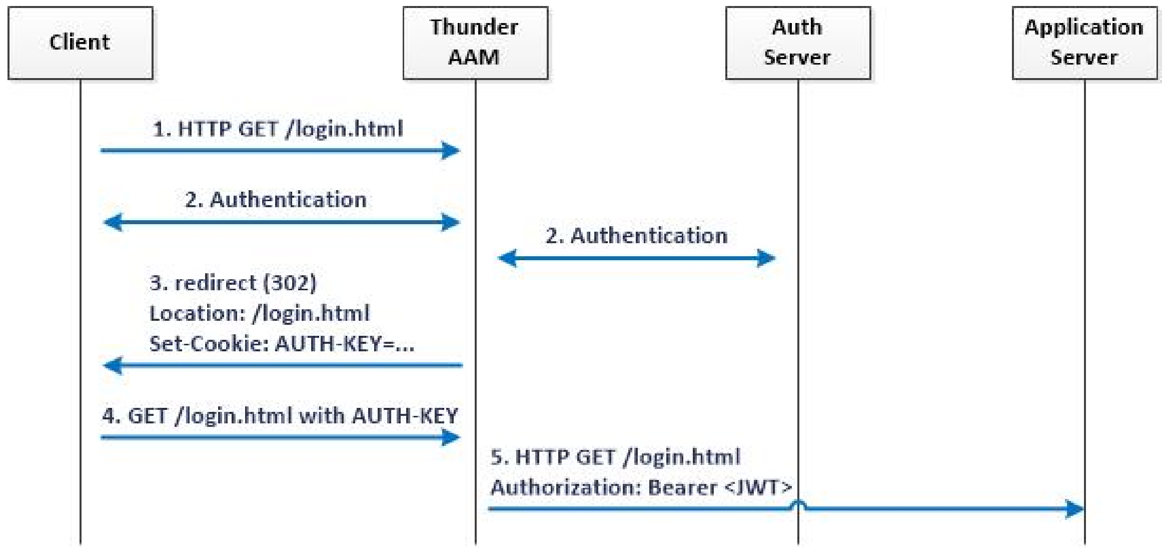
The following example is the result of JWT, which reads with the following values:

```
{
  "iss" : "http://aam.a10-tpalb.com",
  "sub" : "<aam username>",
  "exp" :  1475874757,
  "iat" :  1475874457,
  "jti" : "<unique-id>",
  "nid" : "John",
  "grp" : "ATEN",
  "svc" : 3,
  "flg" : true
}
```

In this example, "**nid**" claims is from the "**User-Name**" attribute from the authorization server (which is "**John**"). Adding to this, add "**grp**" claim as "**ATEN**" if "**Class**" attribute present in authorization attribute. Also, these are same as "**svc**" and "**flag**" claim.

JWT Relay Example (with RADIUS Authentication Server)

In this example, AAM is working with a RADIUS server, and all the client needs to authenticate with AAM before accessing the backend application server. After it is authenticated, client requests which is forwarded by AAM contains the JWT in HTTP Authorization header, for which claims are generated according to the JWT template configuration or conversion from RADIUS attributes.



The following is an configuration example of JWT relay traffic walkthrough:

```
aam authentication jwt test
  issuer http://aam.a10-tplab.com/
  action relay
  token-lifetime 600
  signature-secret password
!
aam authentication server radius ra1
  host 192.168.90.160
  secret password
!
aam authentication template a1
  logon http-basic
  jwt test
  server ra1
!
aam authorization policy p1
  attribute 1 User-Name any
  attribute 2 Class any
  attribute 3 Service-Type any
  attribute 4 Valid any
  jwt-claim-map 1 claim nid
  jwt-claim-map 2 claim grp type string value ATEN
  jwt-claim-map 3 claim svc type number value 3
  jwt-claim-map 4 claim flg type boolean value true
```

```
!
aam aaa-policy http-vip1-aaa-policy
  aaa-rule 1
    action allow
    authentication-template a1
    authorize-policy p1
!
slb virtual-server vip1 192.168.91.56
  port 80 http
    source-nat auto
    service-group http-sg
    aaa-policy http-vip1-aaa-policy
!
```

In this example, the user is able to create the following:

- A relay JWT template which creates JWT with "**`http://aam.a10-tplab.com`**" as "**`iss`**" claim.
- The JWT expiration period is 600 seconds and the secret to sign the JWT signature is "**`password`**".
- An authorization policy "**`p1`**" which converts the RADIUS attributes to JWT claims.
- It binds the JWT template and Authorization template to **`aaa-policy`** rule, then it binds this **`aaa-policy`** to virtual server.

The result of this JWT is as the following:

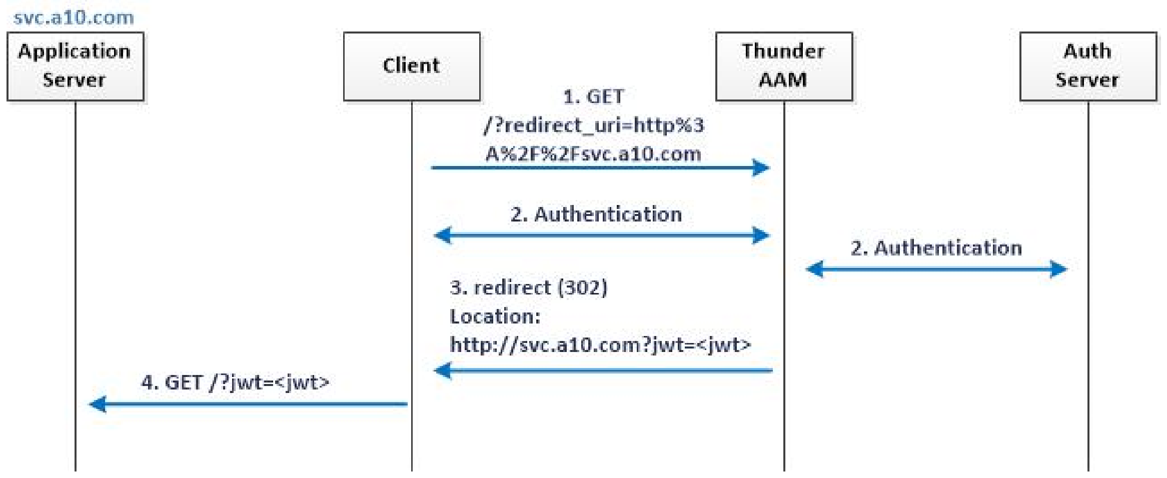
```
{
  "iss" : "http://aam.a10-tpalb.com",
  "sub" : "<aam username>",
  "exp" :  1475874757,
  "iat" :  1475874457,
  "jti" : "<unique-id>",
  "nid" : "John",
  "grp" : "ATEN",
  "svc" : 3,
  "flg" : true
}
```

JWT Redirect Example

Besides JWT relay, the AAM also works as an authentication server and issues JWTs for authentication approved clients. This is performed by redirecting the client to the target application server with the JWT in the URI parameter.

In order to make this work, the URL for the target application server needs to be provided in the first request by **`redirect_uri`** URI parameter.

The traffic walkthrough is shown in the following representation.



1. The client sends the first request to AAM, with **`redirect_uri`** parameter in request URI. (Normally, this request is client redirected by the application server to perform the authentication.)
2. The client authenticates with the AAM, and the later creates the JWT, if the authentication is successful.
3. The AAM redirects client back to the application server according to the **`redirect_uri`** URI parameter of the first request as in the step 1.
4. The JWT is also appended to the redirect **`URL URI`** parameter.
5. Then, the client accesses the application with JWT in the request URI.

The following is an example of the configuration with details:

```
aam authentication jwt test
  issuer http://aam.a10-tplab.com/
  action relay
  token-lifetime 600
  signature-secret password
!
aam authentication server ldap ldap_server
  host 192.168.90.136
  base cn=Users, dc=a10-tplab, dc=com
  admin-dn cn=Administrator, cn=Users, DC=a10-tplab, DC=com
  admin-secret encrypted
rdb9WoR9H3y25a5C93YqpDwQjLjV2wDnPBCMuNXbAOc8EIy41dsA5zwQjLjV2wDn
!
aam authentication template a1
  logon http-basic
  jwt test
  server ldap_server
!
aam authorization policy p1
  attribute 1 cn any
  attribute 2 memberOf any
  jwt-claim-map 1 claim nid
  jwt-claim-map 2 claim grp
!

aam aaa-policy vip1-aaa-policy
  aaa-rule 1
    action allow
    authentication-template a1
    authorize-policy p1
!
slb virtual-server vip1 192.168.91.56
  port 80 http
    source-nat auto
    service-group http-sg
    aaa-policy vip1-aaa-policy
!
```

In this example, the user is able to create the following:

- A redirect JWT template which creates JWT with "**`http://aam.a10-tplab.com`**" as "**`iss`**" claim.
- The JWT expiration period is 600 seconds and the secret to sign the JWT signature is "**`password`**".

- An authorization policy “**p1**” which converts the LDAP attributes to JWT claims.
- Binds the JWT template and authorization template to **aaa-policy** rule, then binds this **aaa-policy** to virtual server.

The following is the result of this JWT:

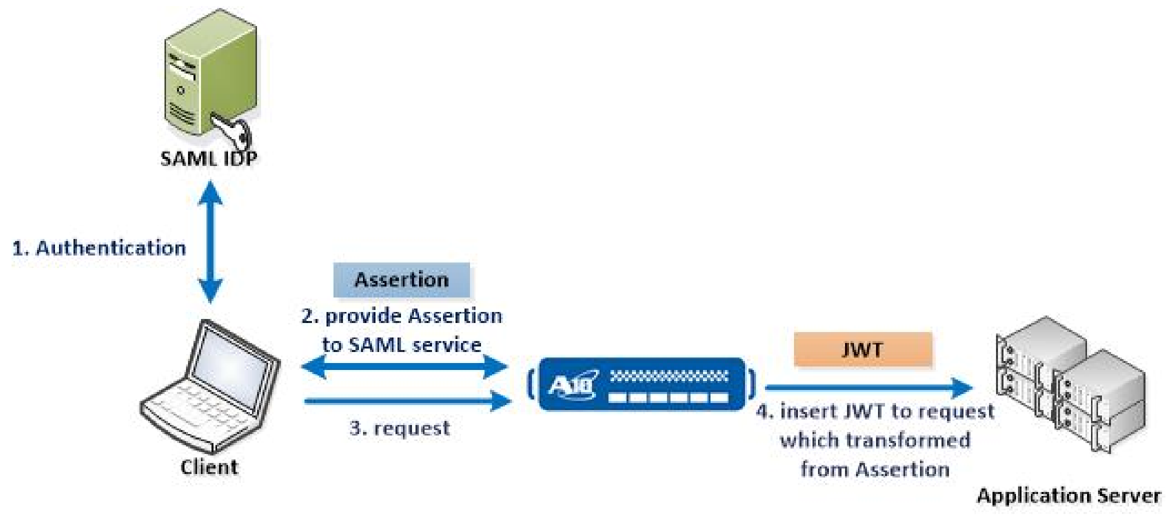
```
{
  "iss" : "http://aam.a10-tpalb.com",
  "sub" : "<aam username>",
  "exp" :  1475874757,
  "iat" :  1475874457,
  "jti" : "<unique-id>",
  "nid" : "John",
  "grp" : "ATEN"
}
```

SAML Assertion to JWT Transformation

The following topics are covered:

Overview

This is for the cases when SAML authentication is already deployed and the application servers need some client information which contains in SAML Assertion. In this case, as the SAML protocol is complicated, it is used. It is a huge effort to make all kinds of application server to understand the SAML Assertion and works with SAML IDP. Thus, the ACOS provides a feature which uses aFleX script to transform the client SAML Assertion to JWT, and the JWT is inserted in HTTP request header, when the forward client's request is requested.


SAML IDP
1. Authentication
Assertion
2. provide Assertion to SAML service
3. request
Client
JWT
4. insert JWT to request which transformed from Assertion
Application Server

The following is the synopsis of the activities:

1. Client is authenticated with the SAML IDP.
2. It provides the SAML assertion and authentication to the Thunder.

NOTE: This is completely based on the mode and type of the configuration. For more details on Security Assertion Markup Language, see Security Assertion Markup Language (SAML).

3. Then the client sends a request to access the resource.
4. The Thunder then generates the JWT, inserts into the client request, and then forwards the client request to the backend application server.
5. Thus, the application server can have necessary information about this client to perform the authorization or provide services.

Assertion to JWT Example

In this example, the Thunder configured as a SAML service provider and it authenticates the client through SAML Assertion. Then an aFleX script is binded to the virtual server port, which then converts the SAML Assertion tothe JWT. Then, the forwarded client request contains the generated JWT.

The following is an example for the SAML Assertion type:

```
<saml:Assertion ID="D-jNU-8HwMcergrNhH9Wv_ggFs9" IssueInstant="2014-07-
30T11:59:16.240Z" Version="2.0"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
. . .
    <saml:Subject>
        <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified">ben
        </saml:NameID>
. . .
    <Conditions NotBefore="2014-07-31T11:06:24.064Z" NotOnOrAfter="2014-
07-31T12:06:24.064Z">
. . .
</saml:Assertion>
The following is a configuration example of the SAML authentication
deployment, with Assertion to the JWT aFleX:
aam authentication saml service-provider sp56
  assertion-consuming-service index 0 location /SAML2/POST binding post
  certificate sp.p12
  entity-id https://192.168.91.56/shibboleth
  service-url http://192.168.91.56/Shibboleth.sso
!
aam authentication saml identity-provider idp
  metadata idp56
!
aam authentication template a1
  type saml
  saml-sp sp56
  saml-idp idp
!
aam authorization policy p1
  attribute-rule 1 or 2 or 3 or 4
  attribute 1 name attr-type string match ben
  attribute 2 Memberof attr-type string match Engineering
  attribute 3 emailaddress attr-type string sub-string a10lab.com
  attribute 4 upn attr-type string sub-string a10lab.com
!
aam aaa-policy aaa-policy-vip1
  aaa-rule 1
    action allow
    authentication-template a1
    authorize-policy p1
!
slb virtual-server vip1 192.168.91.56
  port 80 http
    aflex saml2jwt
    source-nat auto
    service-group http-sg
    aaa-policy aaa-policy-vip1
!
```

The following is an example for the aFleX script:

```
when AAM_AUTHENTICATION_INIT {
  AAM::attribute_collection 1
}

when HTTP_REQUEST_SEND {
  set jwt_msg [AAM::session get jwt]

  if {$jwt_msg equals ""} {
    set name [AAM::attribute get name collection_id 1]

    # hdr
    set jwt_hdr [b64urlencode "{\r\n \"alg\":\"HS256\",\r\n
\"typ\":\"JWT\"\r\n}"]

    # payload (from SAML attributes)
    set raw_payload "{\r\n \"name\": \"$name\""
    set Memberof [AAM::attribute get Memberof]
    set raw_payload "$raw_payload,\r\n \"group\": \"$Memberof\""

    # payload (form SAML Assertion elements)
    set nameId [AAM::saml get Assertion.Subject.NameID]
    set raw_payload "$raw_payload,\r\n \"id\": \"$nameId\""
    set nbf_str [AAM::saml get NotBefore@Assertion.Conditions]
    set nbf [utc_to_numeric_date $nbf_str]

    set raw_payload "$raw_payload,\r\n \"nbf\": \"$nbf\""
    set raw_payload "$raw_payload\r\n}"
    set jwt_payload [b64urlencode $raw_payload]
    # signature

    # HS256
    #set secret "password"
    #set jwt_signature [b64urlencode [hsha256 "$jwt_hdr.$jwt_payload" 
$secret]]

    # RS256
    #set jwt_signature [b64urlencode [rsha256 "$jwt_hdr.$jwt_payload" 
aamtestsrv]]

    # ES256
    set jwt_signature [b64urlencode [esha256 "$jwt_hdr.$jwt_payload" 
secp256k1-key.pem]]

    # jwt
    set jwt_msg "$jwt_hdr.$jwt_payload.$jwt_signature"
    AAM::session set jwt $jwt_msg
  }

  if { not ($jwt_msg equals "") } {
    HTTP::header insert Authorization "Bearer $jwt_msg"
  }
}
```

In the aFleX script, it tries to get the JWT claims from the attribute collection and the SAML Assertion. Then, it constructs the JWT header and payload. Along with this, the JWT signature is also created according to the JWT header and payload with **`HS256`** algorithm.

The example for **`RS256`** and **`ES256`** signature generation is also provided in the script. The aFleX command to get the SAML Assertion element and the element attribute is **`AAM::saml`**.

- The command "`AAM::saml get Assertion.Subject.NameID`" returns **`NameID`** element value which is in the Subject element and the Subject element is in the Assertion element.
- The command "`AAM::saml get NotBefore@Assertion.Conditions`" returns **`NotBefore`** attribute value of Conditions element, which in the Assertion element.

The JWT which is created by the aFleX script, looks like the following:

```
{
 "alg": "HS256",
 "typ": "JWT"
}
{
 "name": "ben",
 "group": "Engineering",
 "id": "ben",
 "nbf": "1401954713"
}
```

aFleX Commands

The following topics are covered:

b64urlencode

Description Encode the specified data with **`base64url`** and return.

Syntax

```
b64urlencode <data-to-encode>
```

Example

```
set buf [b64urlencode "data to encode"]
```

Valid Event

```
All
```

b64urldecode

Description Decode the **`base64url`** data and return.

Syntax

```
b64urldecode <base64url-encoded-data>
```

Feedback

Example

```
Set buf [b64urldecode $base64url_data]
```

Valid Event

```
All
```

utc_to_numeric_date

Description Transform time format from UTC to JASON **NumericDate.**

Syntax

```
utc_to_numeric_date <UTC-time>
```

Example

```
set time [utc_to_numeric_date "2014-07-31T11:06:24.064Z"]
```

Valid Event

```
All
```

hsha256

Description Signing `HS256` signature.

Syntax

```
rsha256 <data> <secret>
```

This returns the signature of data signed by the secret.

Example

```
when HTTP_REQUEST {
    set msg "data to sign"
    set secret "password"
    set signature [hsha256 "$msg" $secret]
}
```

Valid Event

```
All
UTCtoNumericData
b
```

rsha256

Description Signing or verifying RS256 signature.

Syntax

```
rsha256 sign <data> <private-key-file>
```

This returns the signature of data signed by specified private-key.

```
rsha256 <data> <private-key-file>
```

This returns the signature of data signed by specified private-key.

```
rsha256 verify <data> cert <certificate-file> <signature-to-
verify>
```

This verifies the signature-to-verify with data by specified certificate-file.

```
rsha256 verify <data> public-key <modulus> <exponent>
<signature-to-verify>
```

This verifies the signature-to-verify with data by specified public key modulus and exponent.

```
rsha256 verify <data> jwk <jwks-file> <signature-to-verify>
```

This verifies the signature-to-verify with data by specified JWKS file.

Example

```
when HTTP_REQUEST {
    set msg "data to sign"
    set signature [rsha256 sign $msg test.key]
    if { not [rsha256 verify $msg cert test.pem $signature] }
{
        log "signature verify failed"
    }

    set n_b64url "ALsmiIFNcbzhSedzFUW1dn2yiurXgnPZF17PeL_
zPDVkHQirealmWDIf6Rmt0lvz0EamdFrwHLtKqAdrgg9w7fq1Ws_
lK0zFefMXsVI7OA4TXYhQYADnOe0wkUEKCngj7dfejFZ-06iu6Hrza7U1Nb-
CSqM42zDCwvdvUY2K6Vxx"
    set e_b64url "AQAB"
    set n [b64urldecode $n_b64url]
    set e [b64urldecode $e_b64url]

    if { not [rsha256 verify $msg public-key $n $e $signature]
} {
        log "signature verify failed"
    }
    if { not [rsha256 verify $msg jwk test.jwk $signature] } {
        log "signature verify failed"
    }
}
```

Valid Event

```
All
```

esha256

Description Signing **ES256** (ECDSA using **P-256** and **SHA-256**) signature

Syntax

```
esha256 <data> <es256-key>
```

This returns the signature of data signed by specified ES256 key.

Example

```
when HTTP_REQUEST {
    set msg "data to sign"
    set signature [hsha256 "$msg" secp256k-key.pem]
}
```

Valid Event

```
All
```

Authentication Files

This topic provides information about the files that the user might need to create an authentication portal or a Security Assertion Markup Language (SAML) identity provider using the GUI.

NOTE: For more information about SAML identity providers, see Identity and Service Providers.

The following topics are covered:

Exporting the Default Authentication Portal

To export the default authentication portal:

1. Click **AAM** > **Policies and Templates**. Navigate to **Authentication Files > Auth Portals.**
2. Select the **default-portal** check box.
3. Click **Export** to export it.

Creating an Authentication Portal

The following topics are covered:

The user can create an authentication portal, by referring the steps as the following:

NOTE: Steps 4 and 5 apply to creating authentication portal files or SAML identity provider files.

1. Click **AAM** > **Policies and Templates**. Navigate to **Authentication Files > Auth Portals.**
2. Click **Import.**
3. Enter a name.
4. To use the management port, select the **Use Mgmt Port** check box.
5. To overwrite the existing file, select the **Overwrite Existing File** check box.
6. Select a protocol:
   - TFTP
   - FTP

- SCP
- SFTP

7. Complete the relevant fields that appear.

NOTE: The fields that appear depend on the protocol that the user selects.

8. Click **Import**.

Using a Default Portal Image

To use a default portal image:

1. Click **AAM** > **Policies and Templates**. Navigate to **Authentication Files > Auth Portal Images.**
2. On the **Auth Portal Image** tab, select one of the following files:
   - a10bg.gif
   - def_bg.jpg
   - def_logo.png

Creating an Authentication Portal Image

To create authentication portal images:

1. Click **AAM** > **Policies and Templates**. Navigate to **Authentication Files > Auth Portal Images.**
2. Click **Import**.
3. Enter a name.
4. To use the management port, select the **Use Mgmt Port** check box.
5. To overwrite the existing file, select the **Overwrite Existing File** check box.
6. Select a protocol:
   - TFTP
   - FTP
   - SCP
   - SFTP

7. Complete the relevant fields that appear.

NOTE: The fields that appear depend on the protocol that the user selects.

8. Click **Import.**

Creating SAML Identity Provider Files

To create SAML identity provider files:

1. Click **AAM** > **Policies and Templates**. Navigate to **Authentication Files > SAML Identity Providers.**
2. On the SAML Identity Provider tab, click **Import**.
3. Select either “**Local**” or “**Remote**”:

   For **Local**:

   a. Enter a name.
   b. Enter the definition.
   c. Click **Create**.

   For **Remote**:

   a. Enter a name.
   b. To use the management port, select the **Use Mgmt Port** check box.
   c. To overwrite the existing file, select the **Overwrite Existing File** check box.
   d. Select a protocol:
      - TFTP
      - FTP
      - SCP
      - SFTP
   e. Complete the relevant fields that appear.

   NOTE: The fields that appear depend on the protocol that the user

selects.

f. Click **Create**.

Deleting Authentication Portal Files

1. Click **AAM** > **Policies and Templates**. Navigate to **Authentication Files > Auth Portals.**
2. Select the checkbox for the file that the user wants to delete and click **Delete**.

Global Statistics

To display global AAM statistics:

1. Click **AAM** > **Policies and Templates**.
2. Review the statistics on the **Global Stats** page.

Authentication Logs

This chapter describes how to configure authentication logs, store audit logs, and monitor user activity.

The following topics are covered:

Overview

Authentication logs can be used to monitor user activities. This can be defined as the activity of a user or an admin, to track the activities of another user or a particular user. After the authentication logging is enabled, the user or the admin can examine the log files and determine when a particular user logged in, logged out, and other related information. This can help the user or the admin to troubleshoot the issues with the authentication servers. An example can be, if a legitimate user is not being authenticated properly, then the admin or the user can review the logs to determine the cause of the problem.

Authentication audit logs are generated in the following situations:

- Each time an authentication request and response are received
- Each time an authentication session is created or terminated
- Each time an authentication module is initiated

The user can specify whether to enable authentication logs and the facility to use to send these messages when logs are enabled. More granular control is available with the ability to enable or disable the logs at the authentication template level. When the user disables logs for an authentication template, no logs are collected for the authentication requests that come to the VIP to which the authentication template is bound. For more information, see Log Configuration Levels.

NOTE: Logs are disabled by default, and the default facility is `local0`.

Log Collecting Events

The following topics are covered:

Logs are collected during the following events:

Authentication Request

When an authentication request comes in; the username, authentication type, authentication server used, realm or domain name, request id and time of the request are logged.

Authentication Response

When a response comes back from the auth-server; the corresponding request id, username, realm or domain name, authentication type, authentication server, authentication result, authorization result, failure reason if any, authentication session id, VIP for which the request came in and the time of response are logged.

Authentication Session Creation

Upon creation of a new authentication session; the session id, username and time of creation are logged.

Authentication Session Termination

Upon termination of an authentication session; the session id, username, termination reason and time of termination are logged.

Authentication Module Restart

The success or failure of an authentication module restart and time at which it is restarted is logged.

NOTE: For authentication module restart, the local `auth_debug` logs are used. This loginformation is not sent to the external syslog server.

Log Configuration Levels

The user can configure and enable authentication logs at one of the following levels:

- Partition
- Authentication template

Table 2shows the available log configuration options.

Table 2 : Log Configuration Options

Partition-level configuration (authentication-log)	**Template-level configuration (log command in the AAM authentication template)**	**Are logs generated when the VIP that is bound to this template is accessed?**	**Are logs generated when the VIPs that are bound to other templates are accessed?**
`Enable`	`Use-partition-level-config` (default)	Yes	Yes
`Enable`	`Enable`	Yes	Yes
`Enable`	`Disable`	No	Yes
`Disable`	`Use-partition-level-config` (default)	No	No
`Disable`	`Enable`	Yes	No
`Disable`	`Disable`	No	No

NOTE: The log configuration at the template level override the logs that are configured at the partition level.

Log Format for the Authentication Logs

The following topics are covered:

Log Formats

Users can select the log format for the authentication logs. ACOS provides the following log formats for the authentication logs:

The following topics are covered:

Syslog

The logs generated for the authentication (AAM) events are sent to the configured log/syslog server in the syslog format.

Common Event Format (CEF)

The logs generated for the authentication (AAM) events are sent to the configured log/syslog server in the CEF format

NOTE: When Common Event Format (CEF) is selected as the Log Format, the "acos-event" setting must be configured through the Command Line Interface (CLI).

To configure "acos-event" settings, refer to the Monitoring Tools topic in the System Configuration and Administration Guide.

Log Examples

The following topics are covered:

Sample Default Syslog Format for the Authentication Request Event

```
TCP 192.168.91.112:3579 > 192.168.91.110:80  AUTH  REQUEST    username=abc
 request-id=20  auth-server=172.16.1.190  server-type=RADIUS   auth-
type=http-authenticate
```

Sample Common Event Format (CEF) for the Authentication Request Event

```
<CEF prefix>AAM <UUID>|Authentication Request|5|proto=TCP
src=192.168.90.151 spt=3579 dst=192.168.91.110 dpt=80 suser=l2x cn1=20
cs1=RADIUS cs2=172.16.1.190 cs3=http-authenticate cnlLbael=session-id
cs1Label=server-type cs2Label=auth-server cs3Label=auth-type
```

Log Facility for the Log Messages

The log messages that are sent to the Syslog server are of different types. To differentiate each log type messages and store them in distinct log files, the log messages contain the logging facility with the actual message and the IP address of the client for which the log was generated. The user can set the log facility (local0 to local7) for the authentication audit logs by clicking the Facility drop-down list. By default, the log facility is set to local0.

The generated logs are sent to the syslog server configured on the **System** > **Settings** > **Logging** page, Logging Hosts section.

The following table describes the names and values for the authentication logs:

Name	Value	Description
act	aaa-rule action	Action taken by the Authentication policy
cn1	Custom	Retry limit: The number of times the user attempted to

Name	Value	Description
	Number Label 1	authenticate to the requested web application server.
cn2	Custom Number Label 2	Lockup-seconds: The number of seconds the user's account is locked.
cs1	Custom String Label 1	Service Provider name (SAML authentication), Authentication Template Name (MSSQL Request), AAA Policy/AAA Rule index (AAA Rule Match)
cs2	Custom String Label 2	Identity Provider name (IdP) (SAML authentication).
cs3	Custom String Label 3	Authentication Relay Type (SAML Authentication).
cs4	Custom String Label 4	Resource Access (SAML Authentication), Realm (Authentication Response).
cs5	Custom String Label 5	Resource Access (Authentication Response).
dpt	Destination Port	The port number of the destination server.
dst	Destination Address	The IP address of the destination server.
externalId	Request ID	The event ID associated with the generated event.
msg	Message	Event details of the log that is generated.
proto	Protocol	The protocol used during the communication between the client and the server.
src	Source Address	The IP address of the client.
spt	Source Port	The port number of the client.

Name	Value	Description
suser	Source Username	The source username.

Configuring the Authentication Logs

The following topics are covered:

Using the GUI

To configure authentication logs:

1. Go to the **AAM** > **Log Settings**.page.
2. In the Update Log Settings page, do the following:
   a. Select the **Enable** check box.
   b. Select a facility.
   c. Select the format for the authentication logs.
   d. Click **Update**.

Using the CLI

Authentication logging is disabled by default. To enable it:

```
ACOS(config)# aam authentication log enable
```

To configure the logging facility on the syslog server to send authentication logs:

```
ACOS(config)# aam authentication log facility local0
```

To set the log format for the authentication logs:

```
ACOS(config)# aam authentication log format syslog
ACOS(config)# aam authentication log format cef
```

Using Authentication Templates

To configure logs by using the authentication template, first create a template, then enable logging for the template:

```
ACOS(config)# aam authentication template log-template
ACOS(config-auth template:log-template)# log enable
```

AAM Logs

AAM logs require a configured remote syslog server and it must be reachable from the data network.

Limitations

The following are the AAM log limitations:

- Logs cannot be sent through the management network to the remote syslog server.
- Local buffered logging is not supported.
- If multiple remote syslog servers are configured, log messages are sent using the Round Robin method to the set of syslog servers. The log messages are not duplicated.

Debugging Logs

AAM logs can be viewed using the “debug” command. Using the debug command, the user can filter AAM logs by severity level, by a client username, or by a virtual-server name. If both the username and virtual-server name filters are set, then only debug logs that match both criteria are shown.

Severity Level

Any critical error logs that are related to AAM configurations are sent to the syslog and require immediate attention. Detailed information is not included in syslog messages.

AAM debug logs are categorized into two severity levels. Level 1 logs are brief logs, and most of the current AAM authentication logs are placed in this category. Level 2 logs are detailed logs.

NOTE: These commands are mostly used by A10 Networks Support Engineers for troubleshooting. It is not recommended to use the "`debug`" command otherwise.

OAUTH 2.0 and OpenID Connect

ACOS supports OAuth 2.0 and OpenID Connect (OIDC) authentication to authorize and authenticate users with OAuth servers from Facebook, Google, Microsoft, and so on.

The following topics are covered:

OAuth 2.0

OAuth 2.0 is an open-standard authorization protocol that can provide client applications with secure access to server resources without revealing the user's credentials or identity. It uses JSON, which is easy to read and write. OAuth 2.0 uses API calls extensively, so that many modern mobile apps, web apps, gaming consoles, and Internet of Things (IoT) devices find OAuth as a better experience for users. It also uses JSON, which is easy to read and write.

Some of the benefits of OAuth 2.0:

- Enhances the single sign-on (SSO) technology.
- Easy to implement and provides stronger authentication.
- Relies on SSL to save user access tokens and to keep the data safe.

OpenID Connect 1.0

OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It allows clients to verify the identity of the end-user based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the end-user in an interoperable and REST-like manner.

Some of the benefits of OpenID Connect:

- Minimizes password security risks.
- Makes faster sign-ins to websites.
- Provides a complete standardized setup.

Overview

- ACOS works as an OAuth client and OpenID Connect Relying Party and supports the following authorization grant types:
    - **Authorization Code**: ACOS directly creates a session to exchange the information with OAuth server. The authorization server returns a code to the client after authentication of client, the client forwards the code to ACOS, which exchanges for tokens on a secured channel
    - **Implicit**: Clients play intermediate roles to exchange the information between ACOS and OAuth servers. The authorization server returns an access token or ID token to the client.
    - **Hybrid**: This is applicable only for OIDC. It is the combination of the implicit, authorization-code mode, and OpenID. Because the access token and ID token are already in the authorization response, ACOS will not send the access token request in the hybrid processing flow
- ACOS provides the following abilities for OAuth and OpenID Connect support:
    - Allows only HTTPS endpoints of the authorization server and OpenID provider.

  - Performs the client authentication when an access token or ID token is requested from an authorization server.
  - Handles query encoding and fragment encoding response mode.
  - Supports the **Nonce** parameter *(parameter-nonce-enable* option) for some OpenID providers.
  - Supports users to configure the OAuth scope parameter.
- OAuth and OpenID Connect feature support the JSON Web Token (JWT) parsing and validation.
- ACOS supports OAuth and OpenID Connect relay:
  - **OAuth Relay**: ACOS relays access token to the backend in the HTTP Authorization header with bearer scheme.
  - **OpenID Connect Relay**: ACOS relays JWT to the backend in the HTTP Authorization header with the bearer scheme.
- OAuth session supports VRRP, so the standby device can have the OAuth auth session and works properly when it is in the active state. VRRP synchronization is performed when ACOS receives the access token or ID token.
- ACOS OAuth and OpenID Connect work with the following OpenID providers:
  - Google
  - Microsoft
  - Facebook
  - Amazon
  - GitHub
  - Dropbox

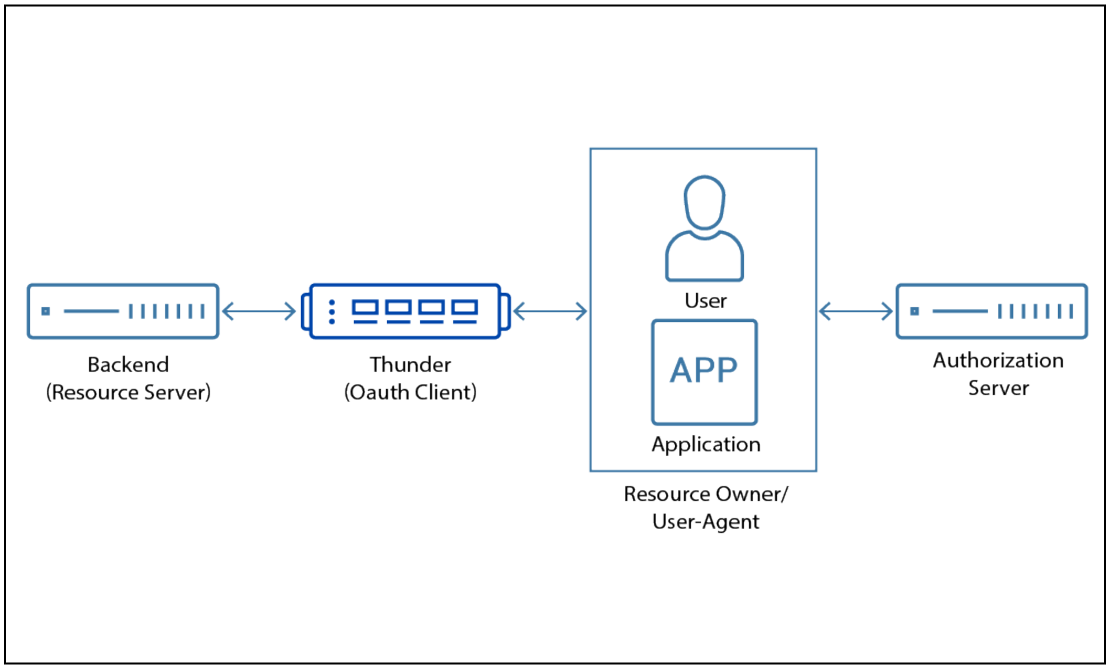
NOTE: Other vendors can also work well if they follow the RFC spec to implement OAuth2 or OpenID.

Network Topology

The following topology illustrates the interaction between the client, ACOS, and the Authorization server for different grant types.

- **Implicit**: Clients play intermediate roles to exchange information between ACOS and OAuth servers.

Figure 24 : Network Topology



Implicit Flow:

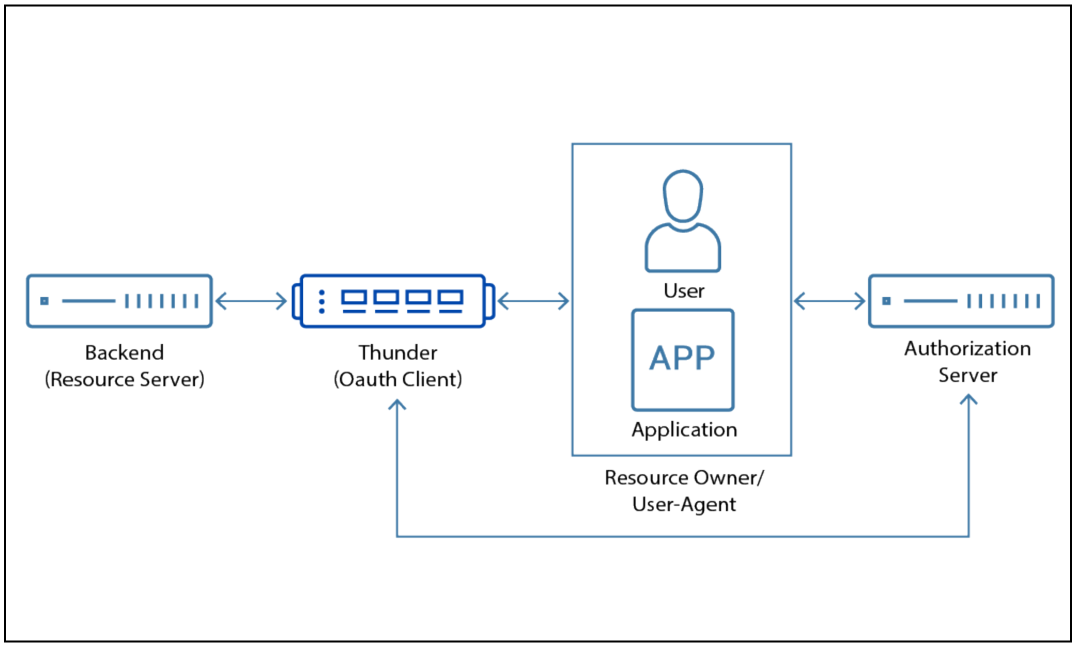
1. When a user accesses the service of backend resource server, it sends a request that will pass to ACOS.
2. ACOS gets the request and checks if the user is already authenticated, and if the user is authenticated, ACOS forwards the request to the Backend server. If the user is not authenticated, ACOS sends a redirects the user to the Authorization server.
3. The Authorization server tries to authenticate the user. If successful, it

prompts the user whether they want to allow the resource server to access the user's data

4. If the user allows to access data, authorization server will redirect the user back to ACOS with an access token.
5. User sends the authorization server response, which includes the access token to ACOS.
6. ACOS creates the authentication session and saves a token in the session.
7. ACOS sends the original user request and the access token to the Backend server.

- **Authorization Code**: ACOS directly creates a session to exchange information.

Figure 25 : Network Topology



Authorization Code Flow:

1. When a user accesses a service on the backend resource server, it sends a request that will pass to ACOS.
2. ACOS gets the request and checks if the user is already authenticated, and if the user is authenticated, ACOS sends the request to the Backend server.

If the user is not authenticated, ACOS redirects the user to the Authorization server.

3. The Authorization server tries to authenticate the user. If successful, it prompts the user whether they want to allow the resource server to access the user's data.
4. If the user allows to access data, authorization server will redirect the user back to ACOS with a code.
5. User sends the authorization server response, which includes the code to ACOS.
6. ACOS connects to authorization server directly and sends an access token request that includes the code to authorization server.
7. Authorization server responses the access token to ACOS.
8. ACOS creates the authentication session and saves token in the session.
9. ACOS sends the original user request and the access token to the Backend server.

Configuring OAuth and OpenID Connect through GUI

Complete the following steps to configure OAuth and OpenID Connect through the ACOS GUI.

For more information about GUI, see *Online Help*.

The following topics are covered:

Configure a DNS

To configure a DNS:

1. Navigate to **System > Settings** and then click **DNS**.
2. On **Configure DNS**, specify the required details.
3. Click **Update DNS** to update the DNS configuration.

Configure an OAuth Authorization Server and Client

To configure an OAuth authorization server and client:

1. Navigate to **AAM > Auth Clients** and then click **OAUTH**.
2. Click the **Authorization Server** tab, click **Create**, and specify the required details.
3. Click **Create** to create an authentication OAuth authorization-server.
4. Configure a client:
   a. Click the **Client** tab, click **Create**, and specify the required details.
   b. Click **Create** to create an authentication OAuth client.

Configure an Authentication Relay

To configure an authentication relay OAuth:

1. Navigate to **AAM > Auth Relays** and then click **OAUTH**.
2. Click **Create** and specify the required details.
3. Click **Create** to create an authentication relay OAuth.

Configure an Authentication Template and Authorization Policies

To configure an authentication template, authorization policy, and AAA policy:

1. Navigate to **AAM > Policies and Templates** and then click **Authentication Templates**.
2. Click **Create** and specify the required details.

3. Click **Create** to create an authentication template.
4. Click the A**uthorization Policies** tab and click **Create**.
5. In the **Create Auth Policy** section, specify the required details.
6. Click **Create** to create an authorization policy.
7. Click the **AAA Policies** tab and click **Create**.
8. In the **Create AAA Policy** section, specify the required details.
9. Click **Create**.
10. In the **AAA Rules** section, click **Create** to set a rule number and bind this rule to the template that the user has created earlier.
11. Click **Create**.

Configure a Virtual Server

To configure a virtual server:

1. Navigate to **ADC > SLB** and click the **Virtual Servers** tab.
2. Click **Create** and specify the required details, including the VIP name, IP address, and virtual port information.
3. In the **Virtual Port** section, click **Create**.
4. Specify the required information and click **Create**.

Configuring OAuth and OpenID Connect through CLI

Complete the following steps to configure OAuth and OpenID Connect through the ACOS CLI.

For more information, see *Command Line Reference Guide*.

