Progress®



Learn about OpenEdge Security Domains

OpenEdge®

Copyright

© 2021 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, DevReach (and design), Icenium, Inspec, Ipswitch, iMacros, Kendo UI, Kinvey, MessageWay, MOVEit, NativeChat, NativeScript, OpenEdge, Powered by Chef, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, Sitefinity (and design), SpeedScript, Stylus Studio, Stylized Design (Arrow/3D Box logo), Styleized Design (C Chef logo), Stylized Design of Samurai, TeamPulse, Telerik, Telerik (and design), Test Studio, WebSpeed, WhatsConfigured, WhatsConnected, WhatsUp, and WS_FTP are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries.

Analytics360, AppServer, BusinessEdge, Chef Automate, Chef Compliance, Chef Desktop, Chef Habitat, Chef WorkStation, Corticon.js, Corticon Rules, Data Access, DataDirect Autonomous REST Connector, DataDirect Spy, DevCraft, Fiddler, Fiddler Everywhere, FiddlerCap, FiddlerCore, FiddlerScript, Hybrid Data Pipeline, iMail, JustAssembly, JustDecompile, JustMock, KendoReact, NativeScript Sidekick, OpenAccess, PASOE, Pro2, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, Push Jobs, SafeSpaceVR, Sitefinity Cloud, Sitefinity CMS, Sitefinity Digital Experience Cloud, Sitefinity Feather, Sitefinity Insight, Sitefinity Thunder, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Supermarket, SupportLink, Unite UX, and WebClient are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

July 2021

Product version: Progress OpenEdge 12.4

Updated: 2021/06/30

Table of Contents

Introduction to OpenEdge security domains.....9

How OpenEdge security domains work in real time to control access...17

OpenEdge security domain configuration tasks.....23

Preface

Purpose

The purpose of this document is to explain basic concepts about OpenEdge security domains, stressing the point that their requirements, configuration, use, and storage depend on the OpenEdge product with which they are configured.

Audience

Any developer, database administrator, security administrator, software architect, or other person responsible for designing, implementing, or managing secure access to an OpenEdge installation.

Organization

- Introduction to OpenEdge security domains on page 9 provides a high-level overview of OpenEdge security domains, describing their architecture and use with OpenEdge products.
- How OpenEdge security domains work in real time to control access on page 17 shows how security domains function in the execution flow of a client request.
- OpenEdge security domain configuration tasks on page 23 gives a high-level summary of the steps to configure an OpenEdge security domain.

Documentation conventions

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

Purpose

The purpose of this document is to explain basic concepts about OpenEdge security domains, stressing the point that their requirements, configuration, use, and storage depend on the OpenEdge product with which they are configured.

Audience

Any developer, database administrator, security administrator, software architect, or other person responsible for designing, implementing, or managing secure access to an OpenEdge installation.

Organization

- Introduction to OpenEdge security domains on page 9 provides a high-level overview of OpenEdge security domains, describing their architecture and use with OpenEdge products.
- How OpenEdge security domains work in real time to control access on page 17 shows how security domains function in the execution flow of a client request.
- OpenEdge security domain configuration tasks on page 23 gives a high-level summary of the steps to configure an OpenEdge security domain.

Documentation conventions

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

1

Introduction to OpenEdge security domains

OpenEdge security domains are modeled on a security domain pattern where servers, applications, and databases all trust a common security token that is issued to an authenticated user account that is recognized as a valid user of the domain. In an OpenEdge environment a OpenEdge security domain (also called an OpenEdge domain, or simply domain) constitutes a collection of PAS for OpenEdge ABL applications, OpenEdge Database instances, SQL-92, and OpenEdge Authentication Gateway components that all trust a common `CLIENT-PRINCIPAL` security token that is issued to users who have been authenticated to a domain-trusted user account. The result is the ability to isolate a user's access to the OpenEdge components of one domain from users of other domains.

The introduction of OpenEdge domains resulted from enabling multi-tenant support throughout the OpenEdge products. OpenEdge domains are an ideal security solution for maintaining the virtual isolation needed for securing applications and their data in multi-tenant SaaS operations. The result is that OpenEdge components now support multiple OpenEdge domains for their run-time user identity management and component access checking. Regardless of whether your OpenEdge application supports multi-tenancy, the multi-domain capability is still available for use. In many OpenEdge applications, the use of OpenEdge domains is simply transparent, and you may choose later how they can be employed to enhance security in your OpenEdge application.

Because support for OpenEdge domains is always present and used, the process of making the transition to one of the SaaS multi-tenancy models, or to a multiple domain security architecture, is made easier because you are able to add and configure OpenEdge domains in the OpenEdge components that require them. The goal of this document is to provide you with a basic understanding of OpenEdge security domains and how they work so that you can use them effectively within your OpenEdge application.

How OpenEdge security domains work

Every OpenEdge domain has two major attributes: a name, and an access code. The domain name is something you choose that is short but meaningful within your OpenEdge application. The access code is a shared secret that permits a CLIENT-PRINCIPAL token to be identified as representing a user of a single OpenEdge domain and therefore able to access the domain's components. When a CLIENT-PRINCIPAL token is presented by a client, its domain name is used by the component's authorization process to first look up the domain's configuration in the component's domain registry. If the domain configuration can be found, and its access code can be used to validate the CLIENT-PRINCIPAL security token, the component will permit access to the user represented by that CLIENT-PRINCIPAL token.

