

Security

Pega Platform protects you against a wide variety of security risks, whether inadvertent or malicious. Use the platform features related to authentication, authorization, and auditing to protect and monitor the use of your application.

Security basics

Security failures can expose your organization to severe consequences, such as a negative perception of your organization's reputation, customer loss, lack of customer trust, and potential legal and financial penalties.

Goal of security

The goal of security is to maintain availability, integrity and confidentiality. This goal is primarily accomplished by implementing authentication, authorization, and auditing. When confidentiality is compromised, unauthorized individuals gain access to systems or data. When integrity is compromised, unauthorized individuals can modify systems or data. When availability is compromised, unauthorized individuals can cause disruption of application or web availability, affecting access timing and uninterrupted access.

The combination of an evolving regulatory environment and threat landscape have put a burden on customer engagement and digital process automation teams. Critical business systems have become more interconnected and need to maintain increasingly sensitive data as regulations expand.

Pega Platform security features

Pega Platform provides powerful capabilities for implementing security in your applications, especially when you deploy guardrail-compliant applications. The Pega Platform model-driven architecture helps you to secure applications in most cases by configuring built-in features, and you do not need to rely on custom code built by developers who are not security experts.

Other Pega Platform security components

In addition to features that explicitly accomplish authentication, authorization, and auditing, other Pega Platform components represent important policies, assets, and safeguards to use with these features.

Certificate, key, and token management

The management of these important assets is critical to the secure functioning of other security features.

Confidentiality and encryption

The confidentiality of your sensitive data at rest, in transit, and in use is extremely important. Pega Platform uses state-of-the-art encryption features that allow you to secure sensitive information at any point in a business process.

Virus checking

Pega Platform allows your application to link to a third-party virus checking program before processing any email or attachment.

Content security policies (CSP)

Use CSP to lock down your application to mitigate the risk of content injection vulnerabilities (such as cross-site scripting) and reduce the privileges required to run your application. Pega Platform only sends these headers on dynamic content requests, not static content requests.

For more information, see:

- [Content security policies](#)
- [Creating a content security policy](#)

- [Security foundations](#)

Security and privacy are concerns at the forefront of every organization. Understanding security foundations helps you to implement a comprehensive security solution. Secure your systems against attack to avoid negative customer perception and potential regulatory sanctions.

- [Security Checklist](#)

The Security Checklist provides Pega's leading practices for securely deploying applications. To assist you in tracking the completion of the tasks in the Security Checklist, Pega Platform™ shows the overall completion on the Dev Studio Home page, and built-in ways to track the status of each task.

- [Authentication](#)

Apply authentication methods to ensure that only users and systems with a verified identity can access your applications, web pages, APIs, and data. Authentication includes verifying user credentials, Pega Platform requests to external services, and external service requests to Pega Platform. You can also authenticate by using an external identity provider.

- [Authorization](#)

Authorization ensures that after logging in, users have access to only the features and data that they need for their work. Pega Platform offers three types of authorization: role-based access control, attribute-based access control, and client-based access control. You can use these authorization features together to provide the strictest level of control.

- [Auditing](#)

With Pega Platform, you can track many types of security events, such as failed logins, password changes, and changes to rules and data. By tracking all of these events, you can understand how your system functions and detect any potential problems.

- [Mitigate common \(OWASP Top 10\) security vulnerabilities](#)

Pega Platform offers policies on the Security Policies landing page, as well as additional security restrictions that control cross-site request forgery (CSRF), content security policies (CSP), cross-origin resource sharing (CORS), and other types of vulnerabilities. Use these features to ensure that your system is as secure as possible.

- [Secure your application user interface](#)

As a security administrator, you permit or restrict groups of users to access various actions in an application, such as having access to a case type, flow action, or button.

- [Verify requests at the application layer](#)

Pega Platform protects access to information in your application by using role-based settings and access control policies. Pega Platform provides additional request verification when you use autogenerated controls.

- [Security operations](#)

Beyond authentication, authorization, and auditing, Pega Platform offers many other configurable security features, such as encryption, HTTP response headers, and Web Service Security profiles. Use these features to ensure that your application is as secure as possible.

- [Security in App Studio](#)

To increase system usability and better accommodate multi-functional teams, some parts of the security framework are now available in App Studio.

Security foundations

Security and privacy are concerns at the forefront of every organization. Understanding security foundations helps you to implement a comprehensive security solution. Secure your systems against attack to avoid negative customer perception and potential regulatory sanctions.

Security objectives

Security policies and standards are the framework that is used to define the requirements and controls used to secure applications and data. Pega Platform must comply to prevent unauthorized access to systems and mitigate attacks that negatively impact the confidentiality, integrity, and availability of client environments. These types of events cost our clients time, money, and brand integrity.

The components of the Pega security framework include:

Policies

The ground level requirements for relevant, in scope functions.

Standards

Detailed requirements for relevant, in scope functions.

Procedures

Documented guidelines and instructions to maintain compliance with the policies and standards.

Pega operates a broad policy stack where corporate functions like HR, legal, IT, and other relevant groups that support the entire company maintain policies and standards that are applicable to the entire workforce and relevant subcontractors. Business groups then use these policies and standards to build additional requirements based on regulatory or contractual obligations in support of the business outcomes, however these requirements cannot be defined in any less restrictive way than the corporate function policies.

Successful implementation of and compliance with the Pega security framework provides the following advantages:

Access control

Prevention of unauthorized access to systems and data.

Availability control

Prevention of attacks on systems that degrade the confidentiality, integrity, or availability of Pega Platform environments.

Audit management

Avoidance of costly and time-consuming audits to determine the source or impact of a security event.

Confidentiality, integrity, and availability triad

Confidentiality, integrity, and availability (CIA) triad, is a model that is designed to guide policies for information security within an organization. The elements of the CIA triad are considered the three classical components of security:

Confidentiality

Refers to protected personal data, and to the way in which all information or data should be secured to never share that information with third or unauthorized parties.

Integrity

Refers to the fact that the service or system is setup in such a way that no information or data can be altered without detection.

Availability

Refers to the services or parts of the systems that, regardless of circumstances, should be up and running and should always respond.

Security features

Pega Platform provides:

- A broad range of security capabilities to prevent malicious use of, and access to, an application.
- Powerful capabilities for implementing security in your applications, especially when you deploy guardrail-compliant software.

You can use the Pega Platform model-driven architecture to secure applications in most cases by configuring built-in features, without relying on custom code that is built by developers who are not security experts.

Authentication

Pega Platform supports multiple types of authentication, including the most common authentication protocols that are performed external to Pega Platform. To perform critical application functions, you must be in an authenticated session. Passwords or other sensitive information between the client and the application are not exchanged other than during the initial sign-on request. Failure messages do not contain sensitive information. Supported authentication protocols include:

- SAML 2.0
- OpenID Connect
- Basic credentials
- Token credentials
- Anonymous
- Custom
- Kerberos

You can also configure multi-factor authentication (MFA).

The built-in authentication capabilities of Pega Platform provide support for the definition of a security policy that covers a range of options. The options that are available while configuring a security policy include:

- Minimum password lengths
- Minimum numeric, alphabetic, and special characters required in the password
- Minimum and maximum password age
- Maximum unique historical passwords
- Number of failed login attempts before lockout
- Initial lockout penalty in seconds
- CAPTCHA authentication settings
- Inactivity disablement

For more information, see [Authentication](#).

Authorization

The authorization model in Pega Platform includes role-based, attribute-based, and client-based access controls. Permission to access data objects and application functions is determined dynamically by the roles and attributes of the user.

You can apply access controls to an entire class of data objects or focus on a particular field in a record. Pega Platform provides tools for the security administrator to ensure that the configuration meets requirements, for example, by running access control simulations.

For more information, see [Authorization](#).

Session management

Pega Platform allocates a session object on behalf of the user by using a randomly generated, unique session value to identify the session object. The session ID contains sufficient entropy (greater than 128 bits) to prevent collisions and successful guessing by attackers.

Data validation

Request processing come standard with several layers of protection against malicious attacks, which often target and attach to input and output data. Pega Platform provides continuous protections at the server level in addition to any that are enforced by the client:

- During input processing, request data is typically assigned to application properties, which are specified to contain well-defined data types. These data types trigger server-side data validation whenever a value is assigned to a property. Some examples of data types include: *integer*, *decimal*, *double*, *DateTime*, *TimeOfDay*, *Date*, and *TrueFalse*.
- Enforced length limits are applied.
- You can configure free text input values for validation against a list of valid entries. For example, you can predefine the list during application development, or you can evaluate the list dynamically by using a database lookup at run time.
- Several validation rule types are available to configure on-site custom validation logic where necessary.
- A cross-site scripting filter is used during input and output processing.
- Validation of the session identifier, content encoding, content type, and other content headers is performed.

Cryptography

Cryptography facilities for Pega Platform are based on the Java Cryptography Extensions API. These facilities rely on cryptography providers, such as those supplied by the Java JDK vendor or the Bouncy Castle JCE, that is included with Pega Platform.

Cryptography facilities provide encryption of sensitive data at rest and protection against unauthorized access. **Note:** Pegasystems does not directly implement the cryptographic algorithm logic.

For more information, see [Encryption](#).

Auditing

Pega Platform audits a complete list of actions, including both successful and unsuccessful attempts to access and modify data. You can also define custom auditing rules. Aggregate the logged data to detect patterns of suspicious behavior. Some of the auditing features include:

- Rule changes
- Security policy changes
- Login failures and successes
- Invalid data access attempts

For more information, see [Auditing](#).

Security alerts

Pega Platform logs security alerts whenever it detects a condition that represents a possible security incident, which includes:

- User-switching attempts
- Access to a restricted activity, stream, or report
- Unauthorized data access
- Session hijacking
- Cross-site request forgery (CSRF) attacks
- Injection attacks
- Content Security Policy violations

For more information, see [Tracking and auditing actions by developers and users](#).

Security Checklist

The Security Checklist provides Pega's leading practices for securely deploying applications. To assist you in tracking the completion of the tasks in the Security Checklist, Pega Platform™ shows the overall completion on the Dev Studio Home page, and built-in ways to track the status of each task.

Pega takes application and system security extremely seriously. Security is a shared responsibility between Pega and our clients.

Each successive release of Pega Platform augments the security features and capabilities available to harden applications and systems against improper access and to protect the data that those applications manage.

The Security Checklist is the key feature of Pega Platform that assists clients in hardening their applications and systems. The Security Checklist does the following:

- Provides Pega's leading practices for securely deploying applications.
- Helps protect the confidentiality, integrity, and availability of your application in production.
- Identifies when each task should be performed: at or near the beginning of development, on an ongoing basis, or just before deployment.
- Helps avoid expensive rework late in your development process.

To assist you in tracking the completion of the tasks in the Security Checklist, Pega Platform automatically installs an application guideline rule instance that includes the tasks in the Security Checklist for each version of your application.

For more information, see [Assessing your application using the Security Checklist](#).

Note: If you are using Deployment Manager, starting in Pega Platform 8.5, completing the Security Checklist is required to move an application into production.

Security is critical

Inadequate security can prevent your application from being deployed in two ways:

- Most clients need approval from the IT security group who review the application's security before the project team can put the application into production.
- If a client is building their application with Deployment Manager, the application will be blocked if the Security Checklist has not been completed.

Your responsibility as an administrator, senior system architect, or lead system architect is to ensure the confidentiality, integrity, and availability of your application. Unauthorized individuals should not be able to access or modify the application or the data it creates and stores. Further, users should only have access to those application functions and data that are necessary to perform their jobs.

Prepare to complete the Security Checklist

At the beginning of application development, determine who is responsible for verifying the completion of the tasks in the Security Checklist, and assign clear responsibility for each task to the Security Administrator.

Determine who is responsible for this Checklist

- Create a Security Administrator work queue.
- Add operators to this work queue who are responsible for verifying the completion of checklist tasks.
- Click the option on each task to create a corresponding user story, give it a high priority, and assign to this work queue.

For more information, see:

- *Notifying operators of work queue activity.*
- [Defining work routing settings for an operator.](#)

Independent security assessment

An independent assessment of your application's security, and sign-off of that assessment, is always recommended and is required by most organizations' IT Security groups.

Guardrail compliance: the basis of a secure application

The most important security requirement for all Pega Platform applications is to maintain guardrail compliance. Pega Platform security features cannot always be successfully enforced in custom code.

Use the built-in security configuration features in Pega Platform to protect your application, and do not rely on custom code built by developers who are not security experts.

Review the **Application Guardrails** landing page weekly and make changes to keep your application rules in compliance.

Do not wait until deploying your application to eliminate non-compliant rules, because applying changes is costlier after deployment.

For more information, see [Improving your compliance score](#).

If your application includes custom Java or custom HTML written by your project team, you must perform special tasks to secure that code.

For more information, see:

- [Security Checklist additional tasks.](#)

Not all security tasks are required for all applications or releases

It's important to understand the nature of the application and how it will be deployed when reviewing the Security Checklist tasks – these tasks are not required for all applications:

- [Security Checklist core tasks](#) are critical and should be performed for *every* application.
- [Security Checklist additional tasks](#) may be required, depending on what Platform features your application uses, how much you customize a Pega application, the amount of sensitive data created and stored within the application, and other factors.
- Special considerations apply for public-facing applications, where not all users are your employees.

For more information, see [Basic requirements for deploying public-facing applications](#).

- Your choice of deployment environment also determines which tasks you need to complete. Pega Cloud configures most deployment environment security for you automatically.

For more information, see:

- [Security Checklist when deploying on Pega Cloud](#)
- [Security Checklist when deploying in on-premises environments](#)

- [Security Checklist core tasks](#)

The Security Checklist provides Pega's leading practices for securely deploying applications. To assist you in tracking the completion of the tasks in the Security Checklist, Pega Platform shows the overall completion on the Dev Studio Home page, and built-in ways to track the status of each task.

- [Security Checklist additional tasks](#)

These tasks are not part of the core Security Checklist because they do not apply to all applications. You should review these additional tasks and determine if they apply to your application.

- [Security Checklist when deploying on Pega Cloud](#)

For applications deployed on Pega Cloud services, there are additional considerations you should address when completing the Security Checklist.

- [Security Checklist when deploying in on-premises environments](#)

When you are deploying on-premises, there are additional considerations you should address when completing the Security Checklist.

- [Assessing your application using the Security Checklist](#)

Use the Security Checklist to prepare your application for deployment. By completing the tasks on this checklist, you can safeguard sensitive data and improve the security of your application.

- [Adding the Security Checklist to an application created before 7.3.1](#)

The Security Checklist is automatically added to applications starting in Pega Platform 7.3.1. You can manually add the Security Checklist to applications that were created in earlier versions. By completing the tasks on the checklist, you can improve the security of your application.

- [Implementing security guidelines for test environments](#)

As a best practice, configure the application server in your test environment so it mirrors a production environment configuration.

Security Checklist core tasks

The Security Checklist provides Pega's leading practices for securely deploying applications. To assist you in tracking the completion of the tasks in the Security Checklist, Pega Platform shows the overall completion on the Dev Studio Home page, and built-in ways to track the status of each task.

The Security Checklist core tasks are critical and should be performed for *every* application.

Tasks to perform during development

Keep application rules guardrail-compliant

The most important security requirement for all Pega Platform applications is to maintain guardrail compliance.

Review the Guardrails landing page weekly and make changes to keep your application rules in compliance. Many security features can only be properly enforced in application rules that comply with Pega Platform guardrails. Applying changes to non-compliant rules is more costly after you deploy your application.

Address security alerts promptly

At least weekly, review run-time security alerts and take appropriate remedial actions to identify, and where possible, eliminate what caused the alerts.

For more information, see

- [Monitoring security alerts and events](#).
- [List of performance and security alerts in Pega Platform](#).

Securely authenticate attempts to access services

Verify that all default service packages and custom authentication services are secured appropriately.

To prevent unauthorized access of services not in use, ensure that the Service Package instances have, at minimum, basic authentication enabled and require TLS.

Ensure that each service package uses a strong authentication profile and requires TLSv1.2 or later.

A new release may deprecate service packages. However, for those with custom applications that connect to a deprecated service, the service package will continue to work and remain available.

Do not put into production services that are unauthenticated or that use only basic authentication.

For more information, see *Service Wizard: Configure Data Records*.

Define appropriate roles and privileges to restrict access

Define roles for the users in your access groups.

For each application function that only certain users need, such as classes, screens, flows, flow actions, or tools, define an appropriate privilege to enable access to that function.

Use the Access Manager to manage the granting of these privileges to roles. Grant access only to users with a real business need for a business function or business data.

Note: It is *critical* to protect access to certain types of rules (such as activities, reports, and flow actions, all of which could provide access to sensitive data, or the ability to do harm within the application) by assigning privileges to rules that are not available to unauthorized individuals. It is not recommended that you wait until application testing and deployment to do this, because it may result in wasteful refactoring and retesting of your application. However, if you wait until production testing and deployment to protect these types of sensitive rules, Pega has provided the Rule Security Mode Utility tool to help you in Pega Platform releases prior to 8.5. This tool analyzes unprotected access to rules allowed by your application and automates the process of adding appropriate privileges.

Review all access groups, especially access groups for unauthenticated users, to verify that the access groups have the minimum required access to rules, case types, and data.

For more information, see

- [Using the Rule Security Mode Utility to secure rules in Pega Applications](#)
- [Authorization](#)
- [Editing authorizations for case type items in a single access group](#)
- [Editing authorizations for case type flows and flow actions in a single access group](#)
- [Viewing authorizations for case type flows and flow actions in a single access group](#)
- [Viewing authorizations for case type items in all access groups](#)
- [Reviewing user privileges for a role by using Access Manager](#)

- [Access Manager](#)
- [Learning about access groups](#)
- [Creating an access group](#)

Define appropriate access control policies to restrict access

Use access control policies to enforce restrictions on access to application data at the row and column level; in other words, to restrict access to specific instances or properties in a class for different operators.

Define policy conditions that dynamically compare properties in each instance of the restricted class to user privileges, credentials, or other information on the clipboard.

For more information, see:

- [Authorization](#)
- [Creating an access control policy](#)
- [Creating an access control policy condition](#)

Appropriately encrypt data

Protect sensitive data within Pega Platform data stores by encrypting all the data in a class or by encrypting individual property values.

For more information, see [Encryption in Pega Platform](#).

Tasks to perform when deploying to production

Set the system production level to 5

To implement the highest restrictive security scheme, set the production level for the application to level 5.

You can change the production level in the navigation pane of Dev Studio by clicking **Configure > System > General > Systems, Nodes, Requestors**. The change takes effect the next time you start the system.

The production level value primarily affects privilege-based authorization through Access of Role Object and Access Deny rules, and your testing mirrors the authorization controls that are set for production. If the system production level setting interferes with access in your development environment, add more focused *Access of Role to Object* rules that grant access, instead of lowering the production level.

For more information, see [Defining values for an application setting](#).

Lock application and rulesets

Lock each ruleset version with a secure password by clicking **Lock and Save** on the **Versión** tab, and entering a hard-to-guess password. In each ruleset rule, click **Use checkout?** on the **Security** tab, and enter three distinct passwords to limit the ability to add versions, update versions, and update the ruleset rule.

For more information, see *Versions tab on the Ruleset form*.

You should also lock the application itself in the Application rule form. If a ruleset is intended to remain unlocked in production (for example, to store rules such as reports that end users can modify), then list it as a Production ruleset in the Advanced section of the Application rule form.

Do not deploy checked-out rules

Run the Checked Out Rule Report and eliminate rules that are checked out.

For more information, see [Reporting on rules](#).

Block unnecessary roles and operators from production

In the production environment, eliminate or block any operators and roles used in development or test environments, but are not needed in production.

For more information, see:

- [Defining security information for an operator](#)
- [Defining work routing settings for an operator](#)
- [Enabling, disabling and deleting operators](#)

In addition, you should delete or disable any Authentication Service rules that are not intended to be usable in production, as these would allow access to your application through unauthorized methods.

Secure passwords

Verify that the system securely hashes and stores all passwords for production use.

- In the database table that holds the operator ID instances, ensure that the column that contains the password property *pyPwdCurrent* is not exposed, and that the value for *pyPwdCurrent* is only in the *pzPVStream* or BLOB column.
- Convert preexisting password hashes to use the bcrypt algorithm.

For more information, see [Password hashing](#).

To prevent unauthorized access with default passwords, change the passwords for *all* default operators. Disable or delete the operator IDs that you do not plan to use.

Note: The passwords should be changed for disabled operators as well.

For more information, see:

- [Defining security information for an operator](#)
- [Enabling, disabling and deleting operators](#)

Configure dynamic system settings for production

Verify that the dynamic system settings are appropriate for a production environment.

For more information, see [Dynamic system settings](#).

Define appropriate Configure Cross-Site Request Forgery (CSRF) settings

Configure Cross-Site Request Forgery (CSRF) settings to prevent unwanted actions on an application in which a user is currently authenticated.

Set these values by using the Cross-Site Request Forgery landing page.

For more information, see [Enabling and configuring Cross-Site Request Forgery settings](#).

For Pega 7 and below, set the CSRF settings as described in [Configuring CSRF protection](#).

Define appropriate Content Security Policies (CSPs)

Review and define an appropriate Content Security Policy (CSPs). Every production application should have a policy specifying which locations the user's browser can load resources from.

For more information, see [Content security policy directives](#).

Define appropriate CORS policies for REST services

Configure cross-origin resource sharing (CORS) policies to control and secure access to the REST services in your application by external systems.

For more information, see [Creating a cross-origin resource sharing policy](#).

Configure logging levels appropriately

Set the appropriate logging levels for production. There are eight logging levels. Error is the default setting. If system is running normally, very little information will be written to the logs. Information is only written to the logs when an exception is thrown.

For more information, see [Logging Level Settings tool](#).

Define and Map Authentication Services to application

Define your application alias.

Map the authentication services used to sign in to the application. For stronger security, list all the valid authentication services for each application.

For more information, see:

- [Application URL patterns for various authentication service types](#)
- [Mapping authentication services in Dev Studio](#)
- [Mapping authentication services in App Studio](#)
- *Service Wizard: Configure Data Records*

Security Checklist additional tasks

These tasks are not part of the core Security Checklist because they do not apply to all applications. You should review these additional tasks and determine if they apply to your application.

These additional tasks may be required, depending on what Platform features your application uses, how much you customize a Pega application, the amount of sensitive data created and stored within the application, and other factors.

Additional tasks in the Security Checklist

The tasks in this checklist are not considered core tasks because:

- They involve changing configuration defaults specified in Pega Platform. These should be acceptable for most organizations and most applications, but your organization's security policies may require different configurations. Examples are session time-out limits and policies for password format and complexity.
- If you are running a Pega application, they depend on how extensively and in what ways you have customized the application. An example is controlling access to REST services you have added to the application by using strong authentication.

Configure authentication security policies

Configure the following authentication security policies for better user authentications and session management.

Authentication security policies	Benefits
Password format policies	Defend your system against brute force attacks in which a hacker tries thousands of randomly generated credentials or popular passwords from a password dictionary to gain application access.
CAPTCHA policies	Guard passwords against brute force attacks by automated processes.
Session lockout policies	Guard against brute force attacks by locking out operator IDs with too many failed login attempts.
Login attempt auditing policies	Can help identify patterns of suspicious behavior.
Multifactor authentication	Increase identity verification by requiring a second, one-time passcode that is sent to the operator from a separate device or account.
Operator access policies	Automatically disable operator IDs that are inactive after a specified period of time.

Configure authentication time-outs

Set an appropriate authentication time-out for each access group according to corporate standards. Configure this setting on the **Advanced** tab of the **Access Group** form. For custom authentication, set this time-out to be longer than the time-out in the external authentication service.

For more information, see [Configuring security settings for an access group](#).

Secure database connections

Secure your database connections.

In the Records Explorer of Dev Studio, expand the **SysAdmin** category, and then click **Database** and open the database instance.

On the **Database** tab, in the **How to connect** field, select use *JDBC Connection Pool* setting. This setting allows the Pega Platform application to access databases through a Java Naming and Directory Interface (JNDI) server. Avoid using the **Use configuration in Preferences** setting to define databases, because it displays credentials in the database as clear text.

Limit the capabilities and roles in the Pega Platform database account to restrict the ability to truncate tables, create or delete tables, or otherwise alter the schema. This limit on capabilities and roles might cause the View/Modify Database Schema tool to operate in read-only mode.

For more information, see [Creating database data instances](#).

For more information, see *Service Wizard: Configure Data Records*.

Audit changes to application data

Enable field-level auditing in *History*- tables, where appropriate, to track changes to key sensitive class properties.

For more information, see [Enabling security auditing for a data class or rule type](#).

Audit other types of user and developer actions

Configure security event logging to track user and developer actions that might be unauthorized or indicate suspicious patterns of behavior. If a security violation or breach occurs, the log can help you determine the level of exposure and risk, and identify remedial actions.

For more information, see [Selecting a security event to monitor](#).

If you are deploying on Pega Cloud, for more information, see:

- [Accessing server logs by using Kibana](#)
- [Streaming Pega logs to Splunk](#)

Test monitoring and analyzing security events and alerts

Define the process for routinely monitoring security events and alerts in production for your application. Test that process by intentionally generating events and alerts to verify that your process identifies and responds to them in a timely manner.

If your application includes custom Java or custom HTML written by your project team, there are special tasks you must perform to secure that code.

Eliminate vulnerabilities in custom code

Run the Rule Security Analyzer weekly to search through custom (non-autogenerated) code in your rules. This utility finds specific JavaScript or SQL coding patterns that might indicate a security vulnerability.

Remove vulnerabilities immediately to avoid wasting time refactoring and retesting your work.

For more information, see:

- [Implementing security guidelines for custom HTML](#).
- [Analyzing security vulnerability search results](#).

Configure rules appropriately

When creating rules:

- Limit the capabilities and roles that are available to the Pega Platform database account to reduce additional capabilities to truncate tables, create or delete tables, or otherwise alter the schema. This limit on capabilities and roles might cause the View/Modify Database Schema tool to operate in read-only mode
- Apply the correct type for all properties
- Review the unauthenticated access group to make sure that it has the minimum required access to rules.

If your application includes custom Java or custom HTML written by your project team, there are special tasks you must perform to secure that code.

Secure HTML if it exists in your application

Keep your application guardrail-compliant and do not include custom (non-autogenerated) HTML. However, if you do include custom HTML, follow Pega guidelines to minimize security vulnerabilities in your application.

For more information, see [Security guidelines for custom HTML](#).

Configure authentication to mirror production

Configure the system to mirror the production authentication scheme. If testing authentication via a browser, verify that all client updates and patches are applied. Also, clear the browser's password history and disable the browser's AutoComplete/Autofill feature.

For more information, see [Implementing security guidelines for test environments](#).

Define appropriate access control for client personal info

Use client-based access policies to define what application data is subject to data privacy regulations like GDPR and how access to that data will be handled.

Secure access to servlets

Use Servlet Management to add, modify, and disable the servlet configurations.

For more information, see [Servlet management](#).

Securely configure SAML authentication

SAML authentication services must be configured so that logins will not succeed when, during login, an unsigned assertion is received by Pega Platform from the identity provider during login.

Security Checklist when deploying on Pega Cloud

For applications deployed on Pega Cloud services, there are additional considerations you should address when completing the Security Checklist.

Secure file uploads

Pega Cloud implements virus scanning to ensure the integrity of our service, but clients are responsible for mitigating the impacts malware can have on users, applications, and data.

If documents can be uploaded into your application on Pega Cloud, we recommend you secure them as follows:

- Use a virus checker to check the files that can be uploaded. You can use an extension point in the *CallVirusCheck* activity to check attachments.
- Regularly update your virus checker to enable detection of new viruses.
- Restrict the file type by adding a when rule or a decision table to the *SetAttachmentProperties* activity to evaluate whether a document type is allowed. If a file type is not allowed (evaluated as false), you can set up a message on the step page that stops the save attachment activity from being performed.
- Verify that the *XML/AllowDocTypes* dynamic system setting is set to false.