The following topics are covered:

Configure a DNS

The following command configures the DNS server:

```
ACOS(config)#ip dns primary 8.8.8.8
```

Configure an OAuth Authorization Server

The following commands configure the AAM authentication server:

```
ACOS(config)#aam authentication oauth authorization-server microsoft
ACOS(config-authorization-server:microsoft)#issuer
https://login.microsoftonline.com/91d27ab9-8c5e-41d4-82e8-
3d1bf81fcb2f/v2
ACOS(config-authorization-server:microsoft)#authorization-endpoint
https://login.microsoftonline.com/common/oauth2/v2.0/authorize post
ACOS(config-authorization-server:microsoft)#token-endpoint
https://login.microsoftonline.com/common/oauth2/v2.0/token post
ACOS(config-authorization-server:microsoft)#verification-jwks
microsoft
```

Configure an OAuth Authorization Client

The following commands configure the AAM authentication client:

```
ACOS(config-authorization-server:microsoft)#aam authentication oauth
client microsoft
ACOS(config-client:microsoft)#client-id 5d0bfd36-0e92-4cb1-a1ed-
13604011e595Dd
```

```
ACOS(config-client:microsoft)#client-secret
jblytinRmCo4eePsPwI3LEnpw9WGFuZkLu/SI2O6pbfJ8CE1LyD5tjwQjLjV2wDn
ACOS(config-client:microsoft)#type openid-connect
ACOS(config-client:microsoft)#grant-type authorization-code
ACOS(config-client:microsoft)#redirection-endpoint
https://jiyen.saml.com:8443
ACOS(config-client:microsoft)#redirection scope email profile openid
```

Configure an Authentication Relay

The following commands configure the AAM authentication relay:

```
ACOS(config)#aam authentication relay oauth r1
ACOS(config-oauth:r1)#relay-type access-token
ACOS(config-oauth:r1)#relay-uri all
```

Configure an Authentication Template

The following commands configure the AAM authentication template:

```
ACOS(config)#aam authentication template t1
ACOS(config-auth template:t1)#type oauth
ACOS(config-auth template:t1)#oauth-authorization-server Microsoft
ACOS(config-auth template:t1)#oauth-client Microsoft
ACOS(config-auth template:t1)#relay r1
```

Configure an Authentication Policy

The following commands configure the AAM authorization policy:

```
ACOS(config)#aam authorization policy p1
ACOS(config-authorization policy:p1)#attribute-rule 1
ACOS(config-authorization policy:p1)#attribute 1 email attr-type
string sub-string @a10networks.com
```

Configure an AAA Policy

The following commands configure an AAA policy:

```
ACOS(config)#aam aaa-policy ap1
ACOS(config-aaa policy:ap1)#aaa-rule 1
ACOS(config-aaa policy:ap1-aaa rule:1)#authentication-template t1
ACOS(config-aaa policy:ap1-aaa rule:1)#authorize-policy p1
```

Configure an SLB Virtual Server

The following commands configure the SLB Virtual server:

```
ACOS(config)#slb virtual-server vs1 192.168.92.168
ACOS(config-slb vserver)#port 443 https
ACOS(config-slb vserver-vport)#source-nat auto
ACOS(config-slb vserver-vport)#service-group sg1
ACOS(config-slb vserver-vport)#template server-ssl ss1
ACOS(config-slb vserver-vport)#template client-ssl cs1
ACOS(config-slb vserver-vport)#aaa-policy ap1
```

Import the JWKS File

The following command imports the JWKS file

```
ACOS(config)#import-periodic auth-jwks microsoft
https://login.microsoftonline.com/common/discovery/v2.0/keys period 60
```

Configuration Example

```
ip dns primary 8.8.8.8
!
aam authentication oauth authorization-server microsoft
  issuer https://login.microsoftonline.com/91d27ab9-8c5e-41d4-82e8-
3d1bf81fcb2f/v2.0
  authorization-endpoint
https://login.microsoftonline.com/common/oauth2/v2.0/authorize post
  token-endpoint
https://login.microsoftonline.com/common/oauth2/v2.0/token post
  verification-jwks microsoft
!
aam authentication oauth client microsoft
```

```
  client-id 5d0bfd36-0e92-4cb1-a1ed-13604011e595
  client-secret encrypted
jblytinRmCo4eePsPwI3LEnpw9WGFuZkLu/SI2O6pbfJ8CE1LyD5tjwQjLjV2wDn
  type openid-connect
  grant-type authorization-code
  redirection-endpoint https://jiyen.saml.com:8443
  scope email profile openid
!
aam authentication relay oauth r1
  relay-type access-token
  relay-uri all
!
aam authentication template t1
  type oauth
  oauth-authorization-server microsoft
  oauth-client microsoft
  relay r1
!
aam authorization policy p1
  attribute-rule 1
  attribute 1 email attr-type string sub-string @a10networks.com
!
aam aaa-policy ap1
  aaa-rule 1
  authentication-template t1
   authorize-policy p1
!
slb virtual-server vs1 192.168.92.168
  port 443 https
   source-nat auto
   service-group sg1
   template server-ssl ss1
   template client-ssl cs1
  aaa-policy ap1
!
import-periodic auth-jwks microsoft
https://login.microsoftonline.com/common/discovery/v2.0/keys period 60
```

Known Issues and Limitations

The following are the known issues and limitations:

- Only OAuth2 implicit, authorization code, and hybrid flow are supported. Resource owner password credentials and client credentials are not implemented because they are not supported in OpenID Connect and they have high security risk.
- ACOS currently does not support the mechanism of refresh tokens since OAuth2 vendors have different specifications.
- For OAuth2, only the Bearer token type is supported.
- In the JWT payload, ACOS receives the client's information. The authentication server provides information based on the Scope field. You can change the value of this field to convey to the server about the required information. The format or value of each vendor is different so, read the guidelines provided by vendors.
- A few vendors do not support the OpenID protocol, such as Facebook. So, ACOS does not perform the authorization policy.

Sharing Authentication Information Between VIPs

This chapter provides information about sharing authentication information between VIPs.

The following topics are covered:

Overview

If an ACOS device configures multiple VIPs with AAM features, and these VIPs are in the same network domain, the user can configure the ACOS device to share (or not to share) authentication information between the VIPs. For example, we have VIP1 and VIP2 on an ACOS device, where VIP1 is for *email.test.com* and VIP2 is for *files.test.com*.

One of the following scenarios exist:

- If the user has already authenticated in VIP1 (*email.test.com*), then the user can access VIP2.
  This scenario means that authentication information is shared between VIP1 and VIP2.
- If the user cannot access VIP2, then the user must first authenticate to access VIP2 resources.
  This scenario means that authentication information is not shared between the VIP1 and VIP2.

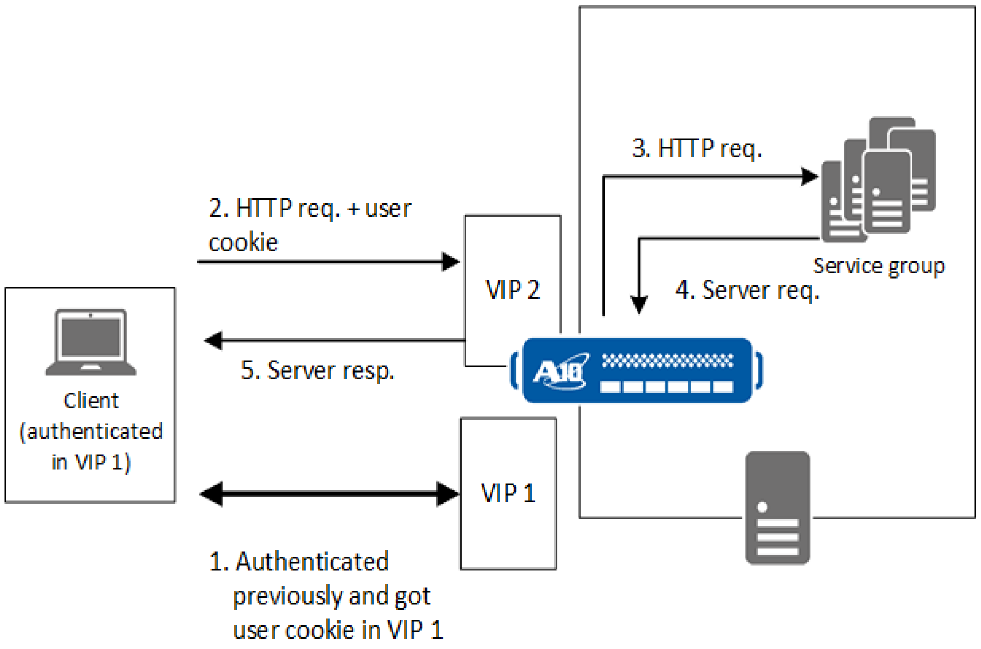
Sharing Authentication Information

The following topics are covered:

Information Details

Authentication information can be shared between VIPs when they are in the same network domain, such that they are configured with the same authentication server and the same domain suffix. If a client is authenticated to one VIP in the network domain, and they receive a user cookie from ACOS, they do not have to authenticate again when they try to access services using a different VIP in that same network domain.

Figure 26 : Sharing Authentication Information Between VIPs in a Domain


3. HTTP req.
2. HTTP req. + user cookie
VIP 2
Service group
4. Server req.
5. Server resp.
Client (authenticated in VIP 1)
VIP 1
1. Authenticated previously and got user cookie in VIP 1

Workflow

The following steps provide a high-level overview of this process:

1. The client is authenticated in VIP1 and receives a cookie with the domain parameter from ACOS.
2. When at the next time that the client requests the same DNS domain or domain suffix, but by using a different VIP, the client sends an HTTP request with the cookie that was received in step 1.
3. ACOS gets the user information from the cookie and determines whether the user can access the other VIP. If the ACOS device allows the request from the user, the device relays the request to the servers.
4. The ACOS device receives a response from the server.
5. The server response is sent to the client.

Not Sharing Authentication Information

In some cases, VIPs that have the same cookie domain don’t share authentication information.

For example, in Figure 27, if the client is authenticated in VIP1, they can access the same domain in VIP2. However, they need to authenticate again to access information in VIP3 and VIP4.

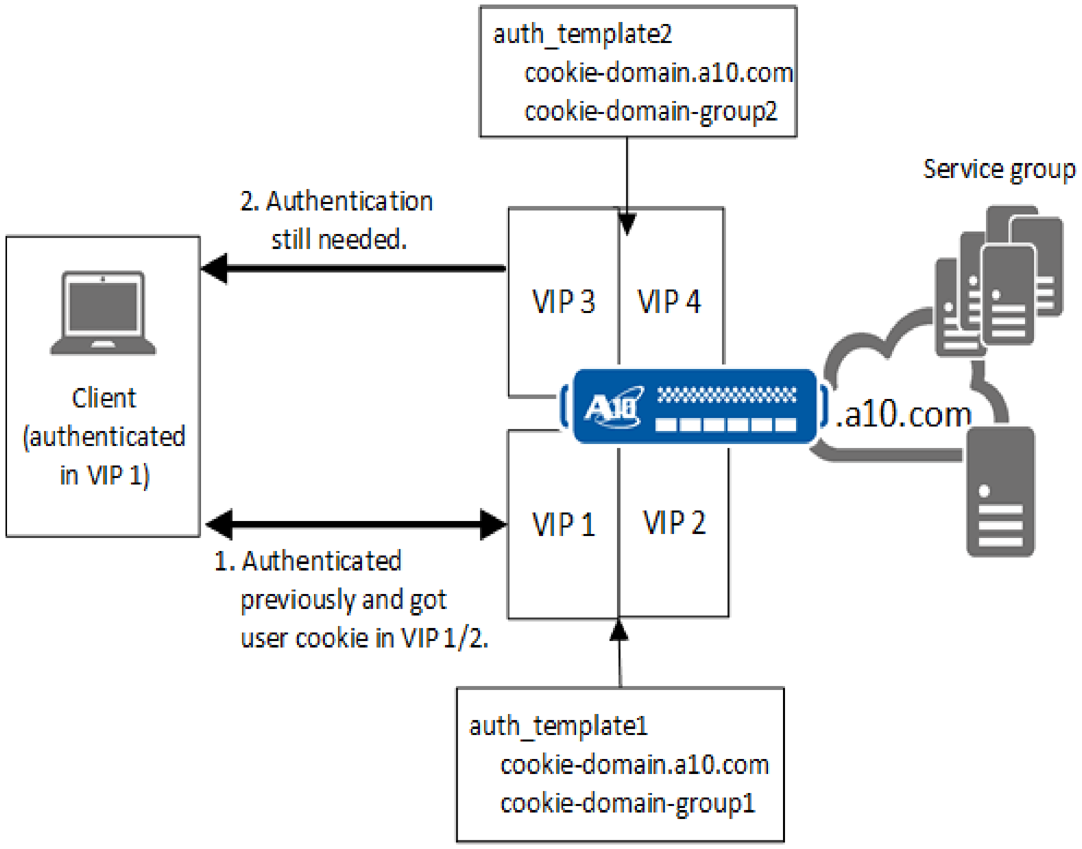
Similarly, if they are authenticated in VIP4, they can access the same domain in VIP3, but they must authenticate in VIP1 and VIP2.

This is configured through the use of a cookie domain group ID when multiple authentication templates are involved.

When a cookie domain group ID is not shared in an aam authentication template, authentication information is not shared.

In this example, two authentication templates share the same cookie domain, but a different cookie domain group.

Figure 27 : No Authentication Information Sharing Between VIPs in a Domain



Configuring a Cookie Domain

The user can configure a cookie domain by using the GUI or CLI.

The following topics are covered:

Adding a Cookie Domain Using the GUI

To add a cookie domain:

1. Click **AAM** > **Policies and Templates**.
2. On the **Authentication Templates** tab, either select an existing template or create a new template.
3. In **Cookie Domains**, enter a domain, and click **Add**.
4. Click **Update**.

Removing a Cookie Domain by Using the GUI

To remove a cookie domain:

1. Click **AAM** > **Policies and Templates**.
2. On the **Authentication Template** tab, select an existing template, and click **Edit**.
3. In **Cookie Domains**, click the **x** next to the cookie domain that the user wants to remove.
4. Click **Update**.

Adding a Cookie Domain Group ID by Using the GUI

To add a cookie domain group ID:

1. Click **AAM** > **Policies and Templates**.
2. On the **Authentication Template** tab, either select an existing template or create a new template
3. In **Cookie Domain Group IDs**, enter a domain group ID.
4. Click **Add**.

Removing a Cookie Domain Group ID by Using the GUI

To add a cookie domain group ID:

1. Click **AAM** > **Policies and Templates**.
2. On the **Authentication Template** tab, select an existing template, and click **Edit**.
3. In **Cookie Domain Group IDs**, click **x** next to the group ID that the user wants to remove.
4. Click **Update**.

Configuring a Cookie Domain by Using the CLI

To configure a cookie domain:

1. Configure a cookie domain in an AAM Authentication Template, using the following command at the AAM Authentication Template configuration level:

```
ACOS(config)# aam authentication template template1
ACOS(confg-auth template:1)# cookie-domain.a10.com
```

2. Bind the AAM Authentication Template to an AAA Rule in an AAA Policy.
3. Configure the same port in different VIPs, and bind the AAA Policy to that port in each respective VIP.
4. To configure a second cookie domain, repeat steps 1-3. However, at the AAM Authentication Template configuration level, enter the additional command:

```
ACOS(confg-auth template:1)# cookie-domain-group num
```

CLI Example

The following commands configure a cookie domain shared by VIP1 and VIP2, as shown in No Authentication Information Sharing Between VIPs in a Domain.

```
aam authentication template auth_template1
cookie-domain .a10.com
  cookie-domain-group 1
  logon lb1
server ra1
!
aam aaa-policy policy1
  aaa-rule 1
action allow
authentication-template auth_template1
```

```
!
slb virtual-server vip1 192.168.91.56
port 80 http
source-nat auto
service-group http_sg
aaa-policy policy1
!
slb virtual-server vip2 192.168.91.57
port 80 http
source-nat auto
service-group http_sg
 aaa-policy policy1
!
```

The following commands configure a second cookie domain group shared by VIP3 and VIP4.

These two VIPs do not share authentication information with VIP1 and VIP2. (See No Authentication Information Sharing Between VIPs in a Domain.)

This is configured by ensuring the cookie-domain-group ID for VIP3 and VIP4's aam authentication template does not share the same cookie-domain-group ID as VIP1 and VIP2, as highlighted in the CLI configuration for *auth_template2*.

```
aam authentication template auth_template2
cookie-domain .a10.com
cookie-domain-group 2
logon lb1
server ra1
!
aam aaa-policy policy2
aaa-rule 1
action allow
authentication-template auth_template2
!
slb virtual-server vip3 192.168.91.58
  port 80 http
source-nat auto
service-group http_sg
aaa-policy policy2
!
slb virtual-server vip4 192.168.91.59
```

```
port 80 http
source-nat auto
service-group http_sg
aaa-policy policy2
!
```

AAM and Kerberos

This chapter describes how to set up a Kerberos authentication logon, server, and relay.

The following topics are covered:

Overview

Kerberos is an authentication protocol that uses “tickets” to allow nodes in a non-secure network to securely communicate and verify their identities to each other. Kerberos uses a client-server model, which allows mutual authentication between the user and the server. Kerberos protocol messages are highly protected, which prevents eavesdropping and replay attacks. By default, Kerberos uses port 88.

AAM supports Linux and Windows Kerberos authentication.

NOTE: In the Windows environment, Kerberos Domain Controller (KDC) supports AAM authentication, password change, and KDC validation. However, In the Linux environment, KDC supports AAM authentication only.

The following topics are covered:

Terms

The following topics are covered:

Realm

The domain of networked services to which the Kerberos server permits or denies access.

Kerberos Domain Controller (KDC)

The Kerberos server, which consists of the following main components:

Authentication Service

Checks the user account database for the username and password from the client.

If there is a match, the authentication service creates a Ticket Granting Ticket (TGT) for the client and sends the ticket to ACOS.

The authenticating service also sends the TGT to the ticket granting service to request a service ticket.

Ticket Granting Service

Creates Service Tickets (STs) for individual client requests.

Ticket-Granting Ticket (TGT)

Master ticket that verifies the identity of the client to ACOS, and ACOS uses the client’s TGT to obtain ST tickets for specific client requests (described below).

Service Ticket (ST)

Ticket that verifies the client’s identify to a specific service for an individual request to that service.

Front-end Server

Kerberos proxy (the ACOS device).

Back-end Server

Server that is running the services that are requested by clients.

Security Principal

Kerberos term for the client.

Service Principal

Kerberos term for the service that is requested by the client.

Protocol Translation

Allows the use of other protocols in addition to Kerberos for the client-server AAA exchange. This capability uses the S4U2self Kerberos extension.

Kerberos Constrained Delegation (KCD)

Allows ACOS, which is operating as a Kerberos front-end server, to use the TGT that is granted to a client by the KDC to issue STs for service requests.

This capability uses the S4U2proxy Kerberos extension. An authentication relay configuration which is shown in this guide is for Constrained Delegation Protocol Transition (CDPT), which utilizes this concept.

ACOS supports Kerberos service configured in the Microsoft Active Directory. When a user attempts to access a secured web application or web service, ACOS communicates with the Kerberos authentication/directory server to authenticate the user accessing the application/service.

Users associated with the Kerberos directory server can update their password. Kerberos password update is one of the solutions that is provided by Application Access Management (AAM).

Password Change Management

Password Change can be initiated by the:

1. Active Directory
    a. When the password has expired.
       OR
    b. When the password is expiring in N days (if prompt-password-change-before-expiration is configured for the user account).
2. User

   The user can change the password by using the "Change Password" link on the logon page.

Guidelines for Kerberos Authentication Using Active Directory

The following are the guidelines for Kerberos authentication using Active Directory.

- The users associated with the LDAP Active Directory can change their password only through the "Logon Form-Based" authentication type. Other logon authentication types are not supported.
- If the Active Directory is configured with the default "Minimum password age" under Account Policies > Password Policy, the user cannot change the password for the duration specified in the policy. To enable password change for users, the administrator can set the "Minimum password age" to zero (0).

Kerberos Authentication Workflow Considerations

In a Kerberos authentication workflow, the user needs an approved user account and a password. All KDCs must be in the same realm and share the same information, such as principals and passwords. This allows ACOS to choose any KDC to authenticate an application server. If the user has multiple domains (realms), the user can configure multiple Authentication templates and use the Kerberos user account to handle the KDC.

The following topics are covered:

Best Practice Workflow

This topic describes the best practice that administrators can follow to deploy the solution.

The following topics are covered:

Authentication Logon Configuration

Administrators configure the authentication logon method.

Authentication Server Configuration

Administrators configure the AAM authentication server parameters. Before configuring this option on the device, administrators must ensure that the authentication server is configured and available.

Authentication Relay Configuration

Administrators configure the authentication relay on the device.

Authentication Template Configuration

Administrators creates the authentication template.

AAA Policy Configuration

Administrators configure the AAA rules to allow or deny access.

Service-Principal Configuration within the Server Port Configuration

Administrators create a service group and add servers within the pool.

Kerberos Authentication Scenarios

The following scenarios describe Windows and Linux Kerberos authentication and topology flow.

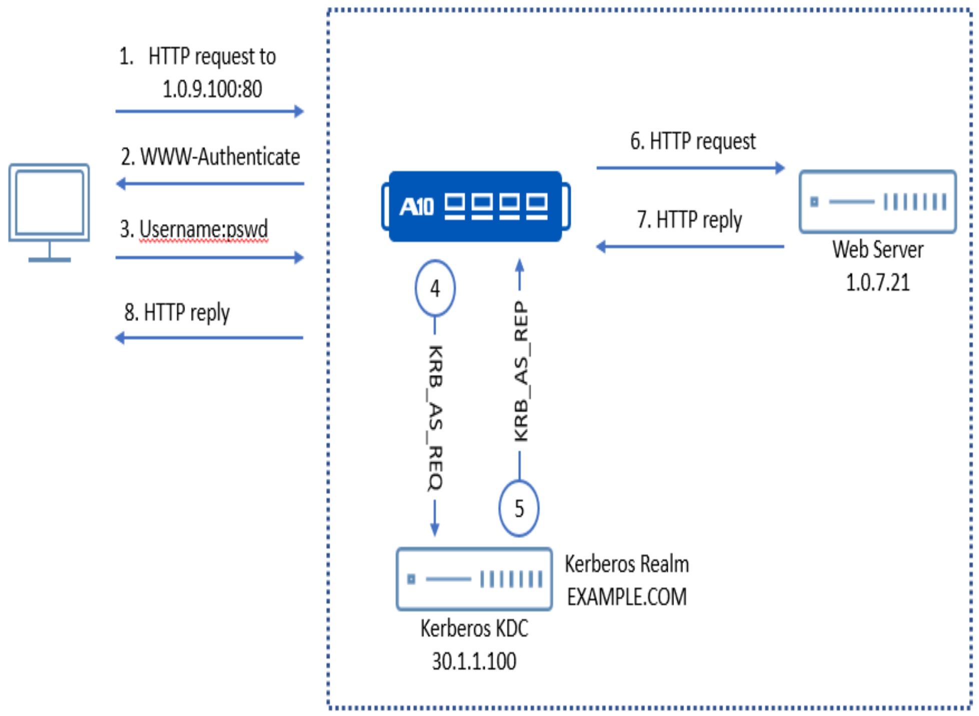
- The first scenario relies on using wildcard VIPs for general domain authentication. The ACOS device is in the AD domain and uses a wildcard VIP. The ACOS device works as a gateway and only clients that have domain accounts are allowed to access the Internet.
- In the second scenario, the ACOS device provides a normal (non-wildcard) SLB service and the client and the ACOS device are members of the same domain.

The following topics are covered:

Basic HTTP Logon

Figure 28 shows how Basic HTTP Logon is used with Kerberos authentication in this deployment.

Figure 28 : Kerberos Single Sign-on - Basic HTTP Login



In this deployment, ACOS uses a backend Kerberos authentication server to authenticate access to virtual port 80 on the VIP.

1. A client sends a request to the HTTP port 80 on the VIP.

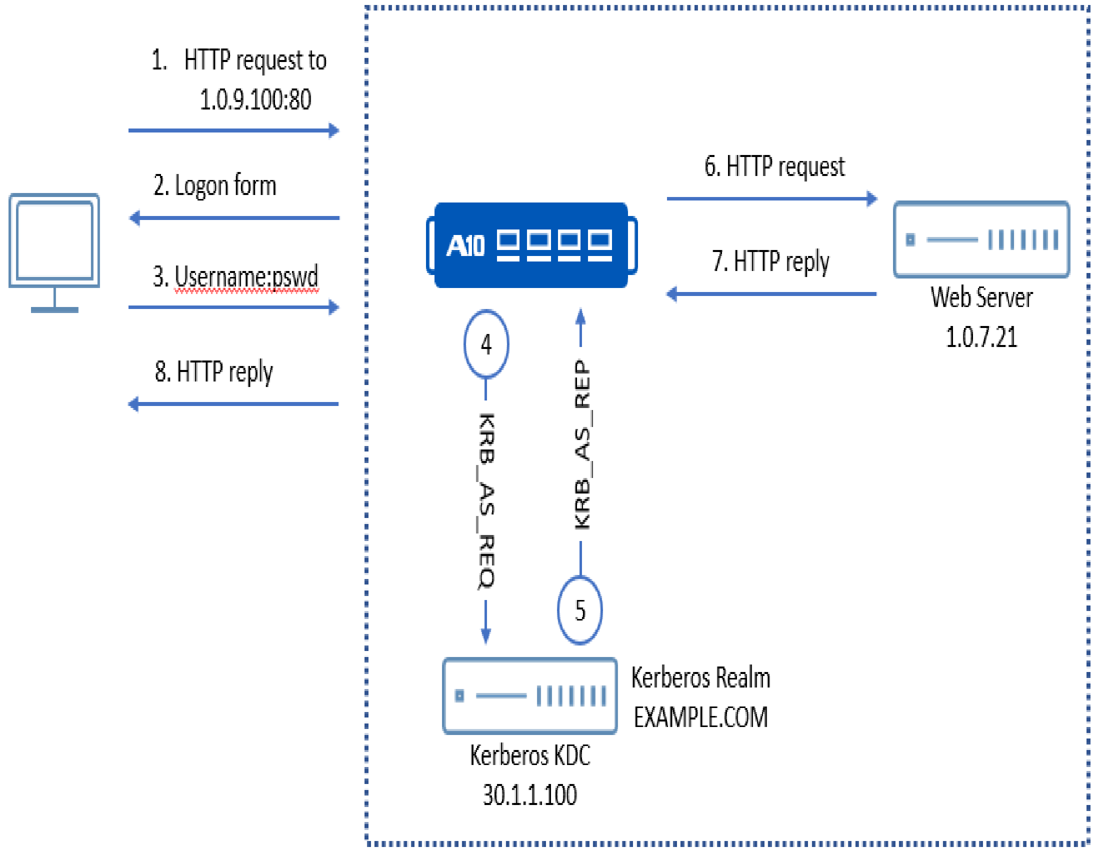
   The 1.0.9.100 IP address is an SLB VIP on the ADC.

2. ACOS sends an HTTP 401 (Not Authorized) message, which contains an authentication header, to the client.
3. The client enters the username and password.
4. ACOS sends the end-user credentials to a backend Kerberos authentication server for authentication.
5. The Kerberos authentication server authenticates the user and sends the response to ACOS.
6. If the authentication is successful, ACOS caches the credential verification from the Kerberos authentication server and forwards the server reply to the client.

Form-based Logon

Figure 29 shows how form-based logon is used with Kerberos authentication in this deployment.

Figure 29 : Kerberos Single Sign-on - Form-based Login



1. Client sends a request to the HTTP port 80 on the VIP.

   The 1.0.9.100 IP address is an SLB VIP on the ADC.

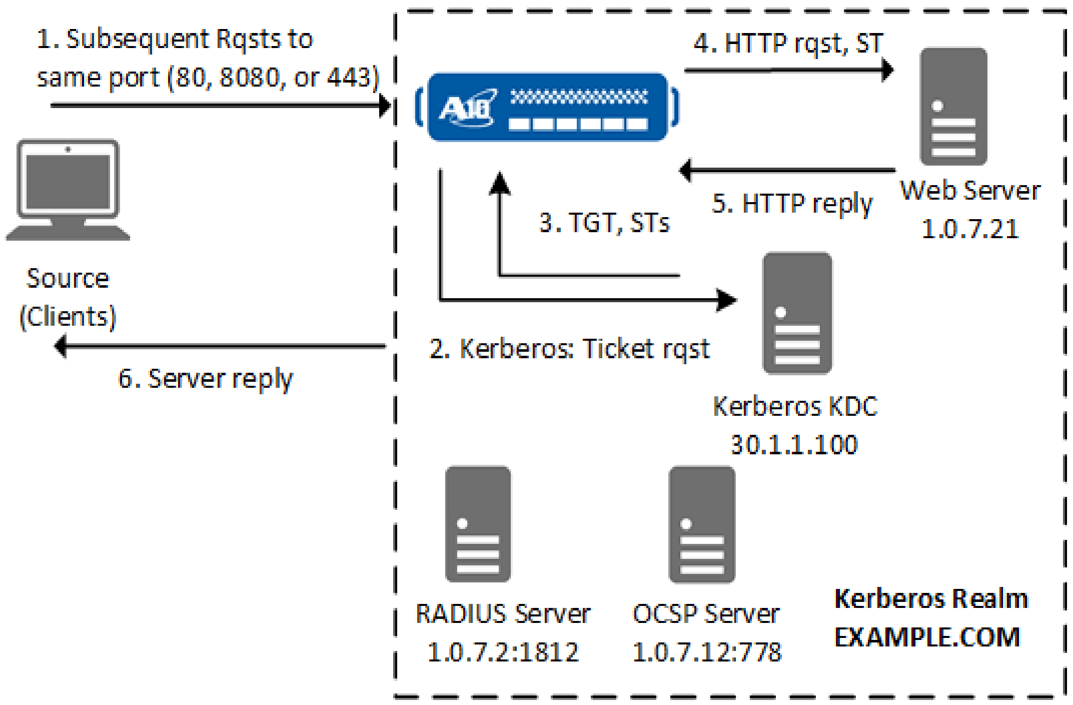
2. ACOS identifies that the web application or service requested by the client is protected by the Kerberos authentication server and sends a web page challenging the user to enter the login credentials (username and password).
3. The client enters the username and password.
4. ACOS transmits the credentials to a backend Kerberos authentication server for authentication.

5. The Kerberos authentication server authenticates the user and sends the response to ACOS.
6. If the authentication is successful, ACOS caches the credential verification from the Kerberos authentication server and forwards the server reply to the client.

Authentication Relay with Kerberos

Figure 30 shows how Kerberos is used in this deployment.

Figure 30 : Kerberos Single sign-on - Kerberos Ticketing



After authenticating a client, ACOS uses a Kerberos Key Distribution Center (KDC) to manage authentication for subsequent requests from the client. ACOS supports Kerberos authentication relay, so clients can authenticate to the ACOS device by using other protocols, such as LDAP and RADIUS, and use the ACOS device to issue the authentication to the Kerberos authentication server.

If the user wants to designate multiple authentication servers for relay, ACOS can load balance and handle multiple KDCs. A primary server can be used to handle most authentication relays, and a secondary server can be used to handle additional requests to ensure the continuous availability of authentication services.

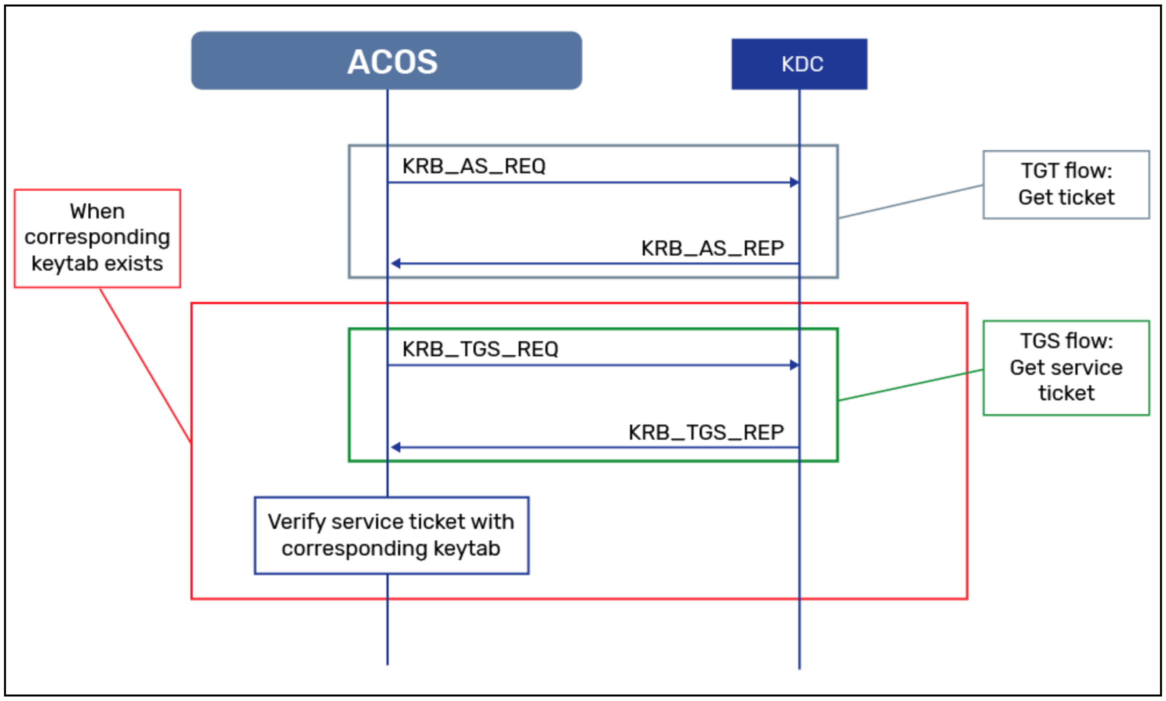
In a multiple KDC environment, the user can only use an AAM service group, which provides Round Robin as the load balancing algorithm. The server that has been configured with the highest priority is chosen first. If the user configures all servers with the same priority, Round Robin is used to select servers in rotation.

Kerberos Key Distribution Centre Service Ticket Validation

The Kerberos Key Distribution Centre (KDC) service ticket validates the KDC with a specified account and password to prevent an attack such as an attacker returning a fake Ticket Granting Ticket (TGT) and bypassing the Kerberos authentication.

NOTE: The KDC validation for KRB5 on Linux is not supported as the encryption type used by ACOS is not supported on KRB5.

Kerberos authentication flow block diagram


ACOS
KDC
KRB_AS_REQ
KRB_AS_REP
TGT flow: Get ticket
When corresponding keytab exists
KRB_TGS_REQ
KRB_TGS_REP
TGS flow: Get service ticket
Verify service ticket with corresponding keytab

Kerberos authentication flow:

1. Create a service account with a specific SPN for the ACOS device.
2. Configure the service account information next to the `kerberos-validate-kdc` command.
3. If the KDC validation is enabled, the KDC validation account will be sent to the auth request. If the corresponding KDC keytab exists, ACOS gets the service ticket for the specified account information and uses the keytab, which is generated from the service account information to verify the service ticket.
4. Kerberos authentication returns success in the following scenarios:
   a. KDC validation is disabled, and the user is authenticated in the TGT flow.
   b. KDC validation is enabled, and the service ticket verification is passed.
5. If KDC validation is enabled, but ACOS cannot find the corresponding keytab, or the service ticket verification fails, the KDC validation failure counter increases and Kerberos authentication fails.

KDC Configuration Example

```
aam authentication logon http-authenticate http-basic
  auth-method basic enable
!
!
aam authentication server windows kerberos4
  host 10.65.18.84
```

Limitation auth-protocol kerberos-validate-kdc spn HTTP/client.lchao.com account client password!QAZ2wsx

```
  realm LCHAO.COM
!
aam authentication template basic_logon-kerberos_authn
 logon http-basic
 server kerberos4
!
aam aaa-policy basic-logon_kerberos-authn
  aaa-rule 1
   authentication-template basic_logon-kerberos_authn
!
slb virtual-server test 5.5.21.201
 port 80 http
  source-nat auto
```

```
service-group sg1
aaa-policy basic-logon_kerberos-authn
```

Configuring Multiple KDC Servers

The user can configure multiple KDC servers to handle additional requests by creating a service group of authentication servers.

The following topics are covered:

Configuration Process for Kerberos Deployment

1. Configure the Kerberos authentication servers to include in the service group
2. Create a service group and bind the authentication servers to the configured service group:
3. Create an authentication template which includes the service group that was configured above:
4. Bind this template to a policy, as done in the previous scenario:
5. Assign the policy to a VIP and server configured in SLB mode:

Instead of using authentication servers, the user can use service groups to load balance multiple Kerberos relays. A revision of step 3 can be completed in the aam authentication relay module, as shown below. The relay that contains the primary KDC server is bound to the template, which allows credential validation and requests to be directed to that server first.

Configuration Example for Kerberos Deployment

To set up credentials and basic logon information in a Kerberos deployment:

1. Set up a Kerberos account by entering the following commands:

```
ACOS(config)# aam authentication account kerberos-spnname
realmdomain
accountusername
```

```
passwordpassword
```

2. Use an HTTP-based authentication mechanism to conduct credential validation and specify an HTTP-based authentication method:

```
aam authentication logon http-authenticatename
auth-method negotiate enable
```

NOTE: Enabling “negotiate” for setting a SPNEGO protocol is a suggested configuration.

3. Set up an authentication server for Kerberos by using the `windows` module:

```
aam authentication server windowsname
hostip-addr
realmdomain
```

NOTE: By default, Kerberos is set up as the authentication protocol

4. Define the authentication template to include the configured logon, server, and Kerberos account:

```
aam authentication templatename
 logon http-auth
servername
accountname
```

5. To configure the set up shown above to port 80, configure a policy template and designate the template settings to the appropriate service groups and ports under SLB configuration:

Configuring the Authentication Logon and Server by Using the GUI

The following topics are covered:

Creating a Kerberos Account

1. Click **AAM** > **Accounts**.
2. Click **Create**.
3. Fill out the template fields for creating a Kerberos account including a name, SPN domain account, the Kerberos realm, service principal name, and secret string.
4. Click **Create** to finish creating the account.

Creating an Authentication Client

1. Click **AAM** > **Auth Clients**.
2. On the **Logon HTTP Auth** tab, select an existing authentication method or click **Create**.
3. Enter a name for the client logon template and a value for the maximum number of retries. Then, select the logon type.

   If the user has selected the **Enable Basic Logon** check box, complete the following steps:

   a. Select a challenge response form.
   b. Enter the new PIN page name, PIN variable name, token page and variable page name, and the realm.
4. Click **Create** to finish creating the client template.

Creating an Authentication Server

1. Click **AAM** > **Auth Servers**.
2. On the **Windows** tab, select **Create** to configure the template for the authentication server.

3. Enter a server name, host name, realm, and timeout value. If the user wishes to disable the server’s health check, select **Disable Health Check** from the drop-down list.
4. Click **Create** to set the authentication server.

Creating an Authentication Policy

1. Select **AAM** > **Policies and Templates**, and click **Create**.
2. Enter a policy name and select a virtual port binding or create a new virtual port:
   a. Select a virtual port binding.
   b. Click **New VP**.
3. Click **Create**.
4. In the **AAA Rule** section, click **Create** to set a rule number and bind this rule to the template that the user sets up earlier.

Creating an SLB Server

1. Click **ADC** > **SLB** and navigate to the **Servers** tab. Select **Create** to pull up the template.
2. Enter the server name, type, host name, action, and determine if the user wants to enable or disable the health check. Select a health monitor, connection limit, and if the user does not wish to not generate logs, select the **No Logging** check box.
3. In the **Port** section, click **Create** to complete the template for the authentication port.
4. Enter the port and select a protocol. Specify the range, action for the health check, connection limit value, and if the user wishes to prevent logs from being generated, select the **No Logging** check box.
5. Expand the **Advanced Fields** section and configure as appropriate. Click **Create** to finish.

Creating a Service Group

1. Click **AAM** > **Service Groups** and click **Create.**
2. Enter a service group name, protocol, and load balancing method. Click **Create** to finish.

Creating a Virtual Server

1. Click **ADC** > **SLB**.
2. On the **Virtual Servers** tab, click **Create** to pull up the virtual server template.
3. Enter the server name, IP address type, netmask, and action. To use a wildcard, select the **Wildcard** check box.
4. In the **Virtual Port** section, click **Create** to pull up the template.
5. Enter a port name, protocol, port number, range, connection limit, action and service group. To use another port, select the **Alternate Port** check box. To reset the connection limit value, select the **Reset** check box. To prevent logs from being generated, select the **No Logging** check box.
6. Expand the **General Fields** section and configure as appropriate.
7. Expand the **Templates** section and select an existing template or create a new template
8. Click **Create** to finish.

Configuring a Kerberos Authentication Relay

The following topics are covered:

Using the GUI

To configure the Kerberos authentication relay:

1. Begin with creating the Windows authentication server by selecting **AAM** > **Auth Servers** and navigating to the **Windows** tab. Click **Create** to begin.

2. Enter the appropriate information and click **Create** to finish.
3. Configure a Kerberos relay by selecting **AAM** > **Auth Relays**, navigating to the **Kerberos** tab, and clicking **Create**.
4. Enter the relevant information and click **Create**.
5. Click **AAM** > **Policies and Templates**.
6. On the **Authentication Templates** tab, click **Create**.
7. Enter the relevant information and click **Create**.
8. On the **AAA Policies** tab, click **Create**.
9. Enter the relevant information and click **Create**.
10. In the **AAA Rules** section, click **Create** to set a rule number and bind this rule to the template that the user has created earlier.
11. Click **Update**.
12. Create the SLB server by clicking **ADC** > **SLB** and navigating to the **Servers** tab. Click **Create** to enter the relevant information and set the IP and port, and click **Create** again to finish.
13. Create the service groups by clicking **AAM** > **Service Groups**.
14. Enter the relevant information to create the server.
15. In the **Service Group Members** section, click **Create**, and complete the following steps:
    a. Enter a port number.
    b. Click **Create**.
16. Click **Update**.
17. To create the virtual server, navigate to **ADC** > **SLB** and on the **Virtual Servers** tab, click **Create.**
18. Enter the appropriate information to create the server, including the VIP name, IP address, and virtual port information.
19. In the **Virtual Port** section, click **Create**.
20. Enter the relevant information.
21. Expand the **General Fields** section, enter the relevant information.

22. Expand the **Templates** section and select a policy template or create a new policy template
23. Click **Create**.

Using the CLI

The following procedure describes how to configure a Kerberos authentication relay. The authentication relay designates the backend Kerberos servers to complete authentication on behalf of the client.

To configure the authentication-relay profile for the Kerberos server:

1. Configure the Kerberos server:

```
ACOS(config)# aam authentication server windowsname
hostip-addr
realmdomain
```

2. Specify the properties of the Kerberos authentication relay, including the realm, KDC server IP address, account and password:

```
ACOS(config)# aam authentication relay kerberosname
 kerberos-realm domain
kerberos-kdcip-addr
kerberos-accountusername
passwordpassword
```

3. Include the logon, relay, and server settings that the user has configured earlier:

```
ACOS(config)# aam authentication template name
logon http-basic
relayname
servername
```

4. Set the policy and bind the configured template to the rule:
5. Configure the set-up shown above to port 80, designate the template settings to the appropriate service groups and ports under SLB configuration:

   After the user sets the appropriate service group and its members with port numbers defined, bind a virtual server to the Kerberos policy template configured earlier.

Feedback

Configuring Health Monitors for Kerberos

The following topics are covered:

Using the GUI

1. Click **ADC** > **Health Monitors**.
2. On the **Health Monitors** tab, click **Create**.
3. Enter a name.
4. Select the Kerberos KDC type.
5. Enter the relevant information.
6. Expand the Kerberos KDC section, and select a Kerberos type.
7. Enter the relevant information.
8. Click **Create Monitor**.
9. Click **ADC** > **SLB**.
10. On the **Virtual Servers** tab, click **Create**.
11. Enter the relevant information and click **Create** to create a real server and port.
12. In the **Virtual Port** section, complete one of the following tasks:
    a. Select an existing port number, and click **Edit** to add the health check and service principal name.
    b. Click **Create**.
13. Expand the **Advanced Fields** section and enter the appropriate information.
14. Click **Create**.

Using the CLI

There are two methods to configure a health monitor by using Kerberos protocol. The example below uses HTTP authentication to check the web server's functionality by providing a Kerberos ticket. Before the user can complete this protocol, configure the Kerberos authentication servers and template designations.

