Progress®

Configure Client Authentication

OpenEdge®

Copyright

March 2020

Product version: Progress OpenEdge 12.2

Updated: 2020/03/19

Table of Contents

Preface

Purpose

This document explains how to configure ABL client authentication in both the client, and also in a PAS for OpenEdge instance.

Audience

This manual is intended for any OpenEdge application server administrators or ABL developers who need to create, configure, or administer security in Progress Application Server for OpenEdge.

Organization

Learn about Client Authentication on page 9

Introduces basic concepts regarding how client authentication works in PAS for OpenEdge, identifies the types of clients for which client authentication is supported, and summarizes the basic configuration tasks that are required.

Configure ABL Client Authentication on page 11

Explains how to set up the certificates and keys that are required in clients for establishing client authentication with a PAS for OpenEdge instance.

Configure a PAS for OpenEdge Instance for Client Authentication on page 13

Explains how to configure a PAS for OpenEdge instance for mutual authentication with an ABL client.

Documentation conventions

See Documentation Conventions for an explanation of the terminology, format, and typographical conventions used throughout the OpenEdge content library.

Purpose

This document explains how to configure ABL client authentication in both the client, and also in a PAS for OpenEdge instance.

Audience

This manual is intended for any OpenEdge application server administrators or ABL developers who need to create, configure, or administer security in Progress Application Server for OpenEdge.

Organization

Learn about Client Authentication on page 9

Introduces basic concepts regarding how client authentication works in PAS for OpenEdge, identifies the types of clients for which client authentication is supported, and summarizes the basic configuration tasks that are required.

Configure ABL Client Authentication on page 11

Explains how to set up the certificates and keys that are required in clients for establishing client authentication with a PAS for OpenEdge instance.

Configure a PAS for OpenEdge Instance for Client Authentication on page 13

Explains how to configure a PAS for OpenEdge instance for mutual authentication with an ABL client.

Documentation conventions

See Documentation Conventions for an explanation of the terminology, format, and typographical conventions used throughout the OpenEdge content library.

2

Learn about Client Authentication

When you use TLS to secure data transfer between a client application and a server (in this case, PAS for OpenEdge), then the server presents its certificate, digitally signed by a third-party certificate authority (CA), to the client application during the TLS handshake. Authenticating and establishing the identity of the server is part of the TLS handshake. This process is how the client validates its connection to the intended server.

Depending upon your business and security needs, you can also configure a client to authenticate its identity to the server. The configuration in which both the server and client authenticate is referred to as mutual authentication. Before you can use client authentication, the intended server must be configured to require a client certificate.

TLS client authentication is supported for SOAP, REST, WEB, and static file access via PAS for OpenEdge, and for ABL clients connecting to PAS for OpenEdge over APSV.

For more information about TLS in PAS for OpenEdge, see Use TLS in PAS for OpenEdge.

How client authentication works

The workflow when using client authentication is as follows:

1. The client sends a message to the server (a PAS for OpenEdge instance). The message contains details about the client identity and other security-related information.
2. The server responds with a message by sending its signed digital certificate and other security information required by the client.
3. Upon successful authentication, the client sends its digital certificate to the server. Note the server must be configured to require the client certificate.
4. The server authenticates the client based on the digital certificate.
5. After the client and server are each authenticated, data transmission between the client and server begins.

Public and private keys, digital certificates, and trusted CAs

Private keys and digital certificates establish and verify identity and trust. TLS uses public key encryption for authentication. With public key encryption, a public key and a private key are generated. Data encrypted with the public key can only be decrypted using the corresponding private key. The public key is embedded in a digital certificate. A private key and a digital certificate provide identity for the ABL client.

The data embedded in a digital certificate is verified by a certificate authority (CA) and is digitally signed with the digital certificate of the CA. The digital certificate of the CA establishes trust.

Main steps for configuring client authentication

The main steps for configuring client authentication are:

1. Set up the client:
   a. Generate a public and private key pair.
   b. Generate a Certificate Signing Request (CSR) and submit it to a CA.
   c. Store the certificate that is signed and returned by the CA.
2. Configure the PAS for OpenEdge instance to require client certificates.