Follow security leading practices for development and testing

The following recommendations are proposed around data used for testing:

- Select test data carefully to ensure that it is protected and controlled through your application authorization and access controls.
- Be mindful of the data you elect to import into sandbox environments (for development and testing purposes).
- Recognize that developers and testers commonly have elevated privileges in "lower" (sandbox) environments and that your users, not Pega, grant and maintain the application privileges.
- If you elect to persist sensitive data to sandbox environments, consider log file implications and check that they do not expose sensitive data.
- Ideally, create test data in a generic form with no relation to live system data. In the exception where live data is needed to perform accurate testing, the live data should be:
 - Anonymized as far as possible.
 - Carefully selected and secured for the period of testing.
 - Securely deleted after you complete your testing.
- Alternatively, you may also consider:
 - Using a production mirror sandbox, which provides an architectural replica of a your scaled production environment (rules and data) and which can be used for production staging and testing, scale benchmark testing, and load performance testing.
 - Performing a Pega product file export/import operation to build a production-like equivalent of your current applications. This will not transfer any of your data but will transfer all rules and schema – and will effectively duplicate your service without including production data.
- Ensure that your developers are informed of and adhere to your organization's internal security practices pertaining to protecting or masking sensitive data used within your Pega application.

If you are not deploying on Pega Cloud, see [Security Checklist when deploying in on-premises environments](#).

Security Checklist when deploying in on-premises environments

When you are deploying on-premises, there are additional considerations you should address when completing the Security Checklist.

Pega advises that on-premises clients practice good network hygiene and block access to all unnecessary network ports. Work with your network security team to ensure environments are configured appropriately.

During production testing, configure your application and the test environment to mirror the intended production environment. Otherwise, your testing might not uncover serious security vulnerabilities.

Secure file uploads

If documents can be uploaded into your application, we recommend you secure them as follows:

- Use a virus checker to check the files that can be uploaded. You can use an extension point in the *CallVirusCheck* activity to check attachments.
- Regularly update your virus checker to enable detection of new viruses.
- Restrict the file type by adding a when rule or a decision table to the *SetAttachmentProperties* activity to evaluate whether a document type is allowed. If a file type is not allowed (evaluated as false), you can set up a message on the step page that stops the save attachment activity from being performed.
- Verify that the *XML/AllowDocTypes* dynamic system setting is set to false.

For more information, see:

- [Extension points and supporting rules for attachments](#)
- [Standard activities — Extension points](#)
- [Restricting user actions for case attachments](#)
- [Steps tab on the Activity form](#)

Apply patches, updates, and hotfixes

Install the latest patches and updates to the operating system, application and web servers, proxies, database, and related applications.

For Pega Platform 8.x releases, you should install the latest patch release. For earlier Pega Platform releases you should be running the latest version, and planning to upgrade to Version 8.x in the near future.

Regardless of what release you are running, you should frequently check for any recommended security updates, which are posted at <https://collaborate.pega.com/discussion/essential-hotfixes>.

Secure *web.xml*

If you are not deploying your application to Pega Cloud, make the following changes to the *web.xml* deployment descriptor file:

- Limit or block access to the Pega Platform servlets that support only testing and debugging, including *HeapDisplay*, *SecManServlet*, and *PRSOAPSimulator*.
- Remove unnecessary resources and servlets.

- Set appropriate time-outs at the application server level and requestor level.
- Block access to the *prweb/PRServlet* servlet, which allows users to log in using the older platform login process instead of the newer PRAuth-based authentication services.

For more information, see [Application URL patterns for various authentication service types](#).

If you are deploying on Pega Cloud, see [Security Checklist when deploying on Pega Cloud](#).

Configure *PRconfig.xml* settings for production

Verify that *prconfig.xml* settings are appropriate for a production environment.

Configure the database and communications to mirror production

Configure the system and database according to your company's security policies and to be the same as in production environment to which the application will be deployed. This configuration should include the use of TLS for all communication between clients and the application.

If you use TLS, remove any cipher suites that have null ciphers. This action prevents the login credentials and password from being sent in clear text format between the client and server even over a TLS connection (if a server and client discover only a null cipher suite in common).

Configure the application server to mirror production

Configure the application server in your test environment to mirror the configuration in your production environment.

For more information, see [Implementing security guidelines for test environments](#).

Assessing your application using the Security Checklist

Use the Security Checklist to prepare your application for deployment. By completing the tasks on this checklist, you can safeguard sensitive data and improve the security of your application.

Before you begin: To mark a task on the Security Checklist as complete, you must have the *pzAdvancedSecurityUser* privilege, which is included in the *PegaRULES:SecurityAdministrator* role. A new Security Checklist is generated for each new version of your application.

1. Review the current status of your application.
 - a. In the header of Dev Studio, click **Configure Application Quality Guardrails Compliance Score**.
 - b. In the **Security status** section, review the number of tasks remaining to be performed in the Security Checklist.
2. To see more information about incomplete tasks, open the **Application Security Checklist**.
 - a. In the **Security status** section of the Dev Studio homepage, click **Application Guides**.
 - b. Click **Application Security Checklist**.
 - c. Follow the instructions for the task, and select the **Check** field to mark the task as complete.
3. Follow the instructions for the task, and then select the **Check** field to mark the task as complete.
4. To keep the **Last Updated by** section current, clear the **Check** field for the tasks that need to be completed periodically before you perform the task again. For example, review the Guardrail landing page or run the Rule Security analyzer.
5. Review and implement the instructions in [Security Checklist core tasks](#).

- [Improving your compliance score](#)
- [Adding the Security Checklist to an application created before 7.3.1](#)

Adding the Security Checklist to an application created before 7.3.1

The Security Checklist is automatically added to applications starting in Pega Platform 7.3.1. You can manually add the Security Checklist to applications that were created in earlier versions. By completing the tasks on the checklist, you can improve the security of your application.

1. In the header of Dev Studio, click the name of the application, and then click **Definition**.
2. Click the **Documentation** tab.
3. In the **Application guides** section, click **Add guide**.
4. In the **Application guide** field, enter *pxApplicationSecurityChecklist*.
5. Click **Save**.

- [Assessing your application using the Security Checklist](#)

Implementing security guidelines for test environments

As a best practice, configure the application server in your test environment so it mirrors a production environment configuration.

Use the following guidelines to minimize security vulnerabilities on the server side in your application:

- Prevent the application server from serving files to unauthorized users.
- Disable application-server directory traversals. For example, eliminate the ability to insert `../` or `..\` into directory paths.
- Disable directory listings on the application server.
- Verify that no extraneous ports are open on the application server or on the firewall that protects the application server.
- Disable HTTP methods that your application does not use, including HEAD, TRACE, and TRACK. By default, Pega Platform uses POST and GET.
- Remove the web server banner from the **Server** field in the HTTP response header. This ensures that you do not share the type and version of your application server with users.
- Disable sample applications, their supporting files, and permissions when they are no longer used. This prevents users from logging in to your system with sample application credentials.

- [Assessing your application using the Security Checklist](#)

Authentication

Apply authentication methods to ensure that only users and systems with a verified identity can access your applications, web pages, APIs, and data. Authentication includes verifying user credentials, Pega Platform requests to external services, and external service requests to Pega Platform. You can also authenticate by using an external identity provider.

User credentials

In a browser, a user logs in to a Pega Platform application, or a developer logs in to Dev Studio to make changes to an application. Authentication services verify these user credentials.

For more information, see [Authentication services](#).

The following table lists the protocols for user logins that Pega Platform supports.

Authentication type	Protocol
SAML 2.0	An external identity provider that supports the SAML 2.0 protocol, such as Microsoft Active Directory. For more information, see Web single sign-on (SSO) with SAML 2.0 .
OpenID Connect	An external identity provider that supports the OpenID Connect (OIDC) protocol.
Basic Credentials	A user ID and password that are stored in the Pega Platform database or in another internal or external data source.

Authentication Type	
Token Credentials	A token that is validated by an external identity provider or by the OAuth 2.0 authorization layer in Pega Platform (often used in offline mobile applications).
Anonymous	Supports activity by guest users, who are prompted to authenticate themselves partway through a session. For example, an unauthenticated user can add items to a shopping cart, and enter credentials when they check out.
Custom	Uses none of the above meets your requirements or meets your use case, you can write your own logic to challenge users for credentials and to validate the credentials. For example, using a Lightweight Directory Access Protocol (LDAP)-compliant directory server.
Kerberos	A computer-network authentication protocol that is based on tickets that can be securely presented by a client or a service on the client's behalf to a server for access to services.

You can configure a custom authentication service to use information that is stored within the identity provider to determine the user roles and privileges in Pega Platform.

Make your application more secure by using simple selections in the authentication service rule form to implement policies such as multi-factor authentication. For example, each time a user logs in, the application can send an authentication code to the user by email. To log in, the user enters that code in addition to a password.

You can use authentication services (including SAML 2.0, OpenID Connect, or token credentials) to implement single sign-on (SSO) solutions. SSO solutions limit repetitive requests for credentials when users access a variety of systems or applications.

For complete control over the login process, you can define custom authentication services.

Pega Platform connectors Requests to external services from Pega Platform connectors

To invoke an external REST service to get information from an external system or application, a Pega Platform application must authenticate to that service. This type of authentication uses an authentication profile and OAuth provider data instances. The supported forms of authentication include basic credentials, NT LAN Manager credentials (NTLM), and OAuth 2.0.

For more information, see:

- [Authentication profiles](#)
- [Setting up an OAuth 2.0 provider](#)

External requests for execution to Pega Platform services

An external system or application can invoke a REST service that is defined in Pega Platform or within a Pega Platform application to get case information. This type of authentication uses a service type and service package instances. Supported forms of authentication include basic credentials, OAuth 2.0, and custom authentication.

For more information, see:

- [Integration Services category](#)
- [Service Packages](#)

Session management

After the initial authentication, the session management features in Pega Platform ensure that requests for access to the system and its data only come from authenticated requestors.

In Pega Platform, you can define various policies to control session time-outs, automatically terminate sessions, deactivate operators after successive days of inactivity, run cross-origin resource sharing (CORS), and detect cross-site request forgery (CSRF).

Authentication process flow

In Pega Platform, during a user-interactive login, the authentication service rules perform the following functions, in order:**Note:** For token-based authentication types, step 3 is skipped. For anonymous authentication types steps 3–7 are skipped.

1. Pega Platform determines if this is the initial user request.
2. Pega Platform initializes an unauthenticated session.
 1. The application context for the session in this state is provided by the access group specified by the browser requestor type:

The access group specified by the browser requestor type

Edit Requestor Type: BROWSER Starting connection
ID: pega • BROWSER RS: No associated ruleset [Edit]

Definition Activities History

Define this requestor type using
an Application-based Access Group ▼

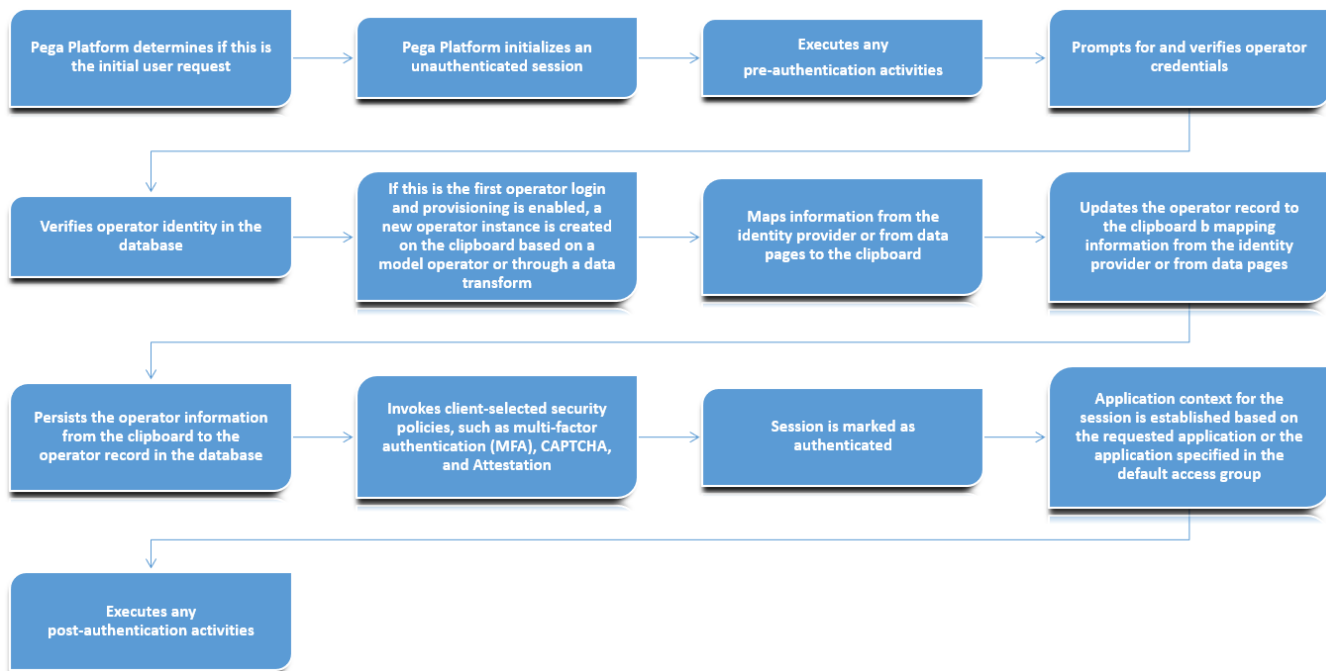
Access Group Name

☒ PRPC:Unauthenticated  

+

3. Executes the pre-authentication activity, if any.
4. Prompts for and verifies operator credentials.
 1. If a Basic authentication service, operator credentials are always prompted and verified.
 2. If a SSO authentication service, the external identity provider may prompt for credentials if needed.
 3. If a Custom authentication service, the behavior depends on authentication activity logic, with the typical response being a login page.
5. Verifies operator identity in the database. If provisioning of new operators is disabled and the operator does not exist, authentication fails.
6. If this is the operator's first login and provisioning is enabled, creates a new operator instance based on a model operator or through a data transform.
7. Maps information from the identity provider or from data pages to the clipboard.
8. Persists the operator information from the clipboard to the operator record in the database.
9. Invokes client-selected security policies, such as multi-factor authentication (MFA), CAPTCHA, and Attestation.
10. Session is marked as authenticated.
11. Application context for the session is established based on the requested application or the application specified in the default access group.
12. Executes the post-authentication activity, if any.**Note:** The session established in step 1 has a short time-out period, which is set to two minutes. If the session sits idle in excess of this time period at any point during login, then the session will be destroyed. This is the only feature within Pega Platform that attempts to mitigate DOS attacks.

A simplified overview of the Authorization process flow



- [Creating an authentication service](#)

To override or extend the default authentication process, create an authentication service. By creating an authentication service, you implement more specialized authentication requirements than the default, for example, to use pre-authentication and post-authentication activities.

- [Authenticating requests in services](#)

You can configure Pega Platform to access external systems to retrieve data and perform application processing. Similarly, you can allow external systems to access services in Pega Platform. By communicating with external systems, you can make use of functionality that has already been configured, and avoid the need to duplicate the same functions in multiple applications.

- [Using JNDI to specify an LDAP server when using an authentication service](#)

You can set up an authentication service to override or extend the default Pega Platform™ authentication process. You can enter a Java Naming and Directory Interface (JNDI) entry, which represents a directory located on the LDAP server. Using JNDI enables you to relocate servers without having to reconfigure the application.

- [Authentication time-out](#)

When users are inactive for a certain period of time, Pega Platform requires users to reauthenticate by entering their login credentials. The browser session cannot resume until the login and password are accepted. Requiring reauthentication helps prevent a malicious or unauthorized user from hijacking the browser session. However, if reauthentication fails or is canceled, some or all of the data on the screen might continue to be displayed.

- [Authentication login failures](#)

When a user fails to authenticate with proper credentials, safeguards ensure that repeated failed attempts to authenticate have repercussions to mitigate automated attempts to gain unauthorized access to the system.

- [Security tab of the Application Definition](#)

Use this tab to define security settings in your application, map authentication services, and enable content, mashups, and digital messaging security.

- [Web Service Security profile](#)

Use a Web Service Security profile (WS-Security) to determine how to authenticate a web service message.

- [Authorization](#)
- [Auditing](#)
- [Security operations](#)
- [Security](#)

Authentication services

To override or extend the default authentication process, create, and configure an authentication service.

You can configure your application to authenticate users by using single sign-on (SSO) and external identity providers. Create an authentication service to configure Pega Platform with one of the following authentication methods:

Authentication types and protocols available in Pega Platform

Authentication type	Protocol
SAML 2.0	An external identity provider that supports the SAML 2.0 protocol, such as Microsoft Active Directory. For more information, see Web single sign-on (SSO) with SAML 2.0 .
OpenID Connect	An external identity provider that supports the OpenID Connect (OIDC) protocol.
Basic Credentials	A user ID and password that are stored in the Pega Platform database or in another internal or external data source.
Token Credentials	A token that is validated by an external identity provider or by the OAuth 2.0 authorization layer in Pega Platform (often used in offline mobile applications).
Anonymous	Supports activity by guest users, who are prompted to authenticate themselves partway through a session. For example, an unauthenticated user can add items to a shopping cart, and enter

Authentication type	credentials when they check out. Protocol
Custom	Uses none of the above meets your requirements or meets your use case, you can write your own logic to challenge users for credentials and to validate the credentials. For example, using a Lightweight Directory Access Protocol (LDAP)-compliant directory server.
Kerberos	A computer-network authentication protocol that is based on tickets that can be securely presented by a client or a service on the client's behalf to a server for access to services.

Default configuration

By default, your system includes a basic authentication service named Platform Authentication. You can save this service with a new name and change it, and you can create any type of authentication service, including the basic type.

Note: As a best practice, ensure that authentication services do not share the same alias. Rename any duplicate authentication service aliases before upgrading to Pega 8.6 or later.

The default servlet, *PRAuth*, provides a unified authentication gateway so that you do not need to edit *prweb.xml* or restart the server for new authentication services.

For more information on URL patterns and servlet names, see [Application URL patterns for various authentication service types](#).

Multi-tenancy

When using multi-tenancy, shared users will not be present, so you must complete the following actions:

- In the navigation pane of Dev Studio, click **Records SysAdmin Authentication Service**.
- Select the Authentication service that you want to add multi-tenancy.
 - Select the **Authentication service** record, of the selected Authentication service.
 - Under the **Operator identification** tab, change the **Model operator** to the tenant username.
- Create an operator identical to the tenant username in the model operator.
 - For more information, see [Creating an operator ID](#).
- [Security policies](#)
- [More about authentication services](#)

Mapping authentication services in Dev Studio

As an administrator, configure authentication services in an application in Dev Studio, to streamline functionality and provide an effective way to manage the authentication services within an application.

This functionality is available in both App Studio and Dev Studio. For more information, see [Mapping authentication services in App Studio](#).

There are two ways to map authentication services in Dev Studio: map an existing authentication service, or create a new authentication service. Both options are explained here.

Note: Any changes that you make to the authentication services in Dev Studio will display in App Studio, and any changes that you make to the authentication services in App Studio will display in Dev Studio. However, you must first save your changes and then refresh both studios to see the changes.

Only administrators with the *pzAdvancedSecurityUser* privilege can map authentication services to an application. By default, the *PegaRULES:SecurityAdministrator* role includes the *pzAdvancedSecurityUser* privilege.

You can map an existing authentication service directly from the application definition by following the steps below.

- In the header of Dev Studio, click the name of the application, and then click **Definition**.
- Click the **Security** tab.
- Click **+ Add authentication service**.
- In the **Name** column, enter the name of the authentication service that you want to map.
- Optional:** To update the authentication service, click the **Configure** icon to the right of the new authentication service.
- Click **Save**.

You can create a new authentication service directly from the application definition by following the steps below.

- In the header of Dev Studio, click the name of the application, and then click **Definition**.
- Click the **Security** tab.
- Click **+ Add authentication service**.
- Click the **Configure** icon to the right of the **Authentication service name** field.
- In the **Authentication type** list, choose an authentication type.
- In the **Name** field, enter a name for your authentication service.
 - The **Short description** field should auto-populate. For example, if you name your authentication service *testservice*, the short description will show: *testservice/prweb/PRAuth*
- Click **Create and open**.

Result: Depending on the type of authentication service that you create, the configuration form will automatically open. **What to do next:** Now, you will need to configure your authentication service. For more information, see:

- [Configuring login authentication with basic credentials](#).
- [Configuring SSO login authentication with a SAML identity provider](#).
- [Configuring the identity provider for an OpenID Connect SSO authentication service](#).
- [Configuring login authentication using LDAP and Active Directory](#).

Creating an authentication service

To override or extend the default authentication process, create an authentication service. By creating an authentication service, you implement more specialized authentication requirements than the default, for example, to use pre-authentication and post-authentication activities.

Before you begin: To create an authentication service, you must have the *pzCanCreateAuthService* privilege, which is included in the *PegaRULES:SecurityAdministrator* role. By default, your system includes a basic authentication service named Platform Authentication. You can save this service with a new name and change it, and you can create any type of authentication service, including the basic type of authentication service. **Note:** As a best practice, ensure that authentication services do not share the same alias. Rename any duplicate authentication service aliases before upgrading to Pega 8.6 or later.

- In the header of Dev Studio, click **Configure Org & Security Authentication Create Authentication Service**.
- In the **Authentication Type** list, click the authentication service type.
 - Basic credentials** – Authentication using a user ID and password, which can be stored in the Pega Platform database or an external source that is accessed by using a data page
 - SAML 2.0** – SAML 2.0 web SSO-based authentication
 - Custom** – LDAP authentication or custom authentication protocol
 - Kerberos** – Kerberos user credentials
 - OpenID Connect** – OpenID Connect SSO-based authentication
 - Anonymous** – Unauthenticated access that uses a model operator
 - Token credentials** – Useful for offline mobile applications
- Enter a name and short description.
- Click **Create and open**.

5. Configure your authentication service.

- [Configuring login authentication with basic credentials](#)

After you create a basic authentication service, configure it so that Pega Platform uses the specified security policies for authenticating users. You can also configure optional features such as preauthentication and postauthentication activities.

- [Configuring SSO login authentication with a SAML identity provider](#)

After you create a SAML SSO authentication service, configure it so that Pega Platform uses the specified identity provider for authenticating users. You can map attributes from the identity repository to properties in Pega Platform, and also configure optional features such as preauthentication and postauthentication activities and operator provisioning.

- [Configuring SSO login authentication with an OpenID Connect identity provider](#)

After you create an OpenID Connect SSO authentication service, configure it so that Pega Platform uses the specified identity provider for authenticating users. You can map claims from the OpenID Connect provider to properties in Pega Platform, and configure optional features such as preauthentication and postauthentication activities and operator provisioning.

- [Configuring login authentication for an anonymous operator](#)

After you create an anonymous authentication service, configure it so that Pega Platform can support guest users. You can map attributes from the model operator to properties in Pega Platform, and also configure preauthentication and postauthentication activities.

- [Configuring custom or Kerberos login authentication](#)

After you create a custom or Kerberos authentication service, configure it so that Pega Platform can connect to the repository and find the operator credentials. You can map attributes from the repository to properties in Pega Platform, and can also configure optional features such as authentication and time-out activities.

- [Testing login authentication services](#)

You can test and debug an authentication service in a development or staging environment by setting the appropriate log level.

- [Configuring login policies such as multi-factor authentication, CAPTCHA, and attestation](#)

You can make user authentication more secure by defining login policies for password requirements, multi-factor authentication, lockout policies, and other similar restrictions.

- [Attestation](#)

Depending on the security requirements for your application, you may need to use attestation to do business.

- [Configuring a token credentials authentication service](#)

After you create a token credentials authentication service, configure it so that Pega Platform uses the specified token provider for authenticating users. Select this type of service for offline mobile applications. You can map claims from the token to properties in Pega Platform, and configure optional features such as preauthentication and postauthentication activities.

- [Authentication services](#)
- [More about authentication services](#)
- [Security policies](#)
- [Configuring an application to use an anonymous authentication service](#)

Configuring login authentication with basic credentials

After you create a basic authentication service, configure it so that Pega Platform uses the specified security policies for authenticating users. You can also configure optional features such as preauthentication and postauthentication activities.

Before you begin: You must complete the following task before you can configure a basic authentication service: [Creating an authentication service](#). The default means of authentication for Pega Platform is a basic authentication service that is named *Platform Authentication*. All basic authentication services include support for mobile OAuth 2.0 authentication with proof key for code exchange (PKCE).

1. Create a basic authentication service, or open an existing service from the navigation panel in Dev Studio by clicking **Records > SysAdmin > Authentication Service** and selecting a basic credentials authentication service from the instance list.
2. In the **Authentication service alias** field, specify an alias to represent a unique value for this service. This value becomes the final part of the URL path for users to access Pega Platform.
Login URL is a read-only field that displays the URL that accesses Pega Platform and uses this service for user authentication.
3. **Optional:** In the **Provider logo** field, specify an image to display on the login screen that identifies this provider.
4. **Optional:** To authenticate new sessions against an external data source instead of the Pega Platform database, select the **Verify credentials using external identity store** check box and enter a name for **Data page for credentials verification**. For example, to verify the identities of external customers, follow these steps:
 - a. Create a requestor-scope read-only data page, with object type equal to *Data-Admin-Operator-ID*. Save the data page to the unauthenticated ruleset.
 - b. Create a data transform with an applies to class equal to *Data-Admin-Operator-ID* and having input parameters for user name and password. Validate the user name and password against the external data source. In the data transform, when the input parameters are valid, set *.pyApproveStatus* to true. Save the data transform to the unauthenticated ruleset.
 - c. On the data page, set the data source equal to the data transform that you just created.
 - d. On the authentication service, set **Data page for credentials verification** equal to the name of the data page you just created.
 - e. At run time, if the operator authenticates against a data page and the operator does not exist in the Pega Platform database, the operator must be provisioned (or added to the Pega Platform database). For information about operator provisioning, see [Configuring operator provisioning for a basic authentication service](#).
5. In the **Map Operator Id** field, provide an expression for deriving the operator ID from the user name that is entered at the time of authentication. To use the Expression Builder, click the **Build an expression** icon.
For example, a user could log in with an email address such as *User123@Something.YourCo.com*, but the operator ID is *User123*. Use the Expression Builder to use all of the characters before the "@" sign.
6. **Optional:** Configure the optional parameters of the service.
 - [Map operator information](#).
 - [Specify preauthentication and postauthentication activities](#).
 - [Require reauthentication for new and expired sessions](#).
 - [Configure operator provisioning](#).
 - [Enforce policies from the Security Policies landing page](#).
7. [Activate your basic authentication service](#).

- [Customizing authentication screens in your Pega application](#)

To customize the login screen of your authentication service, edit the source HTML in the *pyDisplayAuthScreenSelector*.

- [Authentication services](#)
- [Authentication services and rule availability](#)

Mapping operator information for a basic authentication service

You can map attributes to selected properties and pages that are used by your preauthentication and postauthentication activities or by other Pega Platform features such as access control policies. Mapping complements the use of preauthentication and postauthentication activities, operator provisioning by data transform, and other actions.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
2. Click the **Mapping** tab.
3. Map the attributes from your login process to properties and pages that are used by your preauthentication and postauthentication activities or elsewhere in Pega Platform.
 - a. You can use the following properties and pages in the **Map from** field. You can also reference custom properties and pages that are used in a login flow, and you can

use the Expression Builder.

Page name	Description
<i>pxRequestor</i>	The requestor page

- b. You can use the following properties and pages in the **Map to** field. You can also reference custom properties and pages that are used in a login flow.

Page name	Description
<i>OperatorID</i>	Properties of the operator ID
<i>D_pyOperatorAttributes</i>	Requestor-scoped data page for caching operator attributes
<i>D_pyOperatorDeviceInformation</i>	Requestor-scoped data page for caching operator device information

4. Click **Save**.

What to do next: [Specifying preauthentication and postauthentication activities for a basic authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring login authentication with basic credentials](#)

Specifying preauthentication and postauthentication activities for a basic authentication service

To customize the login process, you can write activities that are triggered before and after basic authentication. For example, a postauthentication activity can update the operator record or refuse an automatically provisioned user access to an application.