1. Go to the `health monitor` mode and enter a name for the Kerberos health monitor:

```
health monitorname
```

2. Select a method type and go into `http`:

   To use a Kerberos authentication mechanism, select `kerberos-auth`:

```
method http expect response-codecode
usernameusernamepasswordpasswordkerberos-auth realmdomainkdcip-
addrportnum
```

NOTE: The user needs to specify the realm of the user’s Kerberos server, the port number configured earlier under the server profile, the principal authentication server, a username and password for the administrative credentials.

3. Set up a real server and port:
4. Under the port, configure the health monitor that was set up in step 3, and a service name.

Configuring a Kerberos Health-Check

The user can also configure a Kerberos health check by entering the **`kerberos-kdc health check`** command, which checks Kerberos KDC's functionality.

To configure a Kerberos health check:

1. Go to the `health monitor` mode and enter a name for the Kerberos health monitor:

```
health monitorname
```

2. Select a method type and go into `kerberos-kdc`.

   There are three options for health monitoring checks which the user can select.

```
method kerberos-kdc {kadmin | kinit | kpassword} domainpassword
```

3. Assign the configured health monitor to one of the KDCs that the user has configured.
4. Go into the `aam` mode, and under the server profile, enter the name of the configured monitor under health-check.

```
aam authentication server windowsname
health-checkname
```

Consider the following information:

- **kadmin** health monitoring is completed when administrators can access an accounts and successfully validate the credentials.
- **kinit** health monitoring is completed when, by using the provided username and password, the ACOS device can secure a ticket for the user.
- **kpassword** allows the user to change the password for the principal server that was selected in the Kerberos commands above.
- **http-kerberos-auth** allows the ACOS device to conduct a health check for a principal server under an HTTP authentication protocol. In this scenario, the user is using an HTTP/HTTPS negotiate service to secure tickets and validate a user.

NOTE: For a Kerberos health monitor, the user must select the **Kerberos KDC** method.

Configuration Example

The following is a representation of the examples of configuring Kerberos authentication logon, server, and relay.

```
aam authentication logon http-authenticate hbasic
 auth-method basic enable
aam authentication server ldap ldap_serv
 host 192.0.2.150
 base cn=Users,dc=a10lab,dc=com
 admin-dn cn=Administrator,cn=Users,dc=a10lab,dc=com
 admin-secret *****
 default-domain a10lab
aam authentication relay kerberos kerberos-relay
 kerberos-realm A10LAB.COM
 kerberos-kdc 192.0.2.150
 kerberos-account ax/cdpt
 password ******
slb server sptest1 192.0.2.100
 port 80 tcp
 service-principal-name HTTP/sptest1.a10lab.com
 port 443 tcp
 service-principal-name HTTPS/sptest1ssl.a10lab.com
 port 8888 tcp
 service-principal-name HTTP/sptest1.a10lab.com
aam authentication template kerberos-tmpl
 logon hbasic
 relay kerberos-relay
 server ldap_serv
aam aaa-policy kerb-relay
 aaa-rule 1
 action allow
 authentication-template kerberos-tmpl
slb service-group mywsu-sg-443 tcp
 member sptest1 443
!
slb service-group mywsu-sg-80 tcp
 member sptest1 80
!
slb service-group mywsu-sg-5555 tcp
 member sptest1 5555
slb virtual-server sharepoint-vs 192.0.2.234
 port 80 https
 source-nat auto
 service-group mywsu-sg-80
 template client-ssl cssl
 aaa-policy my-aaa-policy
 port 443 https
 source-nat auto
service-group mywsu-sg-443
 template server-ssl s1
 template client-ssl cssl
 aaa-policy my-aaa-policy
 port 8888 https
 source-nat auto
```

```
service-group mywsu-sg-5555
template client-ssl cssl
aaa-policy my-aaa-policy
```

Windows Integrated Authentication

This chapter describes how to set up a Windows Integrated Authentication (WIA) logon, server, and relay and combine Kerberos protocol by using ACOS.

The following topics are covered:

NOTE: Windows Integrated Authentication can also be referred to by a variety of common industry-standard names. For more information, see https://en.wikipedia.org/wiki/Integrated_Windows_Authentication.

Overview

Windows Integrated Authentication (WIA) is an authentication mechanism that is specific to the SPNEGO, and Kerberos authentication protocols. This mechanism combines approaches to deliver security for Microsoft applications and is specific to Windows NT-based operating systems. WIA leverages the security features of Windows clients and servers, but is different from the logon mechanism in Basic and Digest authentication, because initially users are not prompted to provide a username and password.

The web browser supplies the current Windows user's information on the client computer by using a cryptographic exchange that involves hashing with the web server. If the authentication exchange is unsuccessful, the web browser prompts the user for the Windows account credentials.

This feature is targeted at adding HTTP-negotiate between the client browser and the ACOS device, where the ACOS device is a member of an Active Directory (AD) domain, and the client and device communicate with the KDC. By using Kerberos, HTTP-negotiate is supported between the client browser and device.

The following major components are found in a WIA scenario:

- A client, which needs a domain account to use WIA.

  If client the does not login by using a domain account, the browser will not use WIA, even if WIA is enabled.

- An application server, which might be in the AD domain.

  If the server asks the client to authenticate, the user can configure Authentication-Relay on the ACOS device.

- An ACOS device.

NOTE: Although some of the graphics in this chapter show a device, WIA can be used on all ACOS devices.

WIA Workflow Considerations

Depending on the network requirements, the user can configure the following scenarios by using WIA:

NOTE: The user can consider these scenarios in a set up that does (and does not) include Wildcard VIPs.

- The ACOS device is not in the AD domain and uses a Wildcard VIP.

  In this scenario, the ACOS device acts as a gateway and cannot provide WIA service for the client.

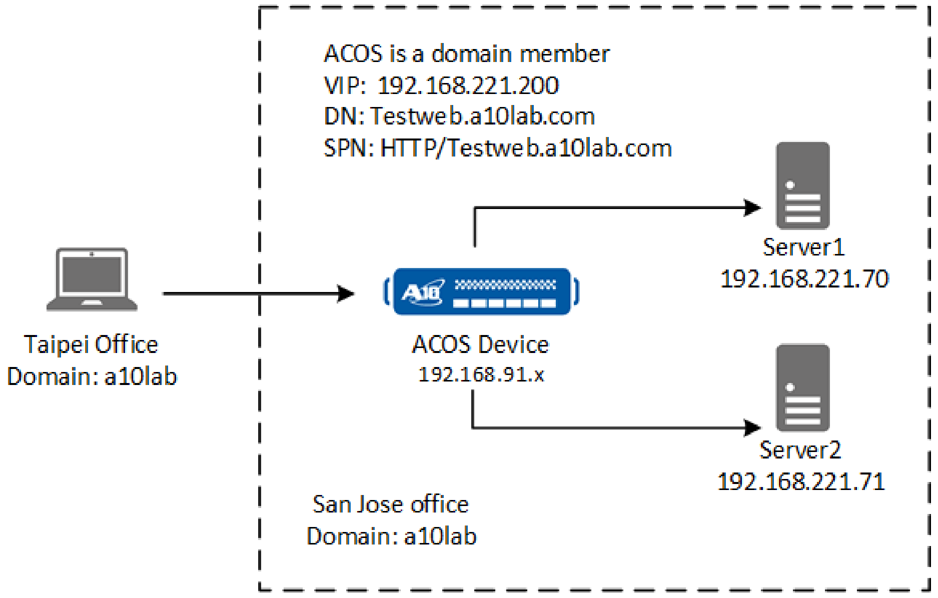
- The ACOS device is not in the AD domain and does not use a Wildcard VIP.

  Here, the ACOS device has no information to complete the authentication for the client, unless the ACOS device has the server’s password (Kerberos).

- The ACOS device is in the AD domain and provides a normal (Non-Wildcard) SLB service.

  In this scenario, the ACOS device is used in an enterprise. It is similar to a normal SLB case, except both client and the ACOS device are the same domain members. The client can use WIA to achieve authentication.

Figure 31 : Sample Topology of a WIA Environment: An ACOS Device in an AD Domain, with Typical SLB Service



This scenario can occur when the ACOS device is used in an enterprise. It is similar to a typical SLB case, except both the client and the ACOS device are the same domain members. In this case, the client can use WIA to achieve authentication.

The following procedure illustrates the workflow in Figure 32:

1. The client sends an HTTP request to *Testweb.a10lab.com*.
2. The ACOS device returns a HTTP 401 packet with *WWW-authenticate: Negotiate* header.
3. The client's browser tries to use WIA.
4. The client's browser requests a service ticket for *HTTP/Testweb.a10lab.com* from AD and sends it to the ACOS device.
5. The ACOS device verifies the ticket and starts to create a back-end connection.

- The ACOS device is in the AD domain and uses a Wildcard VIP.

The ACOS device works as a gateway, and only clients that have a domain account are allowed to access the Internet. The ACOS device can do the WIA with the client since it is in the AD domain. However, in a Wildcard situation, the ACOS device cannot send an HTTP 401 packet directly to the client.

Figure 32 : Sample topology by Using WIA in a Wildcard Setup

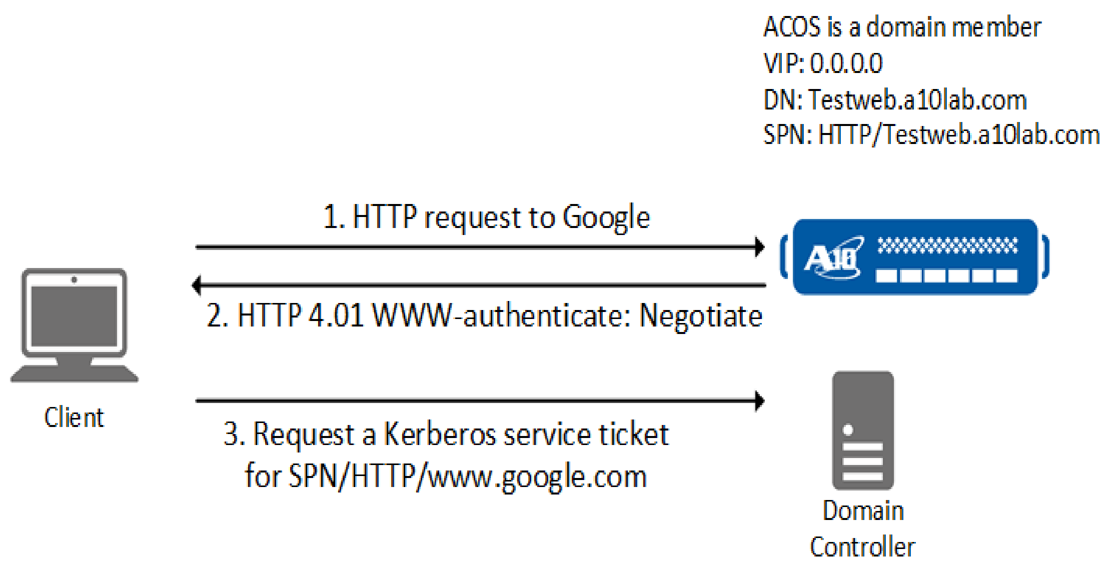

ACOS is a domain member
VIP: 0.0.0.0
DN: Testweb.a10lab.com
SPN: HTTP/Testweb.a10lab.com
1. HTTP request to Google
2. HTTP 4.01 WWW-authenticate: Negotiate
3. Request a Kerberos service ticket for SPN/HTTP/www.google.com
Client
Domain Controller

Figure 33shows a Wildcard case, where a 401 response forces the client to find the appropriate service ticket. However, the client uses a URL to compose a SPN. Since the SPN is not the ACOS device’s SPN, the client gets no feedback or the ACOS device cannot recognize the service ticket. To support WIA in the Wildcard case, the ACOS device must prompt the client to get a service ticket for the ACOS device’s SPN.

Figure 33 : Connections and HTTP-request Protocol in WIA with Wildcard VIP Setup

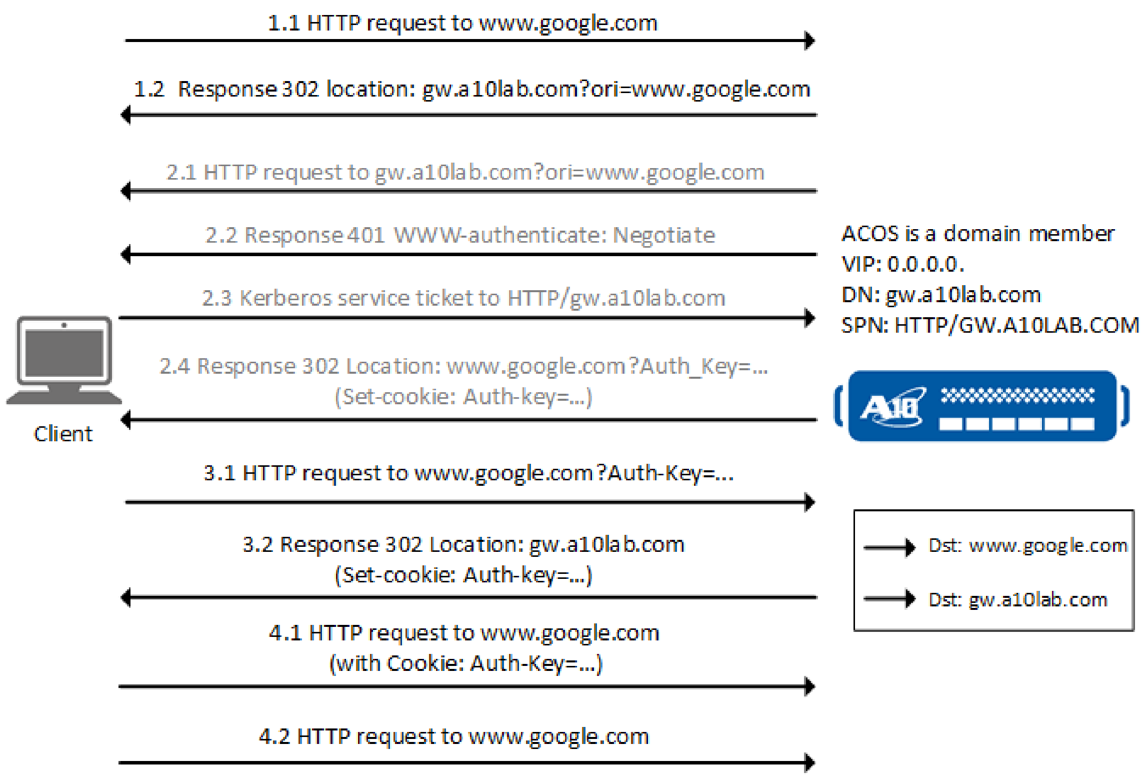

1.1 HTTP request to www.google.com
1.2 Response 302 location: gw.a10lab.com?ori=www.google.com
2.1 HTTP request to gw.a10lab.com?ori=www.google.com
2.2 Response 401 WWW-authenticate: Negotiate
2.3 Kerberos service ticket to HTTP/gw.a10lab.com
2.4 Response 302 Location: www.google.com?Auth_Key=... (Set-cookie: Auth-key=...)
3.1 HTTP request to www.google.com?Auth-Key=...
3.2 Response 302 Location: gw.a10lab.com (Set-cookie: Auth-key=...)
4.1 HTTP request to www.google.com (with Cookie: Auth-Key=...)
4.2 HTTP request to www.google.com
Client
ACOS is a domain member
VIP: 0.0.0.0.
DN: gw.a10lab.com
SPN: HTTP/GW.A10LAB.COM
Dst: www.google.com
Dst: gw.a10lab.com

Figure 31 illustrates the connections that are available to use WIA with a Wildcard VIP:

- In the first connection, the ACOS device receives a new request with no credential (auth-cookie), so the ACOS device makes the client redirect to itself.

  The client uses the ACOS device's SPN to get service ticket, and The original destination must be saved as an argument for future usage.

- In the second connection, the ACOS device finds the destination equal to its SPN so it stores the original destination and sends a 401 packet back to client.

  After verifying the service ticket from the client, the ACOS device generates a cookie to the client. The cookie would be stored in the auth-session table and also sent back to the client in redirecting the packet's argument and in the set-cookie header.

- In the third connection, the client sends a request to its original destination with the auth-cookie as the argument.

The ACOS device sees the cookie, verifies cookie with auth-session table, and sends a 302 packet to the client for setting the auth cookie.

- In the fourth connection, the client sends a request with authentication cookie.

  The ACOS device forwards the client’s request after verifying the cookie.

- Client creates another connection.

  The browser has the information about the authentication cookie for that domain and uses it. The ACOS device can use the cookie to confirm the client’s credential. The ACOS device forwards the request to the server (the fourth connection) and skips connections 1-3.

- Client creates a connection to another domain.

  In the second connection, the client browser accesses *gw.a10lab.com* with an authentication cookie. (The ACOS device sets the cookie in step 2.4). The ACOS device can use the cookie to skip the authentication process (steps 2.2 and 2.3) and speed up the service.

NOTE: For the wildcard case, in an auth-session table, a client might have multiple cookies to different destinations. To reduce the size of the auth-session table, a cookie must be reused in the wildcard scenario. Before generating a new cookie, the ACOS device checks the auth-session table to determine whether a cookie exists with same username and source IP. If the cookie exists, and it can be reused, it is reused.

Configuring a Basic Logon

The user can configure a basic logon by using the GUI or the CLI.

The following topics are covered:

Using the GUI

To configure a basic logon in the GUI mode, perform the following steps:

1. Click **AAM** > **Auth Clients**.
2. On the **logon HTTP Auth** tab, click **Create**.
3. Enter a name.
4. Enter the maximum number of retries.
5. Determine which logon type the user wants to enable.
6. Click **AAM** > **Auth Servers**.
7. On the **LDAP** tab, click **Create**.
8. Enter the relevant information to define the host and realm for the authentication server and click **Create**.
9. Click **AAM** > **Policies and Templates**.
10. On the **AAA Policies** tab, click **Create**.
11. Enter a name.
12. Select an existing virtual port binding or create a new virtual port.
13. Click **Create**.
14. In the **AAA Rules** section, click **Create**.
15. Enter an index number.
16. In Authentication Template, select the template that the user has set up earlier, and click **Create**.
17. In the **Update AAA Policy** page, click **Update**.
18. Click **ADC** > **SLB**. On the **Servers** tab, click **Create**.
19. Enter a name, address type, and IP address.
20. In the **Port** section, click **Create**.Enter the port number and select the protocol. Enter other relevant information as necessary and click **Create** to finish.
21. Expand the **Advanced Fields** section, enter the relevant information, and click **Create**.
22. On the **Service Groups** tab, click **Create**.
23. Enter the relevant information to bind the server that the user created earlier and click **Create**.
24. On the **Virtual Servers** tab, click **Create**.

25. Enter the relevant information, such as the VIP name, IP address, and virtual port information, and click **Create**.
26. In the **Virtual Port** section, click **Create**.
27. Enter the relevant information.
28. Expand the **Advanced Fields** section.
29. Enter the appropriate information and select the AAA policy.
30. Select an existing template or create a new template to bind
31. Click **Create**.

Using the CLI

To configure a basic logon in the CLI mode, perform the following steps:

1. Set up an HTTP-based authentication logon through the AAM module and enable basic authentication:
2. Set up an authentication server (LDAP or RADIUS) and its required settings:
3. Bind this server and logon mechanism to a template:
4. Bind this template to an AAA-policy under the AAM module
5. Configure a policy template and designate the template settings to the appropriate service groups and ports under SLB configuration.

Configuring an Active Directory under the Kerberos Protocol

For more information about Kerberos logon and server configuration, see the following chapter:

- *AAM and Kerberos*

Configuring a Non-Wildcard Setup for WIA

The following topics are covered:

Using the GUI

To configure a Non-Wildcard setup for WIA in the GUI mode, perform the following steps:

1. Begin by creating the Kerberos account by clicking **AAM** > **Accounts** and then clicking **Create**.
2. Fill out the template fields for creating a Kerberos account including a name, SPN domain account, the Kerberos realm, service principal name, and secret string. Click **Create** to finish creating the account
3. Configure the client by clicking **AAM** > **Auth Clients**.
4. On the **logon HTTP Auth** tab, select an existing authentication method or click **Create**.
5. Enter a name for the client logon template, a value for the maximum number of retries, and select the logon types as desired
6. Enter the new PIN page name, PIN variable name, token page and variable page name, and the realm. Click **Create** to finish.
7. To create the authentication server, click **AAM** > **Auth Servers** and select **Create** on the **Windows** tab to pull up the template for the authentication server.
8. Enter a server name, host name, realm, and timeout value. If the user opts to disable the server’s health check, select **Disable Health Check** from the drop-down list.
9. Expand the **Auth Protocol** section and configure the appropriate options. Click **Create** to finish.
10. Click **AAM** > **Policies and Templates**. On the **Authentication Templates** tab, click **Create**.
11. Enter the relevant information and click **Create**.
12. On the **AAA Policy** tab, click **Create**.
13. Enter a name, template type, idle logout time, and the logout URL.

14. Select an existing authentication logon or create a new one.
15. Select an existing authentication relay or create a new one.
16. Select either **Authentication Server or Service Group**

    For account setting, do one of the following:

    a. Select an existing authentication account.

    b. Click **New Account.**
17. Select a logging option, enter a cookie domain and a cookie domain group ID and click **Add**. Click **Create** to finish.
18. To create the authentication server, select **ADC** > **SLB** and navigate to the **Servers** tab. Select **Create** to pull up the template.
19. Enter the server name, type, host name, action, and determine if the user wants to enable or disable the health check. Select a health monitor, connection limit, and if the user opts to not generate logs, select the **No Logging** check box.
20. In the **Port** section, click **Create** to complete the template for the authentication port.
21. Enter a port number, protocol, range, health check option, and connection limit. To prevent logs from being generated, select the **No Logging** check box. Click **Create** to finish.
22. Click **AAM** > **Service Groups** and click **Create**.
23. Enter a name, protocol, and load balancing method. Click **Create** to finish.
24. To create the virtual servers, click **ADC** > **SLB**. On the **Virtual Servers** tab, click **Create** to pull up the virtual server template.
25. Enter the server name, IP address type, Netmask, and action. To use a Wildcard, select the **Wildcard** check box.
26. In the **Virtual Port** section, click **Create** to pull up the template.
27. Expand the **Advanced Fields** section.
28. Select a policy template or click **Add+** to create a new template. Click **Create** to finish.

Using the CLI

To configure a Non-Wildcard setup for WIA in the CLI mode, perform the following steps:

1. Set up an Active Directory account under the AAM module through `kerberos-spn`:

   `aam authentication account kerberos-spn`*`name`*

   ```
   accountusername
   passwordpassword
   ```

2. Configure a logon and specify the logon mechanism:

   NOTE: For a Non-Wildcard setup, the negotiate mechanism must be enabled in the logon. (Kerberos SPNEGO protocol is enabled under `auth-method negotiate`.)

3. Set up a Windows authentication server and its specifications.
4. Set up the WIA authentication template to include the logon credentials and server settings that the user has configured.
5. Bind this template to a policy under the AAA-policy.
6. Set up the appropriate server, service group and VIP to bind the policy and enable the configured template and settings for WIA.

To configure a ***Wildcard*** setup, the process explained above will stay the same except for one additional configuration: the user will have to enable promiscuous VIPs.

reCAPTCHA Challenge for Authentication

The following topics are covered:

Overview

The reCAPTCHA challenge is added to the form-based logon and works with authentication/authorization methods supported by AAM. This challenge-response is used to determine if the user is human or not. This feature supports the usage of third-party CAPTCHA APIs for challenging the user. And the enable CAPTCHA widget is available on the default logon page with the correct type and site-key.

- reCAPTCHAv2-checkbox
- reCAPTCHAv2-invisible
- reCAPTCHAv3

CLI Example

```
aam authentication portal default-portal
logon
enable-CAPTCHA type reCAPTCHAv2-checkbox site-key XXXX
!
```

Default Portal Configuration and Customization

The reCAPTCHA widget can be customized in the following ways:

- **`reCAPTCHA-size`**

  Normal (default) or compact; only available on reCAPTCHAv2-checkbox type.

- **`reCAPTCHA-theme`**

  Light (default) or dark; only available on reCAPTCHAv2-checkbox type.

- **`reCAPTCHA-badge`**

  Bottom-right (default) or bottom-left; only available on reCAPTCHAv2-invisible and reCAPTCHAv3 types.

- **`reCAPTCHA-action`**

  Optional action string to verify the client user, the default value is “A10_DEFAULT_LOGON”; only available on reCAPTCHAv3 type.

Custom Portal Configuration

Import the custom portal configurations which have a CAPTCHA widget and use it with the form-based logon.

CLI example: Set captcha-variable which is the variable name of CAPTCHA token.

```
aam authentication logon form-based hcaptcha_logon
portal sample-custom-portal logon logon-hcaptcha.html failpage
fail.html changepasswordpage cp.html
action-url logon.fo
username-variable user
password-variable pwd
captcha-variable h-captcha-response
changepassword-url cp.fo
changepassword-username-variable cpusr
changepassword-old-password-variable cpold
changepassword-new-password-variable cpnew
changepassword-password-confirm-variable cpcfm
```

Captcha Profile Configuration

The aam authentication captcha profile can be configured in the following ways:

- **`secret-key`**: secret-key used for token verification
- **`url`**: CAPTCHA token verification API
- **`method`**: API method, POST (default), or SET
- **`timeout`**: 1 to 255 seconds, default 10 seconds
- **`secret-key-param-name`**: parameter name used in verification for secret-key
- **`token-param-name`**: parameter name used in verification for token
- **`resp-result-field-name`**: JSON field name used in verification response
- **`resp-error-code-field-name`**: JSON field name used in verification response
- **`send-client-ip`**: send client IP address in verification, default disabled

CLI Example

```
aam authentication captcha recaptchav2-checkbox
secret-key XXXXX
url https://www.google.com/recaptcha/api/siteverify
method POST
timeout 33
secret-key-param-name secret
token-param-name response
resp-result-field-name success
resp-error-code-field-name error-codes
send-client-ip param-name remoteip
!
```

Captcha Vendors Configuration

	Google reCAPTCHA	hCaptcha	MTCaptcha
API	https://www.google.com/recaptcha/api/siteverify	https://hcaptcha.com/siteverify	https://service.mtcaptcha.com/mtcv1/api/checktoken

Method	POST	POST	GET
Secret-key parameter name	secret	secret	privatekey
Token parameter name	response	response	token
JSON response: verification result field name	success	success	success
JSON response:	error-codes	error-codes	fail_codes

error-code field name			
Send client IP address	Support	Not supported	Not supported

Configuring Authentication Template

Bind the CAPTCHA profile in the authentication template for using it with the form-based logon.

CLI example

aam authentication template captcha

logon hcaptcha_logon

server ad

captcha hcaptcha

!

Configuring Authorization Policy

Additional checking of the fields/attributes in the verification JSON response and new attribute type number for decimal numbers.

CLI example:

```
aam authorization policy recaptchav3_score_action
attribute-rule 1 and 2
```

```
attribute 1 score attr-type number more-than 0.5
attribute 2 action attr-type string match ck132_reCAPTCHAv3
!
```

Custom Portal View

- Google CapatchAv2 Checkbox
- hCaptcha
- MTCaptcha

Limitation

- CAPTCHA challenge only works on form-based logon.
- CAPTCHA challenge only works on the logon page.
- The Default-portal logon page only supports Google reCAPTCHA (including reCAPTCHAv2 checkbox/invisible and reCAPTCHAv3).
- CAPTCHA challenge may have an issue when working with RADIUS MFA (Multi-Factor Authentication.
- CAPTCHA challenge cannot work with EP/TP.
- Maximum CAPTCHA API URL length is 127.
- Maximum reCAPTCA site-key length is 63.
- Maximum CAPTCHA secret-key length is 127.
- Maximum CAPTCHA token length is 511.

AAM with LDAP

Lightweight Directory Access Protocol (LDAP) is an application protocol that is used to access and manage user information in a directory service, such as OpenLDAP, Microsoft Active Directory or OpenDL. ACOS supports OpenLDAP and Microsoft Active Directory. When a user attempts to access a secured web application or web service, ACOS communicates with the LDAP directory server to authenticate the user accessing the application/service. Users associated with the LDAP directory server can update their password. LDAP password update is one of the solutions that is provided by Application Access Management (AAM).

The following topics are covered:

Overview

The AAM Authentication Proxy features can use Basic-HTTP authentication or web-based forms to obtain end-user credentials, and query back-end LDAP servers to verify the credentials. If a valid end-user's password has expired, the Login Proxy enables the end-user to change the password and regain access.

Clients do not need to understand LDAP. Instead, when a client sends a request to a VIP for a secured service, ACOS uses Basic-HTTP authentication or a web form to obtain end-user credentials. ACOS then sends the credentials to an LDAP server for verification.

- If the client is authenticated by the LDAP server, ACOS allows the client to access the requested service.
- If the LDAP server returns an error message instead, ACOS does not immediately deny the end-user's request. Instead, ACOS logs onto the LDAP server as an account administrator. ACOS queries the account database for the password entered by the end-user, and the password's expiration. ACOS uses this information to determine whether the password, although currently invalid, used to be valid but has expired.
- If the password is valid but has expired, ACOS sends a web form to the end-user, to allow the password change.
- If the password is invalid, ACOS denies the request.

Guidelines for LDAP Authentication Using Active Directory

- The users associated with the LDAP Active Directory can change their password only through the "logon Form-Based" authentication type. Other logon authentication types are not supported.
- Before the expiration of a password, the user is notified with the number of days remaining for the password to expire.
- If the password retry limit is configured for a user account, the user with the same name but in different domains are treated as the same account. For example, if

"Bill" is the username in example1.com and example2.com domains, Bill will be treated as one user account.

- If the Active Directory is configured with the default "Minimum password age" under Account Policies > Password Policy, the user cannot change the password for the duration specified in the policy. To enable password change for users, the administrator can set the "Minimum password age" to zero (0).

NOTE: LDAP over SSL (LDAPS) MUST be enabled for the LDAP Directory server, as Windows require the client to have SSL-TLS encrypted connection to modify the password attribute.

The following topics are covered:

Configuring LDAP over SSL

To configure LDAP over SSL (LDAPS) for the LDAP Directory server, navigate to the AAM > Auth Servers > LDAP page. The default port number for LDAP over SSL is 636. Ensure that the user configures the port number after setting the Protocol to LDAPS or StartTLS.

- Click Create to add a new LDAP server with Protocol set to LDAPS or StartTLS.
- To modify an existing LDAP server, click Edit next to the LDAP server and set the Protocol to LDAPS or StartTLS.

Password Change can be initiated by the:

1. Active Directory
   a. When the password has expired.
      OR
   b. When the password is expiring in N days (if prompt-password-change-before-expiration is configured for the user account).
2. User

The user can change the password by using the “Change Password” link on the logon page.

Password Expiration Notification

When the “**`prompt-password-change-before-expiration`**” is configured and the authentication type is Active Directory (AD), ACOS notifies the user if the password is expiring. ACOS displays the change password notification page with the number of days for the password to expire and prompts the user to change the password. The user can change the password by clicking the “Change Password” link, or continue accessing the application without changing the password. By default, ACOS displays the default change password notification page to the users. If the user wants to display the custom change password notification URL/page, do the following:

Import the custom logon form to ACOS by using the “import auth-portal <Name of the custom logon form>” CLI command.

Configure the imported custom logon form using the “**`notifychangepassword-change-url`**” and “**`notifychangepassword-continue-url`**” CLI commands.

After the above configuration, ACOS displays the custom change password notification page to the users.

Custom Notification Password Change Page

The user can use JavaScript to submit "**`Custom Logon Form`**" to ACOS based on the user’s action.

If the user clicks "**`Continue`**" (that is, continue accessing the application without changing the password), the “`/custom_continue.fo`” action is triggered.

If the user clicks "**`Change`**" (that is, the user confirms to change the password), “`/custom_notify_change.fo`” action is triggered.

The custom notification password change page must be configured with these parameters before importing the Custom Logon Form to ACOS.

The following is an example of a Custom Notification Password Change page that includes `/custom_continue.fo` and `/custom_notify_change.fo parameters`:

Example

```
<html>
<script type="text/javascript">
function submitForm(action)
{
  document.getElementById('notify_change').action = action;
  document.getElementById('notify_change').submit();
}
</script>
....
<form id="notify_change" name="notify_change" method="POST" action="">
<!-- continue-action-url -->
<!-- continue-text -->
<input type="button" onclick="submitForm('/custom_continue.fo')"
value="Continue">
<!-- change-action-url -->
<!-- change-text -->
<input type="button" onclick="submitForm('/custom_notify_change.fo')"
value="Change">
</p>
…
</form>
```

Password Retry for User Accounts

ACOS enables the administrator to configure the retry option to the user accounts. If the user fails to authenticate within the configured number of retries, the account will be locked for the specified duration.

Example

Consider user "Bill" is configured with 3 retry entries and the account lock duration is set to 10 minutes. Also, consider https://www.abc.com is the secured web application. When Bill attempts to access the secured web application, ACOS identifies the user needs to be authenticated to access the application and challenges the user to provide the credentials for authentication. Bill fails to authenticate, ACOS provides three (3) retry chances. After the failed 3 retry chances, Bill's account will be locked for 10 minutes.

Logon Proxy

The following topics are covered:

Form-Based Login with LDAP

To obtain end-user credentials, the login proxy supports both form-based and http-basic authentication. The credentials are verified by using a back-end LDAP server.

The typical login name credential is used as the first value of the distinguished name (DN). By default, the first attribute of the DN is the Common Name (CN). The end-user object’s full DN is constructed by prepending the base DN to the first attribute.

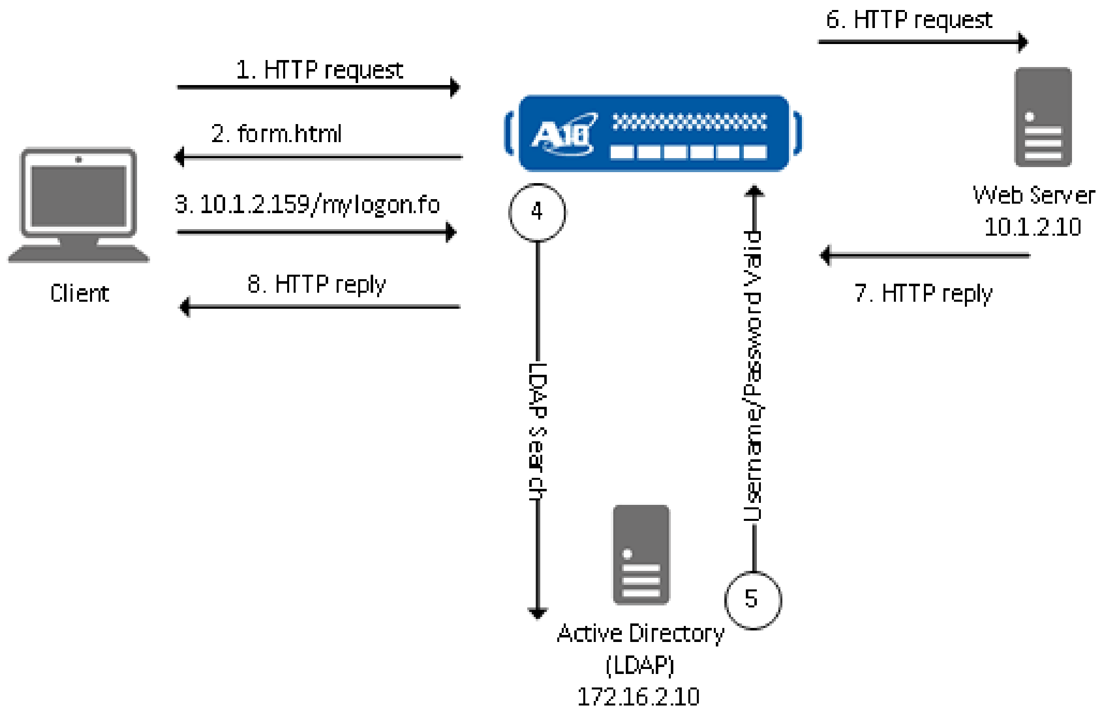
Because the end-user object’s CN might not correspond to the login name that is used in Active Directory (AD), the following alternate LDAP bin login names are also accepted:

- username@domain.com
- Domain\username

When an end-user enters a username in one of these formats, ACOS accepts and uses this format instead of the bind DN form.

Figure 34 shows an example deployment of this solution.

Figure 34 : Form-Based Login with LDAP



Traffic Walkthrough (A Valid Username and Password)

1. The client sends an initial HTTP request to VIP.
2. ACOS replies with a form that contains the username and password fields.
3. The client browser displays the form and end-users enter their username and password.
4. ACOS extracts the username and password from the form and sends the credentials to the LDAP server in a Search request.
5. The LDAP server replies.

   In this example, the username and password exist in the LDAP server's user database. ACOS caches the reply, so that the cached verification of credentials can be used for the next request from the same end-user.
6. ACOS uses SLB to select a server from the Web-server service group, and sends the client's HTTP request to the server. (This example assumes SLB is used.)
7. The server replies.

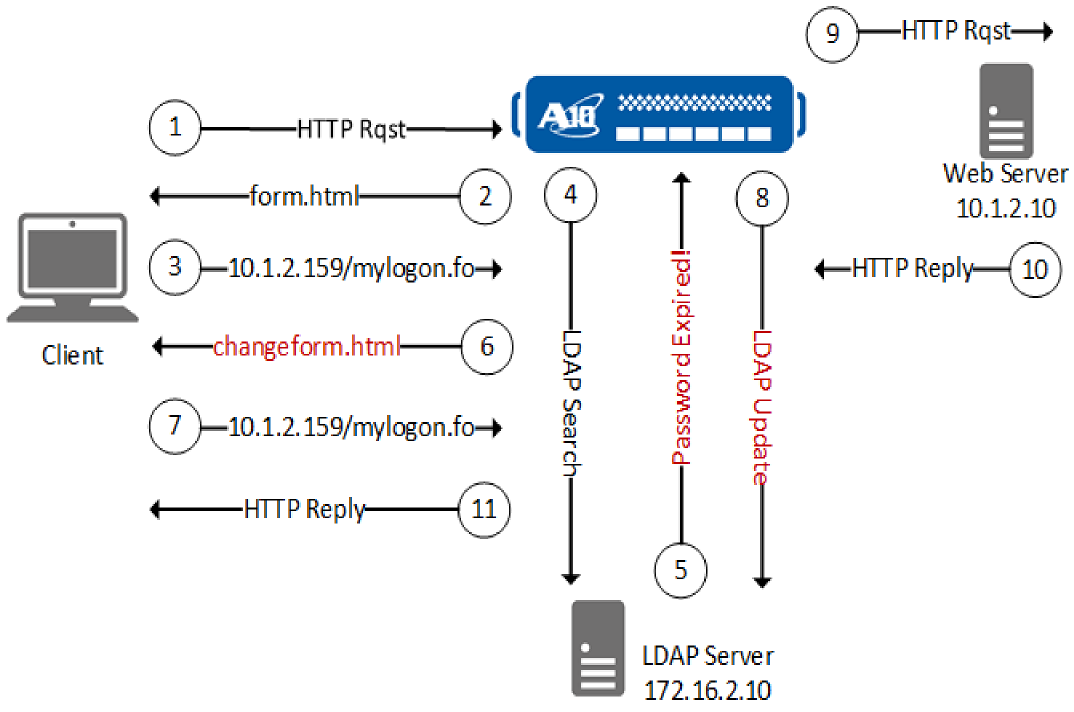
In this example, the server has the requested content and sends it in the reply.

8. ACOS forwards the server reply to the client.

Traffic Walkthrough (A Valid Username and an Expired Password)

This example shows the traffic flow when the username that is entered by an end-user is valid, but the password has expired.

Figure 35 : Form-based logon with LDAP (Change Password)



1. The client sends initial HTTP request to VIP.
2. ACOS replies with a form that contains fields for the user credentials (username and password).
3. The client browser displays the form in which users enter their username and password.
4. ACOS extracts the username and password from the form, and sends the credentials to the LDAP server in a Search request.

5. The LDAP server finds an entry for the username and password, but the password has expired.
6. ACOS sends another form to the client, which contains input fields to change the password.
7. The user enters a username, the old password, and the new password.
8. ACOS sends the updated password to the LDAP server.
9. After the client is authenticated, ACOS uses SLB to select a server, and sends the client's HTTP request to the server.
10. The server replies.

    In this example, the server has the requested content and sends it in the reply.
11. ACOS caches the credential verification from the LDAP server, and forwards the server reply to the client.

Error Codes

The following table lists all the error codes as applicable.

Error Code	**Error Message**	**Description**
Code 49 (Invalid Credentials)	`ERROR_ LOGON_ FAILURE`	Unknown username or bad password.
Code 49 (Invalid Credentials)	`ERROR_ INVALID_ LOGON_HOURS`	Account logon time restriction violation.
Code 49 (Invalid Credentials)	`ERROR_ INVALID_ WORKSTATION`	User not allowed to logon to this computer.
Code 49 (Invalid Credentials)	`ERROR_ ACCOUNT_ DISABLED`	Account has been disabled. Contact your administrator.

Error Code	Error Message	Description
Code 49 (Invalid Credentials)	`ERROR_ WRONG_ LOGON_TYPE`	The user has not been granted the requested logon type at this machine.
Code 49 (Invalid Credentials)	`ERROR_ ACCOUNT_ LOCKED_OUT`	The referenced account is currently locked out and may not be logged on to.
Code 17 (Constraint Violation)	NA	Failed to change the password due to one of following reasons: • Your new password doesn't meet complexity requirements or is too similar to a password previously used. • Your new password can't be changed because it was recently changed. • Your new password has been used before.
Code 53 (Unwilling To Perform)	NA	“The server refused to change your password. Please contact your administrator.”

Creating an LDAP Authentication Server

NOTE: To create an LDAP authentication server by using the GUI or the CLI, see Creating Authentication Servers.

LDAP Name Attribute

When LDAP is used as the authentication server, ACOS acts as a client and tries to bind to the LDAP server by using the base distinguished name (DN) that is specified in the configuration and the user’s username. The user can specify which name attribute is used when ACOS issues a bind request to the LDAP server. If no attribute is specified, the default common name (CN) will be used.

For example, if the user enters the following information:

- Name Attribute: `krbContainer`
- Base DN: `dc=example, dc=com`

The bind request that is sent to the LDAP server looks like this:

```
krbContainer = username,dc=example,dc=com
```

NOTE: The DN attribute can have a maximum of 32 characters.

Configuring the Name Attribute

The following topics are covered:

Using the GUI

The user can configure the name attribute by using the GUI or the CLI.

To configure the name attribute for a new LDAP server:

1. Click **AAM** > **Auth Servers**.
2. On the **LDAP** tab, click **Create**.
3. In **DN Attribute**, enter the name attribute type.
4. Click **Create**.

Feedback

Using the CLI

Configuration Process

The following topics are covered:

- LDAP Authentication Server
- DN Attribute
- DN Attribute Modifications
- Confirmation of the Configuration
- Unconfigure the DN Attribute
- Selection of the Protocol
- CA Certificate for LDAPS
- LDAPS Connection Idle Timeout
- LDAPS Configuration

The following example shows how to configure an LDAP authentication server, and then a dn-attribute.