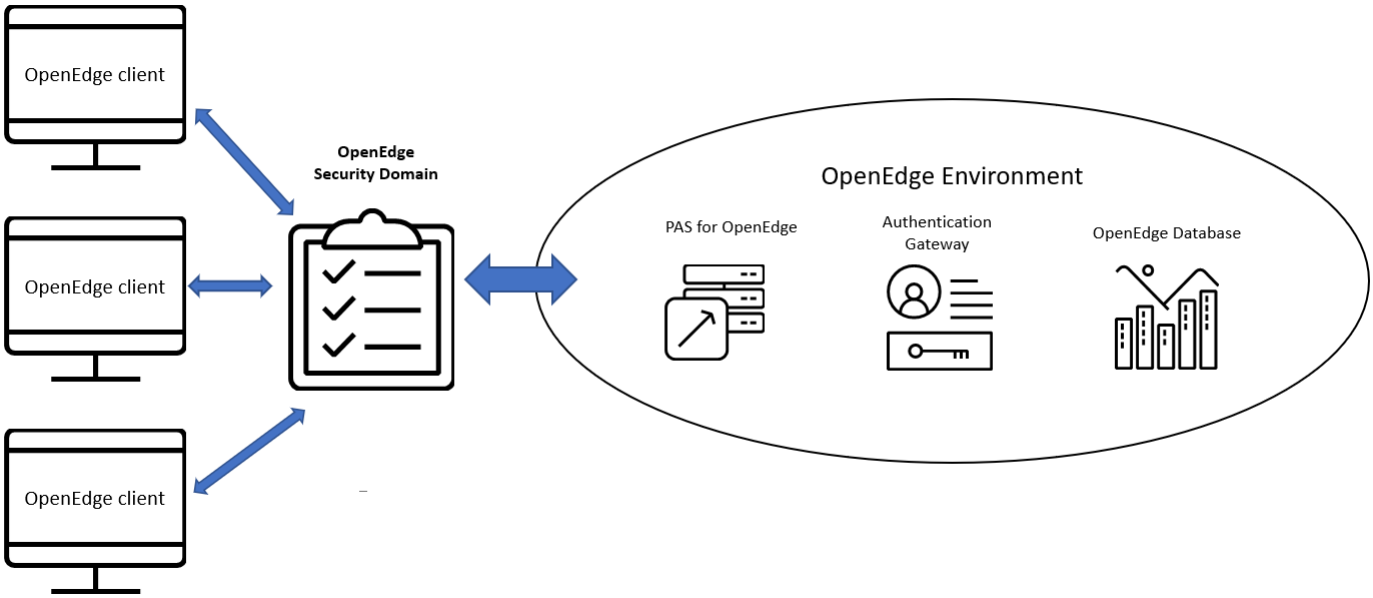
Therefore, accessing a domain component's function and data requires a CLIENT-PRINCIPAL token that can be validated by the OpenEdge component. Consequently, the user who wants access to the OpenEdge domain must first obtain a CLIENT-PRINCIPAL token by authenticating to one of the domain component's trusted authentication systems. Those domain component authentication systems validate user IDs and passwords that are located in secured and trusted locations. When the user account validation is successful, the client is issued a CLIENT-PRINCIPAL token that is sealed with the domain's access code, which makes the CLIENT-PRINCIPAL token usable only in a single OpenEdge domain. Alternately, when a client presents the domain component's authentication system with a non-OpenEdge security token, the authentication system will validate it using keys obtained from the token's source and the tokens expiration date. If that validation is successful, that token is then reissued as a sealed CLIENT-PRINCIPAL token in the same way as the authentication of a user ID and password.

After a CLIENT-PRINCIPAL token is obtained for an OpenEdge domain, the token may be used to access any OpenEdge component that is a member of that user's domain.

Architecture of a security domain configuration

After you understand the basics of how OpenEdge domains operate, the next step is to understand how an OpenEdge component becomes added or removed as member of a domain. OpenEdge components become members of a domain when you add a domain's configuration to the component, and are removed from a domain when you remove that domain's configuration. To enable support for multiple multi-tenant architecture models or security domains in existing OpenEdge applications, you may choose to isolate a single component to a single OpenEdge domain, or permit a single component to be a member of multiple domains. You accomplish that by managing the OpenEdge domain configurations in each OpenEdge component.

An OpenEdge security domain configuration applies to a single domain name and contains properties that are used by the component's authentication and authorization processes when operating in that domain. To continue support of OpenEdge products operating as standalone components, each member of an OpenEdge domain can individually be configured to be an authentication point that produces CLIENT-PRINCIPAL tokens for a domain, or an authorization point that controls user access based on a CLIENT-PRINCIPAL token's domain, or both.



OpenEdge has three separate architectures for storing a security domain configuration: one for a PAS for OpenEdge ABL application, one for OpenEdge Database, and another for OpenEdge Authentication Gateway. The main differences among these architectures is in how the domain configuration data is stored for use by the component's authentication or authorization processes, as follows:

- In OpenEdge Authentication Gateway, the security domain configurations control the authentication processes and specify which source of user accounts are used to authenticate a user to a domain and subsequently produce `CLIENT-PRINCIPAL` tokens for that domain. The domain configuration is split between a JSON storage file for non-sensitive domain properties, and an encrypted keystore for the sensitive domain properties. The OpenEdge Authentication Gateway authorization process, when enabled, also uses the domain configuration to isolate the OpenEdge installations that may access it and receive `CLIENT-PRINCIPAL` tokens.
- In a PAS for OpenEdge ABL application, the default authentication process operates using a single security domain configuration that specifies the source of user accounts that is used to validate user IDs and passwords. If the validation is successful, the domain configuration is used to control the creation of `CLIENT-PRINCIPAL` tokens that the user may use to access other members of its domain. The security domain configurations are also used in the ABL web application's authorization process to validate `CLIENT-PRINCIPAL` tokens that are passed by its clients before they are granted access to the application's functions and data. The domain configuration data is split between property files that hold non-sensitive properties, and an encrypted keystore for the sensitive domain properties.
- In OpenEdge Database, the security domain configurations are used in the authentication process to control the user account sources that may be used to validate a user's ID and password, and subsequently the creation of a sealed `CLIENT-PRINCIPAL` token. The security domain configuration is also used by the database authorization process to validate a client-asserted `CLIENT-PRINCIPAL` token before granting the user permission to use the database. This is especially the case where the database is enabled for multi-tenancy and each OpenEdge domain represents one tenant. The domain name (also known as the tenant name) is used to physically isolate tenant-accessible data from users in any other tenant domain. The domain configuration data is contained in restricted access database tables whose fields contain clear-text non-sensitive data and encrypted sensitive data.

Despite these architectural differences, the primary function of a security domain is consistent across the suite of OpenEdge products: to trust a common security token for authentication and authorization. However, the way in which domain configurations are specified and stored is contextual to each product. Therefore, the process of setting up and storing a security domain configuration depends on the OpenEdge product you are configuring.