1. Create your preauthentication and postauthentication activities. For more information, see the sample activities *pySSOPreAuthenticationActivity* and *pySSOPostAuthenticationActivity*. When you write these activities, consider the following:
 - Update the *pega Browser* requestor type's default access group to equal an access group that includes the ruleset of the preauthentication activity. The preauthentication activity must be accessible to the *pega Browser* requestor type, which is used for the unauthenticated user session. By default, this requestor type is assigned to the *PRPC:Unauthenticated* access group.
 - Create the postauthentication activity in a ruleset that is accessible to the user's default access group. The postauthentication activity must be accessible to the user who has just been authenticated.
 - Set the Applies To key part of the activities to *Code-Security*.
 - Set the authentication result in *pyAuthenticationPolicyResult* to *true* to proceed with authentication, or set to *false* to terminate the request; for example:

```
tools.getRequestor().getRequestorPage().putString("pyAuthenticationPolicyResult", "true");
```

- You can dynamically update the user's role list by using a postauthentication activity. This is useful if you want to dynamically add roles to the user context at login without maintaining multiple access groups in the database for all combinations of roles. In the postauthentication activity, update the list of roles by calling *setRoles*, as shown in the following example.

```
tools.getAuthorizationHandle().setRoles(tools, tools.getAuthorizationHandle().UPDATE_APPEND, rolesList);
```

Caution: Roles that you set in the postauthentication activity without saving the access group to the database lose their effect if the user switches applications. To dynamically add roles in a way that supports switching applications, see [How to dynamically add roles during user authentication](#).

2. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
3. On the **Basic credentials** tab, expand the **Advanced configuration settings** section.
4. In the **Pre-authentication activity** field, enter the name of the preauthentication activity.
5. In the **Post-authentication activity** field, enter the name of the postauthentication activity.
6. Click **Save**.

What to do next: [Requiring reauthentication for new and expired sessions for a basic authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring login authentication with basic credentials](#)
- [Creating an activity](#)

Requiring reauthentication for new and expired sessions for a basic authentication service

To add an extra layer of security, configure Pega Platform to reauthenticate new sessions and to reauthenticate when reactivating an expired session. The time-out period is specified in the operator's access group or is managed by the application server or another external facility.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
2. On the **Basic credentials** tab, expand the **Advanced configuration settings** section.
3. Select the **Use access group timeout** checkbox to use the authentication time-out values that are specified in the user access group. This setting determines how long user sessions can remain inactive before users are prompted to identify themselves again. Clear the **Use access group timeout** checkbox if the time-out is managed by the application server or another external facility.
4. Click **Save**.

What to do next: [Configuring operator provisioning for a basic authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring login authentication with basic credentials](#)

Configuring operator provisioning for a basic authentication service

When an operator who is authenticated by using a data page and who is not defined in the Pega Platform database logs in for the first time, Pega Platform can automatically create the operator record. To do this, you configure operator provisioning for the authentication service. This configuration eliminates the manual step of adding the operator record.

Note: If the operator record is already defined in Pega Platform, you can update it by creating a postauthentication activity.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list, and then click the **Basic credentials** tab.
2. In the **Operator identification** section, select **Enable operator provisioning using model operator**.
3. To construct an operator by copying a specific model, click **By name** and enter a value for **Model operator**. This is the operator ID for the operator record to be copied. If the value contains a period ("."), enclose the value in double quotation marks, for example, "abc.def".
4. To identify the model operator by using an expression, click **By name**, and next to the **Model operator** field, click the **Build an expression** icon and create an expression that returns the operator ID to be copied.
5. To identify the model operator by using organizational information, click **By organization hierarchy** and enter values for **Org** (organization), **Div** (division), and **Unit**. You can also provide an expression for each of these fields. The Model User value in the Unit instance provides the model operator for constructing the new operator.
6. To create the operator by using a data transform, click **By data transform** and enter the data transform name. The applies to class of the data transform must be *Data-Admin-Operator-ID*. For an example data transform, see *pyDefaultForNonPegaOperator*. If the data transform copies property values from a model operator, you must first create an operator instance in the database for the model operator. Otherwise, you do not need to create a model operator.
7. At a minimum, you must provide values for the properties listed below before the postauthentication activity (if any) is run. You can populate them from the model operator or the data transform, or you can explicitly map to them by using the **Mapping** tab. Do not change them in the postauthentication activity.

Property name	Description
<i>OperatorID.pyAccessGroup</i>	Operator's default access group

<i>OperatorID.pyAccessGroupsAdditional</i>	List of all of the operator's access groups, including the default access group
<i>OperatorID.pyOrganization</i>	Organization
<i>OperatorID.pyOrgDivision</i>	Division
<i>OperatorID.pyOrgUnit</i>	Unit

Note: The value for the newly created operator ID (*OperatorID.pyUserIdentifier*) is the value from the main tab in the **Map operator id from** section.

8. Click **Save**.

What to do next: [Enforcing policies from the Security Policies landing page](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring login authentication with basic credentials](#)
- [Organization Unit data instances](#)
- [Expression Builder](#)

Enforcing policies from the Security Policies landing page

To enforce security policies that are defined on the Security Policies landing page, configure your basic authentication service.

1. Open the service from the navigation pane in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list, and then click the **Security policies** tab.
2. Click the **Add item** icon to specify a security policy.
3. In the **Policy type** list, select the policy to enforce.

Note:

- If you select **Multi-factor authentication**, you must map properties such as email address or phone number from the email or SMS receiver account by using the **Mapping** tab.
- If you select **User consent**, the section *pyPostAuthConsentScreen* is displayed to the user after authentication. You can customize this section.
- If you select **CAPTCHA**, the settings on the **Security Policies** landing page take effect as described below.
 - If you set the value of **Enable presentation of CAPTCHA upon initial login** to **Enabled**, the CAPTCHA is displayed for every login.
 - If you set the value of **Enable presentation of CAPTCHA upon initial login** to **Disabled**, the CAPTCHA is displayed according to the probability that you set in the **Probability that CAPTCHA will be presented upon authentication failure (%)** field.

4. Click **Save**.

What to do next: [Activating your basic authentication service](#)

- [Authentication services](#)
- [Authentication services and security policies](#)
- [Configuring login authentication with basic credentials](#)
- [Security policies](#)

Customizing authentication screens in your Pega application

To customize the login screen of your authentication service, edit the source HTML in the *pyDisplayAuthScreenSelector*.

Before you begin: Editing the HTML of the login screen requires configuring an HTML rule. For more information about HTML rules for the user interface, see [About HTML rules](#).

You must complete the following task before you can configure a basic authentication service: [Creating an authentication service](#).

Before you customize pages, you must create an alternate access group and ruleset for the custom authentication screen, then apply the access group and ruleset to the *pyDisplayAuthScreenSelector* rule. For more information, see the following topics:

- [Creating a ruleset and a ruleset version](#).
- [Creating an access group](#)

To edit the source HTML of your authentication service login page:

1. Open the existing authentication HTML rule from the navigation panel in Dev Studio by clicking **Records Technical HTML**.
2. In the **Stream name** column, click the search icon (▼).
3. In the Search Text field, enter *pyDisplayAuthScreenSelector*.
4. Click **Apply**.
5. Click the row containing *pyDisplayAuthScreenSelector*.
6. In the Generate For list, select whether your customized login screen uses **HTML** directives or JavaServer Pages (**JSP**) tags.
7. In the **Browser Support** list, select whether **Modern browsers** or **IE only** support your customized login screen.
8. In the **Accessibility** list, select to tag whether or not your customized login screens meets your accessibility requirements.
9. In the **HTML source** field, enter your customized login page HTML. Observe Pega Guardrails and analyze potential security risks of the customized login page.
10. If your customized login references clipboard pages, click the **Pages and Classes** tab.
11. Enter the required clipboard pages and the classes to which they belong.
12. Click **Save**.

The customized HTML of your login page is built into your application instance of Pega-Engine.

13. In the **Actions** list, select **Preview**.
14. Confirm the login page reflects your customized HTML.

- [Customizing the login screen](#)

Activating your basic authentication service

After you create and configure a basic authentication service, enable the service for Pega Platform.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
2. On the **Basic credentials** tab, select the **Enable this authentication service** check box.
3. Click **Save**.

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring login authentication with basic credentials](#)

Configuring SSO login authentication with a SAML identity provider

After you create a SAML SSO authentication service, configure it so that Pega Platform uses the specified identity provider for authenticating users. You can map attributes from the identity repository to properties in Pega Platform, and also configure optional features such as preauthentication and postauthentication activities and operator provisioning.

Before you begin: You must complete the following task before you can configure a SAML SSO authentication service: [Creating an authentication service](#). **Note:** If you access Pega Platform using a reverse proxy server, you must ensure that the reverse proxy is configured before configuring a SAML SSO authentication service. For more information, see [Deployment behind a reverse proxy](#).

1. Create a SAML SSO authentication service, or open an existing service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and selecting a SAML SSO authentication service from the instance list.
2. In the **Authentication service alias** field, specify an alias to represent a unique value for this service. This value becomes the final part of the URL path for users to access Pega Platform.

Login URL is a read-only field that displays the URL that accesses Pega Platform and uses this service for user authentication.

3. **Optional:** In the **Provider logo** field, specify an image to display on the login screen that identifies this provider.
4. [Configure the identity provider.](#)
5. [Map the operator ID.](#)
6. **Optional:** Configure the optional parameters of the service.
 - [Specify preauthentication and postauthentication activities.](#)
 - [Require reauthentication for new and expired sessions.](#)
 - [Configure operator provisioning.](#)
 - [Override the service provider settings.](#)
 - [Enforce policies from the Security Policies landing page.](#)
7. [Activate your SAML SSO authentication service.](#)

- [Updating an expired identity provider certificate in a SAML Authentication Service](#)

If you are using a SAML Authentication service and your Identity Provider (IdP) certificate has expired, you need to reimport the Identity Provider metadata. This is the easiest way to update an expired IdP certificate.

- [Updating an expired Service Provider certificate in a SAML Authentication Service](#)

If you are using a SAML Authentication service and your Service provider certificate has expired, then you need to create a new keystore, select that keystore under Service Provider details, and re-export the service provider metadata to the identity provider.

- [Authentication services](#)
- [More about authentication services](#)

Configuring the identity provider for a SAML SSO authentication service

To enable the system to verify the identity of requestors, configure the identity provider (IdP) for your SAML SSO authentication service. You configure the identity provider by importing values from a file or by entering them manually.

Note: For the SAML ruleform, Global Resource Settings are supported. For more information, see [Fields that support the Global Resource Settings syntax](#).

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list. On the **SAML 2.0** tab, navigate to the **Identity Provider (IdP) information** section.
 2. If you are configuring the identity provider by importing the configuration, complete these steps:
 - a. Click **Import IdP metadata**.
 - b. Select the source of the metadata (**via URL** or **via file**), and then enter the URL or file path.
 - c. Click **Submit**.
 3. If you are configuring the identity provider by entering values manually, complete these steps:
 - a. In the **Entity Identification (Issuer)** field, enter the entity ID for the identity provider.
 - b. In the **Login (SSO) protocol binding** list, select the standard communication protocol that is supported for the response message:
 - **HTTP POST** – SAML protocol messages are transmitted in an HTML form with base64-encoded content.
 - **HTTP Artifact** – SAML protocol messages are transmitted using a unique identifier called an artifact. Select this protocol if you do not want to expose the content of the SAML message during connection.
 - **HTTP Redirect** – SAML protocol messages are transmitted within URL parameters.
 - c. In the **Login location** field, enter a login service.
 - d. In the **Logout (SLO) protocol binding** list, select the standard communication protocol that is supported for the response message: **HTTP Redirect** or **SOAP**.
 - e. **Optional:** In the **Logout location** field, enter a logout service.
 - f. If your **Login (SSO) protocol binding** is **HTTP Artifact**, then in the **Artifact Resolution Service (ARS) location** field, enter the URL that is used by the service provider to send the artifact resolve request to the Identity Provider. Otherwise, leave the field blank.
 - g. In the **Verification certificate** section, click the Pencil icon to enter the certificate alias.
 - h. In the **CERTIFICATE STORE** field, press the Down Arrow key and select the keystore that contains the IdP public key that is used for verifying the signature of the SAML assertion.
- Note:**
- If you imported IdP metadata, or if the **CERTIFICATE STORE** field is blank, the system creates a keystore instance and adds the IdP certificate to it. The system sets the alias of the entry in the keystore to the certificate's issuer name and sets the keystore password to *rules*.
 - If the **CERTIFICATE STORE** field is not blank and points to a valid keystore instance when you import the IdP metadata, the system adds the IdP certificate to the existing keystore instance and sets the alias of the entry to the certificate's issuer name.
- i. Click **Submit**.
 - j. **Optional:** In the **TLS/SSL truststore** field, press the Down Arrow key and select the truststore record that contains the server certificate for a TLS or SSL handshake.

Note: Select a truststore record if you have provided a secure Logout location or a secure Artifact Resolution Service (ARS) location, and you have not added the TLS or SSL certificate to the truststore of the Java virtual machine.
4. Click **Save**.

What to do next: [Mapping operator information for a SAML SSO authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring SSO login authentication with a SAML identity provider](#)

Mapping operator information for a SAML SSO authentication service

To enable the login process to authenticate the requestor, specify the attribute returned in the identity provider's SAML assertion that corresponds to the Pega Platform operator ID. You can also map other attributes from the SAML assertion to selected properties and pages that are used by your preauthentication and postauthentication activities or by other Pega Platform features such as access control policies.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list. On the **SAML 2.0** tab, navigate to the **Operator identification** section.
2. In the **Map operator id from** section, select one of the following.
 - **Name identifier in the subject**
 - **Attribute**, then specify the attribute surrounded by braces, for example, {http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress}
3. **Optional:** To map additional attributes, click the **Mapping** tab.
4. **Optional:** Map the attributes from your login process to properties and pages that are used by your preauthentication and postauthentication activities or elsewhere in Pega Platform.

- a. You can use the following properties and pages in the **Map from** field. You can also reference custom properties and pages that are used in a login flow, and you can use the Expression Builder.

Page name	Description
<i>pxRequestor</i>	The requestor page

- b. You can use the following properties and pages in the **Map to** field. You can also reference custom properties and pages that are used in a login flow.

Page name	Description
<i>OperatorID</i>	Properties of the operator ID
<i>D_pyOperatorAttributes</i>	Requestor-scoped data page for caching operator attributes
<i>D_pyOperatorDeviceInformation</i>	Requestor-scoped data page for caching operator device information

5. You can also use the following properties and pages in the **Map from** field for a SAML authentication service.

Page name	Description
<i>D_SAMLAAssertionDataPage</i>	The SAML assertion.

6. Click **Save**.

What to do next: [Specifying preauthentication and postauthentication activities for a SAML SSO authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring SSO login authentication with a SAML identity provider](#)
- [Expression Builder](#)

Specifying preauthentication and postauthentication activities for a SAML SSO authentication service

To customize the login process, you can write activities that are triggered before and after SAML SSO authentication. For example, a postauthentication activity can update the operator record with values from the service provider or refuse an automatically provisioned user access to an application.

1. Create your preauthentication and postauthentication activities. For more information, see the sample activities *pySSOPreAuthenticationActivity* and *pySSOPostAuthenticationActivity*. When you write these activities, consider the following:
 - Update the *pega Browser* requestor type's default access group to equal an access group that includes the ruleset of the preauthentication activity. The preauthentication activity must be accessible to the *pega Browser* requestor type, which is used for the unauthenticated user session. By default, this requestor type is assigned to the *PRPC:Unauthenticated* access group.
 - Create the postauthentication activity in a ruleset that is accessible to the user's default access group. The postauthentication activity must be accessible to the user who has just been authenticated.
 - Set the Applies To key part of the activities to *Code-Security*.
 - Set the authentication result in *pyAuthenticationPolicyResult* to *true* to proceed with authentication, or set to *false* to terminate the request; for example:

```
tools.getRequestor().getRequestorPage().putString("pyAuthenticationPolicyResult", "true");
```
 - You can dynamically update the user's role list by using a postauthentication activity. This is useful if you want to dynamically add roles to the user context at login without maintaining multiple access groups in the database for all combinations of roles. In the postauthentication activity, update the list of roles by calling *setRoles*, as shown in the following example.

```
tools.getAuthorizationHandle().setRoles(tools, tools.getAuthorizationHandle().UPDATE_APPEND, rolesList);
```

Caution: Roles that you set in the postauthentication activity without saving the access group to the database lose their effect if the user switches applications. To dynamically add roles in a way that supports switching applications, see [How to dynamically add roles during user authentication](#).

2. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
3. On the **SAML 2.0** tab, expand the **Advanced configuration settings** section.
4. In the **Pre-authentication activity** field, enter the name of the preauthentication activity.
5. In the **Post-authentication activity** field, enter the name of the postauthentication activity.
6. Click **Save**.

What to do next: [Requiring reauthentication for new and expired sessions for a SAML SSO authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring SSO login authentication with a SAML identity provider](#)
- [Creating an activity](#)

Requiring reauthentication for new and expired sessions for a SAML SSO authentication service

To add an extra layer of security, you configure Pega Platform to reauthenticate new sessions and to reauthenticate when reactivating an expired session. The time-out period is specified on the operator's access group or is managed by the application server or another external facility.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
2. On the **SAML 2.0** tab, expand the **Advanced configuration settings** section.
3. Select the **Use access group timeout** checkbox to use the authentication time-out values that are specified in the user access group. This setting determines how long user sessions can remain inactive before users are prompted to identify themselves again. Clear the **Use access group timeout** checkbox if the time-out is managed by the application server or another external facility.
4. To require reauthentication for new and reactivated sessions, select the **Force authentication** check box.
5. Click **Save**.

What to do next: [Configuring operator provisioning for a SAML SSO authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring SSO login authentication with a SAML identity provider](#)

Configuring operator provisioning for a SAML SSO authentication service

When an operator who is not defined in Pega Platform logs in for the first time, Pega Platform can automatically create the operator record with values from the service provider. To do this, you configure operator provisioning for the authentication service. This configuration eliminates the manual step of adding the operator record.

Note:

- If authentication fails, operator provisioning does not occur.
- If authentication succeeds and the operator record is already defined in Pega Platform, you can update it with values from the authentication provider by creating a postauthentication activity.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list, and then click the **SAML 2.0** tab.
2. In the **Operator identification** section, select **Enable operator provisioning using model operator**.
3. To construct an operator by copying a specific model, click **By name** and enter a value for **Model operator**. This is the operator ID for the operator record to be copied. If the value contains a period ("."), enclose the value in double quotation marks, for example, "abc.def".
4. To identify the model operator by using an expression, click **By name**, and next to the **Model operator** field, click the **Build an expression** icon and create an expression that returns the operator ID to be copied.
5. To identify the model operator by using organizational information, click **By organization hierarchy** and enter values for **Org** (organization), **Div** (division), and **Unit**. You can also provide an expression for each of these fields. The Model User value in the Unit instance provides the model operator for constructing the new operator.
6. To create the operator by using a data transform, click **By data transform** and enter the data transform name. The Applies to class of the data transform must be *Data-Admin-Operator-ID*. For an example data transform, see *pyDefaultForNonPegaOperator*. If the data transform copies property values from a model operator, you must first create an operator instance in the database for the model operator. Otherwise, you do not need to create a model operator.
7. At a minimum, you must provide values for the properties listed below before the postauthentication activity (if any) is run. You can populate them from the model operator or the data transform, or you can explicitly map to them by using the **Mapping** tab. Do not change them in the postauthentication activity.

Property name	Description
<i>OperatorID.pyAccessGroup</i>	Operator's default access group
<i>OperatorID.pyAccessGroupsAdditional</i>	List of all of the operator's access groups, including the default access group
<i>OperatorID.pyOrganization</i>	Organization
<i>OperatorID.pyOrgDivision</i>	Division
<i>OperatorID.pyOrgUnit</i>	Unit

Note: The value for the newly created operator ID (*OperatorID.pyUserIdentifier*) is the value from the main tab in the **Map operator id from** section.

8. Click **Save**.

What to do next: [Overriding the service provider settings for a SAML SSO authentication service](#)

- [Mapping operator information for a SAML SSO authentication service](#)
- [Configuring SSO login authentication with a SAML identity provider](#)
- [Organization Unit data instances](#)
- [Expression Builder](#)

Overriding the service provider settings for a SAML SSO authentication service

The service provider settings for a SAML SSO authentication service are automatically populated when you create the authentication service. You can override the default values.

Note: For the SAML ruleform, Global Resource Settings are supported. For more information, see [Fields that support the Global Resource Settings syntax](#).

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list. On the **SAML 2.0** tab, expand the **Service Provider (SP) settings** section.
2. In the **Entity identification** field, enter an entity ID that is auto-populated in the new authentication services.
3. In the **Login (SSO) protocol binding** list, the system provides a default protocol binding. You can change the binding protocol to one of the following.
 - **HTTP Post** – SAML protocol messages are transmitted in an HTML form with base64-encoded content.
 - **HTTP Artifact** – SAML protocol messages are transmitted using a unique identifier called an artifact. Select this protocol if you do not want to expose the content of the SAML message during connection.
 - **HTTP Redirect** – SAML protocol messages are transmitted within URL parameters.
4. In the **Assertion Consumer Service (ACS) location** field, override the system-provided URL of the standard ACS REST service URL.
5. In the **Redirect logout location** field, override the system-provided URL of the standard logout REST service.
6. In the **SOAP logout location** field, override the system-provided URL of the standard logout SOAP service.
7. In the **Artifact Resolution Service (ARS) location** field, override the system-provided URL of the standard ARS to send the artifact resolve request to the IdP.
8. To disable the signing of authentication and logout requests from your application to the Identity Provider, select the **Disable request signing** check box.
9. To reject all unsigned SAML assertions, select the **Reject unsigned assertion** check box.
10. To select the SP Private Key to sign the SAML authentication and logout requests, in the **Signing certificate** section, click the Pencil icon.
 - a. In the **KEYSTORE NAME** field, press the Down Arrow key and select the keystore that contains the private key, private key alias, and password to use.
 - b. Click **Submit**.
11. In the **Decryption certificate** section, click the Pencil icon to select the SP Private Key to decrypt the response from the IdP for authentication and logout requests.
 - a. In the **KEYSTORE NAME** field, press the Down Arrow key and select the keystore that contains the private key, private key alias, and password to use.
 - b. Click **Submit**.
12. To download the service provider metadata, click **Save**, and then click **Download SP metadata**.
13. Click **Save**.

What to do next: [Enforcing policies from the Security Policies landing page](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring SSO login authentication with a SAML identity provider](#)

Enforcing policies from the Security Policies landing page

You can configure your SAML SSO authentication service to enforce security policies that are defined on the Security Policies landing page.

1. Open the service from the navigation pane in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list, and then click the **Security policies** tab.
2. Click the **Add item** icon to specify a security policy.
3. In the **Policy type** list, select the policy to enforce.

Note:

 - If you select **Multi-factor authentication**, you must map properties such as email address or phone number from the email or SMS receiver account by using the **Mapping** tab.
 - If you select **User consent**, the section *pyPostAuthConsentScreen* is displayed to the user after authentication. You can customize this section.
 - If you select **CAPTCHA**, the settings on the **Security Policies** landing page take effect as described below.
 - If you set the value of **Enable presentation of CAPTCHA upon initial login** to **Enabled**, the CAPTCHA is displayed for every login.
 - If you set the value of **Enable presentation of CAPTCHA upon initial login** to **Disabled**, the CAPTCHA is displayed according to the probability that you set in the **Probability that CAPTCHA will be presented upon authentication failure (%)** field.
4. Click **Save**.

What to do next: [Activating your SAML SSO authentication service](#)

- [Authentication services](#)
- [Authentication services and security policies](#)
- [Configuring SSO login authentication with a SAML identity provider](#)
- [Security policies](#)

Activating your SAML SSO authentication service

After you create and configure a SAML SSO authentication service, enable the service for Pega Platform.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
 2. On the **SAML 2.0** tab, select the **Enable this authentication service** check box.
 3. Click **Save**.
- [Authentication services](#)
 - [More about authentication services](#)
 - [Configuring SSO login authentication with a SAML identity provider](#)

Updating an expired identity provider certificate in a SAML Authentication Service

If you are using a SAML Authentication service and your Identity Provider (IdP) certificate has expired, you need to reimport the Identity Provider metadata. This is the easiest way to update an expired IdP certificate.

1. In the side panel of Dev Studio, click **Records SysAdmin Authentication service**.
2. Select the SAML authentication service that you need to update.
3. In the **Identity Provider (IdP) information** section of the *SAML 2.0* tab, click the **Import IdP metadata** text to import the Identity Provider metadata with one of the following options:

Upload option	Description
via URL	Enter the IdP's metadata download URL.
via file	Enter the name of a file containing the downloaded metadata from the IdP.

4. Click **Save**.

Updating an expired Service Provider certificate in a SAML Authentication Service

If you are using a SAML Authentication service and your Service provider certificate has expired, then you need to create a new keystore, select that keystore under Service Provider details, and re-export the service provider metadata to the identity provider.

To complete this process, you must complete the following 3 tasks in this order:

1. Create a keystore with a valid expiration.
2. Configure the keystore.
3. Re-export the service provider metadata into the Identity Provider metadata.

Create a keystore with a valid expiration

A keystore is a file that contains keys and certificates that you use for encryption, authentication, and serving content over HTTPS. A valid expiration is any expiration date that is in the future.

1. Follow the steps in [Creating a keystore for application data encryption](#).

Configure the keystore

Note: There are several ways to configure a keystore in Pega Platform. In this example, we will be uploading a keystore. If you are using an alternative method for creating a keystore, see [Keystores](#).

Now that you have created a keystore, you need to configure it.

Open a keystore you just created by:

1. In the navigation panel of Dev Studio, click **Records Security Keystore** and select a keystore from the instance list.
2. Click **Upload file**.
3. Click **Choose File**, browse to the keystore file, and select it.
4. Click **Upload file**.
5. In the **Keystore type** field, enter the keystore file type: JKS, JWK, PKCS12, KEYTAB, or KEY.
6. In the **Keystore password** field, enter the password to the keystore file.
7. Click **Save**.

Re-export the Service Provider metadata into the Identity Provider metadata

The Service Provider and Identity Provider need to communicate with one another. To finalize the process, you need to export the updated SP metadata, and then add it into the Identity Provider.

1. In the side panel of Dev Studio, click **Records SysAdmin Authentication service**.
2. Select the SAML authentication service that you need to update.
3. In the **Service Provider (SP) settings** section of the *SAML 2.0* tab, click the **Download SP metadata** text.**Note:** This will open a new tab with the system's metadata in XML.
4. Import the SP metadata into your Identity Provider.

Configuring SSO login authentication with an OpenID Connect identity provider

After you create an OpenID Connect SSO authentication service, configure it so that Pega Platform uses the specified identity provider for authenticating users. You can map claims from the OpenID Connect provider to properties in Pega Platform, and configure optional features such as preauthentication and postauthentication activities and operator provisioning.

Before you begin: You must complete the following task before you can configure an OpenID Connect authentication service: [Creating an authentication service](#) **Note:** If you access Pega Platform using a reverse proxy server, you must ensure that the reverse proxy is configured before configuring an OIDC SSO authentication service. For more information, see [Deployment behind a reverse proxy](#).