3

Configure ABL Client Authentication

When an ABL client initiates an HTTPS connection request with a PAS for OpenEdge instance, the client must include certain details in its request so that, in addition to the mandatory server authentication, client authentication is performed during the TLS handshake.

Note: You can configure an ABL client for either TLS client or HTTP Basic authentication (using `user ID` and `password`) but not both.

The steps in this procedure involve the use of the `pkiutil` command utility, which provides all the operations necessary to create and manage keystore entries for OpenEdge clients and PAS for OpenEdge instances. These operations include the ability to generate a Privacy Enhance Mail (`.pem`)-formatted file using the private certificate. A PEM file is an encrypted file that contains keystore information. For details about the syntax and usage of the `pkiutl` utility, see pkiutil in *Manage Keys and Certificates*.

Perform the following steps to obtain a private key and digital certificate for an ABL client:

1. Create the private and public key pair for the ABL client:

   ```
   pkiutil -keysize 2048 -newreq client_alias
   ```

   In the preceding command, *`client_alias`* represents the name that corresponds to the keystore entry for the certificate and keys for the ABL client. When you execute this command, you are prompted for a password. You must later use this password to gain access to that keystore entry.

2. Submit the public key file (*`client_alias`*`.pk10`) to a certificate authority (CA) to request a signed certificate.

   The CA returns both your public key that the CA has signed (your signed certificate), and the public certificate of the CA (the root certificate). These certificates have either a `.crt` or a `.cer` extension. For example, *`client_private`*`.cer` and *`client_public`*`.cer`. You can rename the certificates as desired.

   Make sure that you save all certificates in a directory outside the OpenEdge installation directory. This ensures that the certificates are not deleted when you uninstall or re-install OpenEdge.

3. Import the client certificate (*client_private*.cer) into the client keystore using the following command:

   ```
   pkiutil -import client_alias client_private_certificate_dir\client_private.cer
   ```

   In the preceding command, *client_private_certificate_dir* represents the location where you have stored the client certificate that you received from the CA.

   In this procedure, *client_private*.pem is generated.

   Note: The keystore in the OpenEdge installation directory is the client identity keystore.

4. When you are prompted, enter the password you used when you created the certificate and keys (see Step 1). The pkiutil utility creates a certificate in a file, with a .pem extension, in the *OpenEdge-install-dir*\keys directory.
5. Use the following parameters in the CONNECT( ) method for the SERVER HANDLE in the ABL client procedure:

   Table 1: ABL client parameters for TLS client authentication

Parameter	Description
-sslAuth *authentication_type*	• Specifies if access to HTTPS requires SSL/TLS client authentication. • Set this parameter to ssl to enable client authentication for HTTPS access. • If the parameter is set to basic, the CONNECT( ) method for the SERVER HANDLE does not perform client authentication. The default is basic.
-sslKeyFile *filename*	• Specifies the location of the client certificate (*client_private*.pem) file. • If you do not specify the absolute path of *client_certificate_filename*.pem, the connection operation searches for the certificate file in the *OpenEdge_install-dir*\keys directory. • If -sslAuth is set to ssl, the -sslKeyFile parameter must be specified. Otherwise, client authentication is not performed.
-sslKeyPwd *password*	• Specifies the password used to decrypt and use the private key of the client found in *client_certificate_filename*.pem. • The password may be in clear text or encoded using the GENPASSWORD utility.

4

Configure a PAS for OpenEdge Instance for Client Authentication

Before you can use client authentication, you must complete a set of configuration tasks in PAS for OpenEdge. These tasks include:

1. Importing the CA root certificate of the ABL client into the trust keystore used by the PAS for OpenEdge instance
2. Enabling TLS client authentication in the HTTPS port
3. Configuring Tomcat user accounts
4. Setting the **container** login model of the ABL web application
5. Ensuring that HTTPS support is enabled in your production instance

Complete the following steps to configure a PAS for OpenEdge instance for client authentication:

1. Import the CA root certificate of the ABL client into the trust keystore of the instance:

   ```
   keytool -importcert -alias CA_certificate-alias -keystore
   instance-name\conf\tomcat-certstore.jks -trustcacerts -file
   CA_certificate_filepath.cer -storepass storepass  -noprompt
   ```

   The CA root certificate is required in establishing trust for the client certificate during an HTTPS connection.