Components of a security domain configuration

All OpenEdge security domain configurations are stored in the component's domain registry and have a common set of properties that are used by its authentication and authorization processes:

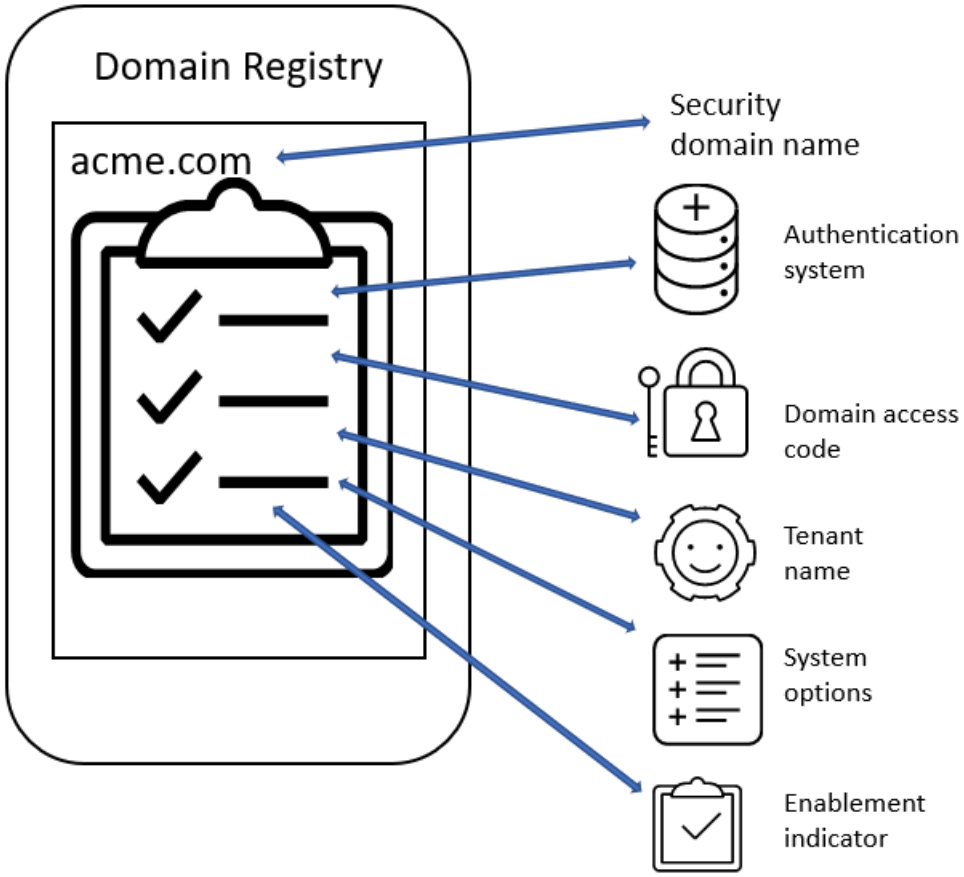
- **Domain name**—An OpenEdge domain is identified by a name that is unique across the application for all domains.
- **Domain access code**—A secret value used by OpenEdge to cryptographically seal a security token during user authentication and to validate a user's sealed security token. For example, to assert their identity in a single sign-on (SSO) operation. When specified, this is typically an encrypted value that is contained in a long character string whose encryption key is known only by OpenEdge.

  Note: This is not a user account password and should not be used as such.

- **Enablement indicator**—Specifies whether the domain is enabled for use at run time (the default). For example, the indicator allows you to configure domains for deployment while not making them available to an OpenEdge application until its deployment is complete.

Depending on the OpenEdge component, security domain configurations may also include the following types of properties for controlling the resource's authentication process:

- **The authentication system type**—The type of authentication system that the domain relies on for authenticating the user ID and password of all users of that domain; for example, the local operating system user accounts, or those stored in Microsoft Active Directory. OpenEdge supports several built-in authentication system types and allows you to define new ones, including ones that allow you to define your own authentication system. Each type of authentication system supports a particular set of user accounts and specific operations to validate and set a given user identity.
- **A tenant name**—For a multi-tenant database component, this property identifies the physical tenant to which the domain belongs, and through which all users in the domain have access to the tenant's data. For a non-multi-tenant database, this setting is ignored.
- **System options**—Allows you to specify options that can be used by the component's authentication process to connect to, and interoperate with, the authentication system type. Therefore the system options content is contextual to the authentication system type.
- **Additional settings**—To support auditing and other options that currently have limited or no effect on user validation.
- **`CLIENT-PRINCIPAL` settings**—Optional properties that provide fine-grained control of `CLIENT-PRINCIPAL` attributes in some components.



OpenEdge products that support security domains

Any OpenEdge component that authenticates and authorizes access requires a minimum of one security domain configuration. These products include the following:

- OpenEdge Database and select database utilities
- SQL-92
- PAS for OpenEdge
- OpenEdge Authentication Gateway

In brief, anywhere in OpenEdge where you can seal a `CLIENT-PRINCIPAL` token, and anywhere where you can validate a `CLIENT-PRINCIPAL` seal, must have a security domain registry.

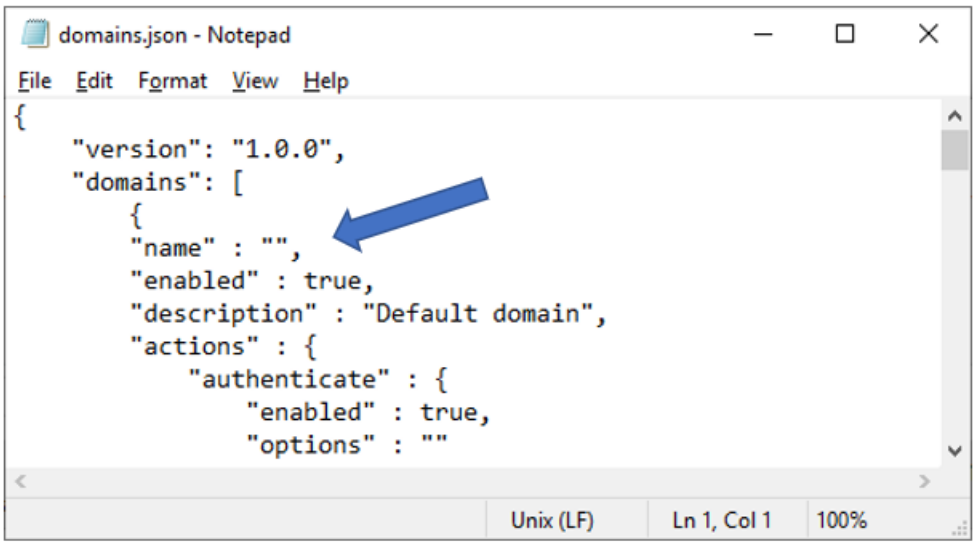
Default domain configurations and the blank domain

Each OpenEdge product whose authentication or authorization processes support using OpenEdge domains that requires a domain configuration will include a default OpenEdge domain configuration that is automatically set up during installation. This default configuration enables OpenEdge to work out-of-the-box and is suitable for quickly setting up an application environment in which the security requirements involve only a single group of users.