LDAP Authentication Server

First, configure the LDAP authentication server and verify that it is running:

```
ACOS(config)# aam authentication server ldap test
ACOS(config-ldap auth server:test)# show aam authentication server ldap
test
LDAP Authentication Server
Name                                    test            ldap
Type                                    ldap
Host
Base DN
Admin DN
Admin Secret                                            389
Port                                    389
Password Expiration Time                0
Timeout                                 10
DN attribute name                       cn
```

```
Default domain name                                    No
Bind with DN                                   No
Username attribute name
```

DN Attribute

Use the **dn-attribute** command to configure the DN attribute:

```
ACOS(config-ldap auth server:test)# dn-attribute test
ACOS(config-ldap auth server:test)# show aam authentication server ldap
test
LDAP Authentication Server
Name                                           test            ldap
Type                                           ldap
Host
Base DN
Admin DN
Admin Secret                                                   389
Port                                           389
Password Expiration Time                       0
Timeout                                        10
DN attribute name                              test
Default domain name                                            No
Bind with DN                                   No
Username attribute name
```

DN Attribute Modifications

If the user wants to change the DN attribute, issue the same command with the desired name to overwrite the previous one:

```
ACOS(config-ldap auth server:test)# dn-attribute uid
ACOS(config-ldap auth server:test)# show aam authentication server ldap
test
LDAP Authentication Server
Name                                           test            ldap
Type                                           ldap
Host
Base DN
Admin DN
Admin Secret                                                   389
```

```
Port                                    389
Password Expiration Time                0
Timeout                                 10
DN attribute name                       uid
Default domain name                                     No
Bind with DN                            No
Username attribute name
```

Confirmation of the Configuration

To confirm and view the user's configuration:

```
ACOS(config-ldap auth server:test)# show running-config aam authentication
server ldap
!Section configuration: 62 bytes
!
aam authentication server ldap test
  dn-attribute test
!
!
```

Unconfigure the DN Attribute

The following example shows how to unconfigure the DN attribute:

```
ACOS(config-ldap auth server:test)# no dn-attribute uid
ACOS(config-ldap auth server:test)# show aam authentication server ldap
test
LDAP Authentication Server
Name                                    test            ldap
Type                                    ldap
Host
Base DN
Admin DN
Admin Secret                                            389
Port                                    389
Password Expiration Time                0
Timeout                                 10
DN attribute name                       cn
Default domain name                                     No
Bind with DN                            No
```

```
Username attribute name
ACOS(config-ldap auth server:test)# show running-config aam authentication
server ldap
!Section configuration: 41 bytes
!
aam authentication server ldap test
!
!
```

NOTE: When unconfigured, the DN attribute changes back to the default value of "**CN**."

Selection of the Protocol

The command to select the protocol:

```
ACOS(config)(NOLICENSE)#aam authentication server ldap test1
ACOS(config-ldap auth server:test1)(NOLICENSE)#protocol ?
  ldap      Use LDAP (default)
ldaps     Use LDAP over SSL
starttls  Use LDAP StartTLS
ACOS(config-ldap auth server:test1)(NOLICENSE)#protocol ldaps
The default port number of LDAP over SSL is 636; please configure port
number if needed
ACOS(config-ldap auth server:test1)(NOLICENSE)#show run aam authentication
server ldap
!Section configuration: 60 bytes
!
!
aam authentication server ldap test1
  protocol ldaps
```

CA Certificate for LDAPS

The command to configure CA certificate for LDAPS:

```
ACOS(config-ldap auth server:test1)# ca-cert ?
NAME<length:1-245>  Trusted LDAPS CA cert filename
ACOS(config-ldap auth server:test1)#ca-cert a10-tplab_rootca_b64.cer ?
  <cr>
ACOS(config-ldap auth server:test1)#ca-cert a10-tplab_rootca_b64.cer
ACOS(config-ldap auth server:tttt)#show run aam authentication server ldap
!Section configuration: 112 bytes
!
!
aam authentication server ldap test1
  ca-cert a10-tplab_rootca_b64.cer
!
ACOS(config-ldap auth server:test1)#!
```

LDAPS Connection Idle Timeout

The command to configure LDAPS connection idle timeout for connection reuse:

```
ACOS(config-ldap auth server:test1)#ldaps-conn-reuse-idle-timeout ?
<0-86400>  Specify idle timeout value (in seconds), default is 0 (not
reuse LDAPS connection)
ACOS(config-ldap auth server:test1)#ldaps-conn-reuse-idle-timeout 600
ACOS(config-ldap auth server:test1)#show run aam authentication server
ldap | sec test1
aam authentication server ldap test1
  ldaps-conn-reuse-idle-timeout 600
ACOS(config-ldap auth server:test1)#
```

LDAPS Configuration

Example of a CLI output that displays LDAPS configuration:

```
ACOS#show run aam authentication server ldap
!Section configuration: 328 bytes
!
!
aam authentication server ldap ldap
  host 1.1.1.1
!
aam authentication server ldap ldaps-ssl
  host 3.3.3.3
  port 636
  protocol ldaps
  ca-cert a10-tplab_rootca_b64.cer
!
aam authentication server ldap ldaps-starttls
```

```
  host 2.2.2.2
  protocol starttls
  ca-cert a10-tplab_rootca_b64.cer
  ldaps-conn-reuse-idle-timeout 600
!
ACOS#show aam authentication server ldap
LDAP Authentication Server
Name                                    ldap
Type                                    ldap
Host                                    1.1.1.1
Base DN
Admin DN
Admin Secret
Protocol                                LDAP
Server Health Monitor
Port                                    389
Port Health Monitor
Password Expiration Time                0
Timeout                                 10
DN Attribute Name                       cn
Default Domain Name
Bind with DN                            No
Username Attribute Name
CA certificate
LDAPS Connection Reuse Idle Timeout     0
---------------------------------------------------------------------------
------
Name                                    ldaps-ssl
Type                                    ldap
Host                                    3.3.3.3
Base DN
Admin DN
Admin Secret
Protocol                                LDAP over SSL
Server Health Monitor
Port                                    636
Port Health Monitor
Password Expiration Time                0
Timeout                                 10
DN Attribute Name                       cn
```

```
Default Domain Name
Bind with DN                            No
Username Attribute Name
CA certificate                          a10-tplab_rootca_b64.cer
LDAPS Connection Reuse Idle Timeout     0
---------------------------------------------------------------------------
------
Name                                    ldaps-starttls
Type                                    ldap
Host                                    2.2.2.2
Base DN
Admin DN
Admin Secret
Protocol                                LDAP StartTLS
Server Health Monitor
Port                                    389
Port Health Monitor
Password Expiration Time                0
Timeout                                 10
DN Attribute Name                       cn
Default Domain Name
Bind with DN                            No
Username Attribute Name
CA certificate                          a10-tplab_rootca_b64.cer
LDAPS Connection Reuse Idle Timeout     600
ACOS#
```

Configuration

The following topics are covered:

- Resources
- CLI Examples

Resources

The deployment requires the following resources:

- Zip archive of the web portal files required for end-user logon
- Logon-portal profile
- Authentication-server profile for the back-end LDAP server
- Health monitor, server configuration, and service group for the back-end LDAP server

NOTE: For more information about LDAP health monitoring, see LDAP Health Monitoring.

- Authentication template containing the service group for the authentication server, and the authentication-logon profile
- Server configuration and service group for the application server
- VIP configuration

CLI Examples

The following topics are covered:

- Logon-Portal Files
- Logon-Portal Profile
- Authentication-Server Profile for the LDAP Server
- LDAP Health Monitor
- SLB Server Configuration for the LDAP Server
- Authentication Template
- SLB Configuration

The following are the examples of various commands used for configuration.

Logon-Portal Files

The following commands import the logon-portal files to the ACOS device:

```
ACOS(config)# import auth-portal portal.zip use-mgmt-port sftp:
Address or name of remote host []?fileserver1
Username []?admin1
Password []?********
```

```
File name [/]?portal.zip
...
```

Logon-Portal Profile

The following commands configure the logon-portal profile.

```
ACOS(config)# aam authentication logon
ACOS(config)# authentication form-based f1
ACOS(config-form-based config-form-based auth logon:f1)# portal portal.zip
logon form.html failpage error.html changepasswordpage changeform.html
ACOS(config-form-based config-form-based auth logon:f1)# action-url
/mylogon.fo
ACOS(config-form-based config-form-based auth logon:f1)# username-variable
username
ACOS(config-form-based config-form-based auth logon:f1)# password-variable
pwd
ACOS(config-form-based config-form-based auth logon:f1)# exit
ACOS(config)#
```

Authentication-Server Profile for the LDAP Server

The following commands create an authentication-server profile for the LDAP server:

```
ACOS(config)# aam authentication server ldap ldap11
ACOS(config-ldap auth server:ldap11)# host 172.16.2.10
ACOS(config-ldap auth server:ldap11)# base cn=Users,dc=umin,dc=com
```

LDAP Health Monitor

The following commands configure an LDAP health monitor:

```
ACOS(config-ldap server)# exit
ACOS(config)# health monitor ldap-sr
ACOS(config-health:monitor)# method ldap run-search BaseDN
dc=a10networks,dc=com query (objectclass=*) AcceptNotFound
ACOS(config-health:monitor)# exit
```

SLB Server Configuration for the LDAP Server

The following commands create an SLB server configuration for the LDAP server, and add it to a service group.

```
ACOS(config)# ?
ACOS(config)# slb server ldap-sr 172.16.2.10
ACOS(config-real server)# port 389 tcp
ACOS(config-real server-node port)# health-check ldap-sr
ACOS(config-real server-node port)# exit
ACOS(config-real server)# exit
ACOS(config)# aam authentication server
ACOS(config-real server-node port)# aam authentication server ldap ldapl1
ACOS(config)# exit
ACOS(config)# slb service-group sg tcp
ACOS(config-slb svc group)# member ldap-sr 389
ACOS(config-slb svc group-member:389)# exit
ACOS(config-slb svc group)# exit
ACOS(config)# ?
```

Authentication Template

The following commands configure the authentication template:

```
ACOS(config)#
ACOS(config)# aam authentication template t1
ACOS(config-auth template:t1)# logon f1
ACOS(config-auth template:t1)# service-group sg
ACOS(config-auth template:t1)# exit
ACOS(config)#
```

The **`service-group`** command binds the template to the service group for the LDAP server. If the configuration did not use a health monitor, server configuration, and service group for LDAP health checking, use the **`server`** command to bind the authentication template directly to the authentication-server profile.

SLB Configuration

The following commands add the SLB configuration.

```
ACOS(config)# slb server rs_http 10.1.2.10
ACOS(config-real server)# port 80 tcp
ACOS(config-real server-node port)# no health-check monitor_name
ACOS(config-real server-node port)# exit
ACOS(config-real server)# exit
ACOS(config)# slb service-group http_g_1 tcp
ACOS(config-slb svc group)# member rs_http 80
ACOS(config-slb svc group-member:80)# exit
ACOS(config-slb svc group)# exit
ACOS(config)# slb virtual-server vip_auth 10.1.2.159
ACOS(config-slb vserver)# port 80 http
ACOS(config-slb vserver-vport)# service-group http_g_1
ACOS(config-slb vserver-vport)# exit
ACOS(config-slb vserver)# exit
ACOS(config)# aam authentication template t1
```

Authentication Relay with LDAP

The following topics are covered:

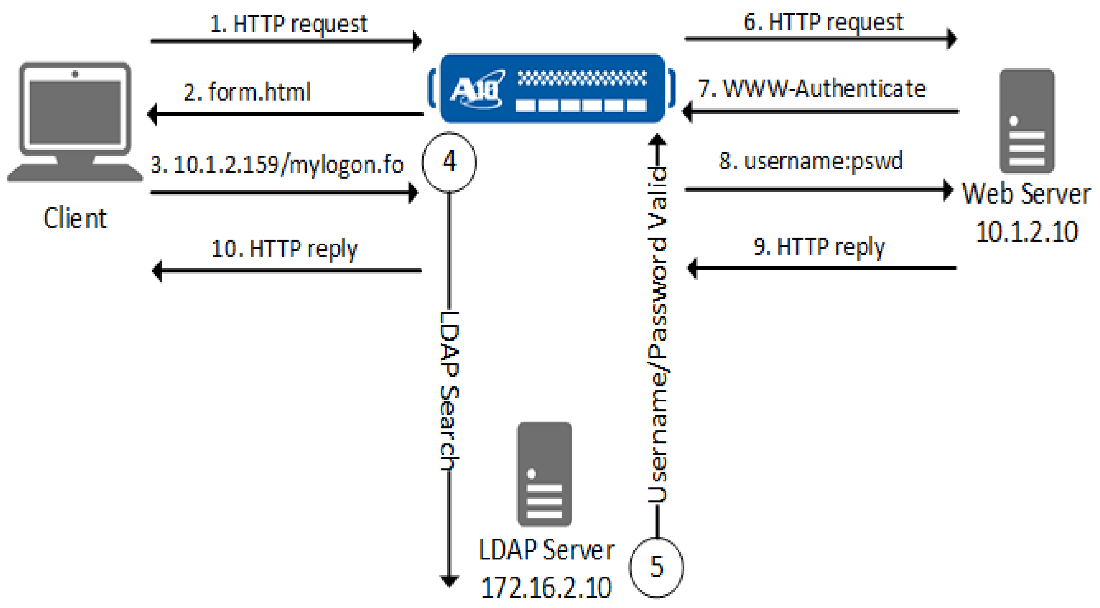
Form-Based Logon with LDAP and Authentication Relay

This AAM solution uses a backend LDAP server to verify an end-user's credentials when they first logon, then reuses those credentials to logonto load-balanced content servers on behalf of that end-user.

In this deployment, the content servers do not need to understand LDAP. Instead, Basic-HTTP authentication is used between ACOS and the servers.

Figure 36 shows an example.

Figure 36 : Form-Based Logon with LDAP and Authentication Relay



Traffic Walkthrough (A Username and a Valid Password)

1. The client sends an initial HTTP request to the VIP.
2. ACOS replies with a form that contains input fields for the user credentials (username and password).
3. The client browser displays the form into which the user enters the credentials.
4. ACOS extracts the username and password from the form, and sends them to the LDAP server in a Search request.
5. The LDAP server replies. In this example, the username and password are present in the LDAP server’s user database.
6. ACOS uses SLB to select a server, then sends the client’s HTTP request to the server.
7. The server replies with an HTTP 401 (Not Authorized) message with response code 4, containing an Authentication header such as the following:
```
WWW-Authenticate: Basic realm=”realm-name”
```
8. ACOS sends an HTTP reply that includes the username and password, in Base64-encoded form, in an Authorization header:
```
Authorization: Basic QTEwOlRodW5kZXI=
```

9. If the username and password are valid, the server replies with the requested content.
10. ACOS caches the credential verification from the LDAP server, and forwards the server reply to the client.

Configuration

The deployment requires the following resources:

- Zip archive of the web portal files required for end-user login
- Login-portal profile
- Authentication-server profile for the backend LDAP server
- Health monitor, server configuration, and service group for the backend LDAP server
- Authentication-logon profile for form-based collection of end-user credentials
- Authentication-relay profile for sending end-user credentials to content servers
- Authentication template containing the authentication-server profile and authentication-login profile
- Server configuration and service group for the application server
- VIP configuration

CLI Example

The following commands import the login-portal files onto the ACOS device:

```
ACOS(config)# import auth-portal portal.zip use-mgmt-port sftp:
Address or name of remote host []?fileserver1
Username []?admin1
Password []?********
File name [/]?portal.zip
...
```

The following commands configure the logon-portal profile. This is the same logon-portal profile used in Logon Proxy.

```
ACOS(config)# aam authentication logon form-based f1
ACOS(config-form-based auth logon:f1)# portal portal.zip logon form.html
failpage error.html changepasswordpage changeform.html
ACOS(config-form-based auth logon:f1)# action-url /mylogon.fo
ACOS(config-form-based auth logon:f1)# username-variable username
ACOS(config-form-based auth logon:f1)# password-variable pwd
ACOS(config-form-based auth logon:f1)# exit
```

The following commands create an authentication-server profile for the LDAP server:

```
ACOS(config)# aam authentication server ldap ldapl1
ACOS(config-ldap auth server:ldap11)# host 172.16.2.10
ACOS(config-ldap auth server:ldap11)# base cn=Users,dc=umin,dc=com
```

The following commands create the SLB configuration for the LDAP server. This is the same configuration used in Logon Proxy.

```
ACOS(config-ldap auth server:ldap11)# exit
ACOS(config)# health monitor ldap-sr
ACOS(config-health:monitor)# method ldap run-search BaseDN
dc=a10networks,dc=com query (objectclass=*) AcceptNotFound
ACOS(config-health:monitor)# exit
ACOS(config)# slb server ldap-sr 172.16.2.10
ACOS(config-real server)# port 389 tcp
ACOS(config-real server-node port)# health-check ldap-sr
ACOS(config-real server-node port)# exit
ACOS(config-real server)# exit
ACOS(config)# aam authentication server ldap ldapl1
ACOS(config-ldap auth server:ldap11)# exit
ACOS(config)# slb service-group sg tcp
ACOS(config-slb svc group)# member ldap-sr 389
ACOS((config-slb svc group-member:389)# exit
ACOS(config-slb svc group)# exit
```

The following commands configure the logon-portal profile:

```
ACOS(config)# aam authentication logon form-based f1
ACOS(config-form-based auth logon:f1)# portal portal.zip logon form.html
failpage error.html changepasswordpage changeform.html
ACOS(config-form-based auth logon:f1)# action-url /mylogon.fo
ACOS(config-form-based auth logon:f1)# username-variable username
ACOS(config-form-based auth logon:f1)# password-variable pwd
ACOS(config-form-based auth logon:f1)# exit
```

The following command creates an HTTP-basic logon profile, and specify the AAA realm name:

```
ACOS(config)# aam authentication relay http-basic r1
ACOS(config-http basic auth relay:r1)# exit
```

The following commands configure the authentication template:

```
ACOS(config)# aam authentication template t1
ACOS(config-auth template:t1)# logon f1
ACOS(config-auth template:t1)# service-group sg tcp
ACOS(config-auth template:t1)# exit
ACOS(config)# aam authentication relay http-basic r1
ACOS(config-http basic auth relay:r1)# exit
```

The following commands add the SLB configuration. This is the same configuration used in the other AAM examples in this topic.

```
ACOS(config)# slb server rs_http 10.1.2.10
ACOS(config-real server)# port 80 tcp
ACOS(config-real server-node port)# no health-check monitor_name
ACOS(config-real server-node port)# exit
ACOS(config-real server)# exit
ACOS(config)# slb service-group http_g_1 tcp
ACOS(config-slb svc group)# member rs_http 80
ACOS(config-slb svc group-member:80)# exit
ACOS(config-slb svc group)# exit
ACOS(config)# slb virtual-server vip_auth 10.1.2.159
ACOS(config-slb vserver)# port 80 http
ACOS(config-slb vserver-vport)# service-group http_g_1
ACOS(config-slb vserver-vport)# template authentication t1
```

Binding to the Distinguished Name

To bind to the LDAP server as an active directory, the ACOS client expects that usernames are entered as a common name (CN), where the format is Last Name + First Name.

An example of this format is Smith John. When LDAP is an Active Directory (AD), the ACOS device cannot bind, because the distinguished name (DN) is not correct.

Starting with release 4.0, the user can configure the ACOS device to do an LDAP search and get the value of the CN. This process allows a DN to be created that can be used to bind to the LDAP server.

The following topics are covered:

Using the GUI

To bind to a distinguished name when LDAP is an AD:

1. Click **AAM** > **Auth Servers**.
2. On the **LDAP** tab, select an LDAP server, and click **Edit**.
3. Select the **Bind with DN** check box.
4. In **Default Domain**, enter the domain.

The ACOS device uses the *default-domain\username* format of the domain name as login name in LDAP bind. The *default-domain* is the domain to which the user's username belongs, and the *username* is entered by the user to login to the ACOS device.

Using the CLI

The following topics are covered:

Binding Steps

To bind to the distinguished name when LDAP is an AD:

1. To access the user's LDAP server, enter the following command:

   ```
   aam authentication server ldap ldap_server_name
   ```

2. To configure the default domain, enter the following command:

   ```
   [no] default-domaindomain_name
   ```

   NOTE: This domain is not entered in the login form.

   This command allows the user to configure default domain. By default, this value is not set.

   The ACOS devices uses the following format for the domain name as login name in LDAP bind:

   default-domain\username

3. To use the DN as login name, enter the following command:

   ```
   [no] bind-with-dn
   ```

   ACOS will ignore the form of the username and use the login name to create the DN. By default, this option is disabled.

4. To configure the attribute name of username, enter the following command:

   ```
   [no] derive-bind-dnusername-attrattr_name
   ```

   The **`username-attr`** command allows the user to specify an attribute that is used as the filter in the LDAP search. *`attr_name`* is the attribute name that the user wants to use, for example, **`uid`**.

ACOS uses the attribute name as the filter in LDAP search to get the CN value, and use this value to create the DN. By default, the attribute is not set.

CLI Example

```
ACOS(config)# aam authentication server ldap svr jsmith

ACOS(config-ldap auth server:svr)# derive-bind-dn username-attr
```

NOTE:

- The maximum length of the username field in login form is 256 characters.
- The maximum length of the default domain is 63 characters.
- Maximum length of the attribute name of username is 31 characters.

About the Work-flow

The following procedure is a high-level overview of this process:

1. Before creating the DN, ACOS checks whether the **`derive-bind-dn`** command has been configured.

   This command allows the user to use the DN as the login name.

2. If the DN has been configured to be used as the login name, ACOS binds the DN as admin first.
3. After the binding is complete, ACOS uses the *`user_attribute_name`* and the *`dn_attribute`* values as the filters to search entire subtree of the LDAP search base.

   `dn_attribute` is the attribute name that is used in the DN, and could be configured in *`dn-attribute`* when the user enters the **`authentication-server ldap`** command. If the search is successful, ACOS locates an object with the following configuration:

   `user_attribute-name = username`

4. The value of *`dn_attribute`* is used to create DN.

Example for the Work-flow

The following procedure is an example of this process:

1. Configure the user’s LDAP server as appropriate for the environment.
2. Enter **`jsmith`** as the username in the login form.
3. Since **`derive-bind-dn`** is configured, ACOS binds as admin first.
4. After this binding is complete, ACOS searches the subtree of LDAP base.

Since `dn-attribute` is not set, the search filter will be the default values of `account` and `cn`.

5. ACOS receives the LDAP search response and finds an object whose `account` is `jsmith`.
6. ACOS gets the value of `cn`, which is `John Smith`.

   As a result, the DN is `cn=John Smith,dc=a10lab,dc=com`.
7. In this example, if user configures `dn-attribute` with the `uid` option, the search filter is `account` and `uid`, and DN is `uid=John Smith,dc=a10lab,dc=com`.

Notes

The following are the important points:

- If ACOS finds more than one object, ACOS uses the value in the first object.
- If ACOS fails to find the corresponding object, the authentication procedure fails.
- If the `derive-bind-dn` feature is not configured, ACOS acts as in the phase 2.

AAM with RADIUS

This topic provides some examples of RADIUS AAA solutions that use Application Access Management (AAM) features. For examples that use Basic HTTP logon and form-based logon can be found in the topics.

The following topics are covered:

Basic HTTP Logon with RADIUS

The following topics are covered:

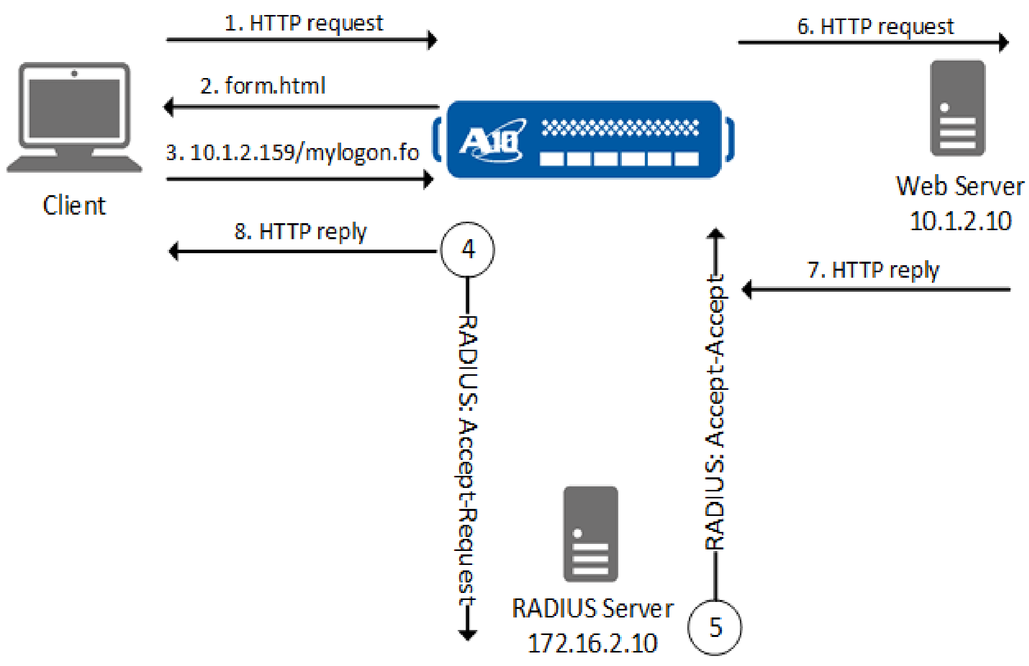
Deployment

This solution uses a RADIUS server to verify end-user credentials before allowing access to a web server.

ACOS uses basic HTTP logon to obtain the username and password from the client and sends these credentials to the backend RADIUS server for verification.

Figure 37 shows an example deployment of this solution.

Figure 37 : Basic HTTP Logon by Using RADIUS



This example illustrates the successful authentication of a client request.

Traffic Walkthrough

1. The client sends an initial HTTP request to the VIP.
2. ACOS replies with an HTTP 401 (Not Authorized) message with response code 4, which contains an Authentication header. The following text is an example of this header:
3. WWW-Authenticate: Basic realm=”*realm-name*”
4. The client browser displays the logon dialog to the end-user.
5. The end-user enters their credentials.
6. The client browser encodes the credentials in the Base64 format and sends them to ACOS.
7. ACOS decodes the username and password and sends the credentials to the RADIUS server in an Access-Request message.
8. The RADIUS server replies.
9. In this example, the username and password are present in the RADIUS server’s user database. The RADIUS server responds with an Access-Accept message.
10. ACOS caches the reply, so that the cached verification of credentials can be used again for the next request from the same end-user.
11. ACOS uses SLB to select a server from the web-server service group and then sends the client’s HTTP request to the server.
12. This example assumes that SLB is used.
13. The server replies.
14. In this example, the server has the requested content and sends the content in the reply.
15. ACOS forwards the server reply to the client.

Configuration

The following topics are covered:

Resources

The deployment in this example requires the following resources:

- An authentication-server profile for the backend RADIUS server
- An authentication-logon profile for Basic HTTP logon
- An authentication template containing the authentication-server profile and authentication-login profile
- A server configuration and a service group for the application server
- A VIP configuration

CLI Example

The following topics are covered:

- Login Request
- Authentication Request
- AAM Template
- AAA Policy
- Server and a Service Group

Login Request

The following commands configure a simple login request:

```
ACOS(config)# aam authentication logon http-authenticate httpbasic
ACOS(config-http-authenticate auth logon:...)# auth-method basic enable
```

Authentication Server

The following commands configure the authentication server:

```
ACOS(config)# aam authentication server radius radius_server
ACOS(config-radius auth server:radius_ser...)# host 10.1.1.12
ACOS(config-radius auth server:radius_ser...)# secret encrypted
oRePnFnDFQhb12zJNmvSCDwQjLjV2wDnPBCMuNXbAOc8EIy41dsA5zwQjLjV2wDn
```

AAM Template

The following commands configure an AAM template that will be part of aaa-policy:

```
ACOS(config)# aam authentication template auth_temp
ACOS(config-auth template:auth-template)# type standard
ACOS(config-auth template:auth-template)# logon httpbasic
ACOS(config-auth template:auth-template)# server radius_server
ACOS(config-auth template:auth-template)# log enable
```

AAA Policy

The following commands configure the AAA policy that will be attached to Virtual service:

```
ACOS(config)# aam aaa-policy aaa_policy
ACOS(config-aaa policy:aaa_policy)# aaa-rule 3
ACOS(config-aaa policy:aaa_policy-aaa rul...)# action allow
ACOS(config-aaa policy:aaa_policy-aaa rul...)# authentication-template
auth_temp
```

Server and a Service Group

The following commands configure a basic server and a service group:

```
ACOS(confg)# slb server u1 10.1.1.11
ACOS(config-real server)# port 80 tcp
ACOS(config)# slb service-group sg1 tcp
ACOS(config-slb svc group)# member u1 80
ACOS(config)# slb virtual-server vip 10.1.1.101
ACOS(config-slb vserver)# port 80 http
ACOS(config-slb vserver-vport)# source-nat auto
ACOS(config-slb vserver-vport)# service-group sg1
ACOS(config-slb vserver-vport)# aaa-policy aaa_policy
```

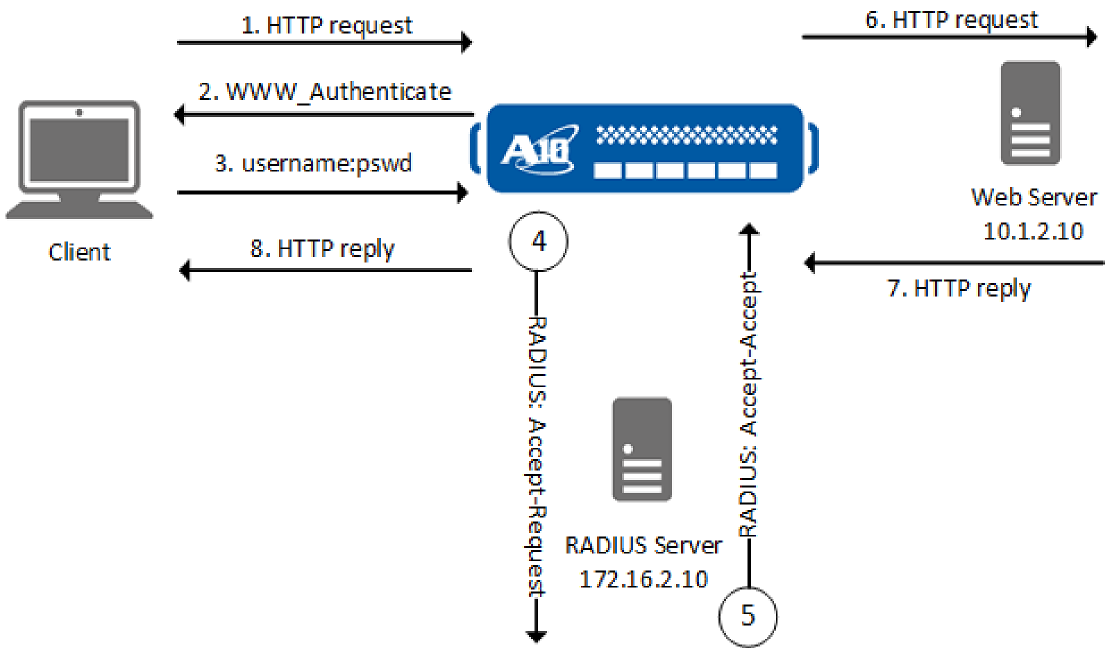
Form-Based Logon with RADIUS

The following topics are covered:

Deployment

This solution is similar to the one in Basic HTTP Logon with RADIUS, except form-based logon is used instead of basic-HTTP logon.

Figure 38 : Form-based logon by Using RADIUS



This example shows successful authentication of a client request. All steps except 2 and 3 are the same as those in Basic HTTP Logon with RADIUS.

Traffic Walkthrough

1. The client sends an initial HTTP request to the VIP.
2. ACOS replies with a form that contains the input fields for the end-user username and password.
3. The client browser displays the form, into which the end-user enters their credentials.
4. ACOS extracts the credentials from the form and sends this information to the RADIUS server in an Access-Request message.
5. The RADIUS server replies.

   In this example, the username and password are present in the RADIUS server's user database. The RADIUS server responds with an Access-Accept message.

   ACOS caches the reply, so that the cached verification of credentials can be used again for the next request from the same end-user.
6. ACOS uses SLB to select a server from the web-server service group and sends the client's HTTP request to the server.

   This example assumes that SLB is used.
7. The server replies.

   In this example, the server has the requested content and sends this content in the reply.
8. ACOS forwards the server reply to the client.

Unsuccessful Logon Messages

If a previous logon attempt failed, when you try to login again, the Logon form includes a customizable error message, instead of just the username and password fields.

The error page that is returned by ACOS displays entry fields so that the end-user can enter their username and password again.

Source Code Example

The following text is an example of the source code for the error page:

```
<form name="logon" action="mylogon-aaa.fo" method="POST">
   <!-- <p><font size="5" color="red">$a10_login_fail_errmsg$</font></p> -
->
   Username: <input type="text" name="username"><br>
   Password: <input type="password" name="pwd">
   <input type="submit" value="Submit">
</form>
```

If the `$a10_login_fail_errmsg$` variable is used, but is commented out as shown above, ACOS includes the logon failure message in the form only when applicable. If a client logon failure occurs, to make the message visible on the new logon page that is displayed to the client, ACOS inserts a message and negates the HTML comment in the form that is sent to the client.

The default error message string for logon failures is “Invalid username or password. Please try again.”

You can customize the string by using the **`login-failure-message`** command.

Creating a RADIUS Authentication Server

The following topics are covered:

NOTE: To create a RADIUS authentication server by using the GUI or the CLI, see Creating Authentication Servers.

Configuration

The deployment requires the following resources:

- A Zip archive of the web portal files that are required for end-user logon
- A Logon-portal profile

- An authentication-server profile for the backend RADIUS server
- An authentication template that contains the authentication-server profile and login-portal profile
- A server configuration and a service group for the application server
- A VIP configuration

CLI Example

The following topics are covered:

- Importing the Login-Portal Files
- Configuring the Login-Portal Files
- Configuring Authentication Template
- Configuring Authentication Template
- Adding the SLB Configuration

Importing the Login-Portal Files

The following commands import the login-portal files to the ACOS device:

```
ACOS(config)# import auth-portal portal.zip use-mgmt-port sftp:
Address or name of remote host []?fileserver1
Username []?admin1
Password []?********
File name [/]?portal.zip
```

Configuring the Login-Portal Files

The following commands configure the login-portal profile:

```
ACOS(config)# aam authentication logon form-based f1
ACOS(config-form-based auth logon:f1)# portal portal.zip logon form.html
failpage error.html changepasswordpage changeform.html
ACOS(config-form-based auth logon:f1)# action-url /mylogon.fo
ACOS(config-form-based auth logon:f1)# username-variable username
ACOS(config-form-based auth logon:f1)# password-variable pwd
ACOS(config)# exit
```

Creating Authentication-Server Profile

The following commands create an authentication-server profile for the RADIUS server:

```
ACOS(config)# aam authentication server radius radius_server
ACOS(config-radius auth server:radius_server)# host 172.16.2.10
ACOS(config-radius auth server:radius_server)# secret a10networks
ACOS(config)# exit
```

This is the same authentication-server profile used in Basic HTTP Logon with RADIUS.

Configuring Authentication Template

The following commands configure the authentication template:

```
ACOS(config)# aam authentication template t1
ACOS(config-auth template:t1)# logon f1
ACOS(config-auth template:t1)# server radius_server
ACOS(config)# exit
```

Adding the SLB Configuration

The following commands add the SLB configuration.

This is the same server, service-group, and VIP configuration that were used in Basic HTTP Logon with RADIUS.

```
ACOS(config)# slb server rs_http 10.1.2.10
ACOS(config-real server)# port 80 tcp
ACOS(config-real server-node port)# no health-check monitor_name
ACOS(config-real server-node port)# exit
ACOS(config-real server)# exit
ACOS(config)# aam authentication service-group http_g_1 tcp
ACOS(config-aam svc group:http_g_1 tcp)# member rs_http 80
ACOS(config-aam svc group:http_g_1 tcp-member:rs_http 80)# exit
ACOS(config-aam svc group:http_g_1 tcp)# exit
ACOS(config)# slb virtual-server vip_auth 10.1.2.159
ACOS(config-slb vserver)# port 80 http
ACOS(config-slb vserver-vport)# exit
ACOS(config)# aam authentication service-group http_g_1 tcp
ACOS(config-aam svc group:http_g_1 tc)# exit
ACOS(config)# aam authentication template t1
```

RADIUS Accounting

The following topics are covered:

Enabling RADIUS Accounting Messages

Enabling RADIUS accounting messages allows a start or stop message to be sent whenever an authentication session is created or deleted. To enable accounting messages, configure a RADIUS server accounting port, or a RADIUS accounting service-group, and bind it to an authentication template.

If RADIUS accounting is enabled, whenever a user logs in successfully and a new authentication session is created, an accounting start message is sent to the RADIUS server. When the authentication session is deleted because of a logout, timeout, or manual deletion, then an accounting stop message is sent to the RADIUS server. If a RADIUS accounting service-group is configured, then the start and stop messages may not be sent to the same RADIUS server.

The RADIUS accounting message counters can be seen using the **`show aam authentication statistics radius`** command.

Configuring RADIUS Accounting Messages

To enable RADIUS accounting messages, first configure an AAM authentication RADIUS server with an accounting port. The RADIUS accounting port cannot be the same as the authentication port. An optional health monitor can be configured for the accounting port. Configure a RADIUS accounting port by using the following command at the AAM authentication RADIUS server configuration level:

```
ACOS(config)# aam authentication server radius radius_server
ACOS(config-radius auth server:radius_server)# accounting-port num
[health-check name]
```

Multiple RADIUS accounting servers can be grouped together in a RADIUS accounting service-group.

After configuring the AAM authentication RADIUS server, bind the server or service-group to an authentication template by using the following command at the AAM authentication template configuration level:

```
ACOS(config)# aam authentication template template1 {accounting-server |
accounting-service-group} name
```

An accounting server and an accounting service-group are mutually exclusive in the authentication template, and either can be used for RADIUS accounting. If the authentication template is removed from the configuration before a RADIUS accounting message is sent, then the message will not be sent.

The RADIUS accounting port and server can be verified using the **show aam authentication server radius** and **show aam authentication template** commands.

RADIUS Accounting Configuration Example

The following example configures a RADIUS accounting server and service-group and bind them to separate authentication templates.

```
ACOS(config)# aam authentication server radius accounting1
ACOS(config-radius auth server)# accounting-port 1813

ACOS(config)# aam authentication service-group sg-radius udp
ACOS(config-aam svc group)# member accounting1 1813

ACOS(config)# aam authentication template radius-accounting-sg
ACOS(config-auth template)# server radius-data-port
ACOS(config-auth template)# accounting-service-group sg-radius

ACOS(config)# aam authentication template radius-accounting-svr
ACOS(config-auth template)# accounting-server accounting1
```

The following example displays the output of the show aam authentication statistics radius command:

```
ACOS(config)# sh aam authentication statistics radius
```

```
Name Request Auth-Succ Auth-Fail Author-Succ Author-Fail Acc-Chall Acct-
Req Acct-Succ Acct-Fail Timeout OtherErr
---------------------------------------------------------------------------
-------------------------------------------------------------------
test   1        1         0            0           0            0
   1         1       0         0       0
```

AAM and Entrust IdentityGuard Support

This chapter describes how to configure Entrust IdentityGuard as an authentication server in AAM.

In this feature, the user will be able to configure a passcode variable name for a default portal or a custom three-field portal, provide Entrust IdentityGuard server support, and configure multi-factor authentication server with authentication relay.

There are several setups which the user can adopt, highlighted in the cases from this topic.

The following topics are covered:

Case 1: Using Three-Field Custom Portal

In this case, the user adds a new command within form-based logon to configure a variable name in the logon form and sets up a multi-factor challenge flow. A three-field custom portal is used to configure the multi-factor authentication server with authentication relay support.

The following topics are covered:

Configuration Overview

The following is an overview of the configuration procedure:

1. Add a new passcode field support in custom logon form. Client user can input the first-factor in passcode field for multi-factor authentication server, and input the password of authentication relay in password field.
2. If the client user does not input desired terms in the passcode field, the ACOS device will set the password as the first-factor to authenticate with the multi-factor authentication server.

CLI Configuration

To configure this case using the CLI:

Import a 3-field custom portal.

```
ACOS(config)# import auth-portal portal-eig use-mgmt-port
scp://192.168.99.28/home/cchen/portal-eig.zip
Username []?exampleuser
Password []?
Done.
ACOS(config)#
```

Configure a form-based logon profile for this portal.

```
ACOS(config)# aam authentication logon form-based fbl_eig
  portal portal-eig logon logon.html failpage fail.html changepasswordpage
change.html
  action-url logon.fo
  username-variable user
  password-variable pwd
  passcode-variable otp
  changepassword-url change.fo
  changepassword-username-variable cpusr
  changepassword-old-password-variable oldpwd
  changepassword-new-password-variable newpwd
  changepassword-password-confirm-variable cnfpwd
```

Configure a multi-factor authentication server and an authentication relay, then bind them and the form-based logon to an authentication template.

```
ACOS(config)# aam authentication server radius eig
  host 192.168.90.118
  secret a10
ACOS(config)# aam authentication relay  nr
  domain A10LAB
ACOS(config)# aam authentication template at_eig
  logon fbl_eig
  relay nr
  server eig
```

Bind the template to an aaa-policy and then bind the aaa-policy to a vport.

```
ACOS(config)# aam aaa-policy aaap_eig
  aaa-rule 1
    action allow
    authentication-template at_eig
ACOS(config)# slb virtual-server testvs 192.168.91.30
  port 8888 http
```

```
    source-nat auto
    service-group ntmsg
    aaa-policy aaap_eig
```

Case 2: Using Default Portal

In this case, the user enables passcode field in default portal and sets up a multi-factor challenge flow. The default portal is used to configure the multi-factor authentication server with authentication relay support.

The following topics are covered:

Configuration Overview

The following is an overview of the configuration procedure:

1. Add a new passcode field support in default portal. Client user can input the first-factor in passcode field for multifactor authentication server, and input the password of authentication relay in password field.
2. User can customize the text string, font, text size, or text color of the passcode field. User can also customize the variable name of the passcode field.

CLI Configuration

To configure this case using the CLI:

1. Enable passcode field in default portal.

```
ACOS(config)# aam authentication portal default-portal
  logon
    enable-passcode
Configure a form-based logon using default portal.
ACOS(config)# aam authentication logon form-based fbl_def
  portal default-portal
```

2. Configure a multi-factor authentication server and an authentication relay, then bind them and the form-based logon to an authentication template.

```
ACOS(config)# aam authentication server radius eig
  host 192.168.90.118
  secret a10
ACOS(config)# aam authentication relay  nr
  domain A10LAB
ACOS(config)# aam authentication template at_eig
  logon fbl_def
  relay nr
  server eig
```

3. Bind the template to an aaa-policy and then bind the aaa-policy to a vport.

```
ACOS(config)# aam aaa-policy aaap_eig
  aaa-rule 1
    action allow
    authentication-template at_eig
ACOS(config)# slb virtual-server testvs 192.168.91.30
  port 8888 http
    source-nat auto
    service-group ntmsg
    aaa-policy aaap_eig
```

Case 3: Adding Entrust IdentityGuard Support

In this case, the user designates the Entrust IdentityGuard server as the authentication server and sets up the challenge flow to route client requests to IdentityGuard for authentication.

Configuration Overview

The following is an overview of the configuration procedure:

- Configure Entrust IdentityGuard as an authentication server under RADIUS protocol.