   In the preceding command syntax:

   - *CA_certificate-alias*—Represents the alias of the CA root certificate in the trust keystore
   - *instance-name*—Represents the path of the PAS for OpenEdge instance directory on your machine
   - *CA_certificate_filepath*—Represents the full directory path of the ABL client's CA root certificate that you are importing into the trust keystore of the instance

- *storepass*—Represents the trust keystore password

Note: You must also import all required intermediate certificates into the trust keystore of the PAS for OpenEdge instance.

2. Configure the HTTPS port of the instance:

   a. Enable HTTPS client authentication for the instance by setting the value of the `psc.as.https.clientauth` property to `required`:

   ```
   tcman config psc.as.https.clientauth=required
   ```

   Note that this property has two additional values that can be specified: `none` and `optional`.

   If the property is set to:

   - `none`, which is the default value, then client authentication is disabled.
   - `optional`, then client authentication is performed if a client presents its certificate. But if the client does not present a certificate, then no authentication is performed, and a connection request is accepted.
   - `required`, then client authentication must be performed for all ABL web applications deployed to the instance.

3. Configure a user account (*username*, *password*, and *roles*) in the *instance-name*`\conf\tomcat-users.xml` file for each client that must use client authentication.

   When a client is successfully authenticated, Tomcat validates the client against the corresponding user account configured in `tomcat-users.xml`. Tomcat then grants or denies the client access to the instance according to the role that is defined for the user. The user account consists of comma-separated values, such as country (`C`), state (`ST`), organization (`O`), organizational unit (`OU`), and domain name server (`CN`). Note that you enter these details when you create your client certificate.

   The following is an example user account added to the `tomcat-users.xml` file:

   ```
   <user username="C=US,ST=client,O=Progress,OU=Openedge,CN=www.progress.com"
   password="password" roles="ROLE_PSCAdmin,ROLE_PSCOper,ROLE_PSCUser" />
   ```

Note:

- Use the `pkiutil -list` command on the ABL client to retrieve details of the client username.
- Username details are displayed in the opposite order in the command window, starting with `CN` and ending with `C`. When you add the details to `tomcat-users.xml`, enter them in the following order: `C`, `ST`, `O`, `OU`, and `CN`.
- The user ID in the ClientPrincipal object is set to the Distinguished Name (`DN`) in the client certificate. The following is an example of the ClientPrincipal user ID:

  ```
  "CN=www.progress.com,OU=Openedge,O=Progress,ST=client,C=US"
  ```

4. Configure your ABL web application:

Configure ABL web application security

The Spring Security framework, which is an integral part of PAS for OpenEdge, is adaptable to a wide variety of authentication models and architecture. To configure security for your ABL web application, you must enable Spring Security in the PAS for OpenEdge instance, as described in the following steps:

1. Open the *instance-name*\webapps*webapp-name*\WEB-INF\oeablSecurity.properties file.
2. Enable Spring Security in the instance by setting the apsv.security.enable property to container.
3. Enable HTTPS client authentication in the <login-config> element in the *instance-name*\webapps*webapp-name*\WEB-INF\web.xml deployment descriptor file.

   For example:

   ```
   <login-config>
        <auth-method>CLIENT-CERT</auth-method>
        <realm-name>OpenEdge</realm-name>
   </login-config>
   ```

   For ease of configuration, the web.xml-clientcert deployment descriptor file is provided in the same directory as the web.xml deployment descriptor file. The web.xml-clientcert file includes the required login configuration and security constraints. Make a backup copy of the web.xml file, and then rename the web.xml-clientcert file to web.xml.

   By default, the security constraint defined in the web.xml-clientcert deployment descriptor file applies to all ABL web application transports. Simply customize the <url-pattern> element inside the <security-constraint> element according to your business needs.

   The following is an example <security-constraint> element definition that restricts container security to the APSV transport:

   ```
   <security-constraint>
           <display-name>PASOE oeabl Security Constraint</display-name>
           <web-resource-collection>
               <web-resource-name>Protected Area</web-resource-name>
               <url-pattern>/apsv/*</url-pattern>
           </web-resource-collection>
           <auth-constraint>
               <role-name>ROLE_PSCUser</role-name>
           </auth-constraint>
       </security-constraint>
   ```