This default security domain is commonly referred to as the **blank domain**, and is used whenever a user's ID does not contain a specific domain that is qualified in the form of *@domain-name*. Included in the default blank security domain configuration is a default domain access code (""). The blank domain name is specified within a security domain configuration within two consecutive double-quote characters (""). Anytime a domain name is not a blank string, it is referred to as a **named domain**. The effect of using the blank domain is that all installed OpenEdge components will operate in a single domain, where any component may authenticate a user, obtain a `CLIENT-PRINCIPAL` token for the blank domain, and use that `CLIENT-PRINCIPAL` token to access any other OpenEdge component in that blank domain.

The default blank domain has a domain configuration like any named domain you may define. The blank domain operates the same way as named domains and may be disabled, its authentication system type changed, and its access code changed.

For example, the following code snippet shows the default domain configuration file that is provided with OpenEdge Authentication Gateway:



Notice the `name` attribute, which specifies the domain name, and that it contains a blank value. Note that the domain configuration does not have an access code field. In OpenEdge, all domain access codes are stored in an encrypted form, which for the Authentication Gateway is an encrypted Java keystore file whose default name is `domains.keystore`.

In PAS for OpenEdge ABL application domain configurations, the domain access codes are also stored in an encrypted Java keystore file, using the default name `ABLDomainRegistry.keystore`, which is found in the `conf/` directory of that PAS for OpenEdge ABL application instance.

The security domain configurations for OpenEdge Database are stored in two related tables. The domain access code is, in this case, stored as an encrypted value in one of the table fields.

When you set up OpenEdge to permit a user's access to multiple domains, you must specify full domain names in the user ID within domain configurations that are used to authenticate the user to the domain. The resulting `CLIENT-PRINCIPAL` token then includes the user's domain name so that it can be used to access components of that single domain.

Domain names and multi-tenancy

As stated previously, OpenEdge security domains came into being to support multi-tenancy across the OpenEdge products. One default blank domain is preconfigured in domain-enabled OpenEdge products in preparation for moving to more complex security architectures, such as multi-tenancy. When you are preparing for multi-tenancy, it is a good idea to create a sustainable naming convention for OpenEdge domains. Although you can choose any naming convention for your tenant-related OpenEdge domain names that suite your application, consider the following example to illustrate the options that are available to you.

As we know, businesses often change their names. They are also frequently subsumed by other businesses, and they are sometimes rendered inert as they realign their finances. The domain and tenant naming separation may help you with those business changes and transitions.

For example, a tenant's legal name is Acme General Products Corp and uses `acmeprods.com` as their internet domain name. You can expect tenant users to continue logging in using their internet domain names, and they can do that by using it by making their OpenEdge domain names be the same. But consider what happens when Acme General Products Corp changes its name to Eastern Business Machines LLC and begins using the internet domain `easternbp.com`. Because of the separation between a tenant that is stored in an OpenEdge Database instance and the OpenEdge domain names, the employees can continue to log in using the `acmeprods.com` OpenEdge domain name to access their tenant data. Later, you can add a new OpenEdge domain `easternbp.com` that maps to the same tenant data as the previous `acmeprods.com` did. After users are logging in using the new OpenEdge domain name, the previous OpenEdge domain name can be removed without interruption in tenant access. The same type of process can occur when one business buys another. Although this is a very simple scenario, it serves as an illustration of what is possible when you have the ability to use multiple OpenEdge domain names to access a single tenant organization's data.

In general, however, security domain names can be anything you want, provided that the name is unique and has a unique domain access code. You might find it useful to compose a domain name as a combination of the *business name* and the *application name*. That way, the domain name scheme can provide separation between the users of a given application who are associated with one company and the users of that application from a different company.

Domain access codes

In each OpenEdge domain configuration, every security domain name must have a corresponding domain access code. As mentioned earlier, a domain access code is a shared secret value used by OpenEdge to cryptographically seal a security token during user authentication and to validate a user's sealed security token. If your OpenEdge environment contains two or more domain configurations—for example, one for a tenant named `acme.com`, and another for a tenant named `greatreads.com`—the domain access code for a given security domain name must be the same in each domain component's configuration in order to permit access.

For example, to sign and validate the `CLIENT-PRINCIPAL` token that is created for `jane@acme.com` and that is used in both PAS for OpenEdge and OpenEdge Authentication Gateway components, the domain configurations for those two OpenEdge products must have the same domain access code for `acme.com`.

For multiple domain's security to operate effectively, each domain should have a unique access code. How that uniqueness appears in your OpenEdge application is best implemented as a design of the application and its configuration maintenance.

When you use the default blank domain, its access code is suitable for development operations because you can develop and unit-test your OpenEdge application to use `CLIENT-PRINCIPAL` tokens as an enabler to employ multiple domains without having to define and maintain multiple domains. However, when you move an OpenEdge deployment into a production environment, you should never retain the default blank domain's access code and should change it in every OpenEdge component where the default blank domain is configured the same way that any named domain's access code would be maintained.

Authentication system types

As mentioned earlier, a domain configuration controls the authentication processes. However, OpenEdge products cannot all use the same authentication process implementation, and this is reflected in the types of authentication system types that the components can use. OpenEdge supports the following authentication system types:

- `_oeusertable`—Built-in system that authenticates against the `_User` table accounts, and is supported for OpenEdge databases.
- `_oslocal`—Built-in system that authenticates against the local operating system user accounts on the system where the authentication process runs. This system type is available in OpenEdge Database and Authentication Gateway.