1. Create an OpenID Connect authentication service, or open an existing service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing an OpenID Connect authentication service from the instance list.
 2. In the **Authentication service alias** field, specify an alias to represent a unique value for this service. This value becomes the final part of the URL path for users to access Pega Platform.
 - **Login URL** is a read-only field that displays the URL that accesses Pega Platform and uses this service for user authentication.
 - **Authentication flow** is a read-only field that identifies the OAuth standard flow type for this authentication service.
 3. **Optional:** In the **Provider logo** field, specify an image to display on the login screen that identifies this provider.
 4. [Configure the identity provider](#).
 5. **Optional:** Configure the optional parameters of the service.
 - [Map operator information](#)
 - [Specify preauthentication and postauthentication activities](#)
 - [Require reauthentication for new and expired sessions](#)
 - [Configure operator provisioning](#)
 - [Enforce policies from the Security Policies landing page](#)
 6. [Activate your OpenID Connect authentication service](#).
- [Authentication services](#)
 - [More about authentication services](#)

Configuring the identity provider for an OpenID Connect SSO authentication service

To enable the system to verify the identity of requestors, configure the identity provider for your OpenID SSO authentication service. You configure the identity provider by importing values from a file or by entering them manually.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
2. On the **OpenID Connect** tab, navigate to the **OpenID Connect provider configuration** section.
3. If you are configuring the identity provider by importing the configuration, complete these steps:
 - a. Click **Import metadata**.
 - b. Select the source of the metadata (**via URL or via file**), and then enter the URL or file path.
URL is in the format: `https://<domain name>/well-known/openid-configuration`
 - c. Click **Submit**.

Note: When you import the provider metadata, the platform does the following actions, depending on whether the **Signature truststore** field is blank at the time of import.

 - If the **Signature truststore** field is blank, the system creates a keystore instance and adds the certificate to the new keystore instance. The system sets the alias of the entry in the keystore to the certificate's issuer name and sets the keystore password to `rules`. The system populates the **Signature truststore** field with the new keystore identifier.
 - If the **Signature truststore** field is not blank and refers to a valid keystore instance that was originally created by the system (keystore name starts with "KS" and ends with "OIDCCertStore"), the system adds the certificate to the existing keystore instance and sets the alias of the entry to the certificate's issuer name.
4. If you are configuring the identity provider by entering values manually, complete these steps:
 - a. In the **Authorization endpoint** field, enter the authorization endpoint URL.
 - b. In the **Token endpoint** field, enter the token endpoint URL.
 - c. **Optional:** In the **Userinfo endpoint** field, enter the userinfo endpoint URL.
 - d. **Optional:** In the **Logout endpoint** field, enter the logout endpoint URL.
 - e. In the **Redirect URI** field, enter the redirect URI.
 - f. In the **Issuer** field, enter the certificate issuer.
 - g. In the **Signature truststore** field, press the Down Arrow key and select the keystore that contains the public key that is used for verifying the signature of the authentication assertion.
 - h. To add parameters for any of these fields, click **Add parameters**.

5. Navigate to the **Client information** section and complete the **Client identifier**, **Client secret**, and **Scope** fields.
6. In the **Operator identification** section, in the **Map operator id from claim** field, specify the name of the claim that contains the operator ID. Enclose the attribute name in curly braces, for example, {name}.
7. **Optional:** [Configure the advanced properties of the identity provider.](#)
8. Click **Save**.

What to do next: [Mapping operator information for an OpenID Connect SSO authentication service](#)

- [Configuring advanced features for an OpenID Connect SSO authentication service](#)
- [Authentication services](#)
- [More about authentication services](#)
- [Configuring SSO login authentication with an OpenID Connect identity provider](#)

Configuring advanced features for an OpenID Connect SSO authentication service

You can fine-tune the interaction of Pega Platform and your OpenID Connect authentication provider by supplying additional parameters.

Before you begin: You must complete the following task before you can update the advanced configuration for an OpenID Connect identity provider: [Configuring the identity provider for an OpenID Connect SSO authentication service](#)

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
2. On the **OpenID Connect** tab, expand the **Advanced configurations** section.
3. In the **Client authentication scheme** section, select the authentication scheme.
 - **Basic**
 - **POST**
4. In the **Send access token as** section, select how to send the access token:
 - **Authorization header**
 - **Query string parameter**
5. In the **Secure protocol configuration** section, complete the following fields:
 - a. In the **Lowest allowable SSL/TLS version** field, select the lowest allowable SSL or TLS version for communicating with your identity provider.
 - b. In the **Truststore** field, press the Down Arrow and select the truststore record that is used for secure access.
 - c. In the **Keystore** field, press the Down Arrow and select the keystore record that is used for secure access.
6. Click **Save**.

What to do next: [Mapping operator information for an OpenID Connect SSO authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring SSO login authentication with an OpenID Connect identity provider](#)

Mapping operator information for an OpenID Connect SSO authentication service

You can map attributes obtained from the UserInfo provider to selected properties and pages used by your preauthentication and postauthentication activities or by other Pega Platform features such as access control policies. Mapping complements the use of preauthentication and postauthentication activities, operator provisioning by data transform, authentication against an external repository, and other actions.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
2. Click the **Mapping** tab.
3. Map the attributes from your login process to properties and pages that are used by your preauthentication and postauthentication activities or elsewhere in Pega Platform.
 - a. You can use the following properties and pages in the **Map from** field. You can also reference custom properties and pages that are used in a login flow, and you can use the Expression Builder.

Page name	Description
<i>pxRequestor</i>	The requestor page

- b. You can use the following properties and pages in the **Map to** field. You can also reference custom properties and pages that are used in a login flow.

Page name	Description
<i>OperatorID</i>	Properties of the operator ID
<i>D_pyOperatorAttributes</i>	Requestor-scoped data page for caching operator attributes
<i>D_pyOperatorDeviceInformation</i>	Requestor-scoped data page for caching operator device information

4. You can also use the following properties and pages in the **Map from** field for an OpenID Connect authentication service.

Page name	Description
<i>D_pyUserInfoClaims</i>	The OIDC claim.

Note: This page becomes available and can only be accessed post authentication.

5. Click **Save**.

What to do next: [Specifying preauthentication and postauthentication activities for an OpenID Connect SSO authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring SSO login authentication with an OpenID Connect identity provider](#)
- [Expression Builder](#)

Specifying preauthentication and postauthentication activities for an OpenID Connect SSO authentication service

To customize the login process, you can write activities that will be triggered before and after OpenID Connect authentication. For example, a postauthentication activity can update the operator record with values from the service provider or can refuse an automatically provisioned user access to an application.

1. Create your preauthentication and postauthentication activities. For more information, see the sample activities *pySSOPreAuthenticationActivity* and *pySSOPostAuthenticationActivity*. When you write these activities, consider the following:
 - Update the *pega Browser* requestor type's default access group to equal an access group that includes the ruleset of the preauthentication activity. The preauthentication activity must be accessible to the *pega Browser* requestor type, which is used for the unauthenticated user session. By default, this requestor type is assigned to the *PRPC:Unauthenticated* access group.
 - Create the postauthentication activity in a ruleset that is accessible to the user's default access group. The postauthentication activity must be accessible to the user who has just been authenticated.
 - Set the Applies To key part of the activities to *Code-Security*.
 - Set the authentication result in *pyAuthenticationPolicyResult* to *true* to proceed with authentication, or set to *false* to terminate the request; for example:

```
tools.getRequestor().getRequestorPage().putString("pyAuthenticationPolicyResult", "true");
```

- You can dynamically update the user's role list by using a postauthentication activity. This is useful if you want to dynamically add roles to the user context at login without maintaining multiple access groups in the database for all combinations of roles. In the postauthentication activity, update the list of roles by calling *setRoles*, as shown in the following example.

```
tools.getAuthorizationHandle().setRoles(tools, tools.getAuthorizationHandle().UPDATE_APPEND, rolesList);
```

Caution: Roles that you set in the postauthentication activity without saving the access group to the database lose their effect if the user switches applications. To dynamically add roles in a way that supports switching applications, see [How to dynamically add roles during user authentication](#).

2. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
3. On the **OpenID Connect** tab, expand the **Advanced configuration settings** section.
4. In the **Pre-authentication activity** field, enter the name of the preauthentication activity.
5. In the **Post-authentication activity** field, enter the name of the postauthentication activity.
6. Click **Save**.

What to do next: [Requiring reauthentication for new and expired sessions for an OpenID Connect SSO authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring SSO login authentication with an OpenID Connect identity provider](#)
- [Creating an activity](#)

Requiring reauthentication for new and expired sessions for an OpenID Connect SSO authentication service

To add an extra layer of security, you configure Pega Platform to reauthenticate new sessions and to reauthenticate when reactivating an expired session. The time-out period is specified on the operator's access group or is managed by the application server or another external facility.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
2. On the **OpenID Connect** tab, expand the **Advanced configuration settings** section and the **Processing** subsection.
3. Select the **Use access group timeout** checkbox to use the authentication time-out values that are specified in the user access group. This setting determines how long user sessions can remain inactive before users are prompted to identify themselves again. Clear the **Use access group timeout** checkbox if the time-out is managed by the application server or another external facility.
4. Click **Save**.

What to do next: [Configuring operator provisioning for an OpenID Connect SSO authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring SSO login authentication with an OpenID Connect identity provider](#)

Configuring operator provisioning for an OpenID Connect SSO authentication service

When an operator who is not defined in Pega Platform logs in for the first time, Pega Platform can automatically create the operator record with values from the service provider. To do this, you configure operator provisioning for the authentication service. This configuration eliminates the manual step of adding the operator record.

Note:

- If authentication fails, operator provisioning does not occur.
- If authentication succeeds and the operator record is already defined in Pega Platform, you can update it with values from the authentication provider by creating a postauthentication activity.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list, and then click the **OpenID Connect** tab.
2. In the **Operator identification** section, select **Enable operator provisioning using model operator**.
3. To construct an operator by copying a specific model, click **By name** and enter a value for **Model operator**. This is the operator ID for the operator record to be copied. If the value contains a period ("."), enclose the value in double quotation marks, for example, "abc.def".
4. To identify the model operator by using an expression, click **By name**, and next to the **Model operator** field, click the **Build an expression** icon and create an expression that returns the operator ID to be copied.
5. To identify the model operator by using organizational information, click **By organization hierarchy** and enter values for **Org** (organization), **Div** (division), and **Unit**. You can also provide an expression for each of these fields. The Model User value in the Unit instance provides the model operator for constructing the new operator.
6. To create the operator by using a data transform, click **By data transform** and enter the data transform name. The applies to class of the data transform must be *Data-Admin-Operator-ID*. For an example data transform, see *pyDefaultForNonPegaOperator*. If the data transform copies property values from a model operator, you must first create an operator instance in the database for the model operator. Otherwise, you do not need to create a model operator.
7. At a minimum, you must provide values for the properties listed below before the postauthentication activity (if any) is run. You can populate them from the model operator or the data transform, or you can explicitly map to them by using the **Mapping** tab. Do not change them in the postauthentication activity.

Property name	Description
<i>OperatorID.pyAccessGroup</i>	Operator's default access group
<i>OperatorID.pyAccessGroupsAdditional</i>	List of all of the operator's access groups, including the default access group
<i>OperatorID.pyOrganization</i>	Organization
<i>OperatorID.pyOrgDivision</i>	Division
<i>OperatorID.pyOrgUnit</i>	Unit

Note: The value for the newly created operator ID (*OperatorID.pyUserIdentifier*) is the value from the main tab in the **Map operator id from** section.

8. Click **Save**.

What to do next: [Enforcing policies from the Security Policies landing page](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring SSO login authentication with an OpenID Connect identity provider](#)
- [Organization Unit data instances](#)
- [Expression Builder](#)

Enforcing policies from the Security Policies landing page

You can configure your OpenID Connect SSO authentication service to enforce security policies that are defined on the Security Policies landing page.

1. Open the service from the navigation pane in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list, and then click the **Security policies** tab.
2. Click the **Add item** icon to specify a security policy.
3. In the **Policy type** list, select the policy to enforce.

Note:

- If you select **Multi-factor authentication**, you must map properties such as email address or phone number from the email or SMS receiver account by using the **Mapping** tab.
- If you select **User consent**, the section *pyPostAuthConsentScreen* is displayed to the user after authentication. You can customize this section.
- If you select **CAPTCHA**, the settings on the **Security Policies** landing page take effect as described below.
 - If you set the value of **Enable presentation of CAPTCHA upon initial login** to **Enabled**, the CAPTCHA is displayed for every login.
 - If you set the value of **Enable presentation of CAPTCHA upon initial login** to **Disabled**, the CAPTCHA is displayed according to the probability that you set in the **Probability that CAPTCHA will be presented upon authentication failure (%)** field.

4. Click **Save**.

What to do next: [Activating your OpenID Connect SSO authentication service](#)

- [Authentication services](#)
- [Authentication services and security policies](#)

- [Configuring SSO login authentication with an OpenID Connect identity provider](#)
- [Security policies](#)

Activating your OpenID Connect SSO authentication service

After you create and configure an OpenID Connect SSO authentication service, enable the service for Pega Platform.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
2. On the **OpenID Connect** tab, select the **Enable this authentication service** check box.
3. Click **Save**.

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring SSO login authentication with an OpenID Connect identity provider](#)

Configuring login authentication for an anonymous operator

After you create an anonymous authentication service, configure it so that Pega Platform can support guest users. You can map attributes from the model operator to properties in Pega Platform, and also configure preauthentication and postauthentication activities.

Before you begin: You must complete the following task before you can configure an anonymous authentication service: [Creating an authentication service](#).

1. Create an anonymous authentication service, or open an existing service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and selecting an anonymous authentication service from the instance list.
2. In the **Authentication service alias** field, specify an alias to represent a unique value for this service. This value becomes the final part of the URL path for users to access Pega Platform.
Login URL is a read-only field that displays the URL that accesses Pega Platform and uses this service for user authentication.
3. [Configure the anonymous operator](#).
4. **Optional:** [Specify preauthentication and postauthentication activities](#).
5. **Optional:** [Map operator information](#).
6. [Activate your anonymous authentication service](#).

- [Configuring an application to use an anonymous authentication service](#)
- [Authentication services](#)
- [More about authentication services](#)

Configuring the anonymous operator for an anonymous authentication service

Configure an anonymous operator for use in an anonymous authentication service. When Pega Platform creates a session for an anonymous operator, a unique operator ID is generated and other operator values are populated from a model operator or by using a data transform.

At run time, the access group for the anonymous operator is set initially to the value of the default access group for the *pega Browser* requestor type, as described in [Authentication services and rule availability](#). After the preauthentication activity (if any) is run, the values from the model operator or data transform are applied to the anonymous operator. If the model operator or data transform sets an access group, that access group is used from that point forward during the anonymous session.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list. On the **Anonymous** tab, navigate to the **Operator identification** section.
2. To construct an operator by copying a specific model, click **By name** and enter a value for **Model operator**.
This is the operator ID for the operator record to be copied. If the value contains a period ("."), enclose the value in double quotation marks, for example, "abc.def".
3. To identify the model operator by using an expression, click **By name**, and next to the **Model operator** field, click the **Build an expression** icon and create an expression that returns the operator ID to be copied.
4. To identify the model operator by using organizational information, click **By organization hierarchy** and enter values for **Org** (organization), **Div** (division), and **Unit**. You can also provide an expression for each of these fields.
The Model User value in the Unit instance provides the model operator for constructing the new operator.
5. To create the operator by using a data transform, click **By data transform** and enter the data transform name.
The applies to class of the data transform must be *Data-Admin-Operator-ID*. For an example data transform, see *pyDefaultForNonPegaOperator*. If the data transform copies property values from a model operator, you must first create an operator instance in the database for the model operator. Otherwise, you do not need to create a model operator.
6. Click **Save**.

What to do next: [Specifying preauthentication and postauthentication activities for an anonymous authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring login authentication for an anonymous operator](#)
- [Organization Unit data instances](#)

Specifying preauthentication and postauthentication activities for an anonymous authentication service

You can write activities that are triggered at various points during the process of setting up the anonymous operator.

1. Create your preauthentication and postauthentication activities.
For an anonymous authentication service:
 - The preauthentication activity is run before the anonymous operator is populated from the model operator or data transform. It must be accessible to the default access group for the *pega Browser* requestor type, as described in [Authentication services and rule availability](#).
 - The postauthentication activity is run after the anonymous operator is populated from the model operator or data transform, and affects the anonymous session.
2. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
3. On the **Anonymous** tab, expand the **Advanced configuration settings** section.
4. In the **Pre-authentication activity** field, enter the name of the preauthentication activity.
5. In the **Post-authentication activity** field, enter the name of the postauthentication activity.
6. Click **Save**.

What to do next: [Mapping operator information for an anonymous authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring login authentication for an anonymous operator](#)
- [Creating an activity](#)

Mapping operator information for an anonymous authentication service

You can map attributes to selected properties and pages used by your preauthentication and postauthentication activities or by other Pega Platform features such as access control policies. Mapping complements the use of preauthentication and postauthentication activities, operator provisioning by data transform, and other actions.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
2. Click the **Mapping** tab.
3. Map the attributes from your login process to properties and pages that are used by your preauthentication and postauthentication activities or elsewhere in Pega Platform.
 - a. You can use the following properties and pages in the **Map from** field. You can also reference custom properties and pages that are used in a login flow, and you can use the Expression Builder.

Page name	Description
<i>pxRequestor</i>	The requestor page

- b. You can use the following properties and pages in the **Map to** field. You can also reference custom properties and pages that are used in a login flow.

Page name	Description
<i>OperatorID</i>	Properties of the operator ID
<i>D_pyOperatorAttributes</i>	Requestor-scoped data page for caching operator attributes
<i>D_pyOperatorDeviceInformation</i>	Requestor-scoped data page for caching operator device information

4. Click **Save**.

What to do next: [Activating your anonymous authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring login authentication for an anonymous operator](#)
- [Expression Builder](#)

Activating your anonymous authentication service

After you create and configure an anonymous authentication service, enable the service for Pega Platform.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
2. On the **Anonymous** tab, select the **Enable this authentication service** check box.
3. Click **Save**.

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring login authentication for an anonymous operator](#)

Configuring an application to use an anonymous authentication service

Enhance your application's user experience by enabling anonymous users to connect to the application without entering any login credentials, and prompting them for authentication partway through a session. You can build this type of application by using an anonymous authentication service and the reauthentication gadget.

For example, at most online shopping portals, you can shop without identifying yourself. When you check out, however, you receive a prompt for credentials.

- Create the anonymous authentication service of the **Anonymous** authentication type.
For more information, see [Creating an authentication service](#).
- Create one or more non-anonymous authentication services of any authentication type except **Anonymous**.
For more information, see [Creating an authentication service](#).
- Result:** At run time, when an authentication prompt is displayed, the user selects an authentication service.
- Create an application.
Create the application in a ruleset that is accessible to guest users. Users initially connect as guests by using the **Login URL** that is shown on the authentication service form. For more information, see [Authentication services and rule availability](#).
- Configure the application to trigger the authentication mechanism at the step where you want to prompt the user for credentials, by starting the local flow action *pyReAuthGadget*.
This flow action displays the section *pxReAuthGadget*, where the user authenticates. The target of the *pyReAuthGadget* flow action can be a modal dialog box or an overlay. The Replace Current target, which is displayed in the list of options, is not supported.
For example: A shopping application can have a button that invokes *pyReAuthGadget* when a user checks out.
You can use the when rule *pzIsAnonymousUser* at any time to check whether the user has reauthenticated. A return value of false indicates that the user has provided valid credentials and has been authenticated. Use this when rule to conditionally display a button or link to *pyReAuthGadget*.
Result: At run time, *pxReAuthGadget* displays the names of the enabled authentication services so that the user can select a service and provide their credentials. You can optionally modify the user interface by customizing the gadget.

After the user authenticates, the gadget invokes the activity *pyMigrateDataForReAuthenticated*, while Pega Platform passes the old guest operator ID and the newly authenticated operator ID as input parameters to this activity. **Note:** You can extend this activity to copy data, such as the shopping cart contents, from the original user to the newly authenticated user. For example, a shopping application stores shopping cart information for all active users in the cart table, keyed by operator ID. When the user authenticates, your customized version of *pyMigrateDataForReAuthenticated* copies the relevant cart data from the old user to the new user and saves the cart.

- [Authentication services](#)
- [Creating an authentication service](#)
- [Configuring login authentication for an anonymous operator](#)

Configuring custom or Kerberos login authentication

After you create a custom or Kerberos authentication service, configure it so that Pega Platform can connect to the repository and find the operator credentials. You can map attributes from the repository to properties in Pega Platform, and can also configure optional features such as authentication and time-out activities.

Before you begin: You must complete the following task before you can configure a custom or Kerberos authentication service: [Creating an authentication service](#).

To create a custom or Kerberos authentication service, or open an existing service

1. In the navigation panel of Dev Studio, click **Records SysAdmin Authentication Service**
2. Choose a non-SAML authentication service from the instance list.
3. If you have a custom authentication service, complete the following steps:
 - a. [Specify the binding for a custom service](#)
 - b. [Map the operator for a custom service](#)
4. If you have a Kerberos authentication service, complete the following steps:
 - a. [Specify the binding for a Kerberos service](#)
 - b. [Map the operator for a Kerberos service](#)
5. **Optional:** [Configure the challenge behavior](#)
6. **Optional:** [Configure the time-out behavior](#)
7. **Optional:** [Map additional operator information](#)

- [Authentication services](#)
- [More about authentication services](#)

Configuring login using a custom authentication service

Specify the appropriate fields based on how users will be authenticated in the custom authentication service. If using a directory, specify binding information for your directory. To access this service, you will also need to specify the activities used to validate user credentials, the alias name, and alias URL pattern.

1. In the navigation pane of Dev Studio click **Records SysAdmin Authentication Service** .
2. Select an existing custom authentication service from the instance list, or create a custom authentication service.
3. Click the **Service** tab.
4. In the Login configuration section:
 - a. In the **Authentication service alias** field, enter the application service alias you want to use for this Authentication service.

The Authentication service alias is used for hitting the authorization endpoint.

For example: Enter ClientLogin in the **Authentication service alias** field.

- b. In the **URL pattern** field, enter the servlet name mapped in web.xml for this authentication service.

In this example, when viewing the web.xml, the LDAP authentication service has the **PRWebLDAP1** servlet, which is mapped to the **WebLDAP1** authentication service.

For example: Enter WebLDAP1 in the **URL pattern** field and the following URL will generate **Login URL**: *https://company.com/prweb/PRWebLDAP1*

5. In the **Custom Authentication Activity** section, in the **Authentication Activity** field, press the Down Arrow key and select the authentication activity. The activity must have *Code-Security* as the Applies To key part.
Note: In this example, you would select the *AuthenticationLDAP* activity because this is for an LDAP authentication. However, you could create your own activity to support a custom authentication service, then select it here.
6. In the **JNDI Binding Parameters** section, in the **Initial context factory** field, enter the Java class name of the JNDI initial context factory that you want to use to connect to the directory server, for example, enter `com.sun.jndi.ldap.LdapCtxFactory`.
7. In the **Directory** field, enter an explicit URL or a JNDI entry that represents a directory located on the LDAP server. This approach enables you to relocate servers without having to reconfigure the application.
For an explicit URL, use the following format: `ldap[s]://[servername]:[portnumber]`.
For a JNDI entry, the syntax is dependent upon the server environment.
8. In the **Trust store** field, press the Down Arrow key and select the truststore record that contains the server certificates.
9. In the **Bind distinguished name** field, enter the credentials of the bind user who is allowed to search the directory tree for the credentials of a user who is attempting to log in.
The standard LDAP authentication activities authenticate with the directory server as this user so it can then search the directory for users.
10. To enter a password for the bind user, click **Set Bind Password**.
11. In the **Search Parameters** section, in the **Directory context** field, enter the directory context that defines the branch in the Directory Information Tree (DIT) that holds information about the users who can be authenticated by this authentication server. For example, enter `OU=people, DC=yourco, DC=com`.
12. Click **Save**.
13. To check the connectivity to the server and verify that the configuration JNDI binding parameters are correct, click **Test connectivity**.

What to do next: [Identifying the operator for custom authentication services](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Creating an authentication service](#)
- [Keystores](#)

Identifying the operator for custom authentication services

Specify the operator mapping so that a custom authentication service can find the requestor record in your repository and bring values back to Pega Platform.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list, and then navigate to the **Service** tab.
2. In the **Search filter** field, enter an expression to use to find and validate the user's distinguished name (DN). Enter the following format: `( cn=%V )`, where `cn` is the user and `%V` is the user ID entered by the person logging in. Scans the characters in user ID and password input, and deletes any characters (such as asterisk, equals sign, or vertical stroke) that are not allowed by the LDAP specification before calling the LDAP server. These characters are removed without notifying the user or rejecting the request. No log messages or error messages appear.
Leave the **User name attribute** field blank. This field provides backward compatibility with a LDAP integration feature provided in Version 4.
To ensure maximum security, during sign-on, Pega Platform scans the characters in user ID and password input and deletes any characters (such as asterisk, equals sign, or vertical stroke) that are not allowed by the LDAP specification before calling the LDAP server. These characters are removed without notifying the user or rejecting the request. No log messages or error messages appear.
3. **Optional:** In the **Mapping** tab, map the naming attributes in the directory sever that correspond to the operator ID properties to locate a model user.
Complete this step only if the custom authentication service identifies a directory server.
 - a. Click the **Add item** icon, and add three attribute mapping rows.
 - b. In each row, in the **Property Name** field, enter one of the following properties: `.pyOrganization`, `.pyOrgDivision`, and `.pyOrgUnit`.
 - c. In the **Attribute Name** field, enter the user attribute in the directory that relates to the entered operator property, for example, `o`, `ou`, and `sn`.
4. On the **Custom** tab, in the **Source of operator credentials** list, select where the operator credentials are stored.
 - **Use credentials stored in PegaRULES** – Only the users whose operator ID records do not have **Use external authentication** selected are allowed access through this service.
 - **Use externally stored credentials** – Only the users whose operator ID records have **Use external authentication** selected are allowed access through this service.
5. Click **Save**.

What to do next: [Configuring the challenge behavior for custom or Kerberos authentication services](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Creating an authentication service](#)

Configuring login using a Kerberos authentication service

Specify the appropriate fields based on how users will be authenticated in the Kerberos authentication service. If using a directory, specify binding information for your directory. To access this service, you will also need to specify the activities used to validate user credentials, the alias name, and alias URL pattern.

1. In the navigation pane of Dev Studio click **Records SysAdmin Authentication Service**.
2. Select an existing custom authentication service from the instance list, or create a custom authentication service.
3. Click the **Kerberos** tab.
4. In the Login configuration section:
 - a. In the **Authentication service alias** field, enter the application service alias you want to use for the Authentication service.

The Authentication service alias is used for hitting the authorization endpoint.

For example: Enter `ClientLogin` in the **Authentication service alias** field.

- b. In the **URL pattern** field, enter the servlet name mapped in web.xml for this authentication service.

In this example, when viewing the web.xml, the LDAP authentication service has the **PRWebLDAP1** servlet, which is mapped to the **WebLDAP1** authentication service.

For example: Enter WebLDAP1 in the **URL pattern** field and the following URL will generate **Login URL**: *https://company.com/prweb/PRWebLDAP1*

5. In the **Custom Authentication Activity** section, in the **Authentication activity** field, press the Down Arrow key and select the authentication activity.
pyAuthenticationKerberosCredentials activity.
Note: In this example, you would select the *pyAuthenticationKerberosCredentials* activity because this is for Kerberos authentication. However, you could create your own activity to support a custom authentication service, then select it here.
If you are not using SPNEGO Source Forge, modify this activity code to retrieve GSSCredentials from the *pxRequestor* clipboard page.
The activity must have *Code-Security* as the Applies To key part.
6. In the **JNDI Binding Parameters** section, in the **Initial context factory** field, enter the Java class name of the JNDI initial context factory that you want to use to connect to the directory server, for example, `com.sun.jndi.ldap.LdapCtxFactory`.
7. If you are using LDAP, in the **Directory** field, enter the LDAP server information. Otherwise, leave this field blank.
8. In the **Trust store** field, press the Down Arrow key and select the truststore record that contains the server certificates.
9. Click **Save**.