NOTE: For more information about the CLI procedure for configuring RADIUS as an authentication server, see Authentication Servers.

- Add a new VPN server configuration for ACOS device in Entrust IdentityGuard Properties Editor.

NOTE: For additional steps and for more information, see *Entrust IdentityGuard Server Administration Guide*.

AAM and RSA

This chapter provides information about configuring an ACOS device to authenticate users by using RSA SecureID on a RADIUS server.

The following topics are covered:

Overview

The following topics are covered:

Configuration

You can configure RSA SecurID to serve as the authentication engine for RADIUS. This way, the RSA server acts like a RADIUS server by accepting RADIUS requests that carry RSA-specific attributes. To be authenticated, instead of just a password, you must enter a passcode, which consists of a personal identification number (PIN) and a tokencode. This process is also known as two-factor authentication, which offers a higher level of security compared with password authentication.

ACOS AAM supports the recommendations of RFC 2865 and RFC 2808. See References for further information.

The following topics are covered:

You can use RSA in the following logon types:

HTTP-basic Logon

You can import an RSA portal that contains a new PIN submission page and a next tokencode submission page or use the default pages on the ACOS device. If the RSA portal is imported, you have to bind the RSA portal to the HTTP-basic logon, and specify the variable names that are used in the imported forms.

Form-based Logon

This type of logon contains the new PIN and next tokencode submission pages. If these pages do not exist, ACOS uses the default page. Also the new PIN and next-token variable names must be specified in form-based logon.

RSA SecureID

RSA SecurID authentication consists of a software or hardware token that is assigned to a computer and user and that generates an authentication code at fixed intervals, usually 60 seconds.

When you try to authenticate to a network resource, like logging into your company's network, you must enter a PIN and the token code that is displayed at that moment on your RSA SecurID token. If you do not enter the code that is displayed on the token during the fixed interval, the number expires. You have to wait until the next number is displayed before authenticating to the network.

NOTE: For more information about RSA SecureID, see the following: http://www.emc.com/security/rsa-securid.htm.

RSA Passcode Authentication

To authenticate a user, instead of using a password, the authentication process prompts the user to enter a passcode. Depending on how the RSA server is configured, a passcode can consist of a PIN and a concatenated tokencode or just a tokencode.

Workflow for RSA Authentication

The following topics are covered:

Overview of the Authentication Process

The following steps provide a high-level overview of the authentication process:

1. The user enters a username and a passcode.
2. The server validates these credentials.
   a. If user is in the New PIN Mode, the server sends an Access-Challenge message and prompts the user set a new PIN.
   b. The user enters the new PIN.
   c. The server accepts and updates the new PIN.
3. The server sends an Access-Challenge message, which prompts the user to enter the next token.
4. The user waits for the previous passcode to expire and uses the new PIN to get a new passcode.
5. The user enters the new passcode as the next token.
6. The server validates the passcode, and the user is granted access to the VIP resource.

In addition to an Access-Challenge message, the RADIUS server might respond with one of the following messages:

Access Reject

The user will be unable to receive access to the requested network service. This will message may be sent because the user account is unknown, inactive or fails an identification check.

Access Accept

The user has been authenticated and will be able to access network resources. The RADIUS server will continue to verify that the user is authorized to use the network services requested. This information may be stored locally on the RADIUS server, or it may be looked up using an external source, such as LDAP or Active Directory.

These RADIUS responses may include a Reply-Message attribute which may give a reason for the rejection, a prompt for the challenge, or a welcome message for the accept.

Authentication Modes

The following topics are covered:

You can use one of the following modes for authentication.

RSA Next Token Mode

When the RSA server receives your access request, and wants to challenge the validity of your token, the server can issue access-challenge message, which prompts you to enter the next tokencode.

The access challenge contains the *Waiting for token to change, then enter the next tokencode* replay message and a state string that beings with *SECURID_NEXT*. When the ACOS device receives this string, the device creates a next-token form and sends the form to you. This step allows you to enter the next tokencode.

The ACOS device also encrypts and encodes the state string into an RSA cookie and prompts your computer to set the cookie in the next request.

After the ACOS device receives the client's request again, the device checks whether the request payload contains the next tokencode, and whether the RSA cookie is present.

If the RSA cookie is present, the ACOS device decodes and decrypts the cookie and determines whether the uncoded cookie is a next token state string.

If the cookie is this type of string, the ACOS device sends an RADIUS authentication request, which contains the next tokencode and state, to the RSA server.

If the next tokencode is correct, the server responds with an access-accept message.

Workflow for the Next Token Mode

The following procedure provides a high-level overview of the Next Token Mode authentication process:

1. When you access a restricted source, you enter and submit the username and token code that are generated by token.
2. The server accepts the user credentials and sends an Access-Challenge message to ask for the next token.
3. After the token generates the next token code, you enter and submit this next token code.
4. If the next token code is correct, the server responds with an Access-Accept message.

The RSA cookie accommodates different browser behaviors. For example, some browsers will use one persistent HTTP session during multiple RSA request-challenge interactions, while other browsers, such as Google Chrome, might simultaneously initiate multiple HTTP sessions after one HTTP authentication transaction is completed. The RSA cookie allows the ACOS device to determine whether a new HTTP session is part of RSA authentication process.

RSA New PIN Mode

The new PIN mode means that you are authenticating for the first time, but your PIN has not yet been created. When the token that is associated with a user is in the new PIN mode and is received by the RSA server, the server responds with an Access-challenge message. which may carry the special *state* and *reply-message* attributes. The reply-message is a string that carries the information that the server wants the client to know.

In the new PIN mode, the state string is *SECURID_NPIN* followed by some random numbers. When the ACOS device receives this string, the device creates a new-PIN form and sends this form to you. This step allows you to enter the new PIN. The ACOS device also encrypts and encodes the state string into an RSA cookie and prompts your computer to set the cookie in the next request.

The RSA cookie expires in 10 seconds.

After the ACOS device receives your request again, the device checks for the following:

- Whether the request payload contains the new PIN.
- Whether the RSA cookie exists..

If the RSA cookie exists, ACOS decodes and decrypts the RSA cookie and determines whether the uncoded cookie is a new PIN state string. If yes, the ACOS device ends the RADIUS authentication request, which contains the new PIN and the state, with the RSA server.

Workflow for the New PIN Mode

The following steps provide a high-level overview of the new PIN mode authentication process:

1. The user has a token in the new PIN state.
2. When the user tries to access a restricted source, the user enters and submits the username and token code that is generated by the default PIN code.
3. The server accepts the user credentials and sends an Access-Challenge message to prompt the user to set a new PIN code.
4. The user enters and submits a new PIN.
5. The server sets this new PIN and sends an Access-Challenge message to ask for the next token.
6. The user waits until the token code expires and uses the new PIN code to generate a new token code.
7. The user enters and submits this new token code.
8. If the token code is correct, the server responds with an Access-Accept message.

Configuring 2 Factor Authentication (2FA/MFA) for management plane

The ACOS supports authentication of users using local database or authentication servers (AS) like LDAP, TACACS and RADIUS. This is primary authentication even when 2FA is enabled. Another authentication layer is added as a second factor, which takes client SSL certificate as input and validates it. The validation of certificates is done by CA root certificate that issues the client SSL certificate. The configuration supports 2FA over SSH connection. When a user is asked to provide 2FA, the user has to provide a certificate file name. Based on the certificate store configuration ACOS gets the certificate from the remote server over secure connection and does the further validation. Following are the steps for configuring 2FA:

1. Enabling 2FA for management plane.

```
!
system mfa-management enable
!
```

2. Configuring CA root certificate.

```
!
system mfa-validation-type
CERT CA.pem
!
```

3. For importing CA certificate CLI command is already available in ACOS and CA certificate will be stored under standard path. Command to import CA is as follows:

```
ACOS(config)# import ca-cert ?
NAME<length:1-245> CA Cert File(enter bulk when import an archive file)
Configure Client certificate store.
!
system mfa-cert-store
cert-host 1.1.1.1
protocol scp
cert-store-path /here/is/cert/store
username testuser password encrypted
10AtKRhH1NQ8EIy41dsA5zwQjLjV2wDnPBCMuNXbAOc8EIy41dsA5zwQjLjV2wDn
!
```

Configuring RSA Authentication for HTTP Basic Logon

The following topics are covered:

NOTE: For more information on HTTP logon, see Basic HTTP Logon.

Using GUI for RSA 6.0 HTTP Basic Logon

If the portal file is used for HTTP-basic logon, it contains a new PIN and next token submission pages. If the portal file is used for form-based logon, this file contains the logon, change password, and fail pages.

To configure RSA by using HTTP authentication:

1. Click **AAM** > **Auth Clients**.
2. On the **Logon HTTP Auth** tab, click **Create**.
3. Enter a name for the authentication client.
4. Enter a maximum retry value.
5. Select the **Enable Basic Logon** check box and complete the following steps:
   a. Select the challenge response form.
   b. Enter the name of the new PIN page.
   c. Enter the name of the new PIN variable page.
   d. Enter the name of the next token page.
   e. Enter the name of the next token variable page.
   f. Enter the realm.
6. Click **Create**.

NOTE: For RSA authentication, you need to enable only Basic Logon. You do not need to enable SPENGO.

Using CLI for RSA 6.0 HTTP Basic Logon

1. Enter the following command to specify the name of this basic HTTP logon as "**http-logon**."

```
ACOS(config)# aam authentication logon http-authenticate http-logon
ACOS(config-http-authenticate auth logon:...)#
```

2. You are now in configuration mode for "**http-logon**." Enable HTTP basic logon

process as follows:

```
ACOS(config-http-authenticate auth logon:...)# auth-method basic enable
```

3. Specify the previously configured and imported components of the logon; namely, the challenge-response-form (**rsa-portal**), the new-pin-page (**np.html**), the next-token-page (**nt.html**), the new-pin-variable (**npv**), and the next-token-variable (**ntv**):

```
ACOS(config-http-authenticate auth logon:...)# auth-method basic
challenge-response-form rsa-portal new-pin-page np.html next-token-page
nt.html new-pin-variable npv next-token-variable ntv
```

Using CLI for RSA 8.1 HTTP Basic Logon

1. Enter the following command to specify the name of this basic HTTP logon as "**http-logon**."

```
ACOS(config)# aam authentication logon http-authenticate http-logon
ACOS(config-http-authenticate auth logon:...)#
```

2. You are now in configuration mode for "**http-logon**." Enable HTTP basic logon process as follows:

```
ACOS(config-http-authenticate auth logon:...)# auth-method basic enable
```

3. Specify the previously configured and imported components of the logon; namely, the challenge-response-form (`rsa-portal`), the challenge-page (`cp.html`), and the challenge-variable (`cpv.html`).

```
ACOS(config-http-authenticate auth logon:...)# auth-method basic
challenge-response-form rsa-portal challenge-page cp.html challenge-
variable cpv.html
```

Configuring RSA Authentication for Form-Based Logon

The following topics are covered:

NOTE: For more information on Form-based Logon, see Form-based Logon.

Using GUI for RSA 6.0 Form-Based Logon Form

To configure RSA by using form-based authentication:

1. Click **AAM** >> **Auth Clients >> Logon Form**.
2. Click +**Create**.
3. Enter a name for the logon form you are creating.
4. To configure the default portal, click the Use Default Portal check box, then click **Create**. If you want to create a custom portal, complete the desired fields on this screen, then click **Create**.

See the online help in the GUI for more information about the fields on this screen.

Using CLI for RSA 6.0 Form-Based Logon Form

1. Enter the following command to specify the name of this form-based logon as "**fb-logon**."

   ```
   ACOS(config)# aam authentication logon form-based fb-logon
   ```

2. You are now in configuration mode for "**fb-logon**." Specify the previously configured and imported components of the logon; namely, the portal (**portal.zip**), the logon (**logon.html**), the failpage (**fail.html**), the changepasswordpage (**change.html**), the new-pin-page (**np.html**), and the next-token-page (**nt.html**):

   ```
   ACOS(config-form-based auth logon:fb-logon)# portal portal.zip logon
   logon.html failpage fail.html changepasswordpage change.html new-pin-
   page np.html next-token-page nt.html
   ```

3. Continue and complete the configuration of "**fb-logon**" by specifying the variables required by your forms:

   ```
   ACOS(config-form-based auth logon:fb-logon)# action-url /mylogon.fo
   ACOS(config-form-based auth logon:fb-logon)# username-variable username
   ACOS(config-form-based auth logon:fb-logon)# password-variable pwd
   ```

```
ACOS(config-form-based auth logon:fb-logon)# next-pin-variable npv
ACOS(config-form-based auth logon:fb-logon)# next-token-variable ntv
```

Using CLI for RSA 8.1 Form-Based Logon Form

1. Enter the following command to specify the name of this form-based logon as "**fb-logon**."

```
ACOS(config)# aam authentication logon form-based fb-logon
```

2. You are now in configuration mode for "**fb-logon**." Specify the previously configured and imported components of the logon; namely, the portal (**portal.zip**), the logon (**logon.html**), the failpage (**fail.html**), the changepasswordpage (**change.html**), and the challenge-page (**cp.html**):

```
ACOS(config-form-based auth logon:fb-logon)# portal portal.zip logon
logon.html failpage fail.html changepasswordpage change.html challenge-
page cp.html
```

3. Continue and complete the configuration of "**fb-logon**" by specifying the variables required by your forms:

```
ACOS(config-form-based auth logon:fb-logon)# action-url /mylogon.fo
ACOS(config-form-based auth logon:fb-logon)# username-variable username
ACOS(config-form-based auth logon:fb-logon)# password-variable pwd
ACOS(config-form-based auth logon:fb-logon)# challenge-variable chalv
```

NOTE: The challenge page form for RSA in form-based authentication must not include a form action.
For example, the following challenge page is not supported by AAM because it calls a URL action:

<html>

<body>

<form name="cp.html" ~~action="form/handle.staff.php"~~ method="POST">

$replymsg$: <input type="text" name="chalv">

<input type="submit" value="Submit">

</form>

```
</body>

</html>
```

Remove “**action="form/ handle.staff.php"**” from the challenge form.
The following challenge form is supported:

```
<html>

<body>

<form name="cp.html" method="POST">

$replymsg$: <input type="text" name="chalv">

<input type="submit" value="Submit">

</form>

</body>

</html>
```

References

For more information, refer to the following:

- RFC 2865, *Remote Authentication Dial In User Service (RADIUS)*
- RFC 2808, *The SecurID(r) SASL Mechanism*
- AAM with RADIUS

Security Assertion Markup Language (SAML)

The Security Assertion Markup Language (SAML) is an open standard for sharing security information about identity, authentication, and authorization across different systems, that allows identity providers (IdP) to pass authorization credentials to service providers (SP). SAML is an XML-based framework for authentication and authorization between two entities: a Service Provider and an Identity Provider. It defines how providers can offer both authentication and authorization services.

In this chapter, the user learns about the Security Assertion Markup Language (SAML) and how it can be configured with AAM.

The following topics are covered:

Feedback

Overview

The following topics are covered:

Introduction

SAML is an XML-based standard that is used to facilitate single sign-on (SSO) services for users, so users can sign on once and gain access to one or more affiliated websites and services across domain boundaries.

When SAML is deployed in an enterprise, this essentially outsources to a third-party Identity Provider (IdP)[1] the task of authenticating and authorizing users.

1 Well-known Identity Providers are Ping Identity, Shibboleth, and Microsoft ADFS. (See Identity and Service Providers.)

User credentials are sent to the IdP, which then checks their credentials against a database. If the user is successfully authenticated, then the IdP sends a token to the client to use to authenticate to the requested web site which authorizes the user to access the website services.

In this way, users do not have to repeatedly enter their username and password multiple times throughout the day for a large number of applications or web services, such as email, CRM, or media websites.

SAML Assertions and the ACOS device

The purpose of a SAML assertion is for the back-end application servers (behind the ACOS device) to recognize the identity of a client (authentication). This is done to provide an appropriate level of access to network resources (authorization).

Once the client has been authenticated, the ACOS device relays the client's assertion information to the back-end application server so that the client can access the resources on the application servers without having to repeat the process of authentication.

Different application servers use different authentication protocols to authenticate users.

Supporting token relay is one of key applications for SAML, because any applications capable of accepting SAML assertions as the user's identity can instantly recognize SAML assertions that the ACOS device has relayed to them. This means that the ACOS device can use the client's assertions to communicate with SAML-aware applications, like Microsoft SharePoint, Office 365, or Salesforce.com.

ACOS is capable of transforming SAML assertions to JSON Web Tokens (JWT). If the back-end application server is protected by the SAML authentication server, the SAML assertions/SAML tokens that are sent to the application server after the user authentication are transformed to JSON Web Tokens.

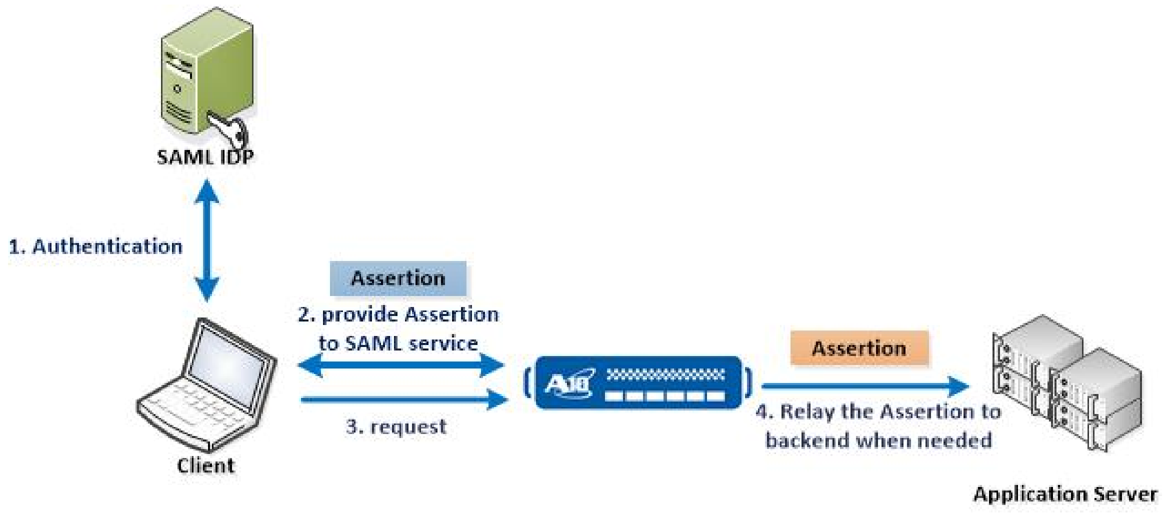
ACOS relays the JWT to the back-end application server.

NOTE: For the aFleX API, refer to the *aFleX Scripting Language Reference Guide*.

AAM and SAML

Along with SAML assertion used for authentication and authorization, it is also required for keeping the assertion and authentication relay to the backend server. The authentication relay SAML assertion to the back-end servers assist to get additional information about the authenticated user. The relay SAML advantage is SAML re-authentication of Application Server is not required as the first Service Provider can relay the SAML response token to Application Server. The relay SAML can be configured using the GUI and CLI.

Figure 39 : AAA and SAML


SAML IDP
1. Authentication
Assertion
2. provide Assertion to SAML service
3. request
Client
Assertion
4. Relay the Assertion to backend when needed
Application Server

The relay SAML is triggered for the below configurations when the following conditions occur:

- **server-login-uri**

  If the backend server response with 302 redirect and location is the server-login-uri then SAML relay will be triggered.

  Thunder will send an HTTP POST to server assertion-consuming-server URI with SAML AuthnResponse and configured RelayState in the payload.

- **server-cookie-name**

The authenticated client sends an HTTP request without a server-cookie-name cookie in HTTP header then SAML relay will be triggered.

Thunder will send an HTTP POST to server assertion-consuming-server URI with SAML AuthnResponse and configured RelayState in the payload.

- **idp-auth-url**

  The backend server response with 302 redirect and location is the SAML service URL of IDP then SAML relay will be triggered.

  Thunder will send an HTTP POST to server assertion-consuming-server URI with SAML AuthnResponse and RelayState in the payload. And in this case, RelayState will provide redirect URI of backend 302 response, and use this RelayState for the SAML relay request.

- **server-login-uri, server-cookie-name and idp-auth-url**

  The SAML relay will be triggered when client URI is matched (to server-login-uri) or backend server cookie (server-cookie-name cookie) is missing in HTTP header or client is redirected to IDP's authentication URL.

- **relay-state get-from-backend**

  Thunder should try to get a RelayState from the backend application server.

- **relay-state get-from-backend with server-login-uri**

  The client HTTP request URI is the server-login-uri then SAML relay will be triggered.

  To get RelayState from the backend, the client request is still be forwarded to the backend server and the Thunder will parse the response from the backend server.

NOTE: Configuring a relay-state option (Get from Backend/Request URI/Value) is mandatory to get the RelayState information in the assertion relays of the backend server.

Using the GUI

Follow the below steps for configuring an authentication relay SAML:

1. Click AAM > Auth Relays.
2. On the SAML tab, click Create. The Create Authentication Relay SAML page opens.
3. Enter the Name for new Authentication Relay SAML.
4. Enter the Relay Assertion Consuming Service URI.
5. Enter the IDP Auth URI to manage the SAML authentication request.
6. Select one option from the drop-down menu for Relay State.
   a. Get from Backend
   b. Request URI
   c. Value
7. The Value field is visible when the Value option is selected in the Relay State. Enter the value.
8. Enter the number from 0 to 10 in the Retry Number field
9. Enter the Server Cookie Name used by the backend server for authentication.
10. Select one of the options from the drop-down menu for finding Server Login URI:
    - Equals
    - Contains
    - Start With
    - Ends With
11. Match URI field is visible when any of the options is selected from Server login URI. Specify the URI to trigger the SAML relay.

Using the CLI

The following command creates an authentication relay SAML

```
ACOS(config)#aam authentication relay saml saml
ACOS(config-saml auth relay:saml)#relay-acs-uri /adfs/ls/
ACOS(config-saml auth relay:saml)#idp-auth-uri https://idp.a10-
tplab.com/adfs/ls/
ACOS(config-saml auth relay:saml)#server-login-uri equals /start
ACOS(config-saml auth relay:saml)#server-cookie-name auth-relay
ACOS(config-saml auth relay:saml)#relay-state value test
ACOS(config-saml auth relay:saml)#retry-number 3
```

NOTE:

- If the ACOS and backend server set the same `assertion-consuming-service` in saml service-provider, then `acs-uri-bypass` option needs to be configured for forwarding POST assertion to the backend rather than always being handled by ACOS
- As currently the different `assertion-consuming-service uri` for the ACOS and backend server is not supported. So client may receive an error `SAML message delivered with POST to incorrect server URL`
- The server-cookie is used to trigger the SAML relay. It is important to set the expected cookie name otherwise the ACOS will relay again assuming it is failed and backend response will have an error
- The IDP (ex. ADFS) may use same URL for all the services. But this may create confusion if the redirected (IDP) is for SAML SSO or SLO and all the services may trigger the SAML relay. For caution this should be used only when the backend server is required to trigger the SSO.

Prerequisites for Deploying SAML

To use SAML, you must have the following components:

- A user
- The Identity Provider (IdP), which authenticates and verifies the user.
- The Service Provider (SP), which uses the verified information from the IdP to determine whether a user is allowed to access the requested web site.
- An application server

In SAML, you can enable multi-domain single sign-on (SSO). When multi-domain services run their authentication by using the same Identity Provider, these services can use SAML to perform federation identity, even if users have a different account name in different domains.

NOTE: For more information on SAML, see the following links:

http://saml.xml.org/wiki/saml-wiki-knowledgebase

https://wiki.oasis-open.org/security/FrontPage

SAML Lite

SAML Lite is a subset of SAML, where the ACOS device acting as the Service Provider, only supports the following profiles:

- Web SSO *AuthnRequest* HTTP redirect
- Web SSO *Response* HTTP POST
- Web SSO *Response* HTTP artifact
- Artifact Resolution, SOAP
- Single Logout (IdP-initiated) HTTP redirect

NOTE: The only difference between SAML and SAML Lite is the number of supported profiles.

For example, Name ID Management is used to manage anonymous IDs that are dynamically created and agreed upon between the Identity Provider and the Service Provider for user identification. A set of messages is used to establish, maintain, and remove these identifiers.

If a Service Provider or Identity Provider does not want to provide user identity federation with other Service Providers or Identity Providers, Name ID Management is not required. As a result, you can use SAML Lite, instead of SAML.

Identity and Service Providers

You can use one of the following Identity Providers:

Shibboleth

Link: https://shibboleth.net

Ping Identity

Link: https://www.pingidentity.com

Microsoft ADFS 2.0

Link: http://technet.microsoft.com/en-us/library/adfs2-step-by-step-guides (v=ws.10).aspx

Microsoft ADFS and Windows Authentication

When you use Microsoft ADFS as your Identity Provider on the Internet Explorer, depending on whether you use Windows authentication, you may have to change some settings in ADFS:

- If you want to use Windows authentication as your authentication policy, modify the following settings:
    - Go to the **Authentication Policies** folder.
    - In **Primary Authentication**, click **Edit** next to **Global Settings**.
    - In **Edit Global Authentication Policy**, in the **Intranet** section, select the **Windows Authentication** check box.
    - Click **OK**.
    - To use the Internet Explorer, modify the relevant Windows authentication settings so that, for example, ADFS and the client join same domain controller, or the client login must use AD credentials.
- If you do not want to use Windows authentication as your authentication policy, modify the following settings:
    - Go to the **Authentication Policies** folder.
    - In **Primary Authentication**, click **Edit** next to **Global Settings**.
    - In **Edit Global Authentication Policy**, in the **Intranet** section, deselect the **Windows Authentication** check box.
    - Verify that the **Forms Authentication** check box is selected.
    - Click **OK**.

NOTE: For information about configuring ADFS by using an Redirect/Artifact binding, see Configuring ADFS Redirect/Artifact Binding in an ACOS Device.

Supported Browsers

You can use the following browsers with SAML:

- Internet Explorer 9 or higher
- Mozilla Firefox 36 or higher
- Google Chrome 31 or higher
- Apple Safari 7 or higher

Single Sign-On

Single sign-on (SSO), is used in the Identity Provider which helps reduce the administrative overhead of distributing multiple authentication tokens to the user.

SAML uses SSO in the following way:

1. The Service Provider redirects the client to the Identity Provider.
2. The Identity Provider exchanges authentication information with the client.
3. After the client is successfully authenticated, a cookie is set for the Identity Provider.

   At the next, the Service Provider redirects the client to the same Identity Provider from a different site, and the client sends the Identity Provider cookie.
4. When the cookie refers to the valid authentication context, an *AuthResponse* message is generated and access is granted to the service (assertion) without having to authenticate again for the different site.

Service Providers

The ACOS device is the Service Provider.

Message Flow

When users requests access to a web site, these users must first be authenticated by the Identity Provider before the Service Provider will grant access to the site.

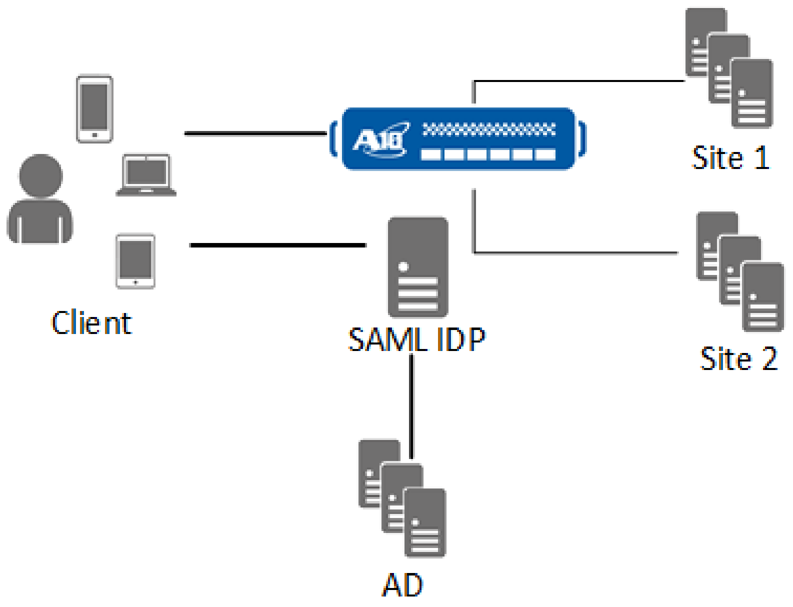
The procedure below provides a high-level overview of the process:

1. The user tries to access to a web site (for example, www.abc.com).
2. The Service Provider determines which Identity Provider is being used.
3. The Service Provider sends an *AuthRequest* message to the Identity Provider to authenticate the user.
4. The Identity Provider displays a logon form to the user and prompts the user to enter a username and password.
5. When Identity Provider verifies the credentials, the Identity Provider sends a *Response* message to the Service Provider stating that the user has been authenticated.

   The Identity Provider includes additional information that verifies that this message is from the specified Identity Provider.
6. The Service Provider provides access the requested web site.

Figure 40 illustrates a multi-domain example of this process, where ACOS is the SP.

Figure 40 : SAML Topology



Best Practice Work-flow

The following topics are covered:

This topic describes the best practice that administrators can follow to deploy the solution.

AAM Authentication SAML Service Provider

Administrators define the SAML Authentication configuration.

This covers the IdP and service provider configuration for SAML, and it includes the configuration of the Thunder ADC and the IdP.

Creating AAM Authentication Template

Administrators define the authentication settings.

This is also a template that can be used for other configurations.

Creating AAM AAA Policy

Administrators create an AAM authentication, authorization and accounting (AAA) policy.

This policy is required for SAML authentication to work. It enables administrators to allow and deny access to application resources.

This policy must be configured and defined before you can deploy authentication.

AAM Policy Binding to VIP

Administrators bind the AAM Authentication template to the virtual IP address (VIP).

NOTE: Once the AAM policy is bound to the VIP address, open up a browser and access the protected application. If the AAM policy allows access, the site must prompt the client on access assuming that the IdP credentials were correct. Likewise, if the AAM policy has deny privileges, the site must prevent users from accessing application resources.

SAML Profiles

The following topics are covered:

Profile Parts

The following topics are covered:

A profile composed of the following parts:

Bindings

Bindings, which contains the mappings of SAML protocols to standard messaging and communication protocols.

Protocols

Protocols, which contains requests and responses for obtaining assertions and completing identity management.

Assertions

Assertions, contains authentication, attributes, and authorization decision-making information.

Bindings, protocols, and assertions help you determine the flow of requests and responses during the authentication process.

Profile Types

The following topics are covered:

- Web Browser Service Provider-initiated
- Web Browser Identity Provider-initiated
- Single Log-out (Identity Provider-initiated)

ACOS 4.0 onwards, you can use one of the following profiles:

Web Browser Service Provider-initiated

This is the most commonly used profile and involves a user, a Service Provider, and an Identity Provider.

Web Browser Identity Provider-initiated

In this profile, the user is logged onto the Identity Provider and is trying to logon to a protected Service Provider web site. The Identity Provider prompts the user for credentials, and when the user has been authenticated, the Service Provider allows the user to access the protected web site.

Single Log-out (Identity Provider-initiated)

Single log-out profile is used to disassociate a session with the user. After a user has authenticated to an Identity Provider, the Identity Provider establishes a session with the user. At some time later, the user can use this profile to terminate the session with the session authority or a session participant.

This profile supports the POST, HTTP Redirect, and SOAP bindings.

Bindings

A SAML binding determine how the request and response flow between the Service Provider and Identity Provider.

NOTE: For more information about bindings, see the following: http://docs.oasis-open.org/security/saml/v2.0/saml-bindings-2.0-os.pdf.

To avoid communication issues, ensure that you specify the same binding for both providers.

NOTE: You must determine which profile and which binding you plan to use before you begin your configuration.

The following topics are covered:

HTTP Redirect Binding

The HTTP Redirect binding defines a mechanism by which SAML protocol messages can be transmitted by using URL parameters. The model is a request-response model, but these messages are sent to the user agent in an HTTP response and delivered to the message recipient in an HTTP request. Both the SAML requester and the SAML responder are assumed to be HTTP responders.

HTTP POST Binding

HTTP POST bindings can be used when the SAML requester and responder need to communicate by using an HTTP user agent as an intermediary. For example, this binding type can be used if the communicating parties do not share a direct path of communication.

HTTP Artifact Binding

In the HTTP Artifact binding, the SAML request or the SAML response, or both are transmitted by reference by using a stand-in, which is called an artifact. This is a synchronously updated binding, similar to the SAML SOAP binding, and this is often used to exchange the artifact for the actual protocol message that is using the artifact resolution protocol defined in the SAML assertions and protocols specification.

SOAP Binding

The SAML SOAP binding defines how to use SOAP to send and receive SAML requests and responses. If a SAML responder cannot process a SAML request, then this typically means that a SOAP fault must be generated.

Reverse SOAP (PAOS) Binding

The HTTP requester can act as a SOAP responder or a SOAP intermediary to a SAML requester. The HTTP requester can support a pattern where a SAML request is sent to the requester it in a SOAP envelope, which is in an HTTP response from the SAML requester, and the HTTP requester responds with a SAML response in a SOAP envelope in a subsequent HTTP request.

Microsoft ADFS and Redirect/Artifact Binding

Consider the following sources of information before you use SAML with a Microsoft ADFS identity provider and an artifact binding on an ACOS device:

- ADFS will not export this certificate in its metadata, and the ACOS device rejects ADFS's artifact.

  ADFS exports only the decrypting and signing certificate in the metadata, but you can use the ""soap-tls-certificate-validate disable"" option to disable the TLS check on the ACOS device.

- In the Service Provider certificate, the common name must be equal to the host URL, for example, `https://10.10.10.22, cn = 10.10.10.22`.

  The binding in the Service Provider is as follows:

```
aam authentication saml service-provider 10.10.10.22-https
artifact-resolution-service index 0 location
/adfs/services/trust/artifactresolution binding soap
assertion-consuming-service index 1 location /adfs/acs/ binding artifact
single-logout-service location /SLO/POST binding post
certificate 10.10.10.22.p12
entity-id https://10.10.10.22/adfs/identifiers
soap-tls-certificate-validate disable
service-url https://10.10.10.22
```

Configuring ADFS Redirect/Artifact Binding in an ACOS Device

Before you can configure your ACOS device to work with an ADFS artifact, the device must have a valid certificate and a correctly configured ADFS.

To create a valid certificate and configure ADFS:

1. To create and prepare the certificate for the Service Provider on the ACOS device:

NOTE: The common name and the alternative name must match the Service Provider entity ID and the ACOS virtual service.

   a. Create an Openssl configure file, for example, `ext_file.txt`, enter the following commands:

```
________________________________________
[ req ]
default_bits = 2048
distinguished_name = req_distinguished_name
```

```
x509_extensions = v3_ca

[ req_distinguished_name ]
countryName = Country Name (2 letter code)
countryName_default = TW
countryName_min = 2
countryName_max = 2

stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = CA

localityName = Locality Name (eg, city)
localityName_default = San Jose

0.organizationName = Organization Name (eg, company)
0.organizationName_default = A10 Networks

#1.organizationName = Second Organization Name (eg,company)
#1.organizationName_default = World Wide Web Pty Ltd

organizationalUnitName = Organizational Unit Name (eg, section)
organizationalUnitName_default = Security
#organizationalUnitName_default =

commonName = Common Name (e.g. server FQDN or YOUR name)
commonName_default = 10.10.10.22
commonName_max = 64

emailAddress = Email Address
emailAddress_default = username@example.com
emailAddress_max = 64

[ v3_ca ]
```

b. To generate a self-signed certificate and private key, enter the following commands:

```
openssl req -x509 -newkey rsa:2048 -keyout 10.10.10.22.key.pem -
keyform PEM -out 10.10.10.22.crt.pem -outform PEM -days 3650 -
config ext_file.txt
```

c. To change the certificate and private key to a pkcs12 format certificate, enter the following commands:

```
openssl pkcs12 -export -in 10.10.10.22.crt.pem -inkey
10.10.10.22.key.pem -out 10.10.10.22.p12
```

2. To configure your ACOS device, run the following commands:

```
aam authentication saml service-provider 10.10.10.22-https
  artifact-resolution-service index 0 location /adfs/artifact binding
soap
  assertion-consuming-service index 1 location /adfs/acs binding
artifact
  single-logout-service location /SLO/POST binding post
  certificate 10.10.10.22.p12
  entity-id https://10.10.10.22/adfs/identifiers
  soap-tls-certificate-validate disable
  service-url https://10.10.10.22
aam authentication saml identity-provider 10.10.10.22-https
  metadata metadata-10.10.10.22-https.xml
aam authentication template 10.10.10.22-https
  type saml
  saml-sp 10.10.10.22-https
  saml-idp 10.10.10.22-https
aam aaa-policy 10.10.10.22-https
  aaa-rule 2
    action allow
    authentication-template 10.10.10.22-https
slb template client-ssl saml-10.10.10.22
  cert 10.10.10.22.cert
  key 10.10.10.22.key
slb virtual-server 10.10.10.22 10.10.10.22
  port 443 https
    source-nat pool saml-ipv4
    source-nat auto
    service-group http-sg
    template client-ssl saml-10.10.10.22
    aaa-policy 10.10.10.22-https
```

3. Complete the following steps in ADFS:

   a. To enable Artifact Resolution service on ADFS:

      - Click **ADFS** > **Endpoints**.
      - In the **Endpoints** section, select an endpoint.
      - In the **Actions** section, which is found on the right-most side, click the endpoint link to enable the endpoint.

   b. Verify that the metadata contains the following artifact resolution service:

      ```
      <ArtifactResolutionService
      Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
      Location="https://adfs.a10lab.com/adfs/services/trust/artifactresol
      ution" index="0"/>
      ```

   c. Export the Identity Provider’s metadata to the ACOS device.

4. To configure the relaying party and the signature certificate and signature algorithm:

   a. Click **ADFS** > **Trust Relationships** > **Relaying Party Trusts**.

   b. Right-click on the certificate you want to modify and click **Properties**.

   c. On the **Signature** tab, click **Add** and select the certificate that you want to add.

   d. On the **Advanced** tab, in **Secure hash algorithm**, enter or select **SHA-1**.

   NOTE: User can only specify SHA-1 as the secure hash algorithm for the relaying party trust. SHA-256 is not supported in SAML.

   e. Click **Apply**, and then **OK**.

Protocols

SAML protocols are used to request and transmit messages between the Identity Provider and the Service Provider. Protocols refer to the content of the transmission and not the method of transmission.

NOTE: For more information about SAML protocols, see the following: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf.

The SAML standard allows Service Providers to perform the following tasks:

- Returning one or more requested assertions.
- This can occur in response to either a direct request for specific assertions or a query for assertions that meet particular criteria.
- Authenticating requests and returning the corresponding assertion.
- Performing registration on a name identifier or terminating a name registration on request.
- Retrieving protocol messages that have been requested using an artifact.
- Performing an almost simultaneous log-out of a group of inter-related sessions (“single logout”) upon request.
- Providing a name identifier mapping upon request.

This can be done by using one of the following protocols:

- Authentication Request
- Artifact Resolution
- Single Logout

Most of these protocols are completely new in SAML 2.0.

Assertions

SAML assertions are usually transferred from Identity Providers to Service Providers and contain statements that Service Providers use to make access-control decisions.

The following topics are covered:

SAML provides the following types of assertion statements.

Authentication Statements

Authentication statements assert or prove to the Service Provider (SP) that the principal has successfully authenticated with the Identity Provider (IdP) at a particular time using one of the supported methods of authentication. This authentication statement may also disclose other information about the authenticated principal (which is called the authentication context).

For example, the following authentication statement can be made.

The principal, which is identified in the *saml:subject* element, was successfully authenticated at time *time_stamp* and this was done using a password sent over a protected channel.

Attribute Statements

An attribute statement simply asserts that a particular subject is associated with a specific set of attributes. Typically, an attribute consists of a name-value pair. Relying parties use these attributes to make decisions about which resources the client can access.

For example, the following hypothetical attribute statement could be made:

The principal, which is identified in the *saml:Subject* element, was a member of a particular organization, which was located at this institution.

Authorization Decision Statements

An authorization decision statement simply asserts that a particular subject (or client) is allowed to perform a particular action upon a given resource, if and only if, evidence exists.

SAML Scenarios

A SAML profile is the message-interchange scenario that you and your federation partner have agreed to use.

The following topics are covered:

Prerequisite

Before you configure your scenario, ensure that a virtual server has been created.

Configuring the Virtual Server

To configure a virtual server:

1. In the left pane, click **ADC** > **SLB**.
2. On the **Virtual Servers** tab, click **Create**.
3. Enter a name.
4. Select an IP address type.
5. Enter an IP address.
6. Enter the netmask if you are not using wildcards.
7. In the **Virtual Port** section, click **Create**.
8. Enter a port name.
9. Select a protocol.
10. Expand the **Advanced Fields** section and configure the appropriate options. Refer to the online help for information on the fields listed on this GUI page.
11. Configure the appropriate options and click **Create**.

SAML Profiles

The following topics are covered:

ACOS is the Service Provider in the following supported profiles.

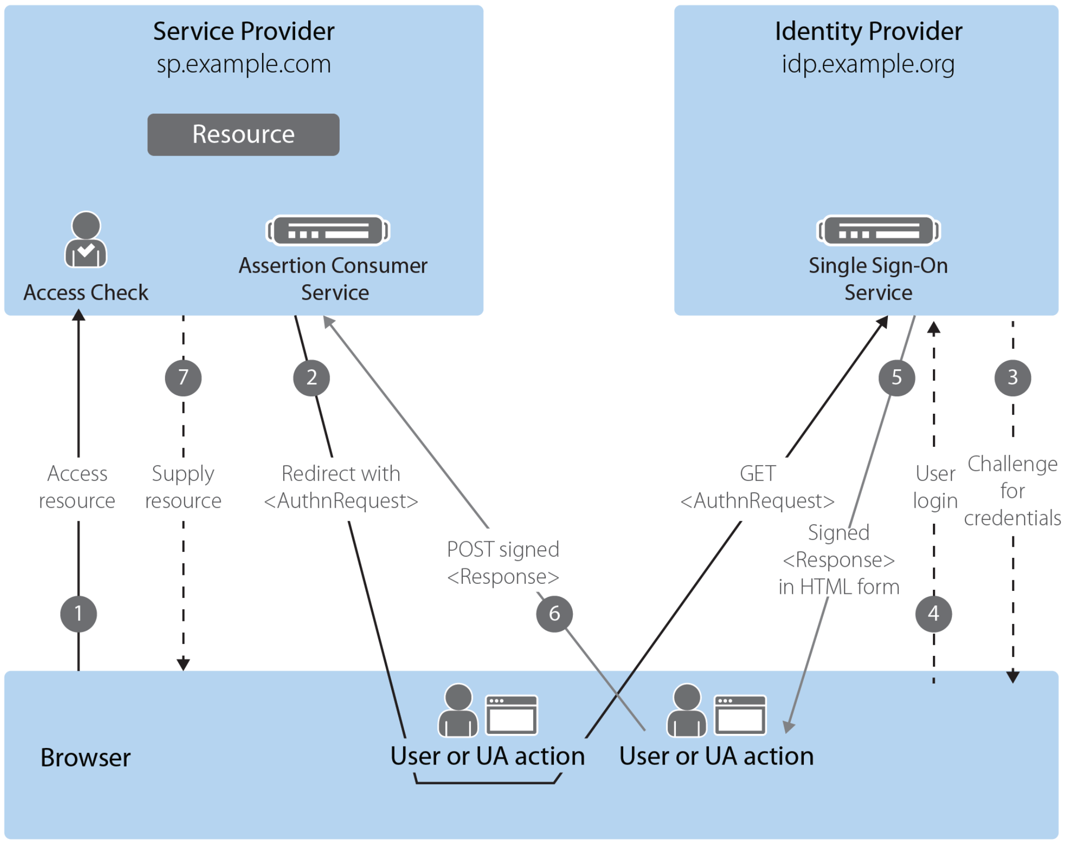
Web Browser Service Provider-Initiated SSO: Redirect/POST Binding

The following topics are covered:

Scenario

In this scenario, the Service Provider initiates the message flow.