- `_extsso`—Built-in system that relies on callbacks into an ABL application to do user authentication, and is supported for OpenEdge databases.
- `properties : local`—Built-in system that authenticates user IDs and passwords against a simple text file. This system type is available in PAS for OpenEdge ABL applications and the OpenEdge Authentication Gateway for use in product development environments.
- `sts`—Built-in system that delegates a PAS for OpenEdge ABL application or OpenEdge Database's user authentication process to use OpenEdge Authentication Gateway to authenticate domain users and produce `CLIENT-PRINCIPAL` tokens.
- `LDAP : Active Directory`—Built-in system that authenticates users to a Directory Service account. This system type is available for the PAS for OpenEdge ABL application and OpenEdge Authentication Gateway.
- `OAuth2 : JWT`—Built-in system that is used to validate client-supplied non-OpenEdge security tokens and produce an equivalent `CLIENT-PRINCIPAL` token. This system type is available in the PAS for OpenEdge ABL application.
- `SAML`—Built-in system that is used to validate client supplied non-OpenEdge security tokens and produce an equivalent `CLIENT-PRINCIPAL` token. This system type is available in the PAS for OpenEdge ABL application.
- `OERealm`—Built-in system that enables a PAS for OpenEdge ABL application or OpenEdge Authentication Gateway authentication processes to call developer-written ABL code that interfaces with external user account systems that are not in OpenEdge's list of built-in Authentication system types. The OERealm does not actually authenticate the user, but does perform the tasks of looking up the user account, validating the account's password, and supplying account information that is folded into the generation of a `CLIENT-PRINCIPAL` token.

Where and how these authentication types are specified is contextual to the OpenEdge product. For example, in OpenEdge Authentication Gateway, you identify the authentication system type by specifying the `authProvider` property in the `domains.json` file. In OpenEdge Database, you use the Data Administration tool. In PAS for OpenEdge ABL applications, you use the `oeablSecurity.properties` file.

The default blank domain includes configuration properties for designating a default authentication system type. The choice of authentication system type is consistent with the type of OpenEdge component and its ability to be a standalone component that is capable of both authenticating users and to granting them access to itself. Those default authentication system types are:

- OpenEdge Database (including ABL and SQL-92 clients)—The `_oeuser` table
- PAS for OpenEdge ABL application—The `user.properties` text file
- OpenEdge Authentication Gateway—The `user.properties` text file

Those default authentication system types are chosen specifically to make it easier for developers to use, but you may change them to another type as necessary. Normally, these authentication system types are recommended to be used only in development environments because they do not support the level of user-account security necessary for production systems. You may continue to use the default blank domain, but you should change the authentication system type before you use it in a production deployment.

For more information, see "Defining and specifying the system type" in *Learn about Identity Management*

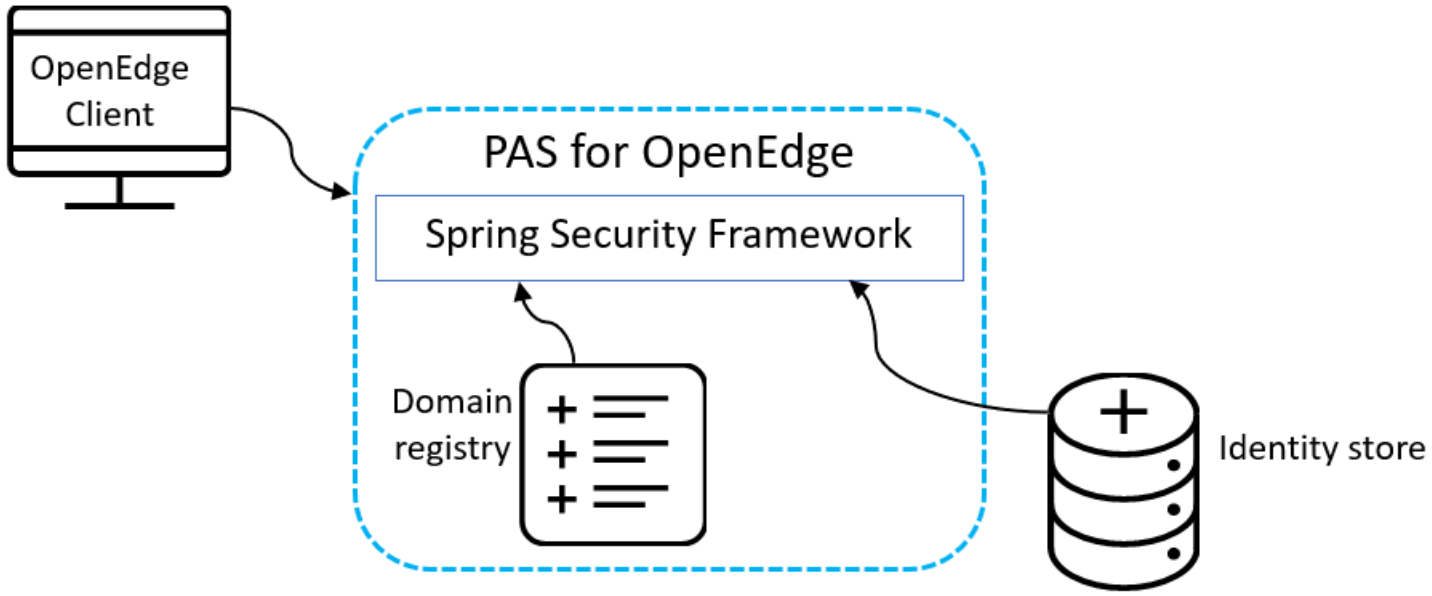
2

How OpenEdge security domains work in real time to control access

Any location in your OpenEdge application architecture where a user can be authenticated, or where a `CLIENT-PRINCIPAL` token can be validated, is called a **point of authentication**. In the OpenEdge environment, each point of authentication must have a minimum of one security domain configuration. Consequently, each source of user identity information that is used in that point of authentication must be specified within a domain configuration. How many domains your OpenEdge environment must have, and understanding where all the points of authentication are, is central to creating a security domain architecture for your environment that can be easily scaled or modified as your requirements change.

Basic security domain scenario with single PAS for OpenEdge ABL application

The following figure shows a basic OpenEdge environment that uses a single security domain that is configured in a domain registry for a PAS for OpenEdge ABL application.