What to do next: [Identifying the operator for Kerberos authentication services](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Creating an authentication service](#)
- [Keystores](#)

Identifying the operator for Kerberos authentication services

Specify the operator mapping so that a Kerberos authentication service can find the requestor record in your repository and bring values back to Pega Platform.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list, and then navigate to the **Kerberos** tab.
2. If you are using LDAP, in the **User principal name attribute** field, enter the LDAP attribute name that is used for the User Principal Name on the LDAP server.
3. On the **Mapping** tab, map the naming attributes in the directory sever that correspond to the operator ID properties.
 - a. Click the **Add item** icon, and add four attribute mapping rows.
 - b. In the **Property Name** field, enter one of the following properties: `.pyUserName`, `.pyOrganization`, `.pyOrgDivision`, and `.pyOrgUnit`.
 - c. In the **Attribute Name** field, enter the LDAP attribute that relates to the entered operator property.
4. On the **Custom** tab, in the **Source of operator credentials** list, select where the operator credentials are stored.
 - **Use credentials stored in PegaRULES** – Only the users whose operator ID records do not have **Use external authentication** selected are allowed access through this service.
 - **Use externally stored credentials** – Only the users whose operator ID records have **Use external authentication** selected are allowed access through this service.
5. Click **Save**.

What to do next: [Configuring the challenge behavior for custom or Kerberos authentication services](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Creating an authentication service](#)

Configuring the challenge behavior for custom or Kerberos authentication services

Configure the way in which your custom or Kerberos authentication service requests the login credentials.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list, and then navigate to the **Custom** tab.
2. In the **Initial challenge stream** field, press the Down Arrow key and select the second key part of an HTML rule to run (where the Applies To key part is `@baseclass`) if user navigates to your system through a non-HTTPS URL. Design the page rendered by this HTML rule to redirect the user to an HTTPS URL where the user is challenged for credentials. See the standard HTML rule *Web-Login-SecuredBasic* for an example. By default, this rule runs when all of the following are true:
 - The Use SSL option is selected.
 - An unauthenticated user navigates to the system through an unsecured port.
 - No rule is specified in the Initial Challenge Stream field.
3. To require users to use a secure port (HTTPS) for authentication, select the **Use SSL** check box. If **Use SSL** is checked, users must use an HTTPS URL for authentication. Verify that the application server that hosts your system uses HTTPS and that a URL is configured with the appropriate security constraints.
4. Complete one of the following steps.
 - To use the Basic authentication browser pop-up window to gather user credentials, select the **Use Basic Authentication for signon** check box.
 - In the **Credential challenge stream** field, press the Down Arrow key and select the second key part of an HTML rule (where the Applies To key part is `@baseclass`) that provides the login form that gathers user credentials.
5. To customize the page that appears when authentication fails, specify an HTML rule in the **Authentication fail stream** field. Enter the second key part of the HTML rule (where the Applies To key part is `@baseclass`) that provides the page that appears when a user's username and password combination does not pass authentication. Commonly, one HTML rule is used for both the challenge stream and the fail stream.
6. Click **Save**.

What to do next: [Configuring the time-out behavior for custom or Kerberos authentication services](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Creating an authentication service](#)
- [About HTML rules](#)

Configuring the time-out behavior for custom or Kerberos authentication services

Configure the way in which your custom or Kerberos authentication service handles operator time-out situations.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
2. For a Kerberos authentication service, on the **Kerberos** tab, in the **Timeout activity** field, press the Down Arrow key and select the *AuthenticationLDAPWebTimeout* activity to control the time-out behavior. The activity must have *Code-Security* as the Applies To key part.
3. For a custom authentication service, on the **Service** tab, in the **Timeout activity** field, press the Down Arrow key and select the *AuthenticationLDAPTimeout* activity to control the time-out behavior. The activity must have *Code-Security* as the Applies To key part.
4. Navigate to the **Custom** tab.
5. Select the **Use PegaRULES Timeout** check box to use the authentication time-out values specified in the user access groups to determine how long a user session can remain inactive before users are prompted to identify themselves again. Leave this check box cleared if time-out is managed by the application server or other external facility.
6. Do one of the following steps.
 - Select the **Use Basic Authentication for Timeout** check box to use the Basic authentication browser pop-up window to gather credentials when user's session times out.
 - In the **Timeout challenge stream** field, enter the second key part of an HTML rule (where the Applies To key part is `@baseclass`) that presents a form for gathering user credentials when the user's session times out.
7. In the **Redirect URL** field, enter a URL to which the user is redirected when the user session times out. This field is not available when the **Use PegaRULES Timeout** option is selected.
8. Click **Save**.

What to do next: [Mapping additional operator information for custom or Kerberos authentication services](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Creating an authentication service](#)
- [About HTML rules](#)

Mapping additional operator information for custom or Kerberos authentication services

In addition to the standard properties that you map for custom or Kerberos authentication, you can map to additional properties in Pega Platform.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
2. Click the **Mapping** tab.
3. In the **Attribute Name** field, enter the name of an attribute used in your authentication process.
4. In the **Property Name** field, enter the name of a property to store the attribute value. You can map to a Single Value Or Value List property that holds a text string.
5. To map additional properties, click the **Add item** icon and enter their names.
6. Click **Save**.

- [Authentication services](#)
- [More about authentication services](#)
- [Creating an authentication service](#)

Testing login authentication services

You can test and debug an authentication service in a development or staging environment by setting the appropriate log level.

1. Set the logger to debug. For performance and security reasons, do not use this setting in a production environment.
 - For a SAML authentication service, the logger is `com.pega.pegarules.integration.engine.internal.util.PRSAMLv2Utils`.
 - For an OIDC authentication service, the loggers are:
 - `com.pega.pegarules.integration.engine.internal.auth.oidc.NimbusOIDCClientHandler`
 - `com.pega.pegarules.integration.engine.internal.auth.oidc.OIDCClientHandler`
2. **Optional:** For an OIDC authentication service, get the claim values by using remote tracing.
 - a. Open a browser window to the application login screen and proceed to the login screen for your identity provider. Do not log in yet.
 - b. In a separate browser window for Dev Studio, do the following steps:
 1. In the developer toolbar, click **Tracer**.
 2. From the **Tracer** dialog, click **Settings**, and under **Pages to Trace**, enter `D_pzSSOAttributes`. Click **Add**, and then click **OK**.
 3. From the **Tracer** dialog, click **Remote Tracer**, select the ID of the unauthenticated requestor, and click **OK**.
3. Attempt to log in by authenticating with the identity provider.
4. Examine the console log by clicking **Configure System Operations Logs Log files** and selecting the Pega log.
5. Use a third-party tool to decode the Base64-encoded assertion from the log.
 - [Logging Level Settings tool](#)
 - [Selecting a session to trace](#)

Configuring login policies such as multi-factor authentication, CAPTCHA, and attestation

You can make user authentication more secure by defining login policies for password requirements, multi-factor authentication, lockout policies, and other similar restrictions.

- [Using the login policies settings](#)

Enable security policies for user authentication and session management to improve application security. You can control the strength of user IDs and passwords, manage session time-outs and the disabling of operator IDs, control the auditing of login events, and implement CAPTCHA and multifactor authentication.
- [Customizing CAPTCHA presentation and function](#)

Pega Platform provides CAPTCHA validation on login as one of the options on the Security Policies landing page. A CAPTCHA (or Reverse Turing Test) creates a challenge that is easy enough for a human user to meet, but which is likely to defeat standard automated software.
- [Multi-factor authentication with a one-time password](#)

Pega Platform supports two-factor authentication by sending a one-time password (OTP) to a user through email. The user must enter this one-time password in your Pega Platform application for verification.
- [Understanding Requestor Type data instances](#)
- [Application ID data instances](#)

Security policies settings

To authenticate users and manage sessions, configure security policies.

Password policies settings

Policy	Notes	Default value	Min value	Max value
Minimum operator identifier (ID) length		8	3	64
Minimum operator password length		8	3	64
Minimum numeric [0-9] characters required in operator password		1	0	64
Minimum alphabetic [a-z, A-Z] characters required in operator password		1	0	64
Minimum lowercase [a-z] characters required in operator password		0	0	64
Minimum uppercase [A-Z] characters required in operator password		0	0	64
Minimum special characters required in operator password	Available special characters include: ` ~ ! @ # \$ % ^ & * () _ + - = { } [] \ : " ; ' < > ? , . /	1	0	64
Maximum unique historical operator passwords	If the value is 5, you cannot change your password to match any of the most recent five passwords that you used.	5	0	128
Maximum operator password age in days	The maximum number of days before an operator must change the password. If you set the value to 0, the password never expires. To set an expiration period for a password, select a value between 1 and 128.	30	0	128
Minimum operator password age in days	The minimum number of days before an operator can change the password.	0	0	128
Minimum number of different characters between current and new operator passwords	Minimum number of characters that should be different between the current password and the new password when changing the password.	0	0	64

CAPTCHA policies settings

Policy	Notes	Default value	Min value	Max value
CAPTCHA implementation	If set to Default, the system displays the CAPTCHA implementation that is included with Pega Platform.	Default		
	If set to Custom, the system displays the custom CAPTCHA implementation enabled for this system. An application can use third-party CAPTCHA solutions on the application login screen. However, a certain amount of developer work is required to prepare the custom ruleset to deliver the third-party resource.			
	If enabled, the system displays the CAPTCHA upon authentication fails, with			

Enable CAPTCHA Reverse Turing test module	a probability set by the following field. Notes	Disabled	Min value	Max value
	If disabled, no CAPTCHA is displayed, even on login failure.			
Probability that CAPTCHA will be presented upon authentication failure (%)	If the CAPTCHA reverse Turing test is enabled, the percentage set here is the likelihood that the CAPTCHA is displayed.	5	0	100
Enable presentation of CAPTCHA upon initial login	If enabled, the CAPTCHA is displayed the first time that the user tries to log in to a new system or from a new computer.	Disabled		

Lockout policies settings

Policy	Notes	Default value	Min value	Max value
Enable authentication lockout penalty	Set to Enabled or Disabled . If enabled, after a specified number of failed login attempts, the system imposes a lengthening delay of a number of seconds between every unsuccessful login attempt and the next login attempt.	Disabled		
Failed login attempts before employing authentication lockout penalty	(Used when Enable authentication lockout penalty is Enabled .) After the set number of failed attempts, the user experiences a delay after each further attempt. The delay gets longer with each attempt.	5	0	128
Initial authentication lockout penalty in seconds	(Used when Enable authentication lockout penalty is Enabled .) Set the initial delay time.	8	0	128
Track login failures duration in minutes	(Used when Enable authentication lockout penalty is Enabled .) After this period of inactivity, the failed login counter is set back to 0.	60		
Failed login attempts before password lockout	(Used when Enable authentication lockout penalty is Disabled .) Set the number of allowed failed login attempts before the account is locked.	0		
Password lockout duration in minutes	(Used when Enable authentication lockout penalty is Disabled .) Set the time period for which the account remains locked after the allowed failed login attempts are exceeded. - Set to a non-zero value to automatically unlock the account after the lockout. - Set to zero to require the account to be unlocked manually from the Unlock Operators landing page.	0		

Audit policies settings

Policy	Notes	Default value	Min value	Max value
Audit log level	Set the Audit log level. The options are: None — No log entry is added Basic — Record failed login attempts only Advanced — Record failed and successful login attempts	Basic		

Multi-factor authentication policies (using one-time password)

Policy	Notes	Default value	Min value	Max value
Maximum one-time password failure attempts	Set the allowed number of failed login attempts before the one-time password becomes invalid and another one-time password must be generated.	3	1	3
Maximum age of one-time password token in seconds	Specify how long a current one-time password can be used to authenticate the user before it becomes invalid and another one-time password must be generated. Note: The maximum age of the one-time password token must be less than the shortlived requestor time-out period, which is defined in minutes in the prconfig setting <i>timeout/requestor/shortlived</i> , and which defaults to 1 minute. If you set the maximum age to be greater than one minute (or accept the default), you must increase the <i>timeout/requestor/shortlived</i> setting.	180		
Validity of one-time password confirmation in minutes	Specify how long a current one-time password confirmation is valid before another one-time password confirmation is required for further transactions in that session.	60		
Email account from which one-time password needs to be sent	Specify an email account that is used to send one-time passwords. If necessary, click the Add icon to edit the selected email account. If you do not define an email account or an SMS account, the multifactor authentication policy is not applied.			
SMS account from which one-time password needs to be sent	Specify an SMS account that is used to send one-time passwords. If necessary, click the Add icon to add the SMS account. If you do not define an SMS account or an email account, the multifactor authentication policy is not applied.			

Policy	Notes	Default value	Min value	Max value
--------	-------	---------------	-----------	-----------

Operator disablement policy settings

Policy	Notes	Default value	Min value	Max value
Number of days of inactivity	Specify how many days a user has to be inactive before being automatically disabled.	90	1	90
Exclusion list of operator IDs	Show a list of operators who are excluded from the policy. Administrator@pega.com is excluded by default. Click Add Operator to exclude additional operators from the policy.			

- [Enforcing policies from the Security Policies landing page](#)
- [Security policies](#)
- [Using the login policies settings](#)
- [Configuring system settings](#)

Security policies

To define security policies for user authentication and session management, use the **Security Policies** tab.

You must have the *pzViewAuthPoliciesLP* privilege to view the **Security Policies** tab. This privilege is part of the *PegaRULES:SecurityAdministrator* role.

Access this tab from the Dev Studio header by clicking **Configure System Settings Security Policies**.

- Select **Enable frequently required policies** to enable and configure password settings, CAPTCHA functionality, lockout settings, and logging levels for auditing logins.
- Enable and configure two-factor authentication.
- Enable and configure automatic disablement of inactive operators.
- Click **Display Audit Log** to display audit logs related to login attempts.
- Click **View History** to see a report of changes to security settings, including the date, the operator who made the change, and what change was made.

- [Using the login policies settings](#)
- [Security events Log](#)
- [Configuring system settings](#)

Using the login policies settings

Enable security policies for user authentication and session management to improve application security. You can control the strength of user IDs and passwords, manage session time-outs and the disabling of operator IDs, control the auditing of login events, and implement CAPTCHA and multifactor authentication.

Before you begin: To manage security policies, you must have the *pzViewAuthPoliciesLP* privilege, which is included in the *PegaRULES:SecurityAdministrator* role. The password, lockout, audit, and operator ID disablement security policies are supported in offline-enabled applications. Multi-factor authentication policies are applied only when two-factor authentication is used in custom authentication policies and in application case flows. The operator disablement policy is not enforced unless the Disable Dormant Operators agent is enabled.

1. In Dev Studio, click **Configure Org & Security Authentication Security Policies**.
2. Configure the following policies:
 - **Password policies**
 - **CAPTCHA policies**
 - **Lockout policies**
 - **Audit policy**
 - **Multi-factor authentication policies (using one-time password)**
 - **Operator disablement policy**

For more information about security policies settings, see [Security policies settings](#)

3. Click **Submit**.

- [Security policies](#)
- [Configuring system settings](#)
- [Pega-RulesEngine agents](#)

Customizing CAPTCHA presentation and function

Pega Platform provides CAPTCHA validation on login as one of the options on the Security Policies landing page. A CAPTCHA (or Reverse Turing Test) creates a challenge that is easy enough for a human user to meet, but which is likely to defeat standard automated software.

For most implementations the default CAPTCHA provided in Pega Platform will provide satisfactory service. However, if you already use another CAPTCHA service (perhaps one your team has developed) in other applications, and would prefer to deploy it for PRPC, you can do so following the steps below. You can also adjust the display of the default CAPTCHA to better match the need of both system security and valid users who wish to log in.

To use any of the options below, you need to create a ruleset and an access group to serve unauthenticated visitors to the site. You use the same ruleset for CAPTCHA refinements and for updates to the images on the login screen.

- [Fine-tuning the display and function of the CAPTCHA default](#)

You can enable or disable the presentation of a CAPTCHA appears to those attempting to log in. You can fine-tune the details of the CAPTCHA, including how many characters drawn from what character set and what sort of background for the image.

- [Substituting another service for the default CAPTCHA](#)

Pega Platform uses the SimpleCaptcha service by default. You can choose to use another third-party service, or a solution you have developed in-house.

- [Substitute a third-party service](#)

Third-party CAPTCHA services provide extensive user guidance: review the documentation of the service you wish to use.

- [Substitute a solution developed in-house](#)

If your team has developed its own CAPTCHA solution, you can use that instead of the default SimpleCaptcha.

Fine-tuning the display and function of the CAPTCHA default

You can enable or disable the presentation of a CAPTCHA appears to those attempting to log in. You can fine-tune the details of the CAPTCHA, including how many characters drawn from what character set and what sort of background for the image.

1. In the navigation of Dev Studio, click **Configure System Settings Security Policies**.
2. On the Security Policies landing page, find the *pyGenerateCaptcha* activity and save a copy with the same name into the ruleset you created for unauthenticated visitors.
3. Expand **step 1** to display the Java code.

4. Make the adjustments you want and
5. Click **Save**.
6. Make sure that, on the *Security Policies* gadget, the **CAPTCHA implementation** policy is set to Default.

Although you have modified the CAPTCHA, you are still using the default CAPTCHA provider.

- Some typical adjustments to the CAPTCHA presentation include:
 - Modify the character set: add characters to, or remove them from, this section of the code:

```
char[] DEFAULT_CHARS = new char[] { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9' };
```

- Change the **number of characters displayed** by modifying the bolded number in this section of the code:

```
nl.captcha.text.producer.DefaultTextProducer textProd = new nl.captcha.text.producer.DefaultTextProducer(6, DEFAULT_CHARS );
```

- Change the **color of the font** by modifying the color specified in the bolded text in this line of the code:

```
DEFAULT_COLORS.add(new java.awt.Color(0,51,102));
```

- For more examples of modification that are available,, visit the SimpleCaptcha website,http://simplecaptcha.sourceforge.net/custom_images.html.
- You can achieve any of the effects demonstrated on the SimpleCaptcha website. However you must use fully-qualified names in Pega Platform to refer to SimpleCaptcha classes. For example, the website demonstrates creating a CAPTCHA using a complex background and Chinese characters.
- The SimpleCaptcha website provides the following code to achieve that CAPTCHA:

```

    Captcha captcha = new Captcha.Builder(200, 50)
        .addText(new ChineseTextProducer(6)) // Answer will be 6 Chinese characters
        .addBackground(new GradiatedBackgroundProducer())
        .addNoise(new StraightLineNoiseProducer())
        .gimp(new FishEyeGimpyRenderer())
        .addBorder()
        .build(); // Required.
```

- To get the same effect in Pega Platform, modify the code by using fully-qualified names, and add it to *pyGenerateCaptcha*:

```

    nl.captcha.Captcha captcha = new nl.captcha.Captcha.Builder(200, 50)
        .addText(new nl.captcha.text.producer.ChineseTextProducer(6)) // Answer will be 6 Chinese characters
        .addBackground(new nl.captcha.backgrounds.GradiatedBackgroundProducer())
        .addNoise(new nl.captcha.noise.StraightLineNoiseProducer())
        .gimp(new nl.captcha.gimpy.FishEyeGimpyRenderer())
        .addBorder()
        .build(); // Required.
```

Substituting another service for the default CAPTCHA

Pega Platform uses the SimpleCaptcha service by default. You can choose to use another third-party service, or a solution you have developed in-house.

In either case, you need to take the following steps:

1. Locate the following rules and save them without changing their names into the ruleset you created for unauthenticated visitors:
 - *pyCustomCaptchaHead*
 - *pyCustomCaptchaBody*
 - *pyValidateCustomCaptcha*
 - *pyCustomCaptchaConnector*
2. Customize the rules as indicated below to serve your custom CAPTCHA solution.
3. In the Security Policies gadget, make sure the *CAPTCHA implementation policy* is set to Custom and that CAPTCHA is enabled.

Substitute a third-party service

Third-party CAPTCHA services provide extensive user guidance: review the documentation of the service you wish to use.

Here is how to implement Google's reCAPTCHA service (<http://www.google.com/recaptcha>) on your Pega Platform login page in place of SimpleCaptcha.

1. Obtain an account for the third-party service. As reCAPTCHA is now part of Google, you must use a (free) Google account to access reCAPTCHA.
2. Create a reCAPTCHA key.
By default, the reCAPTCHA key is restricted to a domain you associate; however, this poses problems as the domain for Pega Platform in development may be different from its domain in the testing environment and from where it is deployed in production. Choose a **global key** that is not tightly tied to a domain.
 1. The key is in two parts, a public key and a private key. The public key is used in the code in the Pega Platform login screen; the private key is what Pega Platform shares with the reCAPTCHA server.
3. Customize the required rules:
 1. *pyCustomCaptchaHead*:

This HTML fragment rule contains the scripts that are embedded in the <head> tag of the login page (Web-login.htm). For reCAPTCHA add the following:

```

<style>

#recaptcha_widget {
background: none repeat scroll 0 0 #FFFFFF;
border: 1px solid #97C7E7;
margin: 16px;
padding: 10px;
}

#recaptcha_widget a {
margin: 5px 0px;
color: #31669A;
text-decoration: none;
}

#recaptcha_widget a:hover {
text-decoration: underline;
}

</style>
<script type="text/javascript">
var RecaptchaOptions = {
theme : 'custom',
custom_theme_widget: 'recaptcha_widget'
};

</script>
```

Refer to the reCAPTCHA website to see the other themes you can choose instead of "custom".

2. *pyCustomCaptchaBody*:

This HTML fragment rule contains the custom code to send the public key to the third-party server. For reCAPTCHA add the following. Include the registered public key you received in the two lines indicated:

```

<div id="recaptcha_widget" style="display:none">
<div id="recaptcha_image"></div>
<div class="recaptcha_only_if_incorrect_sol" style="color:red">Incorrect please try again</div>
<span class="recaptcha_only_if_image">Enter the words above: </span>
```

```

<span class="recaptcha_only_if_audio">Enter the numbers you hear:</span>

<input type="text" id="recaptcha_response_field" name="recaptcha_response_field" />

<div style="padding: 10px; text-align: center; font-family: tahoma; font-size: 11px;"><a href="#">Reload CAPTCHA | <a href="#">Switch Type (audio)" class="recaptcha_only_if_image">Get a
</div>
</div>

<script type="text/javascript" src="http://www.google.com/recaptcha/api/challenge?k=YourPublicKey"> </script>

<noscript>
<iframe src="http://www.google.com/recaptcha/api/noscript?k=YourPublicKey" height="300" width="500" frameborder="0"></iframe>
<br />
<textarea name="recaptcha_challenge_field" rows="3" cols="40"> </textarea>
<input type="hidden" name="recaptcha_response_field" value="manual_challenge">
</noscript>

```

3. pyValidateCustomCaptcha:

When the user enters login credentials and a CAPTCHA answer and clicks the Login button, the system passes the parameters submitted on the login screen, plus other parameters the service requires, to this activity. For reCAPTCHA, the additional parameters are **recaptcha_challenge_field** and **recaptcha_response_field**, referenced in *pyCustomCaptchaBody*. The activity connects to the third-party server to verify whether the user's response matches the CAPTCHA challenge. If they match, the activity sets the value of *pyIsValidCaptcha* to **true**; otherwise the activity sets it to **false**. On the **Parameters** tab, select *pyIsValidCaptcha*:

1. On the **Steps** tab in **Step 1**, set a parameter and four properties in the class *Code-Security* to hold connector-related information.

Property	Value
param.pyFailMessage	""
<i>pyPrivateKey</i>	Provide the registered private key you obtained.
<i>pyResponseCaptcha</i>	param.recaptcha_response_field
<i>pyChallenge</i>	param.recaptcha_challenge_field
<i>pyRemoteip</i>	<i>pxRequestor.pxReqRemoteAddr</i>

2. Provide the private key you obtained from the third-party service as the value for *.pyPrivateKey*.
3. In **Step 2**, invoke *pyCustomCaptchaConnector* (see below):
4. In **Step 3**, set *pyIsValidCaptcha*, making sure the precondition (*pyUserIdentifier* is present) is enabled:

4. pyCustomCaptchaConnector:

The connector connects to the reCAPTCHA service to verify the CAPTCHA answer. This is required for reCAPTCHA; other third-party solutions may have differing requirements. On the **Service** tab, fill in the fields as follows:

Field	Value
Endpoint URL	Provide the URL your third-party service specifies. For reCAPTCHA, enter http://www.google.com/recaptcha/api/verify
HTTP Method	POST
HTTP Version	Select the version your third-party service specifies. For reCAPTCHA, select 1.1.
Allow Redirects	Selected
Response Timeout	0000
Status Value Property	<i>.pyStatusValue</i>
Status Message Property	<i>.pyStatusMessage</i>
Error Handler Flow	Connection Problem
Intended for	Immediate execution

On the **Request** tab, set the query string parameters:

Name	Map From	Map From Key
privatekey	Clipboard	<i>pyPrivateKey</i>
remoteip	Clipboard	<i>pyRemoteip</i>
challenge	Clipboard	<i>pyChallenge</i>
response	Clipboard	<i>pyResponseCaptcha</i>

On the **Response** tab, map the response from the service to the property *pyUserIdentifier* on the clipboard.

When all modifications are complete, the login page uses reCAPTCHA instead of SimpleCaptcha.

Your third-party CAPTCHA solution may allow for extensive customizations. For reCAPTCHA, customization information is available at <http://code.google.com/apis/recaptcha/docs/customization.html>.

Substitute a solution developed in-house

If your team has developed its own CAPTCHA solution, you can use that instead of the default *SimpleCaptcha*.

1. Override *pyGenerateCustomCaptcha* to work with the in-house CAPTCHA solution, following the example for reCAPTCHA, above.
You cannot use any other standard activity beside *pyGenerateCustomCaptcha* with the login screen, as users reaching that screen are not yet authenticated
2. Invoke *pyGenerateCustomCaptcha* from *pyCustomCaptchaBody*, as in the example above
3. Embed in *pyCustomCaptchaHead* any scripts the in-house service requires, as in the example above.
4. Enter appropriate values in *pyCustomCaptchaConnector* and *pyValidateCustomCaptcha*.

Multi-factor authentication with a one-time password

Pega Platform supports two-factor authentication by sending a one-time password (OTP) to a user through email. The user must enter this one-time password in your Pega Platform application for verification.

Two-factor authentication is supported for the following use cases:

- In custom authentication services
- In case flows, to authenticate a user before critical transactions (such as a funds transfer in excess of a certain amount)

You can configure your Pega Platform application to use two-factor authentication by completing the following steps:

1. [Configuring multifactor authentication policies.](#)
2. [Generating a one-time password.](#)

3. [Verifying a one-time password.](#)

1. [Configuring multi-factor authentication policies](#)

To control the behavior of two-factor authentication, configure the multi-factor authentication policy settings on the Security Policies landing page.

2. [Generating a one-time password](#)

You can generate a one-time password by calling an activity. Use this method to use a one-time password in an application that is running on Pega Platform version 7.3 or later.

3. [Verifying a one-time password](#)

You can verify a one-time password by calling an activity. Use this method to verify a one-time password in an application that is running on Pega Platform version 7.3 or later.

Configuring multi-factor authentication policies

To control the behavior of two-factor authentication, configure the multi-factor authentication policy settings on the Security Policies landing page.

1. In the Dev Studio header, click **Dev Studio Org & Security Authentication Security Policies**.
2. In the **Multi-factor authentication policies (using one-time password)** section, configure the following required fields:
 - a. In the **Maximum one-time password failure attempts** list, select a value between 1 and 3 to set the number of failed login attempts that your application allows before the one-time password becomes invalid and another one-time password must be generated. Setting a lower value helps prevent brute force attacks.
 - b. In the **Maximum age of one-time password token in seconds** field, enter the length of time from when the token is generated to when the user must verify it with your application.
The maximum age of the one-time password token must be less than the shortlived requestor timeout period, which is defined in minutes in the `prconfig` setting `timeout/requestor/shortlived`, and which defaults to 1 minute. If you set the maximum age to be greater than one minute, you must increase the `timeout/requestor/shortlived` setting.
 - c. In the **Validity of one-time password confirmation in minutes** field, enter how long a user can work in a single session before being logged out.
 - d. In the **Email account from which one-time password needs to be sent** field, press the Down Arrow key, and then select the name of an email account.
3. Click Submit.