Figure 41 : Web browser Service Provider-initiated SSO: Redirect/POST binding



Traffic Walkthrough

The following procedure provides the steps to understand the message flow in this profile:

NOTE: ACOS must be same use the clock and be in the same timezone as the Identity Provider.

1. The user requests access to a web site.
2. The Service Provider sends an *AuthRequest* message to the Identity Provider to authenticate the user.
3. The Identity Provider sends the user an HTTP logon form.
4. The user enters a username and password and submits the form to the Identity Provider.
5. The Identity Provider sends a signed *Response* HTML form to the user indicating a successful authentication.
6. The response is sent from the user's browser as an HTTP POST Response message to the Assertion Consumer Service in the Service Provider.

   NOTE: The Assertion Consumer Service is the only way for the Service Provider to receive the HTTP POST *Response* message.

7. The Service Provider allows the user's website to access the requested web site.

Reference

NOTE: For more information about this scenario, see the following: http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf

Configuring by Using the GUI

You can configure the Web Browser Service Provider-Initiated SSO: Redirect/POST Binding scenario by using the GUI.

The following topics are covered:

Creating a SAML Service Provider

1. To configure the Service Provider, complete the following steps:
   a. Click **AAM** > **Auth Clients**.
   b. On the **SAML Service Providers** tab, click **Create**.
   c. Enter a name, for example, **www.example.com**.
   d. Enter an entity ID, for example, **https://www.example.com/pingfederate**.
   e. Select a certificate, for example, **splin.cert**.

   NOTE: The Service Provider's certificate must contain a private key. The format of the key is, for example, *pfx (PKCS#12)*.

   f. Enter a service URL, for example, **www.example.com**
   g. To require a signed assertion, select the **Require Assertion Signed** check box.
   h. To disable a signing signature, select the **Disable Signing Signature** check box.
2. To configure single logout, complete the following steps:
   a. Enter an single logout service location, for example, **/SLO/SOAP**.
   b. Select an single logout service binding, for example, **post**.
   c. Click **Add**.
3. To configure an assertion consuming service, complete the following steps:
   a. Enter an assertion location, for example, **/SAML2/POST**.
   b. Enter an assertion index value.
   c. Select an artifact binding, for example, **post**.
   d. Click **Add**.

4. To configure an artifact resolution service, complete the following steps:

   Enter an artifact location.

   Enter an artifact index value.

   Select an artifact binding, for example, **soap**, and click **Add**.

5. Click **Create**.

Deleting a SAML Service Provider

To delete a SAML Service Provider:

1. Click **AAM** > **Auth Clients.**
2. On the **SAML Service Providers** tab, select the Service Provider that you want to delete.
3. Click **Delete**.

Creating a SAML Identity Provider

To create a SAML identity provider:

1. Click **AAM** > **Auth Clients**.
2. On the **SAML Identity Providers** tab, click **Create**.
3. Enter a name.
4. Enter a URL for the metadata file, for example, **http://metadata-wwww.example.com.xml.**
5. Select a certificate such as, **splin.cert**.
6. Click **Create**.

Setting SAML Identity-Provider Limit

With increase or decrease in the SAML environment size, the AAM authentication SAML identity-provider limit can be configured using GUI

To set an SAML identity provider limit:

1. Click **AAM** tab > **Resource Usage.**
2. Click **Config** and set the **Identity Provider Limit**.
3. Click **Save**.

Deleting a SAML Identity Provider

To delete a SAML identity provider:

1. Click **AAM** > **Auth Clients.**
2. On the **SAML Identity Providers** tab, select the identity provider that you want to delete.
3. Click **Delete**.

Creating an Authentication Template

To create an authentication template:

1. In the left pane, click **AAM** > **Policies and Templates**.
2. On the **Authentication Templates** tab, click **Create**.
3. Enter a name, for example, **www.example.com**.
4. Select **SAML** as the type.
5. Configure the fields, as desired. Refer to the online help for information on the fields listed on this GUI page.
6. Click **Create**.

Configuring by Using the CLI

To configure the Web Browser Service Provider-Initiated SSO: Redirect/POST Binding scenario:

1. To configure the service provider, enter the following commands:

```
ACOS(config)# aam authentication saml service-providerwww.example.com
artifact-resolution-serviceindex0location"/saml2/Artifact" binding soap

assertion-consuming-serviceindex0location"/saml2/post" binding post
single-logout-servicelocation"/slo/soap" binding post
```

```
certificatesplin.cert
entity-id "https://www.example.com/pingfederate"
service-url"http://www.example.com"
```

The **location** and **index** commands are required to configure the assertion consumer service in the service provider. Each location you configure must be unique. In our example, a location has been configured for exchange. The **assertion-consuming-service** location is where the artifact for the actual protocol message using the artifact resolution protocol. The **entity-id** identifies the service provider with the identity provider and must be unique between multiple service providers. The binding and location values must be the same between the service provider and the identity provider to prevent communications issues.

2. To configure the identity provider, enter the following commands:

```
ACOS(config)# aam authenticationsamlidentity-providerwww.example.com
metadata"metadata-www.example.com.xml"
```

The **metadata** command allows you to specify the location and name of the metadata file that must be imported to the identity provider. The metadata specifies the binding and prevents communication issues between the service provider and the identity provider. The **provider** command allows you to specify the identity provider you will be using.

Configure SAML Identity-Provider Limit

With increase or decrease in the SAML environment size, the AAM authentication SAML identity-provider limit can be configured using CLI

To set the SAML identity provider limit under AAM, use the following command:

```
ACOS(config)# aam resource-usage identity-provider-limit <64-256>
```

3. To configure the authentication template, enter the following commands:

```
ACOS(config)# aam authentication templatewwww.example.com
typesaml
saml-spwwww.example.com
saml-idpwwww.example.com
aam aaa-policywwww.example.com
aaa-rule256
actionallow
```

```
authentication-templatewwww.example.com
```

4. To configure the virtual server, enter the following commands:

```
ACOS(config)# slb virtual-serverwwww.example.com 10.10.10.13
port 80 http
service-grouphttp-sg
aaa-policywwww.example.com
```

JavaScript Notifications

When using a feature such as file-inspection, the request requires a long time to complete. For
example, when a large HTML page is fetched from server using any browser, it takes a long time with no feedback and blank page. If JavaScript notification is enabled on virtual port during SLB virtual server configuration, user is notified about his request status.

The following topics are covered:

Enabling JavaScript Notifications on the Virtual Server

To enable JavaScript notifications on virtual server for any notifications related to file-inspect, SSLI, WAF and SLB enter the following commands:

```
ACOS(config)# slb virtual-server slb wwww.example.com 10.10.10.13
port 80 http
service-group http-sg
aaa-policy wwww.example.com
jsi-enable file-inspect, ssli, waf, slb
```

Scenario for Configuring JavaScript Notification

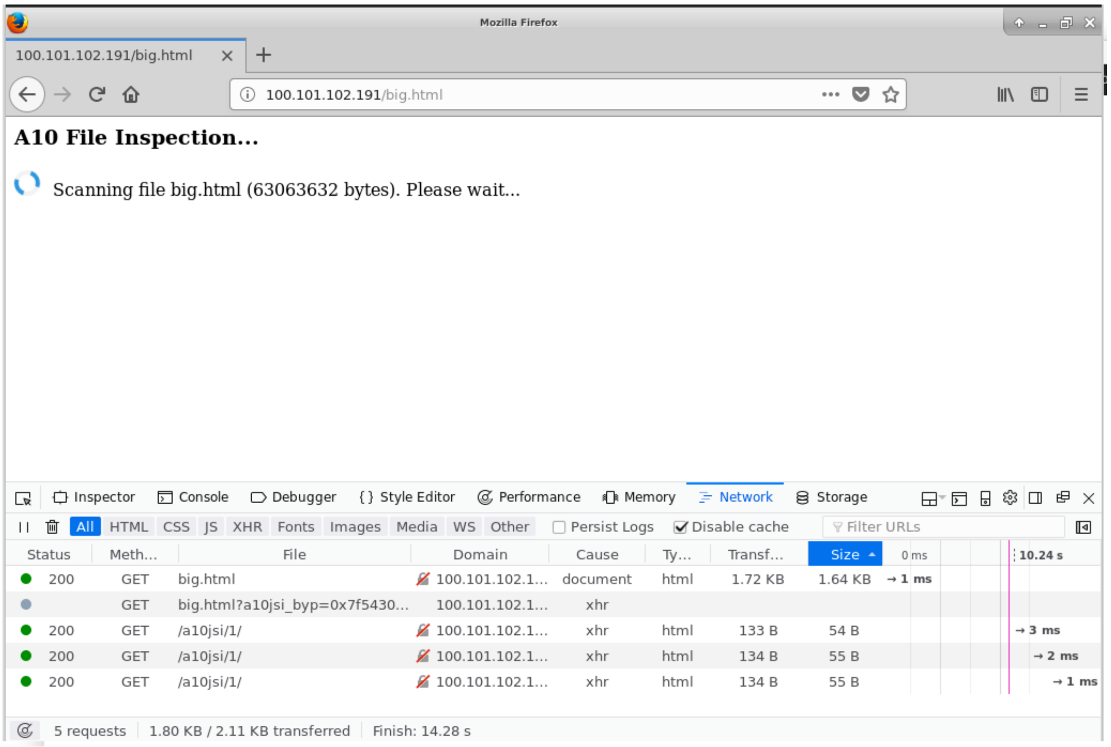
1. When a large HTML page is fetched from a server using browser it takes a long time with no
   feedback
2. If JavaScript notification is enabled on virtual port for file inspection using the

following code,

```
slb virtual-server vip19_100_101_102_191 100.101.102.191
  port 80 http
    jsi-enable file-inspect
```

3. The notifications are enabled and user is notified as displayed in the following image.

   JavaScript notification



A10 File Inspection...

Scanning file big.html (63063632 bytes). Please wait...

Status	Meth...	File	Domain	Cause	Ty...	Transf...	Size	0 ms … 10.24 s
● 200	GET	big.html	100.101.102.1...	document	html	1.72 KB	1.64 KB	→ 1 ms
●	GET	big.html?a10jsi_byp=0x7f5430...	100.101.102.1...	xhr				
● 200	GET	/a10jsi/1/	100.101.102.1...	xhr	html	133 B	54 B	→ 3 ms
● 200	GET	/a10jsi/1/	100.101.102.1...	xhr	html	134 B	55 B	→ 2 ms
● 200	GET	/a10jsi/1/	100.101.102.1...	xhr	html	134 B	55 B	→ 1 ms

5 requests | 1.80 KB / 2.11 KB transferred | Finish: 14.28 s

4. After a while the file is downloaded. Now add the following to have file inspection drop the file if it is corrupted or a “suspect” file.

```
file-inspection template t1
  downloads suspect drop log
```

5. A notification is also received that the “suspect” file was dropped.

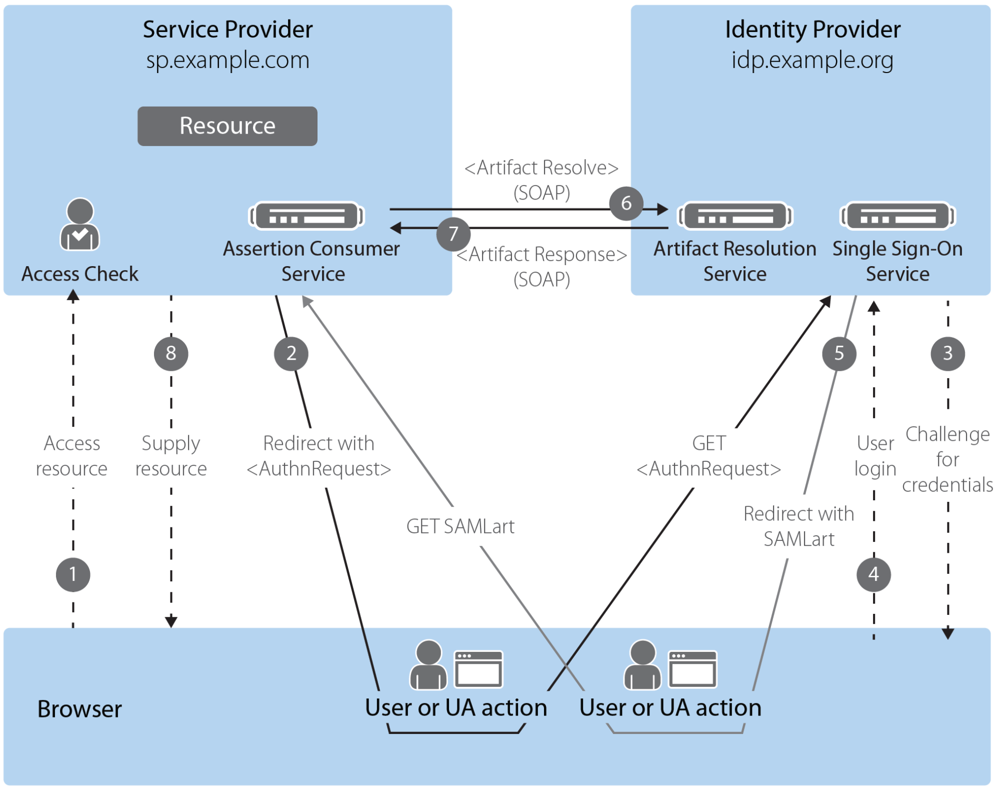
Web Browser Service Provider-Initiated SSO: Artifact/POST Binding

The following topics are covered:

Scenario

In this scenario, the service provider initiates the message flow.

Figure 42 : Web Browser Service Provider-Initiated SSO: Artifact/POST Binding



Traffic Walkthrough

The following procedure provides the steps to understand the message flow in this profile:

1. The user requests access to a web site.
2. The service provider sends an *AuthRequest* message to the identity provider to authenticate the user.
3. The identity provider sends the user an HTTP logon form.
4. The user enters a username and password and submits the form to the identity provider.
5. The identity provider sends a *Redirect SAML Artifact* message to the user, and the message is sent from the user's browser to the Assertion Consumer Service as a *GET SAML Artifact* message.

NOTE: Because it is an Artifact message, a separate, synchronous binding, such as the SAML SOAP binding, is used to exchange the artifact for the actual protocol message.

6. The Assertion Consumer Service sends an *Artifact Resolve* SOAP message to the Artifact Resolution Service.
7. The Artifact Resolution Service replaces the *Artifact Resolve* SOAP message with an *Artifact Response* SOAP message to the Assertion Consumer Service.
8. The service provider allows the user's browser to access the requested website.

Configuring by Using the GUI

You can configure Web Browser Service Provider-Initiated SSO: Artifact/POST Binding by using the GUI.

The following topics are covered:

- Creating a SAML Service Provider
- Deleting a SAML Service Provider
- Creating a SAML Identity Provider
- Deleting a SAML Identity Provider

- Creating an Authentication Template
- Configuring the AAA Policy

Creating a SAML Service Provider

The example below shows how to create a hypothetical SAML service provider:

1. To configure the service provider, complete the following steps:
   a. Click **AAM** > **Auth Clients.**
   b. On the **SAML Service Providers** tab, click **Create**.
   c. Enter the name **www.arti-post-pf.com**.
   d. Enter the entity ID **https://wwww.example.com/pingfederate**.
   e. Select the certificate **splin.cert**.

   NOTE: The service provider's certificate must contain a private key. The format of the key is *pfx (PKCS#12)*.

   f. Enter a service URL **http://www.arti-post-pf.com**.
   g. To require the assertion signed, select the **Require Assertion Signed** check box.
   h. To disable a signing signature, select the **Disable Signing Signature** check box.
2. To configure an assertion consuming service, complete the following steps:
   a. Enter an assertion location **/SLO/POST**.
   b. Enter an assertion index, **0**.
   c. Select an assertion binding, such as, **post**.
   d. Click **Add**.
3. To configure an artifact resolution service, complete the following steps:
   a. Enter an artifact location.
   b. Enter an artifact index, **0**.
   c. Select an assertion binding, such as **post**, and click **Add**.
4. Click **Create**.

Deleting a SAML Service Provider

To delete a SAML Service Provider:

1. Click **AAM** > **Auth Clients.**
2. On the **SAML Service Providers** tab, select the Service Provider that you want to delete.
3. Click **Delete**.

Creating a SAML Identity Provider

To create a SAML identity provider:

1. Click **AAM** > **Auth Clients**.
2. On the **SAML Identity Providers** tab, click **Create**.
3. Enter a name.
4. Enter a URL for the metadata file, for example, **http://metadata-wwww.example.com.xml.**
5. Select a certificate such as, **splin.cert**.
6. Click **Create**.

Deleting a SAML Identity Provider

To delete a SAML identity provider:

1. Click **AAM** > **Auth Clients.**
2. On the **SAML Identity Providers** tab, select the identity provider that you want to delete.
3. Click **Delete**.

Creating an Authentication Template

To create an authentication template:

1. In the left pane, click **AAM** > **Policies and Templates**.
2. On the **Authentication Templates** tab, click **Create**.

3. Enter a name, for example, **www.example.com**.
4. Select **SAML** as the type.
5. Configure the fields, as desired. Refer to the online help for information on the fields listed on this GUI page.
6. Click **Create**.

Configuring the AAA Policy

To configure the AAA policy:

1. In the left pane, click **AAM** > **Policies and Templates**.
2. On the **AAA Policies** tab, click **Create**.
3. Enter a name, for example, **wwww.example.com**.
4. Click **Create**.
5. In **AAA Rules** section, click **Create**.
6. In **Index**, enter a value, for example, **256**.
7. Select an action, for example, **allow**.
8. Select an authentication template, for example, **wwww.example.com**.
9. Click **Create**.

Configuring by Using the CLI

To configure the Web Browser Service Provider-Initiated SSO: POST/Artifact Binding scenario:

1. To configure the service provider, enter the following commands:

```
ACOS(config)# aam authentication saml service-providerwwww.example.com
artifact-resolution-serviceindex0location"/saml2/Artifact" binding soap
assertion-consuming-service index0location"/saml2/post" binding post
single-logout-service location"/slo/soap" binding post
certificatesplin.cert
entity-id"https://wwww.example.com/pingfederate"
service-url"http://wwww.example.com"
bad-request-redirect-uri "http://wwww.example.com"
```

The **location** and **index** commands are required to configure the assertion consumer service in the service provider. Each location you configure has to be unique. In this example, a location has been configured for exchange The **assertion-consuming-service** location is where the artifact for the actual protocol message using the artifact resolution protocol. The **entity-id** identifies the service provider with the identity provider and must be unique between multiple service providers. The binding and location values have to be the same between the service provider and the identity provider to prevent communications issues.

2. To configure the identity provider, enter the following commands:

```
ACOS(config)# aam authentication saml identity-providerwwww.example.com
metadata"metadata-wwww.example.com.xml"
```

The **metadata** command allows you to specify the location and name of the metadata file that must be imported to the identity provider. The metadata specifies the binding and prevents communication issues between the service provider and the identity provider. The **provider** command allows you to specify the identity provider you will be using.

3. To configure the authentication template, enter the following commands:

```
ACOS(config)# aam authentication templatewwww.example.com
typesaml
saml-spwwww.example.com
saml-idpwwww.example.com
```

4. To configure the AAA policy, enter the following commands:

```
ACOS(config)# aam aaa-policywwww.example.com
aaa-rule2
actionallow
authentication-templatewwww.example.com
```

5. To configure the virtual server, enter the following commands:

```
ACOS(config)# slb virtual-serverwwww.example.com 10.10.10.16
port 80 http
        service-grouphttp-sg
aaa-policywwww.example.com
```

Web Browser Identity Provider-Initiated SSO: POST Binding

The following topics are covered:

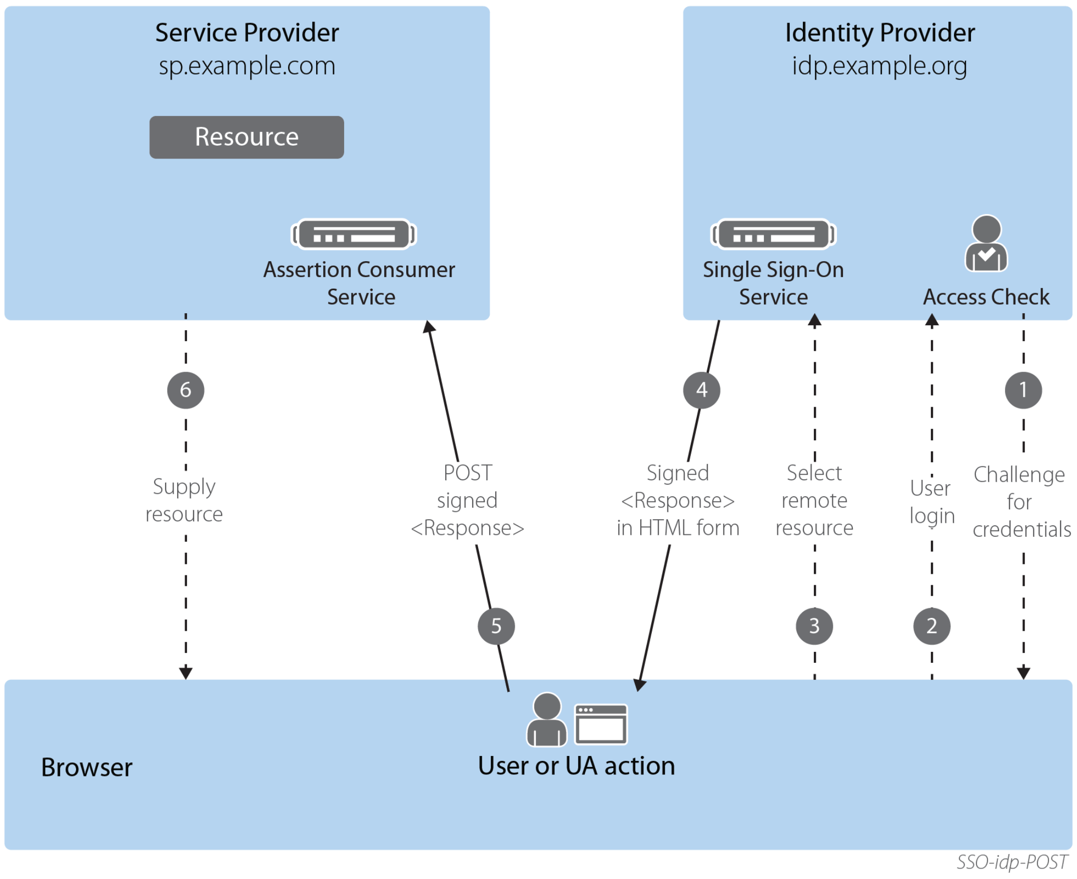
Scenario

In this scenario, the identity provider initiates the message flow.

The following are some of the reasons to use an Identity Provider-initiated scenario:

- In an Identity-Provider initiated scenario, a company might provide restricted resources, such as employee information or internal documents that requires authentication, but the company does not want to provide a resource URL to the location of these resources.
- A company can provide one unified logon URL, which is the Identity Provider’s URL to trigger SAML authentication. Therefore, every employee only needs to type this URL, get authenticated with the Identity Provider, and then the Identity Provider redirects the browser to the location of the restricted resources.

Figure 43 : Web Browser Identity Provider-Initiated SSO: POST Binding



Traffic Walkthrough

The following procedure provides the steps to understand the message flow in this profile:

NOTE: The user is trying to access a website that is located behind a firewall.

1. The identity provider prompts the user for credentials.
2. The user enters a username and a password and submits the form.
3. The user's browser selects a remote resource for authentication.
4. The identity provider sends a signed HTML Response form to the user's browser.
5. The user's browser sends a signed POST response message to the Assertion

Consumer Service.

6. The service provider allows the user’s browser to access the website.

Configuring by Using the GUI

You can configure the Web Browser Identity Provider-Initiated SSO: POST Binding scenario using the GUI.

The following topics are covered:

- Creating a SAML Service Provider
- Creating a SAML Identity Provider
- Creating an Authentication Template
- Configuring the AAA Policy

Creating a SAML Service Provider

To create a SAML service provider:

1. To configure the service provider, complete the following steps:
   a. Click **AAM** > **Auth Clients**.
   b. On the **SAML Service Providers** tab, click **Create**.
   c. Enter a service provider name, for example, **www.arti-post-pf.com**.
   d. Enter an entity ID, for example, **“https://wwww.example.com/pingfederate”**.
   e. Select a certificate, for example, **splin.cert**.

   NOTE: The service provider’s certificate must contain a private key. The format of the key is, for example, *pfx (PKCS#12)*.

   f. Enter a service URL, for example, **“http://wwww.example.com”**.
   g. To require the assertion signed, select the **Require Assertion Signed** checkbox.
   h. To disable a signing signature, select the **Disable Signing Signature** checkbox.
2. To configure an assertion consuming service, complete the following steps:

   a. Enter an assertion location, for example, **"/SLO/POST"**.
   b. Enter an assertion index, for example, **0**.
   c. Select an assertion binding, for example, **post**.
   d. Click **Add**.

3. To configure an artifact resolution service, complete the following steps:
   a. Enter an artifact location.
   b. Enter an artifact index, for example, **0**.
   c. Select an assertion binding, for example, **post**, and click **Add**.
   d. Click **Create**.

Creating a SAML Identity Provider

To create a SAML identity provider:

1. Click **AAM** > **Auth Clients**.
2. On the **SAML Identity Providers** tab, click **Create**.
3. Enter a name.
4. Enter a URL for the metadata file, for example, **"metadata-wwww.example.com.xml".**
5. Select a certificate, for example, **splin.cert**.
6. Click **Create**.

Creating an Authentication Template

To create an authentication template:

1. In the left pane, click **AAM** > **Policies and Templates**.
2. On the **Authentication Template** tab, click **Create**.
3. Enter a name, for example, **wwww.example.com**.
4. Select **SAML** as the type.
5. Configure the fields, as desired. Refer to the online help for information on the fields listed on this GUI page.

6. Select an authentication server or a service group.
7. Depending on the option you selected in step 10, do one of the following:
   - Select an existing server or click **New Authentication Server**.
   - Select an existing service group or click **New Service Group**.
8. Select a log.
9. Enter a cookie domain and click **Add**.
10. Enter a cookie domain group and click **Add**.

Configuring the AAA Policy

To configure the AAA policy:

1. In the left pane, click **AAM** > **Policies and Templates**.
2. On the **AAA Policies** tab, click **Create**.
3. Enter a name, for example, **wwww.example.com**.
4. Click **Create**.
5. In **AAA Rules** section, click **Create**.
6. In **Index**, enter a value, for example, **256**.
7. Select an action, for example, **allow**.
8. Select an authentication template, for example, **wwww.example.com**.
9. Click **Create**.

Configuring by Using the CLI

To configure the Web Browser Identity Provider-Initiated SSO: POST Binding scenario:

1. To configure the service provider, enter the following commands:

```
ACOS(config)# aam authentication saml service-providerwwww.example.com
artifact-resolution-service index0location“/saml2/Artifact” binding
soap
assertion-consuming-service index0location“/saml2/post” binding post
single-logout-service location“/slo/soap” binding post
certificatesplin.cert
```

```
entity-id"https://wwww.example.com/pingfederate"
service-
url"https://10.10.10.3:9031/idp/startSSO.ping?PartnerSpId=http://
wwww.example.com/pingfederate&ACSIdx=0"
```

The **location** and **index** commands are required to configure the assertion consumer service in the service provider. Each location you configure has to be unique. The **assertion-consuming-service** location is where the artifact for the actual protocol message using the artifact resolution protocol. The **entity-id** identifies the service provider with the identity provider and must be unique between multiple service providers. The binding and location values have to be the same between the service provider and the identity provider to prevent communications issues.

2. To configure the identity provider, enter the following commands:

```
ACOS(config)# aam authentication saml identity-providerwwww.example.com
metadata"metadata-wwww.example.com.xml"
```

The **metadata** command allows you to specify the localtion and name of the metadata file that must be imported to the identity provider. The metadata specifies the binding and prevents communication issues between the service provider and the identity provider. The **provider** command allows you to specify the identity provider you will be using.

3. To configure the authentication template, enter the following commands:

```
ACOS(config)# aam authentication templatewwww.example.com
typesaml
saml-spwwww.example.com
saml-idpwwww.example.com
```

4. To configure the AAA policy, enter the following commands:

```
ACOS(config)# aam aaa-policywwww.example.com
aaa-rule256
actionallow
authentication-templatewwww.example.com
```

5. To configure the virtual server, enter the following commands:

```
ACOS(config)# slb virtual-serverwwww.example.com 10.10.10.13
port 80 http
```

```
service-group http-sg
aaa-policywwww.example.com
```

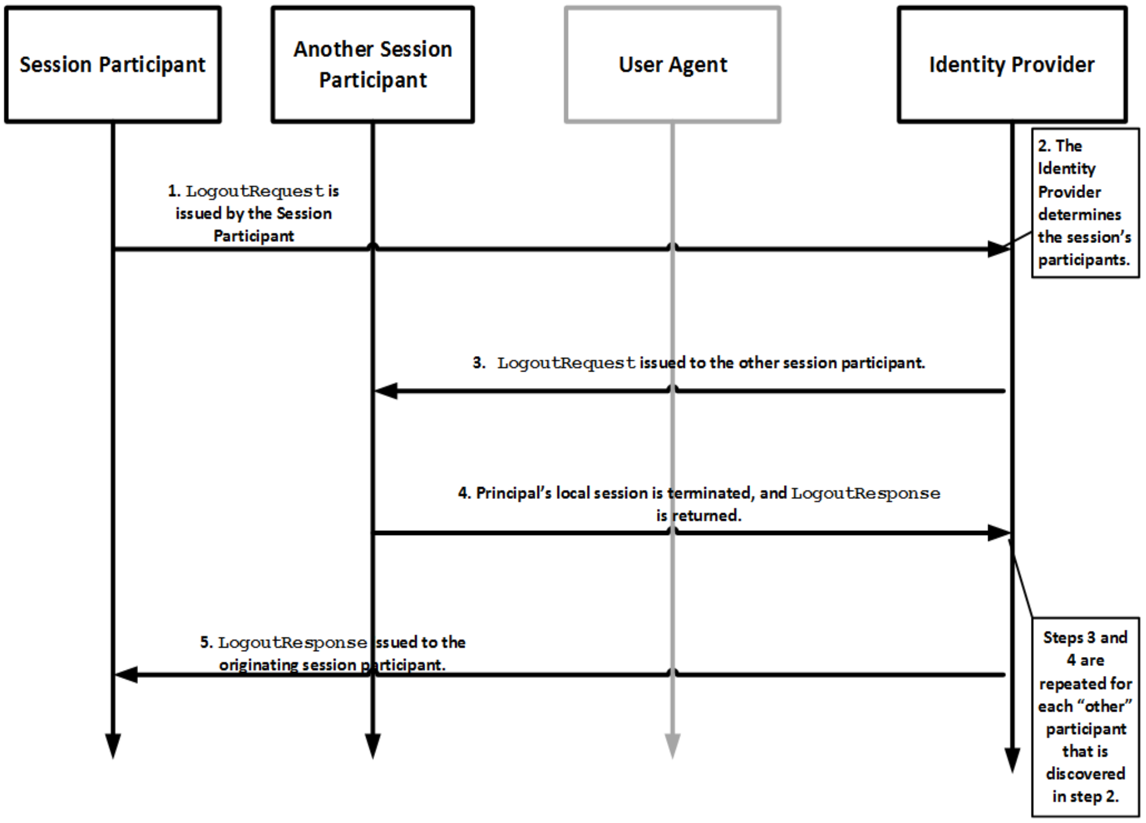
Single Logout (Identity Provider-Initiated): HTTP Redirect, SOAP

The following topics are covered:

Scenario

In this scenario, the single logout profile allows a user to terminate a session with another authenticated user.

Figure 44 : Single Logout (Service Provider-Initiated): HTTP Redirect, SOAP


Session Participant
Another Session Participant
User Agent
Identity Provider
1. LogoutRequest is issued by the Session Participant
2. The Identity Provider determines the session's participants.
3. LogoutRequest issued to the other session participant.
4. Principal's local session is terminated, and LogoutResponse is returned.
5. LogoutResponse issued to the originating session participant.
Steps 3 and 4 are repeated for each "other" participant that is discovered in step 2.

Traffic Walkthrough

The following procedure provides the steps to understand the message flow in this profile:

1. The session participant sends a *LogoutRequest* message to the Identity Provider.
2. The Identity Provider determines the number of existing session participants.
3. The Identity Provider sends a *LogoutRequest* message to the session participant or session authority that is related to the session participant whose session is being terminated.
4. The session participant or session authority sends a *LogoutResponse* message to the Identity Provider.

5. A session participant or session authority terminates the other participant's session(s) as directed by the request (if possible) and returns a *LogoutResponse* to the Identity Provider.
6. The Identity Provider sends a *LogoutResponse* message to the original session participant.

Configuring by Using the GUI

You can configure the Single Logout (Service Provider-Initiated): HTTP Redirect, SOAP scenario using the GUI.

The following topics are covered:

- Creating a SAML Service Provider
- Creating a SAML Identity Provider
- Configuring the AAA Policy
- Creating an Authentication Template

Creating a SAML Service Provider

To create a service provider:

1. Click **AAM** > **Auth Clients**.
2. On the **SAML Service Providers** tab, click **Create.**
3. Enter a name, for example, **www.ecp.com**.
4. Enter an entity ID such as, **https://www.ecp.com/pingfederate**.
5. Select a certificate such as, **splin.cert**.

NOTE: The service provider's certificate must contain a private key. The format of the key is, for example, *pfx (PKCS#12)*.

6. Enter a service URL, for example, **http://www.ecp.com.**
7. To require that the assertion be signed, select the **Require Assertion Signed** checkbox.
8. To disable a signing signature, select the **Disable Signing Signature** checkbox.

9. Enter Metadata Export Service: Location, for example **/A10Sp_Metadata**
10. To enable Metadata Export Service, select **Metadata Export Service: Sign XML** checkbox
11. Enter SP Initiated Single Logout Service Location (Asynchronous) path
12. Enter SP Initiated Single Logout Service Location (Synchronous) path
13. To disable the SOUP TLS service validation, select the **Disable Soup TLS Certificate Validate** checkbox
14. To enable the ADFS WS-Federation, select the **Enable ADFS WS-Federation** checkbox
15. To enable ACS URL, select **ACS URL Bypass** checkbox
16. Select from drop-down the Signature Algorithm
17. To configure the Single Logout Service, complete the following steps:

    Click **Add**.

    Enter the **SLO location**.

    Select **SLO Binding** from the drop-down (artifact or paos or post), for example **post**.

    Click **floppy disk icon** to save.

18. To configure an assertion consuming service, complete the following steps:

    Click **Add**.

    Enter an assertion location, for example, **/SLO/POST**.

    Enter an assertion index such as, 1.

    Select an assertion binding, such as, **post**.

    Click **floppy disk icon** to save.

19. To configure an artifact resolution service, complete the following steps:
    a. Click **Add**.
    b. Enter an artifact location, for example, **/SAML2/Artifact**.
    c. Enter an artifact index, for example, 0.

   d. Select an assertion binding, for example, soap.
   e. Click **floppy disk icon** to save.
20. Click **Create**.

Creating a SAML Identity Provider

To create an identity provider:

1. Click **AAM** > **Auth Clients.**
2. On the **SAML Identity Providers** tab, click **Create**.
3. Enter a name, for example, **www.ecp.com**.
4. Enter a URL for the metadata file, for example, **metadata-www.ecp.com.xml.**
5. Click **Create**.

Configuring the AAA Policy

To configure the AAA policy:

1. Click **AAM** > **Policies and Templates**.
2. On the **AAA Policy** tab, click **Create**.
3. Enter a name, for example, **www.redirect-post-pf.com**.
4. Do one of the following:
   a. Select a virtual port binding.
   b. Click **New VP** to create a new virtual port.
5. Click **Create**.
6. In **AAA Rule**, click **Create**.
7. In **Index**, enter a value, for example, **256**.
8. Select an action, for example, **allow**.
9. Select an authentication template, for example, **www.redirect-post-pf.com**.
10. Click **Create**.

Creating an Authentication Template

To create an authentication template:

1. Click **AAM** > **Policies and Templates**.
2. On the **Authentication Template** tab, click **Create**.
3. Enter a name, for example, **www.arti-post-pf.com**.
4. Select **SAML** as the type.
5. Configure the fields, as desired. Refer to the online help for information on the fields listed on this GUI page.
6. Select an authentication server or a service group.
7. Depending on the option you selected in step 9, do one of the following:
   - Select an existing server or click **New Authentication Server**.
   - Select an existing service group or click **New Service Group**.
8. Select a log.
9. Enter a cookie domain and click **Add**.
10. Enter a cookie domain group ID and click **Add**.
11. Click **Create**.

Configuring by Using the CLI

To configure the Single Logout (Service Provider-Initiated): HTTP Redirect, SOAP scenario:

1. To configure the service provider, enter the following commands:

```
ACOS(config)# aam authentication saml service-provider www.redirect-
post-pf.com
ACOS(config-SAML service provider:www.red...)artifact-resolution-
service index 0 location /SAML2/Artifact binding soap
ACOS(config-SAML service provider:www.red...)assertion-consuming-
service index 0 location /SAML2/POST binding post
ACOS(config-SAML service provider:www.red...)certificate splin.cert
ACOS(config-SAML service provider:www.red...)entity-id
https://www.redirect-post-pf.com/pingfederate
```

```
ACOS(config-SAML service provider:www.red...)service-url
http://www.redirect-post-pf.com
ACOS(config-SAML service provider:www.red...)SP-initiated-single-
logout-service location /Logout2 asynchronous
ACOS(config-SAML service provider:www.red...)SP-initiated-single-
logout-service location /Logout3
```

The **location** and **index** commands are required to configure the assertion consumer service in the service provider. Each location you configure must be unique.

In this example, a location has been configured for exchange. The **assertion-consuming-service** location is that of the artifact for the actual protocol message using the artifact resolution protocol. The **entity-id** identifies the service provider with the identity provider and must be unique between multiple service providers.

The binding and location values must be the same between the service provider and the identity provider to prevent communications issues.

2. To configure the identity provider, enter the following commands:

```
ACOS(config)# aam authenticationsamlidentity-providerwww.redirect-post-
pf.com
ACOS(config-SAML service provider:www.red...)metadata“metadata-
www.redirect-post-pf.com.xml”
```

The **metadata** command allows you to specify the location and name of the metadata file that must be imported to the identity provider. The metadata specifies the binding and prevents communication issues between the service provider and the identity provider. The **provider** command allows you to specify the identity provider you will be using.

3. To configure the authentication template, enter the following commands:

```
ACOS(config)# aam authentication templatewww.redirect-post-pf.com
typesaml
saml-spwww.redirect-post-pf.com
saml-idpwww.redirect-post-pf.com
aam aaa-policywww.redirect-post-pf.com
aaa-rule256
actionallow
```

```
authentication-templatewww.redirect-post-pf.com
```

4. To configure the virtual server, enter the following commands:

```
ACOS(config)# slb virtual-serverwww.redirect-post-pf.com 10.10.10.13
port 80 http
service-grouphttp-sg
aaa-policywww.redirect-post-pf.com
user initial url
https://10.10.10.3:9031/idp/startSSO.ping?PartnerSpId=https://www.redi
rect-post-pf.com/pingfederate&ACSIdx=0
```

NOTE: When you enter the URL, you will be logged out of the Service Provider session.

SAML Authorization

The following topics are covered:

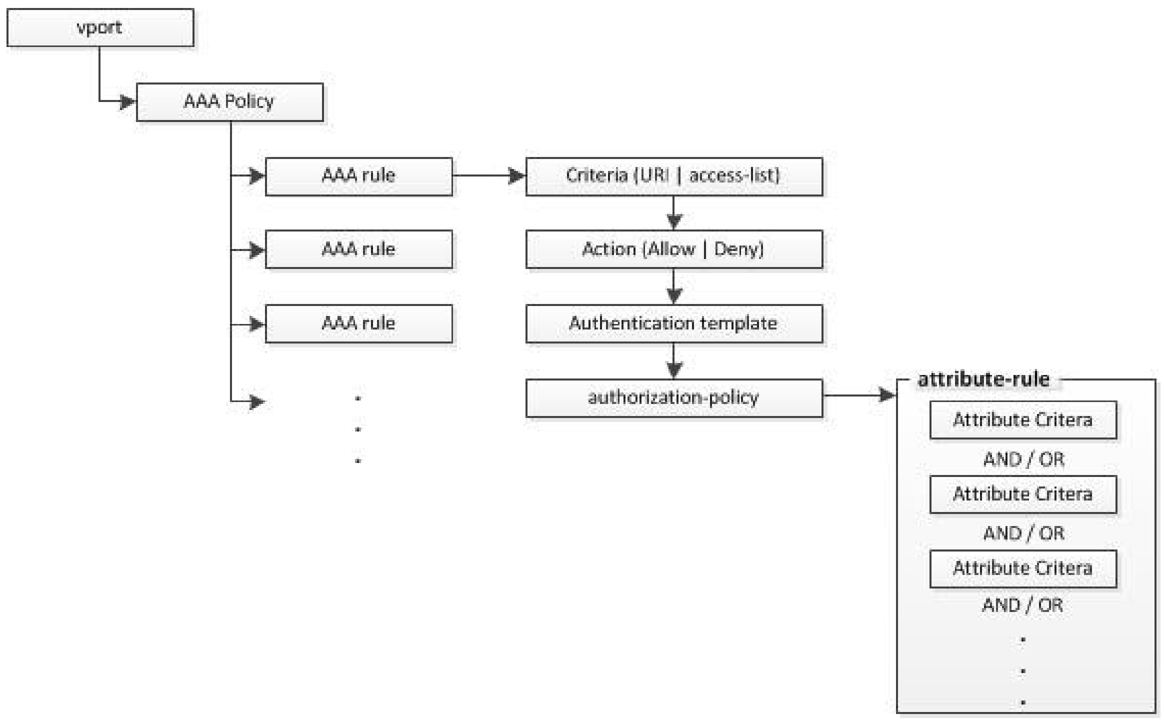
Authenticating and Authorizing Process

The following topics are covered:

Scenario

ACOS supports not only authentication using SAML, but also supports authorization using SAML. As with SAML authentication, SAML authorization is configured as part of an AAA policy, and implemented by binding the policy to a virtual port.

Figure 45 : AAA Policy



As shown in the diagram, a virtual port can bind to an AAA policy and each AAA policy can have multiple AAA rules.

The AAA rules contain the following:

Criteria

The users can configure access list, URI or domain name as criteria to associate client requests to this rule.

Action

It is an action to perform (allow or deny) after client request associates to this rule.

Authentication Template

It applies if client request is allowed access via the AAA rule.

The authentication template performs authentication for the user of the request.

Authorization Policy

If a client authentication is successful, ACOS also can perform authorization to check whether the authenticated user has permission to access the requested resource.