When the OpenEdge client user performs a direct login to invoke a service hosted in the PAS for OpenEdge ABL application, the embedded Spring Security Framework authenticates the client by using the domain configuration to know where and how to look up the user's credentials in an identity store. The identity store can be an LDAP server, such as Microsoft Active Directory, or the local operating system that is located on a network-connected OpenEdge Authentication Gateway instance, for example. If the user can be authenticated (that is, the user account is found, is enabled for use, and its password validates):

1. Spring uses the domain configuration to assign an OpenEdge domain name, if the input user ID does not contain one. That fully qualified user ID is then used throughout the remainder of the authentication process. You may retain the use of the default blank domain, but you can configure another.
2. Spring uses the single authentication system configuration for all user logins and interacts with it to validate the user ID account, its enabled state, its expiration, and its password value.
3. Spring generates an internal token for the user, which contains the user and optionally the user account's data. That is, the user name, roles, and domain.
4. PAS for OpenEdge converts the Spring token into a `CLIENT-PRINCIPAL` object using the `CLIENT-PRINCIPAL` generation properties located in the domain configuration.
5. Spring then reads from the domain registry to determine whether the user's domain to be recorded in the `CLIENT-PRINCIPAL` object is listed in the domain registry, and that the domain is enabled. If there is a match, the user's `CLIENT-PRINCIPAL` token is sealed and access to the service is allowed, otherwise access is denied because the component is not configured as a member of the domain.
6. The `CLIENT-PRINCIPAL` token is then delivered to the ABL business application code on this and every subsequent request.
7. The business application code can use the `CLIENT-PRINCIPAL` token to access the OpenEdge database, if that database's domain configuration includes the user's domain name and the token can be validated using the domain's access code.

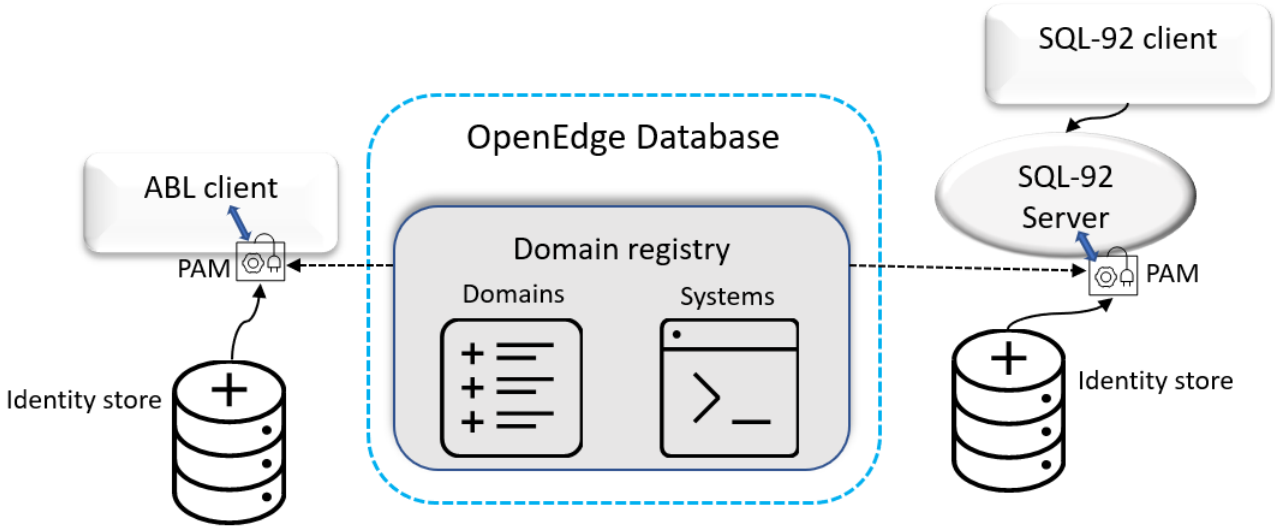
When the PAS for OpenEdge ABL application is configured to use an OpenEdge Authentication Gateway instance, it attains the ability to handle user authentication for multiple domains, each with its own authentication system configuration.

In the case where a non-OpenEdge-issued single sign-on token is presented to PAS for OpenEdge:

1. Spring Security validates the external token using keys that Spring obtains from the issuer of that token.
2. PAS for OpenEdge assigns the token's user to an OpenEdge domain and then converts the token into a `CLIENT-PRINCIPAL` object according to the `CLIENT-PRINCIPAL` generation properties found in the domain configuration.
3. Spring reads from the domain registry to determine whether there is a match with the domain in the `CLIENT-PRINCIPAL` object. As with direct login, access is permitted depending on whether there is a match with a domain that the component is a member of.
4. The `CLIENT-PRINCIPAL` token is then delivered to the ABL business application code on this and every subsequent request.

Basic security domain scenario with OpenEdge Database

The following figure shows OpenEdge Database with an ABL client, a SQL-92 client, and a SQL-92 Server client. The ABL and SQL-92 Server clients are configured with a pluggable authentication module (PAM), which reads domain registry information that is stored in OpenEdge Database.



In OpenEdge Database, the domain registry is made up of two database tables. One is the domain table, which lists all the domains from which access is permitted, and the other table specifies the authentication systems that can be used for clients that create a database connection. Together these tables specify to the PAM for each client which identity store to use.

The authentication process that is used depends on the client type, as follows.

ABL client

1. The PAM authentication process assigns the default blank domain to the user ID if the input user ID does not contain one. That fully qualified user ID is then used through the rest of the authentication process.
2. The PAM reads the domain configuration records from OpenEdge Database and uses the authentication system configuration for the user ID's domain and interacts with it to validate the user ID account, its enabled state, its expiration, and its password value. If the domain is not configured or is disabled, the authentication attempt is rejected.
3. The PAM generates an internal token for the user, which contains the full user specification and optionally the user account's data. That is, the user name, domain, and optionally other user account information.
4. The PAM converts the token into a `CLIENT-PRINCIPAL` token.

5. The PAM seals the user's CLIENT-PRINCIPAL token using the domain-configured access code and stores it in the database connection.
6. The CLIENT-PRINCIPAL token is then made available to the ABL business application that can use it to access other OpenEdge databases or PAS for OpenEdge ABL applications.

When the PAS for OpenEdge ABL application is configured to use an OpenEdge Authentication Gateway instance, the ABL application attains the ability to handle user authentication for multiple domains, each with its own authentication system configuration.

In the case where an OpenEdge CLIENT-PRINCIPAL token is used to single sign-on to the database connection:

1. Spring Security validates the external token using keys that are obtained from the issuer of that token.
2. PAS for OpenEdge assigns the token's user to an OpenEdge domain and then converts the token into a CLIENT-PRINCIPAL token according to the CLIENT-PRINCIPAL generation properties that are found in the domain configuration.
3. Spring reads from the domain registry to determine whether there is a match with the domain in the CLIENT-PRINCIPAL object. As with direct login, access is permitted depending upon whether there is a match with a domain of which the component is a member.