What to do next: Complete the [Generating a one-time password](#) procedure.

Generating a one-time password

The `pxSendOTP` activity generates a one-time password (a fixed-length, eight-digit number), sends it to `pyEmailSettings.pyEmailAccount`, and stores it in memory until it expires. It returns a Reference ID value that identifies a specific one-time password during later verification of a user entry by the `pxVerifyOTP` activity. Input parameters for this activity must be supplied in an instance of the `Embed-OTPIInputs` class, which is supplied as the `OTPIInputs` parameter in the activity.

You can generate a one-time password in the following ways:

- By calling an activity — Choose this method to use a one-time password in an application that is running on Pega Platform version 7.3 or later. For instructions, see [Generating a one-time password by calling an activity](#).
- By using APIs — Choose this method to use a one-time password from an external system or if your application is running on a Pega Platform version earlier than 7.3. For instructions, see [Generating a one-time password by calling an API](#).

• [Generating a one-time password by calling an activity](#)

You can generate a one-time password by calling an activity. Use this method to use a one-time password in an application that is running on Pega Platform version 7.3 or later.

• [Generating a one-time password by calling an API](#)

You can generate a one-time password by calling an API. Use this method to use a one-time password from an external system or if your application is running on a Pega Platform version earlier than 7.3.

Generating a one-time password by calling an activity

You can generate a one-time password by calling an activity. Use this method to use a one-time password in an application that is running on Pega Platform version 7.3 or later.

Before you begin: Complete the [Configuring multifactor authentication policies](#) procedure.

1. Create an activity.

For more information about how to create an activity, see [Creating an activity](#).

2. On the **Pages & Classes** tab of the **Activity** form, define the page that passes the parameters to the `pxSendOTP` activity and an `OperatorID` page to access the Operator instance.
 - a. In the **Page name** field, enter the name of the page that passes parameters to the activity. **For example:** `OTPIInputs`
 - a. In the **Class** field, enter `Embed-OTPIInputs`.
 - b. Click the **Plus** icon to add a new parameter.
 - c. In the **Page** name field, enter `OperatorID`.
 - d. In the **Class** field, enter `Data-Admin-Operator-ID`.
3. On the **Steps** tab, in the first step, create a page.
 - a. In the **Method** list, press the Down Arrow key, and select **Page-New**.
 - b. In the **Step page** field, enter the name of the previously defined page that passes the parameters. **For example:** `OTPIInputs`
4. Add a second step to the activity to configure the page properties
 - a. In the **Method** field, click the Down Arrow key, and select **Property-Set**.
 - b. In the **Step page** field, enter the name of the page that passes parameters to the activity. **For example:** `OTPIInputs`
 - c. Click the **Expand** icon, and then provide the following name-value pairs in the **PropertiesName** and **PropertiesValue** fields:

<code>pyContext</code>	Enter a description of the business context for which the one-time password is being generated.
<code>pySendMode</code>	Enter EMAIL. This method is to send the one-time password to the user.
<code>pyEmailSettings.pyEmailAccount</code>	Enter the email account that is used for outbound email messages.
<code>pyEmailSettings.pyCorrName</code>	Enter the correspondence name for the email message. The <code>.pyOTP</code> property must be inserted in the correspondence content. If no value is provided for this parameter, the default <code>pyDefaultOTPCorr</code> correspondence name that is in <code>Work-</code> class is used.
<code>pyEmailSettings.pyToAddress</code>	Enter the email address to which the one-time password should be sent.
<code>pyEmailSettings.pySubject</code>	Enter a subject line for the generated email message.
<code>pyInputClass</code>	Enter your application level work class so that you are able to refer to a correspondence rule that exists in your application level work class.

5. Add a third step to the activity that calls the `pxSendOTP` activity.
 - a. In the **Method** field, enter `Call pxSendOTP`.
 - b. In the **Step page** field, enter `OperatorID`.
6. Add a fourth step to the activity that reads the Reference ID.
 - a. In the **Method** field, click the Down Arrow key, and select **Property-Set**.
 - b. In the **Step page** field, enter the name of the `OTPIInputs` page. **For example:** `OTPIInputs`
 - c. Click the **Expand** icon, and then provide the following name-value pairs in the **PropertiesName** and **PropertiesValue** fields:

PropertiesName field — Enter the name of the property that stores the Reference ID on the Primary page. For example: Primary.OTPRefID.

PropertiesValue field — Enter .pyReferenceID.

7. Click **Save**.

What to do next: Complete the [Verifying a one-time password](#) procedure.

Generating a one-time password by calling an API

You can generate a one-time password by calling an API. Use this method to use a one-time password from an external system or if your application is running on a Pega Platform version earlier than 7.3.

Before you begin: Complete the [Configuring multifactor authentication policies](#) procedure.

1. Create a REST Connector by using the REST integration wizard.

For more information about how to configure the connector, see [Creating a REST integration](#).

2. In the **Endpoint URL** field, enter the authenticate/one-time password web service on your server instance. **For example:**
`https://localhost:9090/prweb/PRRestService/api/v1/authenticate/otp/email`
3. Select the POST method to invoke the external REST service.
4. In the **Add a REST response** dialog box, in the **Content type** list, select **JSON**.
5. For the response, enter the following text:

```
{
  "Context": "Test 126",
  "EmailSettings": {
    "CorrespondenceName": "pyDefaultOTPCorr",
    "EmailAccount": "Default",
    "Subject": "OTP New",
    "ToAddress": "userid@sampledomain.com"
  }
}
```

6. Click **Save**.

What to do next: Complete the [Verifying a one-time password](#) procedure.

Verifying a one-time password

The *pxVerifyOTP* activity takes as inputs a reference to a previously generated one-time password and a user entry, and verifies that the previously generated one-time password has not expired and matches the user entry.

Input parameters for this activity must be supplied in an instance of the *Embed-OTPInputs* class, which is supplied as the *OTPInputs* parameter in the activity.

You can generate a one-time password in the following ways:

- By calling an activity — Choose this method to verify a one-time password in an application that is running on Pega Platform version 7.3 or later. For instructions, see [Verifying a one-time password by calling an activity](#).
- By using APIs — Choose this method to verify a one-time password from an external system or if your application is running on a Pega Platform version earlier than 7.3. For instructions, see [Verifying a one-time password by calling an API](#).

[Verifying a one-time password by calling an activity](#)

You can verify a one-time password by calling an activity. Use this method to verify a one-time password in an application that is running on Pega Platform version 7.3 or later.

[Verifying a one-time password by calling an API](#)

You can verify a one-time password by calling an API. Use this method to verify a one-time password from an external system or if your application is running on a Pega Platform version earlier than 7.3.

Verifying a one-time password by calling an activity

You can verify a one-time password by calling an activity. Use this method to verify a one-time password in an application that is running on Pega Platform version 7.3 or later.

Before you begin: Complete the [Generating a one-time password](#) procedure.

1. Create an activity.

For more information about how to create an activity, see [Creating an activity](#).

2. On the **Pages & Classes** tab of the **Activity** form, define the page that passes the parameters to the *pxVerifyOTP* activity and an OperatorID page to access the Operator instance.
 - a. In the **Page name** field, enter the name of the page that passes parameters to the activity. **For example:** OTPInputs
 - a. In the **Class** field, enter Embed-OTPInputs.
 - b. Click the **Plus** icon to add a new parameter.
 - c. In the **Page** name field, enter OperatorID.
 - d. In the **Class** field, enter Data-Admin-Operator-ID.
3. On the **Steps** tab, in the first step, create a page.
 - a. In the **Method** list, press the Down Arrow key, and select **Page-New**.
 - b. In the **Step page** field, enter the name of the previously defined page that passes the parameters. **For example:** OTPInputs
4. Add a second step to the activity to configure the page properties
 - a. In the **Method** field, click the Down Arrow key, and select **Property-Set**.
 - b. In the **Step page** field, enter the name of the page that passes parameters to the activity. **For example:** OTPInputs
 - c. Click the **Expand** icon, and then provide the following name-value pairs in the **PropertiesName** and **PropertiesValue** fields:

pyReferenceID — Enter the Reference ID returned by the pxSendOTP activity when it generated the one-time password.

pyOTPValue — Enter the one-time password value entered by the user.

5. Add a third step to the activity that calls the *pxVerifyOTP* activity.
 - a. In the **Method** field, enter Call pxVerifyOTP.
 - b. In the **Step page** field, enter OperatorID.
6. Add a fourth step to the activity that reads the Reference ID.
 - a. In the **Method** field, click the Down Arrow key, and select **Property-Set**.
 - b. In the **Step page** field, enter the name of the OTPInputs page. **For example:** OTPInputs
 - c. Click the **Expand** icon, and then provide the following name-value pairs in the **PropertiesName** and **PropertiesValue** fields:

PropertiesName field — Enter Primary.OTPStatus.

PropertiesValue field — Enter .pyOTPStatus.

- d. Click the **Add item** icon to add another parameter, and then provide the following name-value pairs in the **PropertiesName** and **PropertiesValue** fields:

PropertiesName field — Enter Primary.OTPStatusMessage.

7. Click **Save**.

Verifying a one-time password by calling an API

You can verify a one-time password by calling an API. Use this method to verify a one-time password from an external system or if your application is running on a Pega Platform version earlier than 7.3.

Before you begin: Complete the [Generating a one-time password](#) procedure.

1. Create a REST Connector by using the REST integration wizard.

For more information about how to configure the connector, see [Creating a REST integration](#).

2. In the **Endpoint URL** field, enter the authenticate/one-time password web service on your server instance. **For example:**
`https://localhost:9090/prweb/PRRestService/api/v1/authenticate/otp/verify`
3. Select the GET method to invoke the external REST service.
4. Click **Override** to add query parameters.
5. Enter the **Reference ID** and **OTPValue** parameters.

The **Reference ID** is returned by the *pxSendOTP* activity.

The **OTPValue** is the one-time password entered by the user.

6. Add a REST service, and enter query string parameter values for the Reference ID and OTPValue parameters.
7. Click **Submit**.

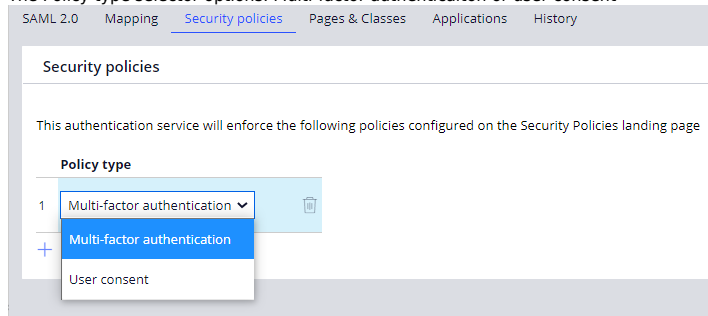
Attestation

Depending on the security requirements for your application, you may need to use attestation to do business.

Attestation is a method for software to prove its identity during normal operations. The goal of attestation is to prove to a remote party that your operating system and application software are intact and trustworthy.

The best method of implementing attestation is through Security policies, specifically using the policy type drop-down and selecting **Multi-factor authentication** or **User consent**.

The Policy type selector options: Multi-factor authentication or user consent



For more information, see:

- [Enforcing policies from the Security Policies landing page](#).
- [Security policies](#).
- [Security policies settings](#).
- [Using the login policies settings](#).

Attestations at Pega

Pega keeps pace with emerging and established international and local standards and regulations, maintaining extensive compliance certifications, attestations, and accessibility, plus third-party assessments.

Pega Platform supports the following types of attestation:

- California Consumer Privacy Act (CCPA).
- United States Food and Drug administration (FDA).
- General Data Protection Regulation (GDPR).
- Health Insurance Portability and Accountability Act (HIPAA).
- Health Information Technology for Economic and Clinical Health (HITECH).
- Privacy Shield Framework.

For more general information about these policies, see their official websites.

For more information on how these are used in Pega Platform, see:

- [Compliance with regulatory standards](#).
- [Health Insurance Portability and Accountability Act \(HIPAA\)](#).
- [General Data Protection Regulation](#).

Configuring a token credentials authentication service

After you create a token credentials authentication service, configure it so that Pega Platform uses the specified token provider for authenticating users. Select this type of service for offline mobile applications. You can map claims from the token to properties in Pega Platform, and configure optional features such as preauthentication and postauthentication activities.

Before you begin: You must complete the following task before you can configure a token credentials authentication service: [Creating an authentication service](#) Note the following best practices when you configure and deploy a token credentials authentication service.

- Derive the operator ID directly from the token that is acquired from the identity provider. Avoid using excessively complex logic for deriving the operator ID.
- To lessen the possibility of phishing attacks, do not update sensitive operator information such as mobile phone number or email address in the preauthentication and postauthentication activities.
- Do not initiate operator provisioning in the postauthentication activity.
- Deploy the authentication service over a secure channel (HTTPS enabled).

To configure a token credentials authentication service, do the following steps.

1. Create a token credentials authentication service, or open an existing service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication**

- Service and choosing a token credentials authentication service from the instance list.
 - In the **Authentication service alias** field, specify an alias to represent a unique value for this service. This value becomes the final part of the URL path for users to access Pega Platform.
 - Login URL** is a read-only field that displays the URL that accesses Pega Platform and uses this service for user authentication.
 - Optional:** In the **Provider logo** field, specify an image that represents the identity provider.
 - Select the token provider.
 - Pega Platform** – The authentication token is issued by the Pega Platform OAuth 2.0 authorization layer.
 - External identity provider** – When you select **External identity provider**, the **Identity mapping** field is displayed, where you enter the key to an identity mapping instance. Some identity mappings are linked to an appropriate token profile; for example, for a JSON Web Token (JWT), the identity mapping instance is linked to the processing token profile.
 - Optional:** Configure the optional parameters of the service.
 - [Map operator information](#)
 - [Specify preauthentication and postauthentication activities](#)
 - [Require reauthentication for new and expired sessions](#)
 - [Activate your token credentials authentication service.](#)
- [Authentication services](#)
 - [More about authentication services](#)
 - [Identity mapping](#)
 - [Creating a processing JSON Web Token profile](#)

Mapping operator information for a token credentials authentication service

You can map attributes to selected properties and pages used by your preauthentication and postauthentication activities or by other Pega Platform features. Mapping complements the use of preauthentication and postauthentication activities and other actions.

- Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
- Click the **Mapping** tab.
- Map the attributes from your login process to properties and pages that are used by your preauthentication and postauthentication activities or elsewhere in Pega Platform.
 - You can use the following properties and pages in the **Map from** field. You can also reference custom properties and pages that are used in a login flow, and you can use the Expression Builder.

Page name	Description
<i>pxRequestor</i>	The requestor page

- You can use the following properties and pages in the **Map to** field. You can also reference custom properties and pages that are used in a login flow.

Page name	Description
<i>OperatorID</i>	Properties of the operator ID
<i>D_pyOperatorAttributes</i>	Requestor-scoped data page for caching operator attributes
<i>D_pyOperatorDeviceInformation</i>	Requestor-scoped data page for caching operator device information

- Click **Save**.

What to do next: [Specifying preauthentication and postauthentication activities for a token credentials authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring a token credentials authentication service](#)
- [Expression Builder](#)

Specifying preauthentication and postauthentication activities for a token credentials authentication service

To customize the login process, you can write activities that will be triggered before and after token profile authentication.

- Create your preauthentication and postauthentication activities. For more information, see the sample activities *pySSOPreAuthenticationActivity* and *pySSOPostAuthenticationActivity*. When you write these activities, consider the following:
 - Update the *pega Browser* requestor type's default access group to equal an access group that includes the ruleset of the preauthentication activity. The preauthentication activity must be accessible to the *pega Browser* requestor type, which is used for the unauthenticated user session. By default, this requestor type is assigned to the *PRPC:Unauthenticated* access group.
 - Create the postauthentication activity in a ruleset that is accessible to the user's default access group. The postauthentication activity must be accessible to the user who has just been authenticated.
 - Set the Applies To key part of the activities to *Code-Security*.
 - Set the authentication result in *pyAuthenticationPolicyResult* to *true* to proceed with authentication, or set to *false* to terminate the request; for example:

```
tools.getRequestor().getRequestorPage().putString("pyAuthenticationPolicyResult", "true");
```

- You can dynamically update the user's role list by using a postauthentication activity. This is useful if you want to dynamically add roles to the user context at login without maintaining multiple access groups in the database for all combinations of roles. In the postauthentication activity, update the list of roles by calling *setRoles*, as shown in the following example.

```
tools.getAuthorizationHandle().setRoles(tools, tools.getAuthorizationHandle().UPDATE_APPEND, rolesList);
```

Caution: Roles that you set in the postauthentication activity without saving the access group to the database lose their effect if the user switches applications. To dynamically add roles in a way that supports switching applications, see [How to dynamically add roles during user authentication](#).

- Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
- On the **Token credentials** tab, expand the **Advanced configuration settings** section.
- In the **Pre-authentication activity** field, enter the name of the preauthentication activity.
- In the **Post-authentication activity** field, enter the name of the postauthentication activity.
- Click **Save**.

What to do next: [Requiring reauthentication for new and expired sessions for a token credentials authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring a token credentials authentication service](#)
- [Creating an activity](#)

Requiring reauthentication for new and expired sessions for a token credentials authentication service

To add an extra layer of security, you configure Pega Platform to reauthenticate new sessions and to reauthenticate when reactivating an expired session. The time-out period is specified on the operator's access group or is managed by the application server or another external facility.

- Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
- On the **Token credentials** tab, expand the **Advanced configuration settings** section.
- Select the **Use access group timeout** checkbox to use the authentication time-out values that are specified in the user access group. This setting determines how long a user session can remain inactive before a client has to present another valid token. The token can be obtained by asking the user to log in, or it can be obtained from the identity provider. Clear the **Use access group timeout** checkbox if the time-out is managed by the application server or another external facility.
- Click **Save**.

What to do next: [Activating your token credentials authentication service](#)

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring a token credentials authentication service](#)

Activating your token credentials authentication service

After you create and configure a token credentials authentication service, enable the service for Pega Platform.

1. Open the service from the navigation panel in Dev Studio by clicking **Records SysAdmin Authentication Service** and choosing a service from the instance list.
2. On the **Token credentials** tab, select the **Enable this authentication service** check box.
3. Click **Save**.

- [Authentication services](#)
- [More about authentication services](#)
- [Configuring a token credentials authentication service](#)

More about authentication services

This page describes additional topics relevant to authentication services that are not directly referenced on the rule form.

- [Authentication services and rule availability](#)

Authentication services are data instances that are available to all requestors. Authentication services can use rules such as preauthentication activities, data pages, and data transforms that need to be available to the requestor at various times during the authentication process. You make these rules available by defining them in the appropriate rulesets.

- [Authentication services and security policies](#)

To improve security in authentication services that support security policies, you select which policies to enable by using the Security policies tab of the authentication service. You define the details of each policy, such as the minimum password length and the duration of a one-time password, on the Security Policies landing page.

- [Application URL patterns for various authentication service types](#)

Authentication services other than Custom and Kerberos use the PRAuth servlet by default. However, for backward compatibility with earlier versions of Pega Platform, you can authenticate by explicitly specifying PRServlet in the application URL instead of PRAuth, or by removing the servlet name.

- [Authentication services](#)

Authentication services and rule availability

Authentication services are data instances that are available to all requestors. Authentication services can use rules such as preauthentication activities, data pages, and data transforms that need to be available to the requestor at various times during the authentication process. You make these rules available by defining them in the appropriate rulesets.

Preauthentication

Rules that are used before authentication must be accessible to the *pega Browser* requestor type, which is used for the unauthenticated user session. By default, this requestor type is assigned to the *PRPC:Unauthenticated* access group.

Depending on your configuration, rules that can be used prior to authentication include the preauthentication activity, the data transform for operator provisioning, and the data page for credentials verification. (Despite their name, data pages are rule instances, not data instances.)

Similarly, any functionality to be accessed by requestors who have connected by using an anonymous authentication service and who have not yet authenticated themselves must be made accessible.

Update the *pega Browser* requestor type's default access group to equal an access group that includes the rulesets for any rules that are used prior to authentication.

Postauthentication

Rules that are used after authentication must be accessible to the user who has just been authenticated. These rules must exist in a ruleset that is accessible to the user's default access group.

Depending on your configuration, rules that can be used after authentication include the postauthentication activity.

- [Authentication services](#)

Authentication services and security policies

To improve security in authentication services that support security policies, you select which policies to enable by using the **Security policies** tab of the authentication service. You define the details of each policy, such as the minimum password length and the duration of a one-time password, on the Security Policies landing page.

All authentication services use the PRAuth servlet. However, for backward compatibility with earlier versions of Pega Platform, it is possible to authenticate by using PRServlet instead of PRAuth (in other words, the login URL includes /prweb/PRServlet). When PRServlet is used, security policies are enabled by using various controls on the Security Policies landing page.

For more information on URL patterns and servlet names, see [Application URL patterns for various authentication service types](#).

For authentication services, enablement of security policies occurs as described below:

- You enable specific policies from the **Security policies** tab for each authentication service, except for some that are always on, as noted below.
- The **Enable frequently required policies** check box on the Security Policies landing page has no effect.
- The **Enable CAPTCHA Reverse Turing test module** setting on the Security Policies landing page has no effect.
- The **Audit policy** on the Security Policies landing page is always in effect, as are the security alerts that are configured on the Security Event Configuration landing page.
- The **Operator disablement policy** on the **Security Policies** landing page is always in effect.
- The **User consent** policy is enabled and disabled by using the **Security policies** tab of the authentication service, but it does not appear on the Security Policies landing page.

- [Authentication services](#)
- [Security policies](#)
- [Selecting a security event to monitor](#)

Application URL patterns for various authentication service types

Authentication services other than Custom and Kerberos use the PRAuth servlet by default. However, for backward compatibility with earlier versions of Pega Platform, you can authenticate by explicitly specifying PRServlet in the application URL instead of PRAuth, or by removing the servlet name.

Web logins are in the format `https:// <hostname> : <port> / <context> / <servlet name>` . The following table shows the context and servlet portions of the URL for various authentication service types, for various releases of Pega Platform.

Authentication type	URL pattern for 8.2 and later	URL pattern prior to 8.2
Authentication service selection screen (if more than one		

Authentication type	URL pattern for 8.1 and later	URL pattern prior to 8.2
service exists), or default authentication (if only Platform Authentication Service exists)	/prweb/PRAuth (8.1 and later)	Not available prior to 8.1
Default platform authentication	See listing for Basic credentials service type	/prweb Or /prweb/PRServlet
Basic credentials and Anonymous authentication services	/prweb/PRAuth/pega	Not available
OpenID Connect and SAML 2.0 authentication services	/prweb/PRAuth/<service alias>	/prweb/PRAuth/<service alias> or for SAML in 7.X, /prweb/sso*
Custom authentication service	/prweb/<as configured in web.xml>	/prweb/<as configured in web.xml>
Kerberos authentication service	/prweb/PRWebKerberos (The default servlet name can be replaced with a different servlet name.)	/prweb/PRWebKerberos (The default servlet name can be replaced with a different servlet name.)
Container-based authentication	/prweb/PRServletContainerAuth (The default servlet name can be replaced with a different servlet name.)	/prweb/PRServletContainerAuth (The default servlet name can be replaced with a different servlet name.)

Customizing the PRAuth servlet name

To customize the PRAuth servlet name that appears in the authentication URL, add code similar to the following example to your `web.xml` file. Do not remove the existing PRAuth mapping.

```
<servlet-mapping>
  <servlet-name>PRAuth</servlet-name>
  <url-pattern>/PRAuth</url-pattern>
</servlet-mapping>
```

If you customize the PRAuth servlet name that appears in the authentication URL, you will need to create a dynamic system setting. The authorization code flow always uses *PegaAuthentication* type, therefore modifying the URL pattern means you must manually change the activity.

You will need to create a dynamic system setting with the following attributes:

Purpose - pegaauthentication/servletname.

Ruleset - Pega-IntegrationEngine.

Value - By default, this value should be PRAuth.

For more information, see [Creating a dynamic system setting](#). **Note:** If you modify the URL pattern for **PegaAuthentication** then you can update the dynamic system setting.

If you configure the PRAuth as the default servlet, then when you configure the dynamic system setting **Value** should be blank.

- [Adding an application URL alias](#)

Help your users work simultaneously in multiple Pega Platform applications by creating an URL alias. The alias, which creates a custom URL for your application, enables users to log into several applications from the same browser, while also providing your business with an address that is easy to remember.

- [Authentication services](#)
- [Adding an application URL alias](#)

Adding an application URL alias

Applicable to [Cosmos React](#) and [Theme Cosmos](#) applications

Help your users work simultaneously in multiple Pega Platform applications by creating an URL alias. The alias, which creates a custom URL for your application, enables users to log into several applications from the same browser, while also providing your business with an address that is easy to remember.

1. In the header of Dev Studio, click the name of the application, and then click **Definition**.
2. On the **Definition** tab, in the **Application URL alias** section, click **Change URL alias**.
3. In the text box, enter the application alias. **For example:** Enter `loanmgmt`. **Note:** The application URL alias must be unique to your application and limited to eight alphanumeric characters. Application URL aliases **cannot** contain periods or underscores.
4. Click **Submit**.
5. Click **Save**.

- [Application URL patterns for various authentication service types](#)

Adding custom error messages in a post-authentication activity

Default error messages in a post-authentication activity can be customized to provide users with more specific, helpful feedback. You can set custom error messages based on the post-authentication logic.

Before you begin: Before you can add custom error messages in a post-authentication activity, you must first create an authentication service. For more information, see [Creating an authentication service](#).

1. In the navigation pane of DevStudio, click **Records > Data Model > Field Value**.
2. Click **Create**.
3. In the **Label** field, enter a label for this message, for example, `customErrorMsgByMe`.
4. In the **Field Name** field, enter `pyMessageLabel`.
5. In the **Context** section, set the **Apply to** field to `@baseclass`, and then click **Create and open**.
6. In the **To** field, enter the custom error message that you want to display when this error occurs.
7. Click **Save**.
8. In the navigation pane of Dev Studio, click **Records > SysAdmin > Authentication Service**, and then click the authentication service that you previously created.
9. In the **Advanced Configuration Settings** section, you can select the previously created custom activity. If you have not created one, you could create a new one based on the OOTB activity `pySSOPostAuthenticationActivity`.
10. Click the **Configure** icon next to the **Post-authentication activity** field, and then

if creating a new activity, click	Create and open
if copying the OOTB activity, click	Save As

11. Set the **Method** field to `Java`.
12. Click the **Method Parameters** section, and then in the **Java Source** field, set the value of `pyPostAuthErrorFieldLabel` to `tools.getParameterPage().putParamValue("pyPostAuthErrorFieldLabel", "<custom field value name>")`. **For example:** `tools.getParameterPage().putParamValue("pyPostAuthErrorFieldLabel", "customErrorMsgByMe")`

Authenticating requests in services

You can configure Pega Platform to access external systems to retrieve data and perform application processing. Similarly, you can allow external systems to access services in Pega Platform. By communicating with external systems, you can make use of functionality that has already been configured, and avoid the need to duplicate the same functions in multiple applications.

The following topics describe the authentication features of Pega Platform. For information about the topology of relationships with external systems, see [Integrating your application with external resources](#).

- [Authenticating requests in connectors](#)

You can configure Pega Platform to access external systems to retrieve data and perform application processing. Similarly, you can allow external systems to access services in Pega Platform. By communicating with external systems, you can make use of functionality that has already been configured, and avoid the need to duplicate the same functions in multiple applications.

- [Creating an authentication profile](#)

Create an authentication profile to move messages securely to and from your application with a variety of connector and server rules.