Process

Within the authorization policy, the authorization decision is based on attribute criteria and rules, which are configured in ACOS, and based on attributes retrieved from external authentication/authorization servers.

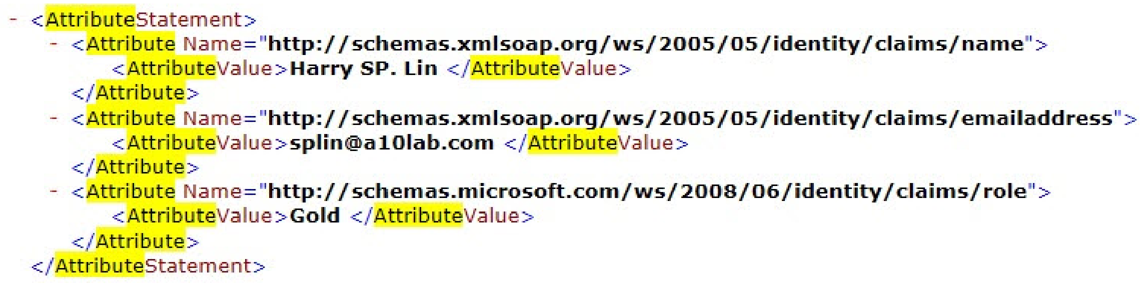
For example, the user can configure an attribute rule so that if attribute “username” is “abc” and attribute email ID is abc@a10lab.com, then access is permitted.

Later, when the client accesses the virtual port and authenticates successfully and gets attributes from the authentication server, then ACOS can check whether the attributes match the configured criteria.

Example

```
ACOS(config)# aam authorization policy p1
  attribute-rule 1 and 2 or 3
  attribute 1 user attr-type string match Ben
  attribute 2 emailaddress attr-type string match ben@a10lab.com
  attribute 3 upn attr-type string match ben@a10lab.com
  attribute 4 role attr-type string match Gold
!
```

In the case of SAML Attribute-based Authorization, after the user authenticates successfully, ACOS will get an Assertion Token for this user. The identity provider will provide attributes for this user in the Assertion, and ACOS can use these attributes to authorize the user. Within a SAML Assertion, the attributes are contained in the <AttributeStatement>. For example:



```
- <AttributeStatement>
  - <Attribute Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name">
      <AttributeValue>Harry SP. Lin </AttributeValue>
    </Attribute>
  - <Attribute Name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress">
      <AttributeValue>splin@a10lab.com </AttributeValue>
    </Attribute>
  - <Attribute Name="http://schemas.microsoft.com/ws/2008/06/identity/claims/role">
      <AttributeValue>Gold </AttributeValue>
    </Attribute>
  </AttributeStatement>
```

Notes

- This document assumes both authentication and authorization are performed using SAML.
- SAML “<AuthorizationDecisionStatement>” support is not included. <AuthorizationDecisionStatement> asserts that a subject is permitted to perform action A on resource R given evidence E. The behavior is similar to an ACOS AAA policy that may need to check each user request URI, and ACOS authorization polices are designed for attribute-based authorization.
- XACML support is not included.

SAML Authentication and Authorization Flow

The following topics are covered:

Scenario

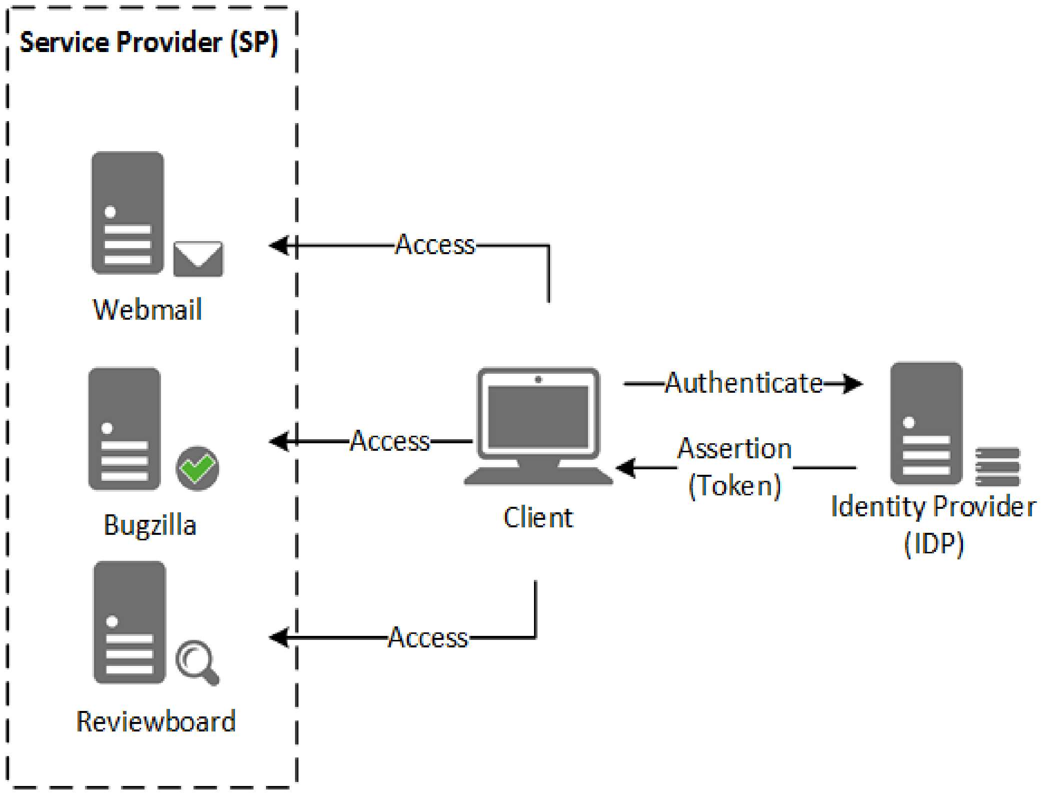
After a user is authenticated to one service, the user can access other services without having to authenticate again. When the client first accesses one of the services (or Service Provider), the Service Provider redirects the client to an authentication server (or Identity Provider). The Identity Provider asks for the user’s account and password to authenticate the client.

Following successful authentication, the Identity Provider redirects the client back to the service with a token (Assertion). T he Assertion will contain enough information ab out the client for the Service Provider to decide whether the client is allowed to access the service.

The next time the client wants to access other services, the client will be redirected to the IdP again. But since the client has already been authenticated by the IdP, the IdP will redirect the client directly back to the service with an Assertion, without having to repeat the authentication process.

Figure 46 illustrates the basic SAML-based SSO process. The user agent is the client's browser, the Service Provider is the ACOS device, and the IDP is the authentication server.

Figure 46 : SAML Process



Process

The following steps describe the process. Within an individual step, there may be one or more actual message exchanges, depending on which binding is used for that step and other implementation-dependent behavior.

HTTP Request is Sent to the Service Provider

The principal, working through an HTTP User Agent, makes an HTTP request for a secured

resource, which is located at the service provider (without a security context).

The Service Provider Determines the Identity Provider

The service provider obtains the location of an endpoint at the identity provider for the

authentication request protocol that supports its preferred binding.

<AuthnRequest> is Issued by the Service Provider and Sent to the Identity Provider

The service provider issues an <AuthnRequest> message, which is delivered by the user agent

to the Identity Provider. The HTTP Redirect, HTTP POST, or HTTP Artifact binding transfers the

message to the IdP through the user agent.

Identity Provider identifies the Principal

The principal is identified by the IdP by some means outside the scope of this profile. This could

possibly require re-authentication, or it could simply re-use an existing pre-authenticated

session.

Identity Provider Issues <Response> to the Service Provider

The identity provider issues a <Response> message, which is delivered via the user agent to the

service provider. Either the HTTP POST, or HTTP Artifact binding can be used to transfer the

message to the service provider through the user agent. The message may indicate an error, or

it could include (at the very least) an authentication assertion. The HTTP Redirect binding must

not have already been used, as the response will typically exceed the URL length permitted by

most user agents.

Service Provider Grants or Denies Access to the Principal

Having received the response from the identity provider, the service provider can respond to the

principal's user agent with its own error.

Configuring by Using the GUI

To configure a SAML authorization policy using the GUI, use the following steps.

The following topics are covered:

- Configuring the Policy
- Binding the Policy to an AAA-Policy Rule

Configuring the Policy

1. Click **AAM** > **Policies and Templates**.
2. On the **Authorization Policies** tab, click **Create**.
3. Click **Create**. (OK to leave Attribute Rule field empty for now.)
4. Define attributes. Refer to the online help for information on the fields listed on this GUI page.
5. After defining the policies, use them to configure the attribute rule. In the Attribute Rule field, enter the attributes and the logical operators to be used to compare them; example: **1 and 2 or 3.**
6. Click **Update**.

Binding the Policy to an AAA-Policy Rule

In the current release, the GUI does not support binding an authorization policy to an AAA policy.

The user can only configure this by CLI mode. For more information, see Configuring by Using the CLI.

Configuring by Using the CLI

To configure a SAML authorization policy, use the following commands.

The following topics are covered:

- Configuring the Policy
- Binding the Policy to an AAA-Policy Rule

Configuring the Policy

To configure the SAML authorization policy, use the following command:

```
ACOS(config)# aam authorization policy policy-name
attribute attr-num attr-name attr-type match-type value
```

The *attr-num* can be 1-32. The *attr-name* is a string. The *attr-type* must be string for SAML. The match type must be **match**. The *value* is the string.

Binding the Policy to an AAA-Policy Rule

To bind the SAML authorization policy to an AAA-policy rule, use the following commands:

```
ACOS(config)# aam aaa-policy policy-name
aaa-rule num
  authorize-policy policy-name
  action {allow | deny}
```

Configuration Example

The following topics are covered:

- SAML Authentication and Authorization
- SAML Authorization Statistics

SAML Authentication and Authorization

The following commands implement SAML authentication and authorization.

```
ACOS(config)# aam aaa-policy saml
  aaa-rule 1
    artifact-resolution-service index 0 location /Artifact/SOAP binding
soap
    assertion-consuming-service index 0 location /SAML2/POST binding post
    assertion-consuming-service index 1 location /SAML2/Artifact binding
artifact
```

```
    single-logout-service location /SLO/POST binding post
    certificate vip1
    entity-id https://vip1.a10.com/adfs
    soap-tls-certificate-validate disable
    service-url https://vip1.a10.com/adfs
!
aam authentication saml identity-provider idp-adfs
  metadata adfs-idp
!
slb server apache1 192.168.221.70
  port 80 tcp
!
slb server apache2 192.168.221.71
  port 80 tcp
!
aam authorization policy p1
  attribute 1 user attr-type string match Ben
  attribute 2 emailaddress attr-type string match ben@a10lab.com
  attribute 3 upn attr-type string match ben@a10lab.com
  attribute 4 role attr-type string match Gold
!
aam authentication template a-adfs
  type saml
  saml-sp sp-adfs
  saml-idp idp-adfs
  logout-idle-timeout 600
!
aam aaa-policy p-adfs
  aaa-rule 1
    action allow
    authentication-template a-adfs
    authorize-policy p1
!
slb template client-ssl c1
  cert ben_ax_cert1.pem
  key ben_ax_key1.pem
!
slb virtual-server vip1 192.168.91.56
  port 443 https
    source-nat auto
    template client-ssl c1
    service-group http-sg
    aaa-policy p-adfs
!
```

SAML Authorization Statistics

The following command shows SAML authorization statistics:

```
ACOS# show aam authentication statistics saml-sp
Name                MD-Export-Req MD-Export-Succ Auth-Req      Auth-Resp SSO-
Req SSO-Succ
                    Authz-Failed  SSO-Err        SLO-Req       SLO-Succ  SLO-
Err SP-SLO-Req
                    Glo-SLO-Succ  Loc-SLO-Succ   Par-SLO-Succ Other-Err
---------------------------------------------------------------------------
------------
asd                 0             0              0             0         0
 0
                    0             0              0             0         0
 0
                    0             0              0             0

ACOS# show aam authentication statistics
...
SAML statistic:
-----------------------
SP metadata export requests: 0
SP metadata export successes: 0
Login authn requests: 3
Login authn responses: 3
SSO requests: 3
SSO successes: 2
SSO authorization fail: 1
SSO errors: 0
SLO requests: 0
SLO succeses: 0
SLO errors: 0
SP-initiated SLO requests: 0
Global Logout successes: 0
```

```
Local Logout successes: 0
Partial Logout successes:0
Relay triggered: 0
Relay success: 0
Relay failed: 0
Relay RelayState error: 0
Other errors: 0
```

SAML Authentication with Kerberos Relay

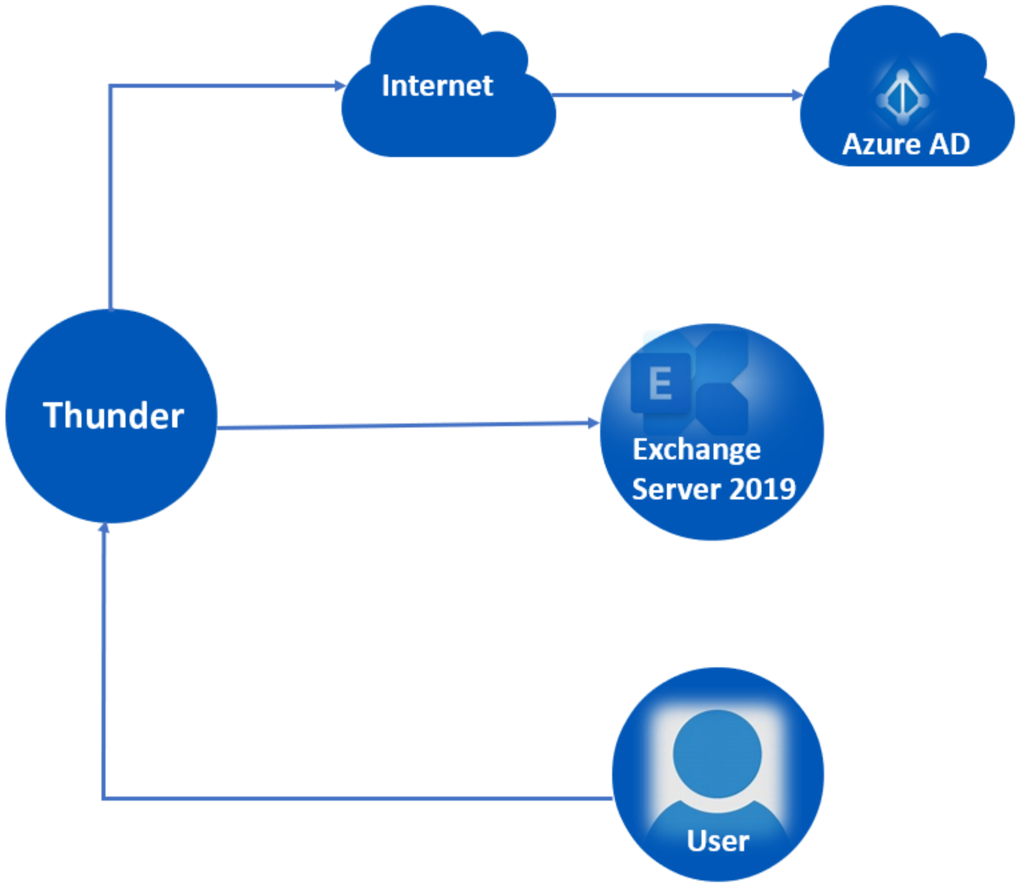
SAML authentication with Kerberos relay provides the capability to integrate Microsoft Azure Active Directory (AD) with AAM SAML authentication and on-premises Exchange 2019 with Kerberos authentication.

The **`username-attribute`** option under the **`aam authentication saml service-provider`** command identifies and differentiates individual users within the database. It retrieves the username used in Kerberos relay and allows the access for both on-premises Exchange 2019 and Microsoft Azure Active Directory (AD) through Single Sign-on.

Topology

The following is a representation of the AAM integration with SAML authentication and Kerberos relay topology:

Figure 47 : SAML Authentication and Kerberos Relay Topology



The following steps indicate the flow of the AAM integration with SAML authentication and Kerberos relay process:

1. A user sends a request to access a Thunder device.
2. The Thunder starts the SAML authentication process on the Microsoft Azure AD through internet.
3. Azure AD uses the SAML authentication for user authentication and validation. After successful authentication, the Thunder extracts the assertion and gets the username information.
4. The Thunder uses the username information and starts the Kerberos authentication process on the on-premises Exchange 2019 server.
5. The Exchange 2019 server authenticates and completes Kerberos relay process.

CLI Configuration

To identify and differentiate individual users within the database, the **`username-attribute`** command is added under **`aam authentication saml service-provider`**.

1. Configure the SAML Service Provider (SP).

   The **`username-attribute`** option is required to retrieve the username used in Kerberos relay.

```
ACOS(config)#aam authentication saml service-provider AD_SAML_SSO_SP
ACOS(config-SAML saml service-provider AD_SAML_SSO_SP)# assertion-
consuming-service index 0 location /saml/login binding post
ACOS(config-SAML saml service-provider AD_SAML_SSO_SP)# SP-initiated-
single-logout-service location /saml/logout
ACOS(config-SAML saml service-provider AD_SAML_SSO_SP)# entity-id
https://ad-saml-sso.a10-tplab.com
ACOS(config-SAML saml service-provider AD_SAML_SSO_SP)# service-url
https://vk-test-adc.a10-tplab.com
ACOS(config-SAML saml service-provider AD_SAML_SSO_SP)# username-
attribute usernamex
```

NOTE: The user account name must be the same in Azure AD and Exchange 2019 to support the Kerberos relay method.

2. Import a SAML Identity Provider (IdP) metadata used in the SAML IdP configuration.

```
ACOS(config)#import auth-saml-idp AD_SAML_SSO_IDP_METADATA
tftp://10.12.10.222:/AD_SAML_SSO_IDP_METADATA.xml
```

3. Configure a SAML Identity Provider (IDP).

```
ACOS(config)#aam authentication saml identity-provider AD_SAML_SSO_IDP
ACOS(config-saml identity-provider AD_SAML_SSO_IDP)#metadata AD_SAML_
SSO_IDP_METADATA
```

4. Configure a Kerberos relay.

```
ACOS(config)#aam authentication relay kerberos KRB
ACOS(config-kerberos auth relay:KRB)# kerberos-realm A10-TPLAB.COM
ACOS(config-kerberos auth relay:KRB)# kerberos-kdc 10.12.10.221
ACOS(config-kerberos auth relay:KRB)# kerberos-account userx
ACOS(config-kerberos auth relay:KRB)# password userP$wd
```

5. Configure an Authentication template.

```
ACOS(config)#aam authentication template AD_SAML_SSO
ACOS(config-auth template:AD_SAML_SSO)# aam authentication template AD_
SAML_SSO
ACOS(config-auth template:AD_SAML_SSO)# type saml
ACOS(config-auth template:AD_SAML_SSO)# saml-sp AD_SAML_SSO_SP
ACOS(config-auth template:AD_SAML_SSO)# saml-idp AD_SAML_SSO_IDP
ACOS(config-auth template:AD_SAML_SSO)# relay KRB
ACOS(config-auth template:AD_SAML_SSO)# log enable
```

6. Configure a Policy.

```
ACOS(config)#aam aaa-policy AD_SAML_SSO
ACOS(config-aaa policy:AD_SAML_SSO)# aaa-rule 1
ACOS(config-aaa policy:AD_SAML_SSO-aaa-rule 1)# action allow
ACOS(config-aaa policy:AD_SAML_SSO-aaa-rule 1)# authentication-template
AD_SAML_SSO
```

7. Configure SLB server that requires Kerberos authentication.

   The **`service-principal-name`** option is required for Kerberos relay.

```
ACOS(config)#slb server KRB_AUTHEN_SERVER 10.12.10.221
ACOS(config-real server)# health-check-disable
ACOS(config-real server)# port 443 tcp
ACOS(config-real server-node port)# health-check-disable
ACOS(config-real server-node port)# service-principal-name
HTTPS/ad01.a10-tplab.com
```

8. Configure SLB service group.

```
ACOS(config)# slb service-group KRB_AUTHEN_SERVER tcp
ACOS(config-slb svc group)# health-check-disable
ACOS(config)# member KRB_AUTHEN_SERVER 443
```

9. Configure SLB templates.

```
ACOS(config)# slb template server-ssl SVR_SSL
ACOS(config-server ssl)# exit
ACOS(config)# slb template client-ssl SSL
ACOS(config-client ssl)# certificate ADC_SSL key ADC_SSL
```

10. Configure SLB virtual server and ports.

```
ACOS(config)# slb virtual-server AD_WEB 10.251.2.81
ACOS(config-slb vserver)# port 443 https
ACOS(config-slb vserver-vport)# source-nat auto
ACOS(config-slb vserver-vport)# service-group KRB_AUTHEN_SERVER
ACOS(config-slb vserver-vport)# template server-ssl SVR_SSL
ACOS(config-slb vserver-vport)# template client-ssl SSL
ACOS(config-slb vserver-vport)# aaa-policy AD_SAML_SSO
```

Use the `aaa-policy` option to bind the AAA policy to the virtual port.

Configuration Example

The following is a representation of the examples of configuring SAML service provider, SAML identity provider, and Kerberos relay:

```
!
aam authentication saml service-provider AD_SAML_SSO_SP
assertion-consuming-service index 0 location /saml/login binding post
SP-initiated-single-logout-service location /saml/logout
entity-id https://ad-saml-sso.a10-tplab.com
service-url https://vk-test-adc.a10-tplab.com
username-attribute username
!
aam authentication saml identity-provider AD_SAML_SSO_IDP
metadata AD_SAML_SSO_IDP_METADATA
!
!
aam authentication relay kerberos KRB
kerberos-realm A10-TPLAB.COM
kerberos-kdc 10.12.10.221
kerberos-account userx
password userP$wd
!
aam authentication template AD_SAML_SSO
type saml
saml-sp AD_SAML_SSO_SP
saml-idp AD_SAML_SSO_IDP
relay KRB
log enable
user-tag "SAML authentication + Kerberos relay SSO"
```

```
!
aam aaa-policy AD_SAML_SSO
aaa-rule 1
action allow
authentication-template AD_SAML_SSO
!
slb server KRB_AUTHEN_SERVER 10.12.10.221
health-check-disable
user-tag "A Server with Kerberos authentication"
port 443 tcp
health-check-disable
service-principal-name HTTP/ad01.a10-tplab.com
!
slb service-group KRB_AUTHEN_SERVER tcp
health-check-disable
member KRB_AUTHEN_SERVER 443
!
slb template server-ssl SVR_SSL
!
slb template client-ssl SSL
certificate ADC_SSL key ADC_SSL
!
slb virtual-server AD_WEB 10.251.2.81
port 443 https
source-nat auto
service-group KRB_AUTHEN_SERVER
template server-ssl SVR_SSL
template client-ssl SSL
aaa-policy AD_SAML_SSO
!
```

SAML Authentication for Forward Proxy Client

The following topics are covered:

Overview

In the forward proxy environment, ACOS supports the SAML authentication for the client to connect to a third-party Identity Provider (IdP) server (Microsoft Azure Active Directory") through ACOS to authenticate and to protect users' data and resources as part of Zero Trust Architecture. You must configure an `ac` class-list that includes the domains of IdP and bind the class-list to the `aaa-policy rule` to allow ACOS to forward the SAML authentication traffic to IdP. The new `domain-whitelist` command is added to bind **class-list** in the **aaa rule**.

The proxy chaining provides a solution for “Cloud Access Proxy”. It controls traffic intended for the cloud services from bypassing the existing proxy. After successful SAML authentication, it creates the authentication session in the forward proxy environment, and then the traffic intended for the cloud services passes through the existing proxy.

Consider the following points:

- When a basic authentication is used with forward proxy, the username and password are not encrypted (because the protocol of forward proxy is "http" instead of "https") though the first access is HTTPS (CONNECT).
- If a form authentication is used with SSLi, the username and password are encrypted only if the client's first access is HTTPS and it can be decrypted using SSLi. If the client's first access is HTTP, the username and password are not encrypted.
- The SAML authentication secures the authentication process with HTTPS. The Multi-Factor Authentication (MFA) can be deployed if the IdP server supports it.

Workflow

Figure-1 and Figure 49, illustrate the flow of the SAML authentication for client to connect to a third-party identity provider.

Figure 48 : SAML authentication for client to connect to a third-party Identity Provider

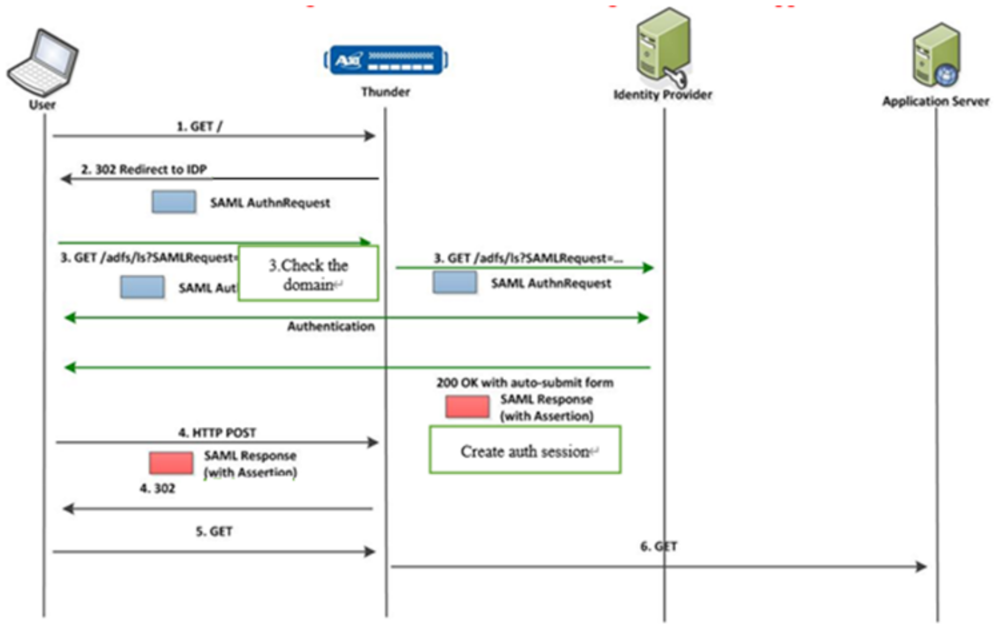
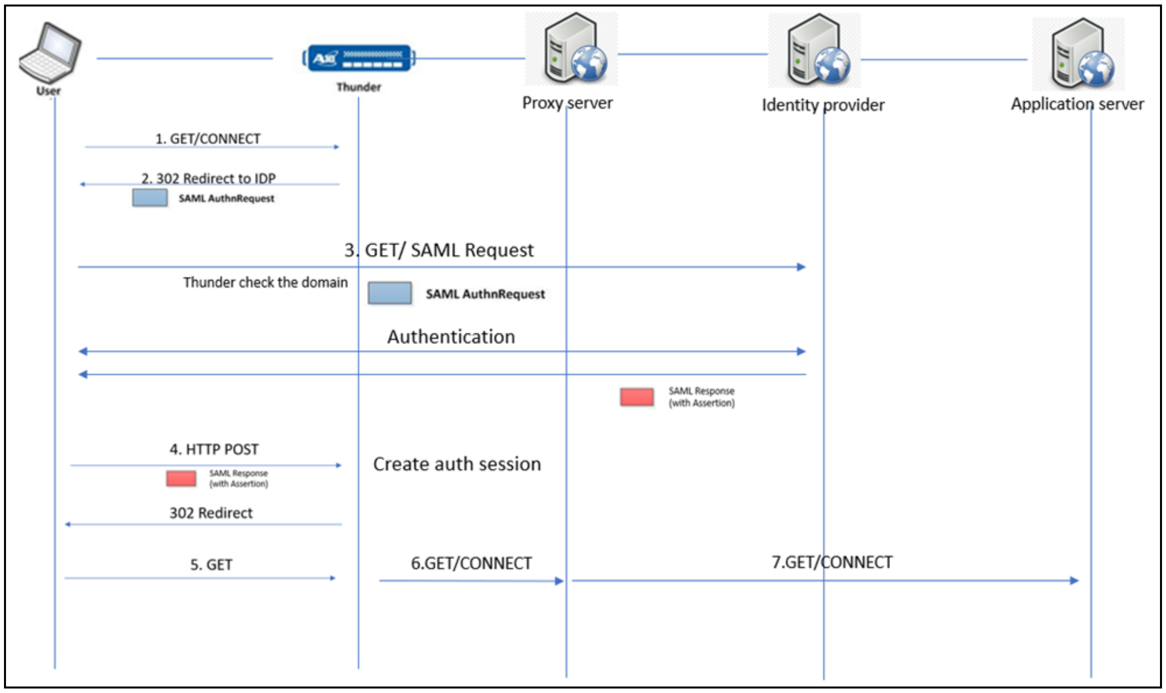


Figure 49 : SAML authentication with proxy chaining



The following steps provide a high-level view of the process:

1. When a user accesses the internet through a proxy, it sends a request to ACOS.
2. AAM sends a redirected 302 packet to the user and redirects the user to the IdP server to authenticate.
3. When the user accesses the IdP server, ACOS verifies the domain, and if the domain is in the allowed list, then traffic is bypassed and the user can access the IdP server.
4. When the user performs the SAML authentication through the IdP server, it sends the SAML response to ACOS. ACOS verifies the response, if the response is valid, sends the 302 packet to the user and redirects to the service provider domain. ACOS creates an authentication session for the user.
5. After SAML authentication, in the forward proxy mode, the traffic will be forwarded to the application server. Whereas in the proxy chaining mode, the traffic will be forwarded to the proxy server.
6. User obtains resource from the internet.

SAML authentication for forward proxy functions with Azure Active Directory (AD). To view and get the domain list for the Azure IdP, click Microsoft listed domains. You can configure the domain list to the class-list for Azure IdP server.

CLI Configuration

Configure the whitelist domain class-list and bind it to the AAM authentication template.

1. To configure the class-list, enter the following commands:

```
ACOS(config)# class-list whitelist1 ac
ACOS(config-class list)# contains domain1
ACOS(config-class list)# contains domain2
```

2. To enable the SAML forward proxy mode, first, configure the class-list with the Microsoft listed domains, and then bind it to the `domain-whitelist` under the **aaa-policy**.

   The CLI configuration is:

```
ACOS(config)# class-list c1 ac
ACOS(config-class list)# contains login.windows.net
ACOS(config-class list)# contains secure.aadcdn.microsoftonline-p.com
ACOS(config-class list)# contains microsoftonline.com
ACOS(config-class list)# contains microsoftonline-p.com
ACOS(config-class list)# contains msauth.net
ACOS(config-class list)# contains msauthimages.net
ACOS(config-class list)# contains msecnd.net
ACOS(config-class list)# contains msftauth.net
ACOS(config-class list)# contains msftauthimages.net
ACOS(config-class list)# contains phonefactor.net
ACOS(config-class list)# contains enterpriseregistration.windows.net
ACOS(config-class list)# contains management.azure.com
ACOS(config-class list)# contains policykeyservice.dc.ad.msft.net
ACOS(config-class list)# contains ctldl.windowsupdate.com
```

Bind the domain whitelist to the **aaa-policy**.

```
ACOS(config)# aam aaa-policy policy1
ACOS(config-aaa policy:policy1)# aaa-rule 1
ACOS(config-aaa policy:policy1-aaa rule:1)# action allow
ACOS(config-aaa policy:policy1-aaa rule:1)# domain-whitelist c1
```

Finally, bind the SAML authentication template to the **aaa-policy**.

```
ACOS(config-aaa policy:policy1)# aaa-rule 2
ACOS(config-aaa policy:policy1-aaa rule:2)# action allow
ACOS(config-aaa policy:policy1-aaa rule:2)# authentication-template
saml
```

3. For the SAML with forward proxy mode, you must set SSLi bypass for the domain whitelist configured using the **forward-proxy-bypass** command.

   For example, see the CLI configuration:

```
ACOS(config)# slb template client-ssl ep
ACOS(config-slb template client-ssl ep)# forward-proxy-bypass class-
list c1
```

Show Command

To view AAM authentication statistics, use the following command:

```
show aam authentication statistics
```

The following new counters are added in the show aam authentication statistics command under the **`Engine statistics`** category.

- `Domain whitelist matched`
- `Domain whitelist unmatched`

If the host of request matches the whitelist, the domain whitelist matched counter will be increased by one. Else, domain whitelist unmatched will be increased by one.

Example:

```
ACOS(config)# show aam authentication statistics
            Engine statistics:
            -----------------------
            Requests to Auth Daemon: 0
            Responses from Auth Daemon: 0
            Requests to SAML: 0
            Responses from SAML: 0
            Misses: 0
            OCSP Stapling Requests to Auth Daemon: 0
            OCSP Stapling Responses from Auth Daemon: 0
            Domain whitelist matched: 0
            Domain whitelist unmatched: 0

            Engine aFleX statistics:
            -----------------------
            Authorize success: 0
            Authorize failure: 0
            ...
```

Limitations

- ACOS will be forced to IP based mode under SAML forward proxy.
- For the SAML with forward proxy mode, you must set SSLi bypass for the domain whitelist configured using the **`forward-proxy-bypass`** command.
- The proxy chaining feature is only supported in the forward proxy mode.

ADFS WS-Federation for Microsoft SharePoint

The following topics are covered:

WS-Federation

The following topics are covered:

Scenario

Popular Microsoft applications, such as SharePoint and Exchange, typically use Web Services Federation Language (WS-Federation) as their single-sign-on (SSO) standard for authenticating clients. WS-Federation is a popular SSO standard which is similar to SAML, but is essentially incompatible with SAML. It was developed by a group of companies, chiefly Microsoft.

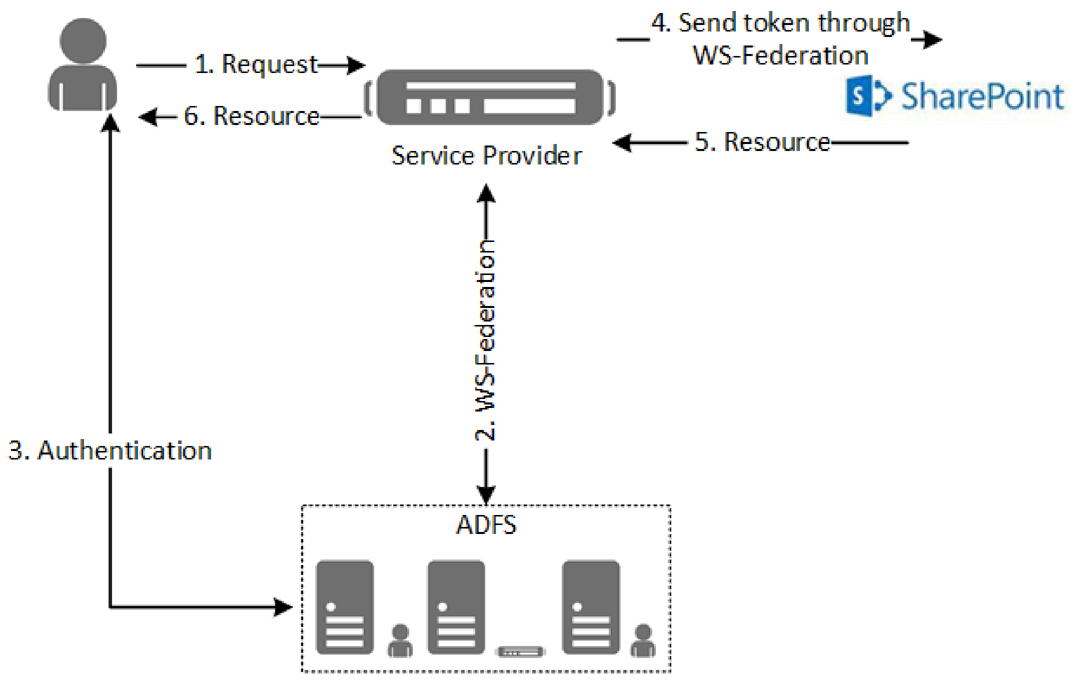
WS-Federation supports various token types provided by identity providers. For example, SAML assertion is a common token type that is accepted by many application services.

While SharePoint and Exchange do not support the ability to receive SAML 2.0 assertions, they do support the ability to receive the older SAML 1.1 tokens. Therefore, in order to support SSO for SharePoint and Exchange, ACOS 4.0.1 adds WS-Federation support for the ability to relay the client's SAML token to the back-end Microsoft applications.

To initiate the WS-Federation process, the ACOS device must communicate with active directory federation services (ADFS) via WS-Federation in order to obtain the SAML version 1.1 token and relay it to SharePoint. In addition, token relay must be configured to ensure that once the ACOS device receives the SAML assertion (either v1.1 or v2.0), the ACOS device will relay the token to the application on the backed server. ADFS will only reply to SAML v1.1 tokens if WS-Federation is configured.

The ACOS device will relay the token to the application on the backed server. The application server can then recognize the client's token and grant the client access to the restricted network resources. This flow is shown in Figure 50.

Figure 50 : WS-Federation with ADFS and token relay with SharePoint



Steps

The following steps describe the process. Within an individual step, there may be one or more actual message exchanges.

1. The client sends a HTTP request to a restricted network resource (such as SharePoint).
2. The ACOS device redirects the client request to ADFS.

3. The client sends the credentials to ADFS, at which point ADFS verifies the client's credentials.
4. The ADFS sends the token to the client, which is then sent to the ACOS device, to be relayed to the Sharepoint server.
5. There is no direct communication between ADFS and the ACOS with this configuration.
6. The SharePoint server verifies the token that was relayed from the ACOS device.
7. If the token verification passes, the SharePoint responds with its portal.
8. The ACOS device forwards the SharePoint portal back to the client.

How ACOS handles the Client Request

1. The client sends an HTTP GET request to the ACOS VIP.
2. ACOS transforms the request message into format that SAML accepts, based on the information configured in the authentication for the SAML service provider and the identity provider.
3. SAML generates a WS-Federation “Requesting Security Tokens” request based on a passive requester profile and sends it back to the ACOS device. This request will redirect the client to the Identity Provider/Security Token Service (IP/STS) which, in this case, is ADFS.
4. The ACOS device forwards the “Requesting Security Tokens” request back to the client.

How ACOS handles the ADFS Response

1. Once ADFS authenticates the client, it sends “Returning Security Tokens” message to the client. This is a redirect message to automatically redirect the client back to the ACOS VIP. The “Returning Security Tokens” message contains SAML V1.1 assertion/token, which can be used by the ACOS device to relay to SharePoint.
2. The ACOS device transforms the request message into a format that SAML accepts, based on the information configured in the authentication template for the SAML service provider and identity provider.

3. SAML verifies “Returning Security Tokens” and sends the token and authentication status back to the ACOS device.
4. The ACOS device relays the token embedded in “Returning Security Tokens” to SharePoint. This message must be transformed into WS-Federation “Returning Security Tokens” format because the ACOS device needs to make it appear to the SharePoint server that this is a trust token from ADFS.
5. SharePoint verifies the token and responds with its portal.
6. The ACOS device forwards the portal back to the client.

Configuring by Using the CLI

The following topics are covered:

To configure a ADFS WS-Federation, use the following commands.

Configuring the Policy

To enable ADFS WS-Federation in the SAML service provider (i.e., the ACOS device), use the following command:

```
ACOS(config)# aam authentication saml service-provider provider-name
adfs-ws-federation enable
```

The *provider-name* must be a string.

Configuring Token Relay with new Type WS-Federation

To configure token relay for WS-Federation, use the following commands:

```
ACOS(config)# aam authentication relay ws-federation relay-name
```

The *relay-name* must be a string.

Configuration Example for Microsoft SharePoint

The following topics are covered:

The following commands are used to implement ADFS WS-Federation and token relay with SharePoint:

```
Configuring AAM
First, you must configure AAM:
ACOS(config)# import auth-saml-idp adfsv2
scp://root:password@192.168.100.168/root/user_tmp/federationmetadata_100_
159.xml
Done.
ACOS(config)# show run aam
!Section configuration: 715 bytes
!
aam authentication saml service-provider sharepoint_sp
  adfs-ws-federation enable
  assertion-consuming-service index 0 location /_trust/ binding post
  entity-id https://sharepoint2010.aamtest.com
  service-url https://sharepoint2010.aamtest.com
!
aam authentication saml identity-provider adfsv2
  metadata adfsv2
!
aam authentication relay ws-federation sharepoint_relay
application-server sharepoint
  authentication-uri /_trust/
!
aam authentication template sharepoint_template
  type saml
  saml-sp sharepoint_sp
  saml-idp adfsv2
  relay sharepoint_relay
!
aam aaa-policy sharepoint_policy
  aaa-rule 1
    authentication-template sharepoint_template
```

Configuring SLB and Applying WS-Federation Relay

Next, you must configure SLB and apply WS-Federation relay onto the virtual port (via the aaa-policy):

```
ACOS(config)# show run ip nat pool
Section configuration: 70 bytes
!
ip nat pool pool3_ipv4 172.128.10.104 172.128.10.104 netmask /24
!
end

ACOS(config)# show run slb
!Section configuration: 836 bytes
!
slb template client-ssl c-ssl
  ca-cert cacert
  cert ACOS
  key ACOS
!
slb template server-ssl s-ssl
  ca-cert cacert
  cert ACOS
  key ACOS
!
slb server s170 172.128.10.170
  port 80 tcp
  port 443 tcp
!
slb service-group aam-sg-http tcp
  member s170 80
!
slb service-group aam-sg-https tcp
  member s170 443
!
slb virtual-server vip4 172.128.9.104
  port 80 http
    source-nat pool pool3_ipv4
    service-group aam-sg-http
  port 443 https
    source-nat pool pool3_ipv4
    service-group aam-sg-https
    template client-ssl c-ssl
    aaa-policy sharepoint_policy
```

Show Commands

The following command shows commands and sample output for checking WS-Federation Relay statistics.

```
ACOS# show aam authentication session
TTL = Session Idle timeout (Sec), Lifetime = SAML Token/Assertion Lifetime
(Sec)
Total Sessions: 1
---------------------------------------------------------------------------
-----
ID      Type   VIP             User            Client IP       Created Time
        VPort  Domain          Domain-group    TTL             Lifetime
---------------------------------------------------------------------------
------
36      SAML   vip4            a@aamtest.com   172.128.9.171   04-03-15
10:08:03
        443    .aamtest.com    31              290             3531

ACOS# show aam authentication saml sp-session
Service Provider: sharepoint_sp
NameID          Client addr          Id Provider
     Auth Instant                 Expiration time
---------------------------------------------------------------------------
------
a@aamtest.com 172.128.9.171
http://adfsv2.aamtest.com/adfs/services/trust 2015-03-04T02:07:50.575Z
2015-03-04 11:07:50

ACOS# show aam authentication statistics
A10LB statistics:
-----------------------
Requests to A10Authd: 0
Responses from A10Authd: 0
Requests to A10SAML: 2
Responses from A10SAML: 2
Misses: 0
OCSP Stapling Requests to A10Authd: 0
OCSP Stapling Responses from A10Authd: 0
...
```

```
SAML statistic:
-----------------------
SP metadata export requests: 0
SP metadata export successes: 0
Login authn requests: 1
Login authn responses: 1
SSO requests: 1
SSO successes: 1
SSO authorization failed: 0
SSO errors: 0
SLO requests: 0
SLO succeses: 0
SLO errors: 0
SP-initiated SLO requests: 0
Global Logout successes: 0
Local Logout successes: 0
Partial Logout successes:0
Relay triggered: 0
Relay success: 0
Relay failed: 0
Relay RelayState error: 0
Other errors: 0
```

Configuration Example for Microsoft Exchange

The following topics are covered:

Scenario

The following commands are used to implement ADFS WS-Federation and token relay with Exchange. The example is mostly the same as the previous example for SharePoint, in terms of the configurations on the ACOS device. The only difference lies in the configuration for the application-server, which is found under the aam authentication relay configuration.