The CLIENT-PRINCIPAL token is then delivered to the ABL business application code on this and every subsequent request.

SQL-92 client

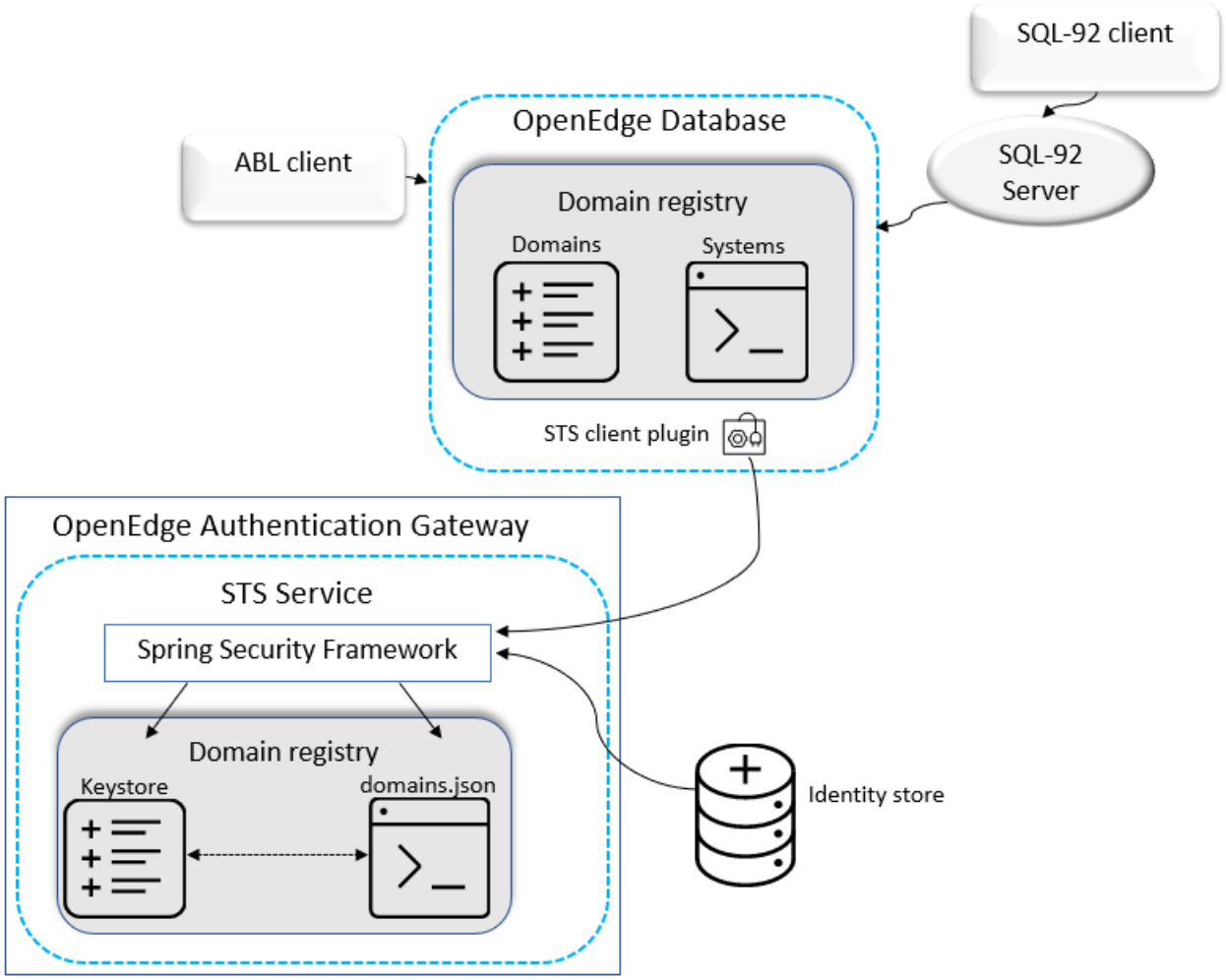
The client login to SQL-92 follows the same direct-login process as described previously for the database's ABL client. The difference is that:

- The sealed CLIENT-PRINCIPAL token is not made available to the SQL-client where it can be used to access other SQL-92 databases.
- The SQL-92 database connection will not accept a sealed CLIENT-PRINCIPAL token to gain access.

Basic security domain scenario with OpenEdge Authentication Gateway

You can configure OpenEdge Database or a PAS for OpenEdge ABL application to delegate its client authentication process to OpenEdge Authentication Gateway. In this scenario, when a client executes a direct login, the login operation is executed in the Authentication Gateway instead of locally. The SSO token authentication to these domain components remains local and continues to use the local OpenEdge domain registry configuration to reduce network overhead during peak operating times.

The Authentication Gateway exists as a single service for all PAS for OpenEdge and OpenEdge Database component instances in your application environment. It is fully multi-domain and multi-tenant enabled for direct user logins so that you maintain a single set of domain configurations in a secured server environment. The additional benefit that you derive from using the Authentication Gateway is that it permits you to supplement the provisioning and auditing of CLIENT-PRINCIPAL token generation using ABL. This single location of domain configuration and its security brings benefits to larger application environments and reduces maintaining many domain configurations in many locations.



In this configuration, whereby OpenEdge Database delegates the authentication process to OpenEdge Authentication Gateway, the PAM plug-ins used by the ABL and SQL-92 Server clients are substituted by an STS client plug-in, which is configured in OpenEdge Database. Whereas the PAM plug-in reads domain information from the domain registry tables in OpenEdge Database, the STS client plug-in delegates domain validation and authentication to the Spring Security Framework, which is part of the STS service in Authentication Gateway. The specifics of the authentication process depend on the database client type.

The PAS for OpenEdge ABL application follows the same pattern as ABL and SQL clients.