- [Setting up an OAuth 2.0 client registration](#)

Configure an OAuth 2.0 client registration data instance to allow an external application or mobile native application to access Pega Platform REST services over HTTPS.

- [OAuth 2.0 Management Services](#)

You can manage the token-based security of user sessions provided in the OAuth 2.0 service packages. OAuth 2.0 is a protocol that grants a third-party web site or application access to the user's protected resources, without necessarily revealing their long-term credentials or even their identity.

- [OAuth 2.0 Scope support when Pega acts as the OAuth2 Provider](#)

- [Understanding authorized access tokens](#)

Authorized access token (AAT) is now the default token format used in Pega Platform for OAuth 2.0 access tokens. AAT replaced the previously used opaque tokens.

- [Creating and configuring an OAuth 2.0 provider](#)

To enable your application to securely access an external application over HTTPS, create an OAuth 2.0 provider data instance.

- [Creating an identity mapping data instance](#)

If you use OAuth 2.0 Client Registration instances that authenticate users through a SAML 2.0 Assertion, JSON Web Token, or custom source, you need to specify how the Pega server identifies an operator and how to map the user identity information for use in the Pega application.

- [Integrating your application with external resources](#)

- [Mitigating common security vulnerabilities](#)

Authenticating requests in connectors

You can configure Pega Platform to access external systems to retrieve data and perform application processing. Similarly, you can allow external systems to access services in Pega Platform. By communicating with external systems, you can make use of functionality that has already been configured, and avoid the need to duplicate the same functions in multiple applications.

The following topics describe the authentication features of Pega Platform. For information about the topology of relationships with external systems, see [Integrating your application with external resources](#).

- [Integrating your application with external resources](#)

- [Mitigating common security vulnerabilities](#)

Creating an authentication profile

Create an authentication profile to move messages securely to and from your application with a variety of connector and server rules.

1. In the navigation pane of Dev Studio, click **Records Security Authentication Profile**.
2. In the list of authentication profiles, click **Create**.
3. In the **Type** list, click the profile type:
 - To pass basic HTTP authentication credentials, select **Basic**.
 - To pass NT LAN Manager credentials, select **NTLM**.
 - To use OAuth 1.0 authentication, select **OAuth 1.0a**.
 - To use OAuth 2.0 authentication, including support for the on-behalf-of (OBO) flow and JWT bearer grant type for Microsoft Azure and other providers, select **OAuth 2.0**.
 - To use Amazon Web Services (AWS) security credentials, for example, for S3 repository storage, select **Amazon Web Services (AWS)**.
 - To use Microsoft Azure storage account access, select **Microsoft Azure**.
4. Enter a name and short description.
The name must begin with a letter.
5. Click **Create and open**.
6. Configure your selected authentication profile.
7. Click **Save**.

- [Configuring a Basic authentication profile](#)

Configure a basic authentication profile so that messages sent to and from your application use basic HTTP authentication credentials.

- [Configuring an NTLM authentication profile](#)

Configure an NTLM authentication profile so that messages sent to and from your application use NT LAN Manager credentials.

- [Configuring an OAuth 2.0 authentication profile](#)

Configure an OAuth 2.0 authentication profile to secure messages that your application sends and receives, by using an OAuth 2.0 token-based authorization process. You define OAuth 2.0 authentication profiles for Pega Platform to securely access external REST service APIs with access tokens.

- [Configuring an Amazon Web Services authentication profile](#)

Configure an Amazon Web Services (AWS) authentication profile so that messages sent to and from your application use AWS authentication.

- [Potential problems with keystores when using AWS KMS](#)

You can integrate Pega Platform with your private Amazon Web Services Key Management Service (AWS KMS) account to manage the keys that encrypt and decrypt BLOBs and property values. Some problems can occur but they can be mitigated or prevented with the proper configuration.

- [Configuring a Microsoft Azure authentication profile](#)

Configure a Microsoft Azure authentication profile so that messages sent to and from your application use Microsoft Azure authentication.

- [Authentication profiles](#)

- [OAuth 1.0 clients](#)

- [OAuth 2.0 providers](#)

Authentication profiles

Create an authentication profile to securely authenticate your application to an external system via an integration connector, such as Connect REST.

You can specify authentication profile data instances on the **Service** tab of the following rules:

- [Connect CMIS](#)
- [Connect dotNet](#)

- [Connect HTTP](#)
- [Connect JMS](#)
- [Connect REST](#)
- [Connect SAP](#)
- [Connect SOAP](#)

The types of authentication services that can be used in all of the authentication profiles are: Basic, NTLM, OAuth 1a, OAuth 2.0, Amazon Web Service (AWS), and Microsoft Azure.

Additionally, you can specify authentication profile data instances on the **Environment** tab of the *FTP Server* rules. **Note:** Amazon Web Services authentication profiles and Microsoft Azure authentication profiles are supported only by Amazon S3 and Azure BLOB storage repositories; these authentication profiles are not supported in connectors.

Use the **Records Explorer** to list the authentication profile data instances in your system.

The *Data-Admin-Security-AuthenticationProfile* class contains authentication profile data instances.

- [Creating an authentication profile](#)

Configuring a Basic authentication profile

Configure a basic authentication profile so that messages sent to and from your application use basic HTTP authentication credentials.

Before you begin: You must complete the following task before you can configure an authentication profile: [Creating an authentication profile](#).

1. Open an authentication profile from the navigation panel by clicking **Records Security Authentication Profile** and selecting a basic profile from the instance list.
2. In the **User name** field, enter a valid user name for the scheme. The default property for this field is *pyUserName*.
This field also supports the Global Resource Settings syntax (`=PageName.PropertyName`).
3. Click **Set password**, enter a password that is valid for the user name you provided, and then click **Submit**. The default property for this field is *pyPassword*.
This field also supports the Global Resource Settings syntax (`=PageName.PropertyName`).
4. In the **Realm** field, enter the realm of the user account on the server to which you are connecting.
5. In the **Host Name** field, enter the host name of the server to which the request is being sent.
6. Select the **Preemptive authentication** check box to have the system send the basic authentication request before the server gives an unauthorized response.
Selecting this check box reduces the overhead of making the connection.
7. Click **Save**.

- [Configuring dynamic references to external systems by using the Global Resource Settings feature](#)
- [Authentication profiles](#)
- [Creating an authentication profile](#)

Configuring an NTLM authentication profile

Configure an NTLM authentication profile so that messages sent to and from your application use NT LAN Manager credentials.

Before you begin: You must complete the following task before you can configure an authentication profile: [Creating an authentication profile](#).

1. Open an authentication profile from the navigation panel by clicking **Records Security Authentication Profile** and selecting an NTLM profile from the instance list.
2. In the **User name** field, enter a valid user name for the scheme. The default property for this field is *pyUserName*.
This field also supports the Global Resource Settings syntax (`=PageName.PropertyName`).
3. Click **Set password**, enter a password that is valid for the user name you provided, and then click **Submit**. The default property for this field is *pyPassword*.
The password field also supports the Global Resource Settings syntax (`=PageName.PropertyName`).
4. In the **Domain** field, enter the domain of the user account that you specified in the **User name** field.
5. In the **Host Name** field, enter the host name of the server to which the request is being sent.
6. Click **Save**.

- [Configuring dynamic references to external systems by using the Global Resource Settings feature](#)
- [Authentication profiles](#)
- [Creating an authentication profile](#)

Configuring an OAuth 2.0 authentication profile

Configure an OAuth 2.0 authentication profile to secure messages that your application sends and receives, by using an OAuth 2.0 token-based authorization process. You define OAuth 2.0 authentication profiles for Pega Platform to securely access external REST service APIs with access tokens.

Before you begin: Add an authentication profile that you want to configure. For more information, see [Creating an authentication profile](#). The OAuth 2.0 provider is an authorization server that provides token-based authorization so that Pega Platform can access external web service APIs. When you use OAuth 2.0, Pega Platform, the provider negotiates a token that Pega Platform uses to access the external application for a defined period. To learn how to obtain the values for the configuration fields on this tab, see the API guide of the authorization server (OAuth 2.0 provider).

1. In the navigation pane of Dev Studio, click **Records Security Authentication Profile**.
2. In the list of instances, select an OAuth 2.0 authentication profile.
3. In the **OAuth 2.0 provider** list, click the name of an OAuth 2.0 provider. You can select from various well-known providers, as well as **Custom**.
4. In the **Grant type** list, select the grant type that defines how to request an access token from the OAuth 2.0 provider:
 - To configure your application to obtain client credentials from the authorization service, select **Client credentials**.

With these credentials, you can access protected resources in the external application for a defined period.

- To configure your application to submit an authorization code request to the external application to which you want to connect, select **Authorization code**.

The authorization server of the external application validates the credentials in the request, and then authenticates you with your user credentials to produce an authorization code. You exchange the code to obtain an access token from the authorization server, which grants you access to your protected resources on the external application. Use this grant type when your application requires manual intervention to authorize access to the protected resources of the external application.

- To configure your application to exchange your user credentials for an access token, select **Password credentials**.

Use this grant type when your application has a trusted relationship with the API provider or when the API provider does not support other authorization grants.

- To configure your application to use a JWT bearer token to request an access token from the authorization server. **JWT bearer**.
5. Enter the credentials to access OAuth 2.0-protected resources in the external application:
 - a. In the **Client identifier** field, enter the client ID that the authorization server provides.
 - b. In the **Client secret** field, enter the client secret that the authorization server provides.
 - c. **Optional:** To define the client scope, in the **Scope** field, enter the scope that is specified in the API guide of the external application and that is configured for this client.
Separate multiple scopes with a space.
 - d. **Optional:** If the service provider supports a refresh token, select the **Use refresh token if available** check box.
The refresh token automatically obtains a new access token when the current token expires.
 - e. **Optional:** To revoke all access tokens that you obtain from the external OAuth 2.0 provider for this OAuth 2.0 client, click **Revoke access tokens**.
 6. **Optional:** To enable single sign-on (SSO) for Box, perform the following actions:
 - a. Select the **Enable SSO** check box.
 - b. In the **Single sign-on (SSO) identity provider (IdP) federation ID** field, enter the SSO identity provider federation ID for the Box account.
You can get this value from the `entityID` attribute of the `EntityDescriptor` element of the IdP metadata XML file with SSO configuration for Box.
This field supports the Global Resource Settings syntax (`=PageName.PropertyName`).
 7. For the JWT bearer grant type, in the **JWT assertion** section, identify the source of the JWT for generating the access token request:
 - To use the JWT from the OAuth 2.0 identity provider from the OpenID Connect identity provider you use to SSO into your Pega application, select **Use token from active SSO session**.

- To use a JWT generation profile, in the **JWT generation template** field, press the Down arrow key to select a JWT generation profile, or click the **Open** icon to create a JWT profile.
 - To use a property reference, in the **Property reference** field, enter the name of the property that stores the encoded JWT.
8. For the password credentials grant type, in the **Password credentials** section, enter the login details:
 - a. In the **Username** field, enter the name of the property that references the value for the username.
 - b. In the **Password** field, enter the name of the property that references the value for the user password.
Ensure that you encrypt the password.
 9. In the **Access token endpoint** field, enter the access token URL that is specified in the API guide of the application to which you want to connect.
Calls to this endpoint produce an access token that you can use to access protected resources in the external application for a defined period.
 10. **Optional:** To enter parameters for an access token or to refresh a token request, click **Add access token and refresh token parameters**, enter parameter names and values, and then click **OK**.
 11. **Optional:** In the **Revoke token endpoint** field, enter the revoke token URL that is specified in the API guide of the application to which you want to connect.
 12. **Optional:** To add request parameters, click **Add parameters**, and then enter the parameter names and values.
Calls to this endpoint revoke access tokens for your application that functions as an OAuth 2.0 client.
 13. For the authorization code grant type, configure the code:
 - a. In the **Authorization code endpoint** field, enter the authorization code endpoint URL that is specified in the API guide of the authorization server to which you want to connect.
 - b. To add parameters, click **Add parameters**, and then enter the parameter names and values.
Calls to this endpoint produce an access token that you can use to access protected resources in the external application for a defined period.
 - c. In the **Redirect URI endpoint** field, specify that the target endpoint redirects to an authority with a URI that exposes a datapage using the Global Resource Settings Syntax (=PageName.PropertyName).
 14. In the **Advanced configuration** section, in the **Client authentication** subsection, select an authentication scheme to use for sending the client credentials in the access token request.

Choices	Actions
Client secret through the Basic scheme	a. In the Authentication scheme list, select Client secret . b. In Method , select Basic . This option sends the client credentials (client ID and client secret) to the external application as a part of the authorization header.
Client secret through the POST request	a. In the Authentication scheme list, select Client secret . b. In Method , select POST . This option sends the client credentials in the body of the POST request.
JSON web token through a JWT generation profile	a. In the Authentication scheme list, click Private Key JSON Web Token (JWT) . b. Select Use JWT generation profile . c. In the JWT generation profile field, press the Down arrow key to select a JWT generation profile, or click the Open icon to create a JWT profile.
JSON web token through a property reference	a. In the Authentication scheme list, click Private Key JSON Web Token (JWT) . b. Select Use property reference . c. In the Property reference field, enter the name of the property that contains the JWT.

15. In the **Send access token as** section, select how to send the access token to the external application:
 - To allow applications, such as mobile apps, to use third-party tokens to authenticate with a server, select **Authorization header**
 - To allow applications to authenticate requests when you want to express a request entirely in a URL, select **Query string parameter**
16. In the **Secure protocol configuration** section, in the **Lowest allowable SSL/TSL version** list, select the SSL or TLS protocol version that you want to use to securely connect to the external application by using OAuth 2.0.
17. **Optional:** To increase authentication security, in the **Security settings** section, configure keys and certificates:
 - a. In the **Truststore** field, press the Down arrow key to select a truststore, or click the **Open** icon to create a truststore.
The truststore contains the server certificate to use in the TLS/SSL handshake.
 - b. In the **Keystore** field, press the Down arrow key to select a keystore, or click the **Open** icon to create a keystore.
The keystore stores the Pega Platform client's private/public key pair the server uses to authenticate the client.
18. Click **Save**.

- [Authentication profiles](#)
- [OAuth 2.0 providers](#)
- [Configuring dynamic references to external systems by using the Global Resource Settings feature](#)
- [Creating an authentication profile](#)

Configuring an Amazon Web Services authentication profile

Configure an Amazon Web Services (AWS) authentication profile so that messages sent to and from your application use AWS authentication.

Before you begin: You must complete the following task before you can configure an authentication profile: [Creating an authentication profile](#). **Note:** Amazon Web Services authentication profiles are supported only by Amazon S3 BLOB storage repositories; this authentication profile is not supported in connectors.

1. Open an authentication profile from the navigation panel by clicking **Records Security Authentication Profile** and selecting an AWS profile from the instance list.
2. In the **Access key ID** field, enter the access key.
3. In the **Secret access key** field, enter the secret access key.
The value you enter is masked by default. To see the value as you type, select the **Provide secret access key by reference in clear text** check box.
4. Click **Save**.

- [Authentication profiles](#)
- [Creating an authentication profile](#)

Potential problems with keystores when using AWS KMS

You can integrate Pega Platform with your private Amazon Web Services Key Management Service (AWS KMS) account to manage the keys that encrypt and decrypt BLOBs and property values. Some problems can occur but they can be mitigated or prevented with the proper configuration.

When integrating Pega Platform with your private AWS KMS, you can create, delete, and control the keys that are used to encrypt your data. By using AWS KMS, you do not need to implement a site-specific cipher to encrypt your data.

When you use AWS KMS, the following problems can occur:

- New Keystore instance is required for key changes
- Unavailability of the customer master key in some situations
- Switching between cipher types can result in data loss

New Keystore instance required for key changes

If a customer master key (CMK) in the keystore is compromised and you want to change it, create a new Keystore instance, and then reference it on the Data Encryption landing page. When you update the Data Encryption landing page, the following changes happen:

- A new customer data key (CDK) is generated for new encryption requests.
- The previous CDKs are encrypted under the new CMK in the new keystore, which you can use to decrypt previously encrypted data.

Unavailability of the customer master key

If the CMK is disabled, deleted, or pending deletion (the deletion process can take from 7 to 30 days), the key is not available. Depending on the reason why the CMK is not available, you can do one of the following solutions:

- If the customer master key (CMK) is disabled, reenable it on your AWS account.
- If the key has a pending deletion status, cancel the deletion request, and enable the key.
- Create a Keystore rule form with active CMKs. Update the Data Encryption landing page with the new keystore name. This action initiates a request to reencrypt the data by using a different master key.

Switching between cipher types

Do not switch between cipher types because you can lose previously encrypted data. **Caution:** If you switch between cipher types and delete the AWS KMS encryption keys, or if a custom cipher class becomes unavailable, you cannot decrypt and reencrypt all your previously encrypted data. For information about how to safely reencrypt data, contact Global Customer Support.

Configuring a Microsoft Azure authentication profile

Configure a Microsoft Azure authentication profile so that messages sent to and from your application use Microsoft Azure authentication.

Before you begin: You must complete the following task before you can configure an authentication profile: [Creating an authentication profile](#). **Note:** Microsoft Azure authentication profiles are supported only by Azure BLOB storage repositories; this authentication profile is not supported in connectors.

1. Open an authentication profile from the navigation panel by clicking **Records Security Authentication Profile** and selecting a Microsoft Azure profile from the instance list.
2. In the **Endpoint protocol** list, select the endpoint protocol:
 - **HTTPS**
 - **HTTP**
3. In the **Account name** field, enter the account name.
4. In the **Account key** field, enter the account key.
The value you enter is masked by default. To see the value as you type, select the **Provide secret access key by reference in clear text** check box.
5. Click **Save**.

- [Authentication profiles](#)
- [Creating an authentication profile](#)

Setting up an OAuth 2.0 client registration

Configure an OAuth 2.0 client registration data instance to allow an external application or mobile native application to access Pega Platform REST services over HTTPS.

Pega Platform can act as both an OAuth 2.0 provider and an OAuth 2.0 client when you use the client credentials grant type. In this grant type, a Pega application can get an access token for its own account, unrelated to a specific user.

To use OAuth 2.0 to protect your REST services, select OAuth 2.0 as the Authentication type in the *service package* to which the REST services belong. For more information, see [Service Package form - Completing the Context tab](#).

Note: Pega Platform does not support the optional scope parameter that is a part of the OAuth 2.0 specification for endpoints other than Userinfo. The default access group for the OAuth 2.0 operator is the default scope.

- [Creating and configuring an OAuth 2.0 client registration](#)

Create an OAuth 2.0 client registration data instance to allow an external application or mobile native application to securely access Pega Platform REST services over HTTPS.

- [OAuth 2.0 client registrations](#)

OAuth 2.0 client registrations

The OAuth 2.0 protocol allows mobile native applications and external applications such as Facebook and Google to communicate securely with Pega Platform over HTTPS. You define OAuth 2.0 client registration data instances to allow external applications to access Pega Platform REST services by using access tokens.

An OAuth 2.0 client data instance is an external application that requests access to Pega Platform. When Pega Platform and the client use OAuth 2.0, they negotiate a token that allows the client to access Pega Platform for a defined period.

The grant types supported by Pega Platform OAuth 2.0 clients are authorization code, client credentials, password credentials, Security Assertion Markup Language (SAML) bearer assertion, and JSON Web Tokens (JWT) bearer assertion. You can use these grant types alone or in combination with each other.

Note: Use OAuth 2.0 to protect the REST services for which the consumer is an application and not an individual user. For example, if an insurance company wants to create new claims adjustment cases, the insurance application can make Pega API REST calls to create the cases. If the Pega API REST services are protected by OAuth 2.0, the client application developer registers the client and uses an access token to make Pega API REST calls. Pega Platform authorizes the application by using the operator ID that is associated with the client during registration, not the operator ID that is associated with the developer.

- [Creating and configuring an OAuth 2.0 client registration](#)
- [Setting up an OAuth 2.0 client registration](#)
- [OAuth 2.0 providers](#)

Creating and configuring an OAuth 2.0 client registration

Create an OAuth 2.0 client registration data instance to allow an external application or mobile native application to securely access Pega Platform REST services over HTTPS.

1. In the Dev Studio header, click **Create Security OAuth 2.0 Client Registration**.
2. Enter a short description and client name for your OAuth 2.0 client registration data instance. Begin the name with a letter, and use only alphanumeric characters.
3. Click **Create and open**. **Result:** The **Client Information** tab is populated with the following information:
 - **Client ID** – The unique ID that is assigned to the client.
 - **Client secret** – The password that is assigned to the client to authenticate with the Pega Platform authorization module.
 - **Authorization endpoint** – The endpoint when you use the Authorization code grant type to configure Pega Platform as an access token provider.
 - **Access token endpoint** – The request URL that you can access to receive an access token for the client as a response after authentication. Your REST service authenticates the client by using the client ID and client secret.
 - **Token revocation endpoint** – The request URL that you can access. After authentication, you can revoke an already issued access token for the client. When the token is revoked, the system returns an HTTP status code 200 as a response, and all subsequent requests that the client sends by using the token result in an error.
 - **Userinfo endpoint** – The endpoint for requesting user profile information.
4. **Optional:** View and manage the client access tokens.
 - To see the client secret, click **View & download**, and then click **Download credentials** in the dialog box to save the client credentials in a .txt file.
 - To generate a new password for the client, click **Regenerate client secret**. The system revokes any active access token that was issued for the client.
 - To revoke all the access tokens for the client, click **Revoke access token**.
5. In the **Supported grant types** section, select one or more grant types that define how an application gets an access token.
 - **Authorization code** – Select this check box to use an authorization code grant type for web and mobile apps, and then enter the following:
 - **List of redirect URIs** – Enter one or more URIs that provide an authorization code that is used to request an access token.
 - **Enable refresh token** – Select this check box to get a refresh token along with an access token from the token endpoint. Use the refresh token to obtain a new access token when your previous access token has expired.
 - **Enable proof key for code exchange** – Select this check box to use the PKCE extension for additional security in mobile applications.
 - **Caution:** As a best practice, do not configure a loopback URL as a redirect URI. If you configure a loopback URL, then at run time, the port number is not validated, and the client application can use any available port on the user's system, including ports that you might not intend to be used.
 - **Client credentials** – Select this check box to choose an operator that your application uses to authorize the OAuth 2.0 client, and then enter the following:
 - **Operator ID** – Press the Down Arrow key to select a unique ID that is assigned to the operator. **Caution:** This operator must not represent a human user who can log in to the system. The operator must be reserved for an OAuth 2.0 client that wants to access resources in Pega Platform. You can modify the scope of the operator's access group to allow the client to access selective resources in Pega Platform.
 - **Password credentials** – Select this check box to use user credentials to authorize the OAuth 2.0 client, and then enter the following:

- **Identity mapping** – Press the Down Arrow key to select an identity mapping that validates the operator credentials and converts them to the OAuth 2.0 access token. The default is the identity mapping data instance named *pyDefaultIdentityMappingForPasswordGrant*. Click the **Open** icon to create a new identity mapping.
 - **Enable refresh token** – Select this check box to get a refresh token along with an access token from the token endpoint. Use the refresh token to obtain a new access token when your previous access token has expired.
 - **SAML bearer** – Select this check box to use a SAML assertion to authorize the OAuth 2.0 client in your application, and then enter the following:
 - **Identity mapping** – Press the Down Arrow key to select an identity mapping that uses a SAML 2.0 assertion, or click the **Open** icon to create a new identity mapping. This extracts user information from the SAML assertion and validates it to issue the OAuth 2.0 access token.
 - **JWT bearer** – Select this check box to use a JSON Web Token assertion to authorize the OAuth 2.0 client, and then enter the following:
 - **Identity mapping** – Press the Down Arrow key to select an identity mapping that uses a JSON web token, or click the **Open** icon to create a new identity mapping. This extracts the user information from the JSON web token (JWT) claims and validates it to issue the OAuth 2.0 access token.
 - 6. In the **Token lifetime** section, define the lifetimes for authorization code and tokens.
 - a. **Authorization code lifetime (in seconds)** – Enter the time period in seconds for which an authorization code issued to an OAuth 2.0 client is valid. The default value is 600 seconds.
 - b. **Access token lifetime (in seconds)** – Enter the time period in seconds for which an access token issued to an OAuth 2.0 client is valid. The default value is 3600 seconds.

For **Authorization code grant type**, the access token lifetime will always be taken from the **Client registration rule** form, even though the refresh token value taken from IdP is less than the configured value in the rule form.

For **SAML and JWT bearer grant type**, the access token lifetime will be the minimum value of assertion validity and the access token value configured in the client registration rule form.

 - c. **Refresh token lifetime (in seconds)** – Enter the time period in seconds for which a refresh token issued to an OAuth 2.0 client is valid. The default value is 86400 seconds.
 - 7. To configure the information returned from the Userinfo endpoint as a Userinfo token, in the **User profile mapping** section, in the **JWT generation profile** field, press the Down Arrow key and select a token profile. The default is *pyDefaultUserInfoMapping*, which is the JSON web token (JWT) generation profile that represents the default Userinfo token structure. To define a different JWT generation profile, click the **Open** icon.
 - 8. Click **Save**.
 - 9. To specify a proxy for your REST connector, in the **Advanced configuration** section, select the **Use proxy** check box, and then configure the remaining proxy parameters:
 1. In the **Host** field, enter the address of the proxy host.
 2. In the **Port** field, enter the port number.
 3. To add in the authentication details, select the **Use Authentication** check box, and:
 - In the **User ID** field, enter the user ID.
 - In the **Password** field, enter the password.
- [OAuth 2.0 client registrations](#)
 - [Identity mapping](#)
 - [Token profile data instance](#)
 - [Operators](#)

OAuth 2.0 Management Services

You can manage the token-based security of user sessions provided in the OAuth 2.0 service packages. OAuth 2.0 is a protocol that grants a third-party web site or application access to the user's protected resources, without necessarily revealing their long-term credentials or even their identity.

OAuth 2.0 REST endpoints

The OAuth 2.0 service package defines a set of Representational State Transfer (REST) service endpoints. An endpoint is typically a Uniform Resource Identifier (URI) on a web server. Pega Platform supports three REST endpoints: keys endpoints, token introspection endpoints, and token deny list endpoints, which are described in the following table:

OAuth 2.0 REST endpoint types

Endpoint type	Purpose	Service endpoint	HTTP method
Keys endpoint	Used to get the public keys that are exposed in the JSON Web Key Set (JWKS) format and to perform signature validation of the authorized access token (AAT) that Pega Platform issues as the OAuth 2.0 provider.	<code>https://<host>:<port>/prweb/PRRestService/oauth2/v1/token/keys</code>	GET
Token introspection endpoint	Used to validate the access token.	<code>https://<host>:<port>/prweb/PRRestService/oauth2/v1/token/introspect</code>	POST
Token deny list endpoint	Used to get the list of denied tokens or to add tokens to the deny list.	<code>https://<host>:<port>/prweb/api/oauth2/v1/token/denylist</code>	GET and POST

For more information about HTTP methods, see [Service REST methods](#).

Keys endpoint

The keys endpoint publishes the public keys that you can use to validate the signature of access tokens that Pega Platform issues as an OAuth 2.0 provider. These public keys are exposed in the JSON Web Key Set (JWKS) format.

The OAuth 2.0 provider issues access tokens in the authorized access token (AAT) format. This token is a signed JSON Web Token (JWS).

For signing, Pega Platform uses a 4096-bit length RSA key-pair. The RSA key pair contains both private and public keys. The private key is used for signature generation of the JWT token, while the public key is used for signature verification of the token.

Key rotation

The default RSA key pair is valid for 15 days. After 15 days, Pega Platform generates a new RSA key-pair for JWT token signing. The keys endpoint provides both old and new RSA public keys in the JWK format.

You can customize the key rotation from anywhere between 1 to 365 days by using the *AccessToken/KeyRotationInterval* dynamic system setting in the *Pega-IntegrationEngine* ruleset. Pega Platform does not include the dynamic system setting by default.

For more information about creating this dynamic system setting, see [Creating a dynamic system setting](#).