This must be configured as “exchange-owa” for MS Exchange instead of “sharepoint” for MS SharePoint, as shown below.

However, to deploy the feature with MS Exchange, some additional configurations are required on the ADFS side and MS Exchange side.

For more information, see the A10 Networks deployment guide, *Deploy OWA WS-Federation SSO with AX*. (Sections: “Changes on ADFS” and “Change on Exchange Server”).

NOTE: For more information on the configurations required on ADFS and Exchange, see the Microsoft website https://www.microsoft.com.

CLI Example

```
aam authentication saml service-provider owa_sp
  adfs-ws-federation enable
  assertion-consuming-service index 0 location /axowa binding post
  entity-id https://webmail.win2008r2.com/axowa
  service-url https://webmail.win2008r2.com
!
aam authentication saml identity-provider adfsv2_idp
  metadata adfsv2_metadadta_v21
!
aam authentication relay ws-federation owa_relay
  application-server exchange-owa       <-- main difference between
SharePoint and Exchange
  authentication-uri /owa
!
aam authentication template owa_template
  type saml
  saml-sp owa_sp
  saml-idp adfsv2_idp
  logout-url /auth/logoff.aspx
  relay owa_relay
  log enable
!
aam aaa-policy owa_policy
  aaa-rule 1
    action allow
    authentication-template owa_template
```

```
!
ip nat pool pool3_ipv4 172.128.10.104 172.128.10.104 netmask /24 vrid 31
!
slb server exchange2010_owa 172.128.10.254
  port 443 tcp
!
slb service-group aam-owa-sg-https tcp
  member exchange2010_owa 443
!
slb template client-ssl c-ssl
  ca-cert cacert
  cert ACOS
  key ACOS
!
slb template server-ssl s-ssl
  ca-cert cacert
  cert ACOS
  server-certificate-error logging
  server-certificate-error ignore
  key ACOS
!
slb virtual-server vip5 172.128.9.105
  vrid 31
  port 443 https
    source-nat pool pool3_ipv4
    service-group aam-owa-sg-https
    template server-ssl s-ssl
    template client-ssl c-ssl
    aaa-policy owa_policy
!
```

SAML Statistics

The following topics are covered:

You can display statistics about your SAML configuration at one of the following levels:

- Globally, as a part of your entire AAM configuration
- Granularly, for each SAML object in your configuration

Displaying SAML Statistics by Using the CLI

To display SAML statistics, enter the following command:

```
show aam authentication statistics saml-sp ?
```

For more information about this CLI command, see the *CLI Reference* guide.

Displaying SAML Statistics by Using the GUI

To display SAML statistics:

1. Click **AAM** > **Auth Client**.
2. On the **SAML Service Provider** tab, under the **Statistics Details** column, click **Stats**.

Clearing SAML Statistics

The following topics are covered:

You can clear SAML statistics by using the CLI or the GUI.

Clearing SAML Statistics by Using the CLI

To clear SAML statistics, enter the following command:

```
clear aam authentication statistics saml-sp ?
```

NOTE: For more information about this CLI command, see the *CLI Reference* guide.

CLI Examples

The following text is an example of SAML statistics at a global level:

```
ACOS# show aam authentication statistics
A10LB statistics:
-----------------------
Requests to A10Authd: 0
Responses from A10Authd: 0
Requests to A10SAML: 0
Responses from A10SAML: 0
Misses: 0
OCSP Stapling Requests to A10Authd: 0
OCSP Stapling Responses from A10Authd: 0
...

SAML statistic:
-----------------------
SP metadata export requests: 0
SP metadata export successes: 0
Login authn requests: 0
Login authn responses: 0
ACS requests: 0
ACS successes: 0
ACS errors: 0
SLO requests: 0
SLO succeses: 0
SLO errors: 0
SP-initiated SLO requests: 0
Global Logout successes: 0
Local Logout successes: 0
Partial Logout successes:0
Relay triggered: 0
Relay success: 0
Relay failed: 0
Relay RelayState error: 0
Other errors: 0
```

Feedback

SAML SP Initiated Single Log Out

The following topics are covered:

Overview

The service provider (SP) can log out from an ACOS device in the SAML SP-initiated SLO endpoint. Both the SP and identity provider (IDP) will log out and clear the sessions. The user requires re-authentication for accessing the server. There are two types of session log out:

- **Global Logout** - Both the IDP server and Thunder device removes their sessions successfully.
- **Partial Logout** – The IDP server may refuse the logging out event or the request from Thunder is lost, so only the Thunder device removes the session.

IDP Server Modes

- Synchronous mode

  When the IDP server is in synchronous mode, it will reply the response to the client for the log out request, and the response is redirected to SP through the client. The ACOS device treats this situation as global logout.

- Asynchronous mode

  When the IDP server is in the asynchronous mode, different types of IDP server behave differently:

  - Type 1

    IDP server manages the log out request event from the client but does not give any response, so the SP does not get the response from IDP as well. Then the ACOS treats this situation as partial logout.

- Type 2

  IDP server may reject the log out request event from the client and give an error message which can be redirected to SP. Then the ACOS treats this situation as partial logout.

- Type 3

  IDP server forces to reply the response to client for log out request, IDP still rejects the log out request. Then the process for SP is the same as synchronous mode, so at last, this situation is treated as global logout.

AAM with OCSP

Online Certificate Status Protocol (OCSP) is a network component that provides certificate verification services.

The following topics are covered:

Feedback

Overview

You can use OCSP to verify client certificates for access to an HTTPS virtual port. ACOS uses OCSP to authenticate the client, instead of a certificate revocation list (CRL) that is imported to the ACOS device.

When this feature is configured, the ACOS device acts as an authentication client in relation to the OCSP responder (server).

NOTE: Transport Layer Security (TLS) 1.3 is not supported in AAM OCSP and OCSP Stapling.

The following topics are covered:

Certificate Verification Process

You can configure ACOS to use external OCSP responders to verify client certificates. When OCSP support is configured, ACOS verifies client certificates in the following way:

1. When a client sends a request to set up an SSL session, ACOS sends the certificate sent by the client to an OCSP responder for validation.

   If the responder is a member of a service group, ACOS first uses the configured load-balancing method to select a responder and sends the request to that responder.

2. The OCSP responder checks its CRL database to determine whether the certificate is valid or has been revoked.
3. The responder sends the verification result to ACOS.

4. ACOS caches the response in one of the following ways:
   - If the certificate is valid, ACOS completes the SSL session setup with the client.
   - If the certificate is invalid, ACOS does not complete the SSL handshake with the client.

ACOS Verification of Replies from OCSP Responder

As part of the session setup with the OCSP responder, ACOS receives a copy of the responder’s certificate. To verify the identity of the responder, ACOS checks the responder’s certificate.

To check the OCSP responder’s certificate, ACOS needs a copy of one of the following certificate types for the responder:

CA-Signed Certificate

OCSP responder’s certificate is signed by a root CA.

Intermediate Certificate

OCSP responder’s certificate is signed by an intermediate CA.

NOTE: As part of configuration for the OCSP support, you must import the certificates to the ACOS device.

Using a root CA as a CA-certificate depends on the OCSP responder’s configuration. The OCSP responder may use different certificate or CA certificate to sign the OCSP response. You must specify the certificate and/or the CA certificate used in the OCSP responder.

You can configure a certificate bundle in `responder-cert`, if the bundle contains the certificate used in the OCSP responder.

Creating an OCSP Authentication Server

To create an OCSP authentication server by using the GUI or CLI, see Creating Authentication Servers.

The following topics are covered:

OCSP Configuration Example with Single Server

This topic provides a configuration example that uses one OCSP server.

Configuration

The following topics are covered:

- Deployment Resources
- CLI Example for Importing Server
- Authentication-Server Profile
- Configuring the Client-SSL Template
- Adding VIP
- CLI Example
- Server Certificate and Key
- Authentication-Server Profile
- Client-SSL Template
- SLB Configuration

Deployment Resources

The deployment requires the following resources:

- Server certificate and key files that ACOS can present to clients during an SSL session set up between ACOS and the clients
- An authentication-server profile for the OCSP server
- A client-SSL template
- A server configuration and a service group for the services that are requested by clients
- A VIP configuration

CLI Example for Importing Server

The following commands import a server certificate and key to the ACOS device:

```
ACOS(config)# import ssl-cert-key bulk use-mgmt-port sftp:
Address or name of remote host []?fileserver1
Username []?admin1
Password []?********
File name [/]?server.pk7
...
```

ACOS presents the server certificate and public key to clients as verification of the server identity. From the client’s perspective, the ACOS device where the VIP requested by the client is configured is the server.

Authentication-Server Profile

The following commands create an authentication-server profile for the OCSP server:

NOTE: The path to an OCSP server in a URL is in the authentication requests.

```
ACOS(config)# aam authentication server ocsp OCSP1
ACOS(config-ocsp server)# url http://10.10.10.5:778/
ACOS(config)# exit
```

The **url** command specifies the IP address and protocol port to which ACOS sends client certificates for verification.

Configuring the Client-SSL Template

The following commands configure the client-SSL template:

```
ACOS(config)# slb template client-ssl ssl1
ACOS(config-client ssl)# cert server
ACOS(config-client ssl)# key server
ACOS(config-client ssl)# ca-cert ca ocsp ocsp1
ACOS(config-client ssl)# client-certificate Require
```

The cert and key commands specify the local filenames for the certificate and key files that are imported to the ACOS device. In this example, both the certificate and key are imported as one file, so the same filename is used for the certificate and key.

Adding VIP

The following commands add the VIP:

```
ACOS(config)# slb virtual-server http 2.1.0.100
ACOS(config-slb vserver)# port 443 https
ACOS(config-slb vserver-vport)# service-group http
ACOS(config-slb vserver-vport)# template client-ssl ssl1
ACOS(config)# exit
```

CLI Example

This configuration is similar to the configuration in OCSP Configuration Example with Single Server, with the addition of SLB server configurations and a service group for the OCSP servers.

ACOS load balances certificate verification requests among the OCSP responders in the service group.

Server Certificate and Key

The following commands import a server certificate and key to the ACOS device:

```
ACOS(config)# import ssl-cert-key bulk use-mgmt-port sftp:
Address or name of remote host []?fileserver1
Username []?admin1
Password []?********
File name [/]?server.pk7
...
```

As verification of the server identity, ACOS presents the server certificate and public key to clients. From the client’s perspective, the ACOS device where the VIP requested by the client is configured is the server.

Authentication-Server Profile

The following commands create an authentication-server profile for each OCSP server:

```
aam authentication server ocsp OCSP1
url http://{host | ip} [:port]/
aam authentication server ocsp OCSP2
```

```
url http://{host | ip} [:port]/
aam authentication service-group ocsp tcp
member OCSP1 {port}
member OCSP2 {port}
```

The `url` command specifies the IP address and protocol port to which ACOS sends client certificates for verification.

Client-SSL Template

The following commands configure the client-SSL template. This is similar to OCSP Configuration Example with Single Server, except that the **ca-cert** command refers to the OCSP service group instead of an individual authentication-server profile for a OCSP server.

```
ACOS(config)# slb template client-ssl ssl1
ACOS(config-client ssl)# cert server
ACOS(config-client ssl)# key server
ACOS(config-client ssl)# ca-cert ca ocsp service-group ocsp
ACOS(config-client ssl)# client-certificate Require
ACOS(config-client ssl)# exit
```

SLB Configuration

The following commands add the SLB configuration for the web servers. This is the same configuration that is used in OCSP Configuration Example with Single Server.

```
ACOS(config)# slb server s1 20.20.20.30
ACOS(config-real server)# port 80 tcp
ACOS(config-real server-node port)# exit
ACOS(config-real server)# exit
ACOS(config)# slb server s2 20.20.20.31
ACOS(config-real server)# port 80 tcp
ACOS(config-real server-node port)# exit
ACOS(config-real server)# exit
ACOS(config)# aam authentication service-group sg1 tcp
ACOS(config-aam svc group:sg1)# member s1 80
ACOS(config-slb svc group:sg1)# member s2 80
ACOS(config-slb svc group:sg1)# exit
ACOS(config)# slb virtual-server http 2.1.0.100
ACOS(config-slb vserver)# port 443 https
ACOS(config-slb vserver-vport)# service-group http
ACOS(config-slb vserver-vport)# template client-ssl ssl1
```

OCSP Configuration Example with Multiple Servers

The configuration example in this topic uses a group of multiple OCSP servers.

Configuration

The following topics are covered:

- Deployment
- CLI Example for the Deployment
- Configuring the OCSP Service-Groups

Deployment

The deployment requires the following resources:

- Server certificate and key files, for ACOS to present to clients during an SSL session set up between ACOS and the clients
- An authentication-server profile for each OCSP server
- Server configurations and service group for the OCSP servers
- A client-SSL template
- Server configurations and service group for the services that are requested by clients
- A VIP configuration

NOTE: The authentication-server profiles in this deployment are required as part of the configuration. But, in this deployment, the OCSP server information is in the SLB server configurations for the OCSP servers, instead of in the authentication-server profiles.

CLI Example for the Deployment

The following commands create an authentication-server profile for each OCSP server:

```
ACOS(config)# aam authentication server ocsp OCSP1
ACOS(config-ocsp auth server:OCSP1)# aam authentication server ocsp OCSP2
ACOS(config-ocsp auth server:OCSP2)# exit
```

Because verification requests are load balanced among multiple OCSP servers, no additional information about the servers is required in the authentication-server profiles. Instead, the following commands create SLB server configurations for the OCSP servers and adds the servers to a service group.

```
ACOS(config)# slb server o1 10.10.10.5
ACOS(config-real server)# port 80 tcp
ACOS(config-real server-node port)# authentication-server OCSP1
ACOS(config-real server-node port)# slb server o2 10.10.10.6
ACOS(config-real server)# port 80 tcp
ACOS(config-real server-node port)# exit
ACOS(config-real server)# exit
ACOS(config)# aam authentication server ocsp OCSP2
```

Configuring the OCSP Service-Groups

The following commands configure the OCSP service-groups:

```
aam authentication server ocsp OCSP1
  url http://{ip}:{port}/
aam authentication server ocsp OCSP2
  url http://{ip}:{port}/
aam authentication service-group ocsp tcp
  member OCSP1 {port}
  member OCSP2 {port}
```

Secure Sockets Layer

Secure Sockets Layer (SSL)-enabled servers can be configured to require client authentication to verify a client’s identity. After the client is authenticated, you can complete an additional optional authorization check. In this additional check, ACOS acts as a client, sends an LDAP query request to the LDAP server, and completes the following tasks:

- Perform client authorization with an LDAP server after CRL or OCSP checks.
- Extract username information from the following fields in the certificate:
  - Subject name
  - User Principal Name (UPN) in Subject Alternative Name other name field. UPN is the name of a system user in an email address format. The username is followed by “@”, then followed by the name of the Internet domain with which the user is associated. Ex: bill@example.com. We must extract “bill” as username in this example.
- Perform LDAP query request to LDAP server. Search success means authorization success.

The following topics are covered:

Authentication Work-flow

The following steps provide a high-level overview of the SSL authentication process:

1. Client sends a *ClientHello* message that includes SSL options.
2. ACOS responds with a *ServerHello* message selecting the SSL options.
3. ACOS sends a *Certificate* message, which contains ACOS’s SSL certificate.

NOTE: When an SSL certificate expires or is near expiration, the ACOS device automatically send a system log warning, rather than a system log notice.

4. ACOS requests the client's certificate in a *CertificateRequest* message, so that the connection can be mutually authenticated.
5. ACOS concludes its part of the negotiation with *ServerHelloDone* message.
6. Client verifies the server’s (ACOS) certificate.

7. Client responds with a *Certificate* message, which contains the client's certificate.
8. If configured, ACOS completes SSL authentication with a CRL or an OCSP server.

   If the server/service group-related information is configured in the client SSL template, ACOS completes LDAP authorization with the LDAP server by sending an LDAP search query. In the LDAP search query, a predefined LDAP search filter can be used to select the entries to be returned.

   Here is an example of an LDAP search filter:

   ```
   (&(objectclass=posixAccount)(uid=subject-uid)
   ```

   ```
   (&(objectclass=person)(myattr='UPN'))
   ```

   UPN is a special string, which means that ACOS must extract this string from the certificate dynamically during run-time. ACOS also tries to get the LDAP search base from the subject DN field in the certificate.
9. LDAP server replies to ACOS's request.
10. ACOS performs authorization checking by verifying LDAP search response.
11. After authentication and authorization succeeds, ACOS will continue SSL packets handling.
12. The client's original request (not specify in below diagram) is forwarded to the HTTPS server.

    Instead, if authorization check is failure, ACOS will stop SSL packets handling.

OCSP Stapling

In standard OCSP protocol, completion of the certificate validation process requires that every certificate passed through the client will need an OCSP request. In situations where multiple browsers may lodge multiple OCSP request synchronously, responding to each of these requests without impacting website performance can be an issue.

To circumvent this, OCSP stapling can allow for multiple OCSP responses to be stapled into a single response within the SSL/TLS handshake to assure the client that the certificate is valid. This avoids requiring multiple sources to contact the OCSP server and compromise the OCSP response which is necessary to ensure uptime of an HTTPS site.

To do this, the OCSP server is regularly queried by the certificate owner, and the response is a signed time-stamped output.

The following topics are covered:

Creating an OCSP Authentication Server by Using the GUI

To create an OCSP authentication server:

1. Click **AAM** > **Auth Servers**.
2. On the **OCSP** tab, click **Create**.
3. Enter a name.
4. To disable the health check, select **Disable Health Check**.
5. Enter the URL for the OSCP server.

NOTE: When you enter the URL for the OCSP server, you must add a **/** at the end of the URL.

6. Select a responder CA certificate file name.
7. Select a responder certificate file name.
8. Click **Create**.

Deleting an OCSP Authentication Server by Using the GUI

To delete an OCSP server:

1. Click **AAM** > **Auth Servers**.
2. Select an OCSP server and click **Delete**.

Creating a Client SSL Template by Using the GUI

To create a client SSL template:

1. Click **ADC** > **Templates**.
2. On the **SSL** tab, click **Create**, and select **Client SSL**.

NOTE: You can import an SSL template by clicking **Import**.

3. Expand the **General Fields** section and complete the following steps:
   a. Enter a template name.
   b. Select the authentication username that you want to use for the template.
4. Configure the fields on this page, as desired. Refer to the online help for information on the fields listed on this GUI page.
5. Click **OK**.

Binding the Client SSL Template to a Virtual Port by Using the GUI

To bind the client SSL template to a virtual port:

1. Click **ADC** > **SLB**.
2. On the **Virtual Servers** tab, select a virtual server, and click **Edit**.
3. In the **Virtual Port** section, click **Create**.
4. Configure the fields, as desired. Refer to the online help for information on the fields listed on this GUI page.
5. Expand the **General Fields** section, select a source NAT pool.
6. Expand the **Templates** section and complete one of the following tasks for each template you want to use:
   a. Select an existing template.
   b. Click **Add+** to create a template.
7. Click **Create**.

Configuring OCSP Stapling by Using the CLI

To configure OCSP stapling:

1. To set up an OCSP server in an ACOS device, enter the following commands:

```
aam authentication server ocspa1
```

```
urlhttp://{host | ip} [:port]/
One of the following fields:
responder-cacert_name#Delegated Trust Model
responder-certcert_name#Directed Trust Model
```

2. To place the *server.crt/server.key/RootCA.crt* file on the ACOS device, enter the following commands:

NOTE: To place the file in the ACOS device, you must load the file in a client SSL template.

```
slb templateclient-ssl c1
certserver_cert
ocsp-staplingca-certca_certocspauth_server | service-groupname (period
[days1-30 | hours1-24 | minutes1-60 ]) (timeout1-65535)
client-certificateRequire
keyserver_key
```

3. To bind this client SSL template to a vport, enter the following commands:

```
slb virtual-servervip-http2 xxx.xxx.xxx.xxx
port 443 https
templateclient-ssl c1
source-natauto
service-grouphttp_g_1
```

CLI Example

The following example shows you how to how to configure OCSP stapling on one server:

```
aam authentication server ocsp a1
url http://[OCSP server IP addr]:7878/
responder-cert RootCA.crt

slb template client-ssl c1
cert server.crt
ocsp-stapling ca-cert RootCA.crt ocsp a1 period minutes 1 timeout 1
client-certificate Require
key server.key

slb virtual-server vip-http2 xxx.xxx.xxx.xxx
```

```
port 443 https
source-nat auto
service-group http_g_1
templateclient-ssl c1
```

If the OCSP responder uses the intermediate certificate, the configuration is as follows:

```
aam authentication server ocsp a1
urlhttp://[OCSP server IP addr]:[OCSP server port]/
responder-cert intermediate.crt
!
slb template client-ssl c1
cert server.crt
ocsp-stapling ca-cert RootCA.crt ocsp a1 period minutes 1 timeout 1
client-certificate Require
key server.key
!
```

SSL Client Certificate Authentication with LDAP

The following topics are covered:

SSL Server

SSL-enabled servers can be configured to require client authentication and verify the client's identity. After the default client authentication is complete, an additional optional authorization check can be performed for SSL client authorization. The ACOS device acts as a client to send the LDAP query request to the LDAP server.

The SSL-configured server can complete the following tasks:

- Complete client authorization with the LDAP server after CRL or OCSP checks.
- Extract the username information from the following in the certificate:
  - Subject name
  - User Principal Name (UPN) in Subject Alternative Name other name field.

    UPN is the name of a system user in an email address format. The username is followed by “@”, then followed by the name of the Internet domain with which the user is associated. In the example, *bill@example.com*, *bill* is extracted as the username.
- Perform LDAP query request to LDAP server. Search success means authorization success.

Sample Work-flow

The following procedure provides an overview of the process:

1. The client sends a *ClientHello* message that includes SSL options.
2. ACOS selects the SSL options and responds with *ServerHello* message.
3. ACOS sends a *Certificate* message, which contains the ACOS’s SSL certificate.
4. ACOS requests the client's certificate in a *CertificateRequest* message.

   The connection is mutually authenticated.
5. ACOS completes the process by sending a *ServerHelloDone* message.
6. The client verifies the server’s (ACOS) certificate.
7. The client responds with a certificate message that contains the client's certificate.
8. If configured, ACOS completes the SSL authentication by using a certification revocation list (CRL) or with the OCSP server.

   If the server/service group-related information is configured in the client SSL template, ACOS will perform LDAP authorization with the LDAP server by sending an LDAP search query. In LDAP search query, a pre-defined LDAP search filter can be used to select the entries to be returned.

   The following is an example of an LDAP search filter:

```
(&(objectclass=posixAccount)(uid=subject-uid)
(&(objectclass=person)(myattr='UPN'))
```

`UPN` is a special string, which means that ACOS must dynamically extract the information from the certificate.

To ensure that ACOS can identify a special attribute like UPN without an ambiguous definition, an apostrophe must be used with this attribute when you configure an LDAP search filter. for example, **`'UPN'`**.

The following procedure is an example of how to configure SSL client certificate authorization with LDAP. The syntax rules for the LDAP search filter string in this example are found in RFC 4515:

```
slb template client-ssl test
cert server.crt
key server.key
client-certificate Require
authorization service-group ldap-service-group ldap-search-filter (&
(objectClass=inetOrgPerson)(uid='UPN')) ç 'UPN'
ca-cert RootCA.crt ocsp test
crl crl
```

In addition to the LDAP search filter, ACOS gets the LDAP search base from the subject DN field in the certificate.

9. The LDAP server replies to ACOS's request.
10. ACOS performs authorization checking by verifying LDAP search response.

    After authentication and authorization succeeds, ACOS continues to handle SSL packets. The client's original request is forwarded to the HTTPS server. But, if the authorization check fails, ACOS stops SSL packets handling.

Configuring SSL Client Certificate Authentication with LDAP

You can configure SSL client certificate authentication by using the GUI or CLI.

NOTE: Before you begin this procedure, ensure that you have created an LDAP server.

The following topics are covered:

Configuring SSL Client Certificate Authentication by Using the GUI

To configure SSL Client Certificate authentication:

1. Click **AAM** > **Auth Servers**.
2. On the **LDAP** tab, select the LDAP server you created, and click **Edit**.
3. Modify the configuration as necessary and click **Update**.
4. Click **AAM** > **Service Groups**.
5. Select a service group and click **Edit**.
6. In the **Service Group Members** section, click **Create**.
7. Select the LDAP server you just created.
8. Enter a port number.
9. Select the member priority.
10. In **Member State**, select **enable**, and click **Create**.
11. Click **ADC** > **SSL Management**.
12. On the **SSL Certificates** tab, you can create or import certificates.

The following topics are covered:

- Creating a Certificate
- Importing a Certificate

Creating a Certificate

1. Click **Create**.
2. Enter the SSL Certificate file name.
3. To generate the CSR, select the **CSR Generate** check box.
4. Enter a common name.

5. Configure the fields, as desired. Refer to the online help for information on the fields listed on this GUI page.
6. Click **Create.**

Importing a Certificate

1. Click **Import**.
2. Enter the certificate file name.
3. Select the option that you want to import.
4. Select the import location.
5. Select the certificate type.
6. To generate the CSR, select the **CSR Generate** check box.
7. Select a certificate format.
8. Click **Choose File**, select the certificate file, and click **Open**.
9. Click **Import**.

Configuring SSL Client Certificate Authentication by Using CLI

1. To create an authentication-server profile for the LDAP server, enter the following commands:

```
ACOS(config)# aam authentication server
ldapauthentication_LDAP_server_name
host192.168.221.71
base ou=People,dc=a10lab,dc=com
dn-attributeuid
aam authentication server
host [IP | hostname]
base [base-dn]
admin-dn [admin-dn]
dn-attributeuid
admin-secret [password]
```

NOTE: Currently, the LDAP client in ACOS only allows you to send an LDAP bind with a user password. If you do not specify the user password, ACOS will not send the bind request. The initial LDAP bind for SSL certificate authorization with LDAP must use the admin DN and the admin password.

The **host** can be an IP address or a host name.

2. To create an SLB server configuration for the LDAP server, enter the following commands:

```
ACOS(config)# aam authentication service-group LDAP_slb_
server192.168.221.71
port 389 tcp
no health-check
authentication serverauthentication_LDAP_server_name
```

3. To add the LDAP server to the service group, enter the following commands:

```
ACOS(config)# aam authentication service-groupLDAP Service_group_name
tcp
member LDAP_slb_server389
```

4. To configure the SSL certificate authentication, enter the following commands:

```
ACOS(config)# slb template client-sslssl-template-name
certserver_certificate_name
client-certificate {Require | Request | Ignore}
ca-certca_cert_nameocsp {ocsp_name | service-groupOCSP_service_group_
name}
authorization {LDAP_server_name | service-groupservice_group_name}
[ldap-base-dn-from-cert | ldap-authorization-filterLDAP_search_filter]
```

Optional Commands

The following commands are optional:

- The `ldap-base-dn-from-cert` command is optional. When it is configured, the corresponding LDAP server must use the subject DN (from client certificate) as the base DN to complete the LDAP search. If it is not configured, ACOS uses the base DN that is configured in the authentication server on ACOS.

`ldap-authorization-filter`

- To ensure that ACOS can identify a special attribute like UPN without an ambiguous definition, an apostrophe must be used to wrap the special attribute when you configure an LDAP search filter, for example, **`‵UPN‵`**.

Configuring SSL Client Certificate Authorization with LDAP

The following procedure is an example of how to configure SSL client certificate authorization with LDAP:

```
ACOS(config)# slb template client-ssl test
ACOS(config-client ssl)# cert my-cert
ACOS(config-client ssl)# key my-cert
ACOS(config-client ssl)# client-certificate Require
ACOS(config-client ssl)# authorization service-group my-ldap-service-group
ldap-search-filter (&(objectClass=inetOrgPerson)(uid='UPN')) ç ‵UPN′
```

AAM in Multi-PU Deployment

This section describes the features and configurations specific to AAM implementation in multi-PU platforms.

For the common multi-PU implementation details, see *Application Delivery Controller Guide*.

The following topics are covered:

Key Considerations

- For seamless traffic distribution in multi-PU, you must configure the **`odd-even-nat-enable`** command in the SLB common mode. This is especially useful when all client side traffic terminates on a single VIP configured in a multi-PU chassis.

NOTE: The **`odd-even-nat-enable`** command in the multi-PU mode is currently not supported with the **`source-nat auto`** command.

- Traffic is sent to the same virtual port from two different sources: odd source IP and even source IP.

  When traffic from an odd source IP is sent to the same VIP, it is sent to PU1. Similarly, when traffic from an even source IP is sent to the same VIP, it is sent to PU2.

- AAM traffic statistics are aggregated from the both PUs to appear as a single device.

CLI Configuration

This section describes the CLI configuration guidelines and examples for implementing AAM multi-PU using IP address to ensure seamless traffic load balancing over multiple PUs.

Use the following guidelines or workflow to ensure seamless traffic load balancing in both PUs (PU1 and PU2):

1. Configure the chassis application type using the **`chassis-application-type adc`** command.
2. Enable the odd-even IP NAT globally using the **`odd-even-nat-enable`** command for NAT seamless traffic distribution across both PUs.
3. Configure IP NAT pools using the **`ip nat pool`** or **`ipv6 nat pool`** commands to source NAT client traffic.
4. Configure client-side VLAN (or interface ethernet) with traffic distribution mode SIP using the **`traffic-distribution-mode`** command.
5. Configure server-side VLAN (or interface ethernet) with traffic distribution mode DIP using the **`traffic-distribution-mode`** command.
6. Configure the `system chassis-port-split enable` command, to split ports for each PU.
7. Configure AAM authentication logon, AAM authentication servers, AAM authentication templates, and AAA policies.
8. Create SLB servers, SLB service groups, SLB templates, and SLB virtual servers and ports.
9. Bind the appropriate templates along with AAA policies to the virtual ports.
10. Bind the source NAT pool to the virtual ports if you are using the NAT seamless traffic distribution.
11. Verify the configuration and view the show commands for traffic distribution.

The following topics are covered:

Application Type Configuration

```
ACOS(config)# chassis-application-type adc
```

NOTE: A reload or reboot is required when the application type is switched from CGN to ADC. Additionally, while switching the box from CGN to ADC, ADC-related configuration should not exist on the box on any partitions (shared/L3V/service).

SLB Global Configuration

```
ACOS(config)# slb common
ACOS  (config-common)# odd-even-nat-enable
```

IP Address and Default Gateway Configuration

```
ACOS(config)# interface management
ACOS(config-if:management)# ip address 10.65.18.137 255.255.255.0
ACOS(config-if:management)# ip default-gateway 10.65.18.1
ACOS(config-if:management)# secondary-ip address 10.65.18.138
255.255.255.0
ACOS(config-if:management)# secondary-ip default-gateway 10.65.18.1
```

Interface and VLAN Configuration

```
ACOS(config)# interface ethernet 1
ACOS(config-if:ethernet:1)# speed-forced-40g
ACOS(config-if:ethernet:1)# enable
ACOS(config-if:ethernet:1)# lldp enable rx tx
ACOS(config-if:ethernet:1)# exit
ACOS(config)# vlan 10
ACOS(config-vlan:10)# tagged ethernet 1
ACOS(config-vlan:10)# router-interface ve 10
ACOS(config-vlan:10)# traffic-distribution-mode sip
ACOS(config-vlan:10)# exit
ACOS(config)# vlan 20
ACOS(config-vlan:20)# tagged ethernet 1
ACOS(config-vlan:20)# router-interface ve 20
ACOS(config-vlan:20)# traffic-distribution-mode dip
ACOS(config-vlan:20)# exit
```

IP Interface on VLAN Configuration

```
ACOS(config)# interface ve 10
ACOS(config-if:ve:10)# ip address 10.31.8.35 255.255.255.0
ACOS(config)# interface ve 20
ACOS(config-if:ve:20)# ip address 10.31.9.35 255.255.255.0
ACOS(config-if:ve:20)# exit
```

Common Configuration

```
ACOS(config)# system chassis-port-split enable
```

Source NAT Pool Configuration

```
ACOS(config)# ip nat pool snat1 10.31.9.48 10.31.9.63 netmask /24
ACOS(config)# ip nat pool snat2 10.31.9.64 10.31.9.79 netmask /24
```

AAM Configuration

```
ACOS(config)# aam authentication logon http-authenticate basic
ACOS(config-http-authenticate auth logon:...)# auth-method basic enable
ACOS(config-http-authenticate auth logon:...)# exit
ACOS(config)# aam authentication server windows krb
ACOS(config-radius auth server:radius)# host 10.31.9.200
ACOS(config-radius auth server:radius)# realm A10-TPLAB.COM
ACOS(config-radius auth server:radius)# exit
ACOS(config)# aam authentication template basic_krb
ACOS(config-auth template:basic_radius)# logon basic
ACOS(config-auth template:basic_radius)# server krb
ACOS(config-auth template:basic_radius)# log enable
ACOS(config-auth template:basic_radius)# exit
ACOS(config)# aam aaa-policy ap1
ACOS(config-aaa policy:ap1)# aaa-rule 1
ACOS(config-aaa policy:ap1-aaa rule:1)# action allow
ACOS(config-aaa policy:ap1-aaa rule:1)# authentication-template basic_krb
```

SLB Configuration

```
ACOS(config)# slb server rs1 10.31.9.200
ACOS(config-real server)# port 80 tcp
ACOS(config-real server-node port)# health-check hm_monitor1
ACOS(config-real server-node port)# exit
ACOS(config)# slb server rs2 10.31.9.201
ACOS(config-real server)# port 80 tcp
ACOS(config-real server-node port)# health-check hm_monitor2
ACOS(config-real server-node port)# exit
ACOS(config-real server)# exit
ACOS(config)# slb service-group sg1 tcp
ACOS(config-slb svc group)# member rs1 80
ACOS(config-slb svc group-member:80)# exit
ACOS(config)# slb service-group sg2 tcp
ACOS(config-slb svc group)# member rs2 80
ACOS(config-slb svc group-member:80)# exit
ACOS(config)# slb virtual-server vs1 10.31.8.52
ACOS(config-slb vserver)# port 80 http
ACOS(config-slb vserver-vport)# source-nat pool snat1
ACOS(config-slb vserver-vport)# service-group sg1
ACOS(config-slb vserver-vport)# aaa-policy ap1
ACOS(config-slb vserver-vport)# exit
```

Show Commands

This section describes the various show commands to view traffic statistics on AAM multi-PU. All the AAM module statistics are either aggregated as a single device (Primary + Blade) or separate for PU1 and PU2.

For more information, see 'Show Commands' section under 'Application Access Management' in the Command Line Interface Reference.

View AAM Authentication Statistics

```
ACOS# show aam authentication statistics
Engine statistics:
```

```
-----------------------
Requests to Auth Daemon:                    4
Responses from Auth Daemon:                 4
Requests to SAML:                           0
Responses from SAML:                        0
Misses:                                     0
OCSP Stapling Requests to Auth Daemon:      1
OCSP Stapling Responses from Auth Daemon:   2
Domain whitelist matched:                   0
Domain whitelist unmatched:                 0
Engine aFleX statistics:
-----------------------
Authorize success:                          0
Authorize failure:                          0
Auth Daemon statistic:
-----------------------
Opened socket:                              11
Open socket failed:                         0
Get socket option failed:                   0
Connect:                                    11
Connect failed:                             0
Created timer:                              11
Create timer failed:                        0
DNS resolve failed:                         0
Total request:                              6
-------------------------------------------------------------------------
-------------
\ statistic     Request   Request   Response  Response  Response
Response  Response
Type\           Normal    Dropped   Success   Failure   Error     Timeout
  Other
-------------------------------------------------------------------------
-------------
OCSP              0         0         0         0         0         0
    0
RADIUS            0         0         0         0         0         0
    0
LDAP              0         0         0         0         0         0
    0
```

```
Windows-KERBEROS   4          0          3          0          0          0
    1
KERBEROS-RELAY     0          0          0          0          0          0
    0
OCSP-STAPLING      2          0          2          0          0          0
    0
SPN-KERBEROS       0          0          0          0          0          0
    0
CAPTCHA            0          0          0          0          0          0
    0
-------------------------------------------------------------------------
-------------
Auth Daemon RADIUS statistic:
-----------------------
Request:                                     0
Authentication success:                      0
Authentication failure:                      0
Authorize success:                           0
Authorize failure:                           0
Access challenge:                            0
Accounting-Request sent:                     0
Accounting success:                          0
Accounting failure:                          0
Timeout error:                               0
Job start error:                             0
Polling control error:                       0
Other error:                                 0
Auth Daemon LDAP statistic:
-----------------------
Request:                                     0
Admin bind success:                          0
Admin bind failure:                          0
Bind success:                                0
Bind failure:                                0
Search success:                              0
Search failure:                              0
Authorize success:                           0
Authorize failure:                           0
Timeout error:                               0
Job start error:                             0
```

```
Polling control error:                      0
TLS/SSL session created:                    0
TLS/SSL session failure:                    0
LDAPS idle connection number:               0
LDAPS in-use connection number:             0
Other error:                                0
Password expiry:                            0
Password change success:                    0
Password change failure:                    0
Auth Daemon Windows-Kerberos statistic:
-----------------------
kerberos request send:                      9
kerberos response get:                      9
Timeout error:                              0
Job start error:                            0
Polling control error:                      0
Other error:                                0
Password expiry:                            0
Password change success:                    1
Password change failure:                    0
KDC validation success:                     0
KDC validation failure:                     0
KDC keytab generation success:              0
KDC keytab generation failure:              0
KDC keytab deletion success:                0
KDC keytab deletion failure:                0
Current KDC keytab count:                   0
Auth Daemon Kerberos-relay statistic:
-----------------------
kerberos-relay request send:                0
kerberos-relay response get:                0
Timeout error:                              0
Job start error:                            0
Polling control error:                      0
Other error:                                0
Auth Daemon Windows-SMB statistics:
-----------------------
Authentication success:                     0
Authentication failure:                     0
SMB proto negotiation success:              0
SMB proto negotiation failure:                0
SMB session setup success:                    0
SMB session setup failed:                     0
Prepare req success:                          0
Prepare req failed:                           0
Timeout error:                                0
Job start error:                              0
Polling control error:                        0
Other error:                                  0
Auth Daemon OCSP-Stapling statistics:
-----------------------------------
Certificate Good:                             2
Certificate Revoked:                          0
Certificate Unknown:                          0
OCSP failed:                                  0
Auth Daemon CAPTCHA statistic:
------------------------------
Request:                                      0
Verification success:                         0
Json response parse failure:                  0
Json response failure:                        0
Attribute check failure:                      0
Timeout error:                                0
Other error:                                  0
Job start error:                              0
Polling control error:                        0
Kerberos SPN statistic:
-----------------------
SPN kerberos request:                         0
SPN kerberos response success:                0
SPN kerberos response failure:                0
SAML statistic:
-----------------------
SP metadata export requests:                  0
SP metadata export successes:                 0
Login authn requests:                         0
Login authn responses:                        0
SSO requests:                                 0
SSO successes:                                0
SSO authorization failed:                     0
SSO errors:                                 0
SLO requests:                               0
SLO successes:                              0
SLO errors:                                 0
SP-initiated SLO requests:                  0
Global Logout successes:                    0
Local Logout successes:                     0
Partial Logout successes:                   0
Relay requests:                             0
Relay successes:                            0
Relay failed:                               0
Relay errors:                               0
Other errors:                               0
OAUTH statistic:
-----------------------
Auth requests:                              0
Auth successes:                             0
Auth failed:                                0
Auth errors:                                0
Relay requests:                             0
Relay successes:                            0
Relay failed:                               0
Other errors:                               0
```

Consolidated Configuration Example

The consolidated configuration example is given below:

```
ACOS#show run
!Current configuration: 1392 bytes
!Configuration last updated at 23:32:06 PDT Wed Jun 4 2022
!Configuration last saved at 17:45:20 PDT Wed Jun 4 2022
!64-bit Advanced Core OS (ACOS) version 5.2.1-P7-dual-chassis, build 21
(Mar-03-2023,03:26)
!
chassis-application-type adc
!
multi-config enable
!
```

```
system chassis-port-split enable
!
terminal idle-timeout 0
!
vlan 10
  tagged ethernet 1
  router-interface ve 10
  traffic-distribution-mode sip
!
vlan 20
  tagged ethernet 1
  router-interface ve 20
  traffic-distribution-mode dip
!
interface management
 ip address 10.65.18.137 255.255.255.0
 ip default-gateway 10.65.18.1
 secondary-ip address 10.65.18.138 255.255.255.0
 secondary-ip default-gateway 10.65.18.1
 enable
```

```
!
interface ethernet 1
  speed-forced-40g
  enable
  lldp enable rx tx
!
interface ethernet 2
!
interface ethernet 3
!
interface ethernet 4
!
interface ethernet 5
  speed-forced-40g
  enable
  lldp enable rx tx
!
interface ethernet 6
  enable
!
interface ethernet 7
  enable
!
interface ethernet 8
!
interface ethernet 9
!
interface ethernet 10
!
interface ethernet 11
!
interface ethernet 12
!
interface ethernet 13
!
interface ethernet 14
!
interface ethernet 15
```

```
!
interface ethernet 16
!
interface ve 10
  ip address 10.31.8.35 255.255.255.0
!
interface ve 20
  ip address 10.31.9.35 255.255.255.0
  ipv6 address 2602:803:c002::500:142/126
  ipv6 enable
!
ip nat pool snat1 10.31.9.48 10.31.9.63 netmask /24
!
ip nat pool snat2 10.31.9.64 10.31.9.79 netmask /24
!
health monitor hm_monitor1
 method http expect response-code 200 url GET /healthcheck
 interval 10
!
health monitor hm_monitor2
 method http port 3000 expect response-code 200 url GET /health
!
slb common
 odd-even-nat-enable
!
aam authentication logon http-authenticate basic
auth-method basic enable
!
aam authentication server windows krb
host 172.16.61.221
realm A10-TPLAB.COM
!
aam authentication template basic_krb
logon basic
server krb
log enable
!
aam aaa-policy ap1
aaa-rule 1
action allow
```

```
authentication-template basic_krb
!
slb server rs1 10.31.9.200
 port 80 tcp
 health-check hm_monitor1
!
slb server rs2 10.31.9.201
 port 80 tcp
 health-check hm_monitor2
!
slb service-group sg1 tcp
 member rs1 80
!
slb service-group sg2 tcp
 member rs2 80
!
slb template tcp-proxy tcp-default
 idle-timeout 60
!
slb template virtual-port vport_template
 reset-l7-on-failover
!
slb template virtual-server vs_template
 conn-limit 5
 conn-rate-limit 10
!
slb template connection-reuse multiplex-default
 keep-alive-conn 1024
!
slb template http http-frp
  insert-client-ip X-Forwarded-For
  request-header-erase X-ClientIP
  keep-client-alive
!
slb virtual-server vs1 10.31.8.52
 port 80 http
    source-nat pool snat1
    service-group sg1
    template connection-reuse multiplex-default
    template http http-frp
    template tcp-proxy tcp-default
!
```

Limitation

AAM will not sync all tables between PU1 and PU2.

Contact Us