ABL client

When a database client executes a `CONNECT` operation to the database, the database forwards the user ID and password, by means of the STS client plug-in, to the STS service's Spring Security Framework in the Authentication Gateway to perform a login. The `domains.json` file contains the information that tells the Spring Security Framework how to connect to the identity store; that is, the location and connection information for the identity store. The Spring Security Framework then compares the domain that is passed in with the user identity to the domains listed in the domain registry. If a match is not found, the authentication process stops and returns an invalid domain exception. Basically, if the domain in the user credentials cannot be found in the domain registry, then the Spring Security Framework cannot find the configuration for the identity store that would be needed to locate the user account. However, if there is a match between the domain in the user identity and the domain registry, the Spring Security Framework reads in the user account information for that user from the identity store. If the user can be successfully authenticated, the Spring Security Framework provisions a `CLIENT-PRINCIPAL` token, seals it, and returns it to the database.

When the database receives the CLIENT-PRINCIPAL token, the database then validates the domain in that token against the domains listed in its domains table. If the database can validate the CLIENT-PRINCIPAL token—that is, the domain access code that was used to seal the token matches the domain access code in the database's domain registry—then the database accepts the ABL client's connection and returns the CLIENT-PRINCIPAL token to that client. Note that the business application for that ABL client can extract that token from the database connection and pass it over to other application servers and database connections. After the application obtains the CLIENT-PRINCIPAL token, the application can use that token to connect to other PAS for OpenEdge instances and perform single sign-on in other locations in the OpenEdge environment rather than having to pass a user ID and password around that continually requires re-authentication. Passing CLIENT-PRINCIPAL tokens among components in this way also facilitates auditing: because the tokens come from a trusted source, they are more secure and easier to trace.

SQL-92 client

The SQL-92 database client follows exactly the same use pattern as the ABL client, with the difference being that the sealed CLIENT-PRINCIPAL token is not available to the SQL-92 client for use with other databases.

3

OpenEdge security domain configuration tasks

The specific steps to configure an OpenEdge security domain depend on the OpenEdge product you are setting up. Those steps are contextual to the architecture of each product, where security domain information is stored, and how that information is consumed with the clients of those products.

For information about names that you can assign to OpenEdge security domains, see "Defining the name of a domain" in *Learn about Identity Management*.

PAS for OpenEdge

The domain configuration's access codes for a PAS for OpenEdge instance exists within a single Java keystore that is generated by the `gendomreg` utility. This keystore file is created from a CSV file that contains a list of each authorized domain name and its corresponding domain access code.

Note that when you install PAS for OpenEdge, `ABLDomainRegistry.keystore` is located by default in the `myInstance/conf` directory. This file contains encrypted values for the blank domain and its corresponding blank domain access code.

To configure a security domain for PAS for OpenEdge:

1. Create a domain registry CSV file that lists each OpenEdge security domain name and its associated domain access code in clear-text. Each security domain name and its access code are separated by a comma. For example:

   ```
   acme1.com,123456
   acme2.com,654321
   acme3.com,098765
   ```

2. Start the Proenv command shell.

3. In the Proenv command shell, run the `gendomreg` utility to create an encrypted Java keystore file from the CSV file. For example, the following command creates the keystore file `ABLDomainRegistry.keystore`.

   ```
   $DLC/bin/gendomreg ABLDomainRegistry.csv ABLDomainRegistry.keystore
   ```

4. Copy the keystore file into the `conf` directory of each PAS for OpenEdge instance in your environment.

Note: The keystore file is initially distributed with the default blank domain and blank domain access code. You have the choice of whether to remove the default blank domain, and thereby disable it, or retain it and change its domain access code. The following examples show how to use these `.csv` file entries (disregard the single quotes that are there to delimit the beginning and end of a `.csv` file line):

```
default domain and access code:                          ','
default domain and a non-blank access code:    ',abcde2'
```

The remaining domain configuration properties are located in the ABL web application's `oeablSecurity.properties` file.

OpenEdge Authentication Gateway

For OpenEdge Authentication Gateway, the security domain configuration comprises two files that are stored in the `webapps/ROOT/WEB-INF/config` directory of an Authentication Gateway server instance:

- `domains.json`—A simple JSON-formatted text file that you modify to specify the domain configuration.
- `domains.keystore`—Domain access codes for the OpenEdge Authentication Gateway are stored separately as a Java keystore. You create the keystore using the OpenEdge `gendomreg` command.

Tip: Use a JSON GUI editor to simplify adding and configuring domain configurations.

To configure a security domain for OpenEdge Authentication Gateway:

1. Make the following updates to the `domains.json` file:
   - Add the OpenEdge security domain names that you want to include.
   - For each security domain you add, specify the corresponding authentication type and provider.

     For more information, see "Configure domains" in *OpenEdge Authentication Gateway Guide*.
   - Enable the domain.
   - Configure the authentication processes that the domain will support.
   - Optionally add to the configuration ABL classes you have written to tailor the `CLIENT-PRINCIPAL` token provisioning and auditing steps.

2. Create a domain registry CSV file that lists each OpenEdge security domain name and its associated domain access code in clear-text. Each security domain name and its access code are separated by a comma. For example:

   ```
   acme1.com,123456
   acme2.com,654321
   acme3.com,098765
   ```

3. Start the Proenv command shell.

4. In the Proenv command shell, run the `gendomreg` utility to create an encrypted Java keystore file from the CSV file. For example, the following command creates the keystore file `domains.keystore`.

   ```
   $DLC/bin/gendomreg domainsks.csv domains.keystore
   ```

5. Copy the keystore file into the directory in which the `domains.json` file is located.

   For more information, see "Configure domain access codes" in *OpenEdge Authentication Gateway Guide*.

OpenEdge Database

For OpenEdge Database, you define and configure security domains in each OpenEdge Database instance. You then enable or disable each security domain for run-time use. A security domain for use with OpenEdge Database is:

- Configured with an authentication system that has access to valid user accounts and supports user authentication
- Authentication-enabled if the domain is enabled for run-time use

OpenEdge Database provides a default domain registry that is stored within two database tables. One table identifies the configured security domain names, and also domain-to-tenant relationships if the database is enabled for multi-tenancy. The second table contains the domain access codes. This arrangement allows OpenEdge Database to access its security domains for authenticating the identities assigned to OpenEdge database connections. These database connections are typically created for the following:

- Database clients, such as ABL applications and database utilities
- AVM
- OpenEdge SQL Server

You configure OpenEdge-supported security domains in an OpenEdge RDBMS using database administration tools, including the:

- Database Administration Console in OpenEdge Management
- Data Administration utility in the OpenEdge program group on Windows
- **Admin** menu of the character-mode Data Dictionary

For more information, see "Domain Maintentance option" in *OpenEdge Database Tools*.