The following table presents the key parameters of the keys endpoint:

Keys endpoint settings

Setting	Value
Cryptographic algorithm	4096-bit RSA key pair
Default key rotation period	15 days
Key rotation period range	1 to 365 days
Short description: Should be descriptive and easy to read, for example, AAT key rotation	

Setting	Value
Dynamic system setting	Setting purpose: <i>AccessToken/KeyRotationInterval</i> . Owning ruleset: <i>Pega-IntegrationEngine</i> . Value: Integer value with the desired key rotation time period.

Service endpoint: `https://<host>:<port>/prweb/PRRestService/oauth2/v1/token/keys`

For example:

The following list shows a sample request and response:

- Sample request: `GET /keys`
- Sample response:

The response is JSON Web Key Set (JWKS).

```
{
  "keys": [
    {
      "kty": "RSA",
      "e": "AQAB",
      "use": "sig",
      "kid": "8928ce9c4eb8f1f02e57cd435cb7a794",
      "alg": "RS256",
      "n": "kVgSt5nVKt5OXRwJBAlgHZ1oBkm8MQhxp-TUG0t9CsHFeu8yDMRDINqJZgwjQPjhc7pjkTt1b0xnxjQEGs6ezxXwBpSmtysLQzuAkTd4TQnIY9OVq78Q9DO0TpTWVhaHYnsGoTswQNyTPsgF9WypCqMX1t10PVvArMR6GXbJuUi5cY7topV",
    },
    {
      "kty": "RSA",
      "e": "AQAB",
      "use": "sig",
      "kid": "02611e4a4cdad7b3861939df5f8cdb76",
      "alg": "RS256",
      "n": "oKeBjxPbIG__GZVt7NsCz5kOJIGudlqE-ZO36sTBBElcwnOBKE_nPbbqPoNMxkb8tDt52_drm1OCFRrarraydhZvvqaiM9hGX3_OTVZBWXD_i1JnUomtZlBe-foyW-330CVEumlpGHiXdA9BKkXYb5W0CBb7vDdJP6AE4vUTDqBdSPFWJX85",
    }
  ]
}
```

Token introspection endpoint

The token introspection endpoint validates the AATs and refresh tokens that the Pega Platform OAuth2.0 provider issues. The endpoint response is either `true` or `false` for the active response parameter. The introspection endpoint supports only the POST method. By default, the endpoint is protected by basic authentication; therefore, you must provide client credentials (client identifier and secret) as an authorization header. You provide the input token in a POST body parameter.

Service endpoint: `https://<host>:<port>/prweb/PRRestService/oauth2/v1/token/introspect`

For example:

The following list shows a sample request and response for POST introspection:

- Sample request: `POST /introspect`**Content-Type:** `application/x-www-form-urlencoded`**Authorization:** Basic `czZCaGRSa3F0MzpnWDFmQmF0M2JW`**Request body:** `token=mF_9hgh.B5f-4hghgh.1JqMhghghgh`
- Sample response:

Response Code: 200

```
{
  "active": true
}
```

Token deny list endpoint

The token deny list endpoint supports GET and POST methods. The GET method retrieves deny listed tokens from the server. You can use the POST method to add tokens to the deny list. The *pzCanAccessBlacklistedTokenService* privilege is required to use this endpoint. This endpoint is protected by OAuth2.0, and you must provide bearer access token in the authorization header.

Service endpoint: `https://<host>:<port>/prweb/PRRestService/oauth2/v1/token/denylist`

The supported GET query parameters are *client_id*, *username*, and *revoked_after*.

For example:

The following list shows a sample request and response for GET HTTP method:

- Sample request: `GET /denylist`**Authorization:** Bearer `czZCaGRSa3F0MzpnWDFmQmF0M2JW...`
- Sample response:

HTTP status: 200

```
{
  "revoked_before": "20200719T132021.66 GMT",
  "jti": [
    "9e69ebde3cd3d78212a66346a48ceb8a",
    "78e21a3aa9c999cab6b7d64dd2ca3f82",
    .....
  ]
}
```

Pagination

The GET method supports pagination and can send up to 1000 JWT identifiers (jti or JWT ID) in a single response. For more than 1000 results, use the pagination parameters to retrieve additional JWT IDs. To get the next page of results, the request must include the *revoked_after* query parameter, and its value should be set to the *revoked_before* value from the previous response.

For example:

The following input shows sample response for pagination: `GET /denylist?client_id=67088366916174900&username=testOperator&revoked_after=20200719T132021.66 GMT`**Authorization:** Bearer `czZCaGRSa3F0MzpnWDFmQmF0M2JW...`

POST method

The POST method adds a token to the deny list based on filters that are POST body parameters. The supported POST body parameters are **client_id**, **jti**, **username**, **issued_before**, and **issued_after**.

A JWT ID provides a unique identifier for the JWT. You can use the **Revoked_before** parameter to get next page results if you have more than 1000 JWT IDs.

For example:

The following list shows a Sample request and response for POST HTTP method:

- Sample request
The sample request invalidates all tokens of a client where the *client_id* is 67088366973216174900.

POST / denylist HTTP

Content-Type: application/x-www-form-urlencoded

Authorization: Bearer czZCaGRSa3F0MzpnWDFmQmF0M2JW

Request body: client_id = 67088366973216174900
- Sample response
HTTP status: 200

{
 "jti": [
 "0c2141b68c4b7ceb888c30be6d95b218",
 "78e21a3aa9c999cab6b7d64dd2ca3f82"
]
}

Understanding authorized access tokens

Authorized access token (AAT) is now the default token format used in Pega Platform for OAuth 2.0 access tokens. AAT replaced the previously used opaque tokens.

Authorized access tokens

AATs are self-contained, compact, and digitally signed to be tamperproof.
Pega Platform manages AATs with autogenerated claims and a built-in key rotation strategy. Pega Platform uses JSON Web tokens (JWT) and JSON Web Signature (JWS) standards for managing authorized access tokens.
Note: AATs are fully backward compatible. They have the same ease of use as opaque tokens, which are used in versions of Pega Platform earlier than 8.5.

Sample AAT

The following image shows a sample AAT with information on what each part of the token contains:

Header : Algorithm & Token Type		
{		
"alg": "RS256",		/* Signing algorithm */
"typ": "JWT",		/* Standard token type for JWT */
"kid": "d8b9c97c4774d4c19e78bd755b701a80"		/* KeyID of the public key certificate for signature verification */
}		
Payload : Token Data		
{		
"iss": "urn:lu-85-hyd.eng.pegacom",		/* Identity of the system issuing the token */
"aud": "urn:12344527397556544345",		/* Identity of intended audience of the token */
"sub": "user@constellation.com",		/* Unique identity of user (or) operator */
"app_name": "SpaceTra",		/* Pega application name the token issued for */
"app_version": "01.01.01",		/* Pega application version the token issued for */
"operator_access": "SpaceTra:Users",		/* Pega access group scope of the token */
"iat": 1592996618,		/* Token issued time */
"nbf": 1592996618,		/* Time before which token can't be used */
"exp": 1592997218,		/* Token expiry time */
"jti": "b4e2d5948d84857f85b0bf5900986409"		/* Unique identifier of the token */
"csrf": "84cd8456ea7dac7cafae7e95360ce24b"		/* unique csrf token for browser based applications */
}		

- [Enhanced refresh token strategy](#)
You now have more precise control over your refresh token expiration strategy. When an OAuth 2.0 client application requests a new access token using the refresh token grant type, the Pega Platform response includes the new access token as well as the refresh token. In the Token Management section, you choose the refresh token issuance mechanism and the expiration of various tokens issued by Pega Platform.
- [Understanding dynamic client registration](#)
Use dynamic client registration (DCR) to dynamically register trusted third-party applications as OAuth 2.0 clients with Pega Platform. DCR create OAuth 2.0 clients for you, using Pega Platform defaults
- [Enhanced refresh token strategy](#)

Enhanced refresh token strategy

You now have more precise control over your refresh token expiration strategy. When an OAuth 2.0 client application requests a new access token using the refresh token grant type, the Pega Platform response includes the new access token as well as the refresh token. In the **Token Management** section, you choose the refresh token issuance mechanism and the expiration of various tokens issued by Pega Platform.

Supporting grant types

The new access token expiry time is set to the value provided in **Access token lifetime (in seconds)**.

Token Management

The **Token Management** section lets you choose the refresh token issuance mechanism.

In the **Token Management** section, the **Token issuance** setting for the **Refresh token** setting has three options from which you can choose. These are described in the following table:

Token Issuance Method	Behavior
Issue once and keep until expiry	Each time a new access token is requested using the <i>Refresh token</i> grant type, Pega Platform issues the same refresh token, with the expiry time updated to the remaining token lifetime using the value provided in Refresh token lifetime (in seconds) .
Issue a new refresh token without changing expiry	Each time a new access token is requested using the <i>Refresh token</i> grant type, Pega Platform issues a new refresh token, with the expiry time updated to the remaining token lifetime using the value provided in Refresh token lifetime (in seconds) .
Issue a new refresh token and reset expiry	Each time a new access token is requested using the <i>Refresh token</i> grant type, Pega Platform issues a new refresh token, with the expiry time reset to the value provided in Refresh token lifetime (in seconds) .

Identity Provider (IdP) session bounded refresh tokens

When refresh tokens are used in combination with the authorization code grant type and single sign on (SSO), using Pega Authentication services, you can choose to set the refresh token expiry time as the session timeout value provided by the IDP during SSO.

When **Authorization code** and **Set refresh token expiry from IdP session expiry** are selected:

- The refresh token expiry value is set to the same value as the IdP session expiry.
- The refresh token lifetime value specified in the **Token management** section of the rule form is not considered.
- [Creating and configuring an OAuth 2.0 client registration](#)
- [Understanding authorized access tokens](#)

Understanding dynamic client registration

Use dynamic client registration (DCR) to dynamically register trusted third-party applications as OAuth 2.0 clients with Pega Platform. DCR create OAuth 2.0 clients for you, using Pega Platform defaults

OAuth 2.0 client registration

There are two ways to register an OAuth 2.0 client application:

Manual registration

Manual registration is typically used when you need to create an OAuth 2.0 client and the defaults do not suit your needs.

You can manually create an OAuth 2.0 client in the navigation pane of Dev Studio, by clicking **Records Security OAuth 2.0 Client registration**.

Then click **Action Create**.

Then, you choose the settings as you go through.

For more information, see *OAuth 2.0 client registrations*.

Dynamic creation

Dynamic client creation can be used to save time in creating OAuth 2.0 clients. The defaults can be changed at any time on the OAuth 2.0 client registration form.

OAuth 2.0 clients can also be registered programmatically by calling the dynamic client registration service endpoint.

The service endpoint is `https://<host>:<port>/prweb/api/oauthclients/v1/client`.

Dynamic client registration

Pega Platform introduced new properties to support enhanced refresh token mechanism in the OAuth2 Client Registration Rule form.

The newly added properties are also supported in the dynamic client creation.

By default, in the *Authorization Code grant type* section, the **Set refresh token expiry from the IDP Session expiry** check box is not selected.

When the **Set refresh token expiry from the IDP Session expiry** check box is selected, the *use_idp_session_expiry* is included in the payload. This value can be set to either `true` or `false`. If set to any other value, the system returns an Invalid Request (400) error. When the payload is not specified, then the default value is `false`.

OAuth 2.0 clients can also be registered programmatically by calling the DCR service endpoint. DCR itself is protected by OAuth 2.0. You must have the *pzCanManageOAuthClients* privilege enabled in order to use DCR to register third-party applications to manually registered as an OAuth 2.0 client.

Registering a client by using dynamic client registration

When you register a client using dynamic client registration, the OAuth 2.0 Client it generates has the default attributes shown in the table below:

Property	Value
Service endpoint	<code>https://<host>:<port>/prweb/api/oauthclients/v1/client</code>
HTTP Method	POST

HTTP headers

When you register a client using dynamic client registration, the OAuth 2.0 Client it generates has the default attributes shown in the table below:

Property	Value	Behavior
Accept	Application/json	
Content-Type	Application/json	
Authorization	Bearer <AccessToken>	Access token issued to the registered client privileged for dynamic client registration.

HTTP request

When you register a client using dynamic client registration, the OAuth 2.0 Client it generates has the default attributes shown in the table below:

HTTP body	DataType	Description
client_name	String	Unique name of the third-party application specified in the DCR request.
client_id	String	Unique client ID of the newly registered OAuth 2.0 client.
client_secret	String	Unique client secret of the newly registered OAuth 2.0 client.
client_secret_expires_at	String	Always returns '0' and indicates a non-expiring client secret.
redirect_uris	Array of Strings	List of URIs set as OAuth 2.0 redirect URIs for the newly registered client.
grant_Types	Array of Strings	List of OAuth 2.0 grant types configured for the newly registered client.

For example:

Example request:

```
{
  "client_name":
  "AllGrants",
  "client_description":
  "AllGrants",
  "grant_types": [
    "authorization_code",
    "password",
    "urn:ietf:params:oauth:grant-type:jwt-bearer",
    "urn:ietf:params:oauth:grant-type:saml2-bearer"
  ],

  "authorization_code": {
    "redirect_uris": [
      "https://localhost",
      "https://localhost.com"
    ],
    "issue_refresh_token":
    "false",
    "enable_pkce": "false",
    "use_idp_session_expiry": "true"
  },
  "urn:ietf:params:oauth:grant-type:jwt-bearer": {
    "identity_mapping_name":
    "testidentity"
  },
  "password": {
    "identity_mapping_name":
    "testidentity"
  },
  "urn:ietf:params:oauth:grant-type:saml2-bearer": {
    "identity_mapping_name":
    "testidentity"
  },

  "refresh_token_strategy": "issueNew",
  "pega_associated_ruleset": "validruleset"
}
```

Error handling

The following table details common errors that can occur during dynamic client registration requests:

"> "> ">

HTTP error code	Error message	Description
409	{ "error": "duplicate_client", "error_description": "Client already exists" }	A registered client already exists with the client_name parameter provided in the request.
400	{ "error": "invalid_client_metadata", "error_description": "grant_types is missing" }	Grant type details are not provided in the request.
400	{ "error": "invalid_request", "error_description": "<grant_type_name> grant type details are missing" }	The Grant type is specified in the grant_types parameter but corresponding details are not provided in the request.
400	{ "error": "invalid_request_data", "error_description": "Request parsing failed" }	A request is not in the correct JSON format. A comma (,) is missing the client_name and client_description properties.
400	{ "error": "invalid_grant_types", "error_description": "Identity profile missing in <grant_type> details" }	The grant_type details does not contain identity profile details. This error applies to the SAML bearer and JWT bearer grant types.
400	{ "error": "invalid_client_metadata", "error_description": "redirect_uris are not found for authorization_code grant" }	Redirect URI details are not provided in the request. This error applies to the Authorization code grant type.

HTTP error code	Error message	Description
400	<pre> { "error": "invalid_request", "error_description": "Invalid issue_refresh_token value" } </pre>	The issue_refresh_token value is not either true or false. This error applies to the authorization_code and password grant types.
400	<pre> { "error": "invalid_request", "error_description": "Invalid refresh_token_strategy value" } </pre>	The refresh_token_strategy is not among: • issueOnce • issueNew • issueNew_ResetExpiry
400	<pre> { "error": "invalid_request", "error_description": "Invalid use_idp_session_expiry value" } </pre>	The use_idp_session_expiry value is not either true or false. This error applies only to the authorization_code grant type.
400	<pre> { "error": "invalid_client_metadata", "error_description": "pyRuleSetName-1 is either invalid RuleSet or user has insufficient privilege to use this RuleSet" } </pre>	The input to the property pega_associated_ruleset is not a valid ruleset
401	<pre> { "errors": [{ "ID": "invalid_authorization_header", "message": "Invalid Authentication Data." }] } </pre>	The authorization header is not provided in the request.
401	<pre> { "errors": [{ "ID": "invalid_token", "message": "Invalid token or expired." }] } </pre>	The invalid token is provided in the request as authorization header.

Creating and configuring an OAuth 2.0 provider

To enable your application to securely access an external application over HTTPS, create an OAuth 2.0 provider data instance.

Note: OAuth 2.0 provider instances are deprecated. OAuth 2.0 provider instances that were created prior to version 8.4 are backward compatible. As a best practice, use unified authentication profiles instead. For more information, see [Configuring an OAuth 2.0 authentication profile](#).

- In the header of Dev Studio, click **Create Security OAuth 2.0 Provider**.
- Enter a name and short description for your OAuth 2.0 provider data instance.
Begin the name with a letter, and use only alphanumeric characters.
- Click **Create and open**.
- In the **Grant types** section, select one or more grant types that define how to request an access token:
 - To configure your application to obtain client credentials from the authorization service, select **Client credentials**
 - To configure your application to submit an authorization code request to the external application to which you want to connect, select **Authorization code**
 - To configure your application to exchange your user credentials for an access token, select **Password credentials**
- In the **Authorization code endpoint** field, enter the authorization code endpoint URL you want to connect from the API guide of the application.
Calls to this endpoint produce an access token that you can use to access protected resources in the external application for a defined period.
- In the **Access token endpoint** field, enter the access token URL you want to connect from the API guide of the application.
Calls to this endpoint produce an access token that you can use to access protected resources in the external application for a defined period.
- In the **Revoke token endpoint** field, enter the revoke token URL you want to connect from the API guide of the application.
Calls to this endpoint revoke the current access tokens for your application.
- In the **Client authentication scheme** section, select how to send the client credentials:
 - To send the client credentials (client ID and client secret) to the external application as a part of the authorization header, select **Basic**.
 - To send the client credentials in the body of the POST request, select **POST**.
- In the **Send access token as** section, select how to send the access token to the external application:
 - To allow applications, such as mobile apps, to use third-party tokens to authenticate with a server, select **Authorization header**
 - To allow applications to authenticate requests when you want to express a request entirely in a URL, select **Query string parameter**
- In the **Secure protocol configuration** section, in the **Lowest allowable SSL/TLS version** field, select the SSL or TLS protocol version that you want to use to securely connect to the external application by using OAuth 2.0.
- In the **Security settings** section, in the **Truststore** field, press the Down arrow key to select a truststore, or click the **Open** icon to create a truststore.
The truststore contains the server certificate to use in the TLS/SSL handshake.
- In the **Security settings** section, in the **Keystore** field, press the Down arrow key to select a keystore, or click the **Open** icon to create a keystore.
The Pega Platform keystore stores the client's private and public key pair, which the server uses to authenticate the client machine.
- Click **Save**.

- [Web Service Security profile](#)
- [Keystores](#)
- [OAuth 2.0 providers](#)

OAuth 2.0 providers

The OAuth 2.0 protocol provides your application with secure authentication in open networks with external applications, such as Twitter, Facebook, and Google, over HTTPS. You define OAuth 2.0 provider data instances so that Pega Platform can access protected external resources by using access tokens.

Note: OAuth 2.0 provider instances are deprecated. As a best practice, use unified authentication profiles instead. OAuth 2.0 provider instances that were created prior to version 8.4 are backward compatible. For more information, see [Configuring an OAuth 2.0 authentication profile](#).

Grant types

An OAuth 2.0 provider data instance is an external application to which your application connects to negotiate a token that provides your application with access to an external application for a defined period.

The grant types that Pega Platform OAuth 2.0 providers support are client credentials, authorization code, and password credentials. You can use these grant types alone or in combination with each other.

Client credentials grant type

With the client credentials grant type, you obtain client credentials from the external application to which you want to connect. Your application can use these credentials to generate an access token. Use this grant type when the OAuth 2.0 consumer is an application and not an individual operator or user.

Authorization code grant type

With the authorization code grant type, your application submits an authorization code request to the external application to which you want to connect. The authorization server of the external application validates the credentials in the request, and then authenticates with your user credentials to produce an authorization code. Your application then uses the code to generate an access token, which grants access to protected resources on the external application. Use this grant type when your application does not have a trusted relationship with the API provider. For example, you can connect to Facebook from your Pega Platform application to access your photos.

Password credentials grant type

With the password credentials grant type, your application exchanges your user credentials for an access token. Use this grant type when your application has a trusted relationship with the API provider or when the provider does not support other authorization grants.

You can ensure that your application securely accesses external REST service APIs by creating an OAuth 2.0 provider data instance. Each external application handles the OAuth 2.0 protocol in a unique way. For more information, see the API guide for your application.

- [Creating and configuring an OAuth 2.0 provider](#)
- [Setting up an OAuth 2.0 provider](#)
- [OAuth 2.0 client registrations](#)
- [Configuring an OAuth 2.0 authentication profile](#)

Creating an identity mapping data instance

If you use OAuth 2.0 Client Registration instances that authenticate users through a SAML 2.0 Assertion, JSON Web Token, or custom source, you need to specify how the Pega server identifies an operator and how to map the user identity information for use in the Pega application.

You also need an identity mapping if you create a token credentials authentication service. Use the Identity Mapping data instance to create a token profile, and map attributes, claims, or password credentials to operator record properties to identify an operator.

1. In the navigation panel of Dev Studio, click **Records Security Identity Mapping**.
2. Click **Create**.
3. In the **Name** field, enter the name of the identity mapping profile.
4. In the **Short description** field, enter a short description of the identity mapping profile.
5. In the **Source** field, choose one of the following options to select the source of the Identity Mapping.
 - **SAML 2.0 Assertion** – Select to map operator attributes from a SAML 2.0 Assertion source.
 1. Click **Create and open**.
 2. In the **Signature verification** section, in the **Truststore** field, press the Down arrow key and select the same keystore that is used by the SAML 2.0 Assertion and that verifies the token.
 3. In the **Operator identification** section, select mapping the operator ID from either **Name identifier in the subject** or **Attribute**.
 - If you select **Attribute**, enter an expression to indicate the attribute name.
 4. In the **Attribute Mappings** section, map one or more attributes to property names. For example, enter an attribute to map the `.pyUserIdentifier` property to identify an existing operator.
 5. In the **Processing options** section, in the **Post processing activity** field, enter the name of your postprocessing activity. The postprocessing activity can connect to any external entity to get additional data, and enrich the **OperatorID** page with attributes that are available on the assertion page. To facilitate writing the postprocessing activity, you can use the `pySAMLBearerIdentityMappingAct` activity as a reference.
 - **JSON Web Token** – Select to map operator attributes from a JSON Web Token source. **Caution:** You need to create a new identity mapping instance and token profile. In the token profile, specify the Issuer and Audience claim in the Claim validation section, the Subject and Expiration Time claim in the Claims mapping section, and a keystore in the Truststore field. For more information, see [Processing a JSON Web Token](#).
 1. Click **Create and open**.
 2. In the **Token validation** section, in the **Token processing profile** field, press the Down arrow key and select the JSON Web Token profile that you created to validate the token.
 3. In the **Operator identification** section, select mapping the operator ID from either **Standard subject (sub) claim** or **Custom claim**.
 - If you select **Custom claim**, enter an expression to indicate the claim.
 - Indicate concatenation with a plus sign ("+"). Do not use @concat.
 - The value that you enter relates to the claims mapping in the processing token profile. For example, if you enter a **Token processing profile** value of `DemoJWT` and you enter a **Custom claim** value of `{test}`, then in the token profile instance for `DemoJWT`, you map the **Claim name** of `test` to a property on a clipboard page.
 4. In the **Processing options** section, in the **Post processing activity** field, press the Down arrow key and select the `pyJWTBearerIdentityMappingAct` activity that processes the JSON Web Token.
 5. Modify the `pyJWTBearerIdentityMappingAct` activity to map the claims in the **Claim mapping** and **Advanced mapping** section in the token profile to the operator page.
 - **Custom** – Select to map operator attributes from a custom source. **Caution:** For external users, modify steps 3, 4, and 5 in the `pyCustomIdentityMappingAct` activity. In addition, you must populate the `pyOperPage` parameter if it is not populated in the custom activity.
 1. Click **Create and open**.
 2. In the **Processing options** section, in the **Post processing activity** field, press the Down arrow key and select the activity name that is used to identify the operator through the password credentials.
6. Click **Save**.

- [Identity mapping](#)

Identity mapping

The Identity Mapping rule form allows you to specify how to identify an operator from a SAML 2.0 Assertion, JSON Web Token, or custom source.

When you register a user through OAuth 2.0 Client Registration, you can choose to authenticate the user with the attribute values that are provided in the SAML 2.0 Assertion or with claims that are provided in the JSON Web Token. To identify the operator, you can map the attributes from the SAML 2.0 Assertion or the claims from a JSON Web Token to the operator record in Pega Platform. You can also choose to use a custom source to identify the operator by providing password credentials as an operator.

The Identity Mapping data instance is referenced in the OAuth 2.0 Client Registration rule form.

- [Setting up an OAuth 2.0 client registration](#)

Using JNDI to specify an LDAP server when using an authentication service

You can set up an authentication service to override or extend the default Pega Platform™ authentication process. You can enter a Java Naming and Directory Interface (JNDI) entry, which represents a directory located on the LDAP server. Using JNDI enables you relocate servers without having to reconfigure the application.

The JNDI syntax is dependent upon the server environment.

Using a JNDI entry with WebSphere

Enter the name used for the JNDI lookup. This value matches the value entered as the Provider URL in the authentication server instance. For example, enter `pega/ldapURL`.

To define the JNDI entry, do the following:

1. Use the WebSphere administrative console and navigate to **Environment Naming Name Space Bindings** .
2. Specify the **Binding identifier**. For example, pega
3. Specify the **Name in name** space. This value matches the value entered as the **Directory** in the authentication service instance. For example, pega/ldapURL
4. Specify the **String value**, which is the actual URL of the directory. For example, ldap://serverx:384

Using a JNDI entry with Tomcat

Enter the full JNDI reference that corresponds to the environment entry specified in the deployment descriptor and the prefix `java:com/env`. For example, enter `java:com/env/pega/ldapURL`

Here is an example of an environment entry:

```
<env-entry>
<description><![CDATA[Generic LDAP URL entry]]> </description>
<env-entry-name>pega/ldapURL</env-entry-name>
<env-entry-value>ldap://serverX:384</env-entry-value>
<env-entry-type>java.lang.String</env-entry-type>
</env-entry>
```

Authentication time-out

When users are inactive for a certain period of time, Pega Platform requires users to reauthenticate by entering their login credentials. The browser session cannot resume until the login and password are accepted. Requiring reauthentication helps prevent a malicious or unauthorized user from hijacking the browser session. However, if reauthentication fails or is canceled, some or all of the data on the screen might continue to be displayed.

Authentication time-out is the length of time between when user activity in a browser session ceases and Pega Platform requires reauthentication. The expired browser session is still displayed during this time.

Authentication time-out configuration

You configure authentication time-out on the **Advanced** tab of each Access Group form. For more information, see [Learning about access groups](#).

Configure the authentication time-out according to your organization's security policies. Make sure that the authentication time-out is consistent with your organization's policy so that you can set how long a user's browser session can be idle before it requires reauthentication.

If your organization uses a custom authentication scheme such as single sign-on (SSO), the session time-out might be handled outside Pega Platform. In this case, compare the internal settings to the external settings. Determine the authentication time-out of your custom authentication scheme and verify that the Pega Platform authentication time-out is consistent with the external time-outs.

If authentication is handled by an external system, you can turn off the Pega Platform authentication time-out feature by leaving the authentication time-out entry blank on the **Advanced** tab of the Access Group form.

You also can set a warning message that is displayed to the user whose session is about to expire. The user's response to this message resets the user activity timer. The response to the warning renders the browser session no longer idle. To customize the message, update the localized value for the field value rule that has a field value equal to `TimeoutWarning` and a field name equal to `pyMessageLabel`.

Authentication login failures

When a user fails to authenticate with proper credentials, safeguards ensure that repeated failed attempts to authenticate have repercussions to mitigate automated attempts to gain unauthorized access to the system.

Two security measures that impact the impact of behavior of one another are excessive login attempts and lockout policies. Excessive login attempts and lockout policies are primarily involved in Pega Platform authentication, such as in *PRServlet* and the basic credentials authentication service. These policies typically do not apply to other types of *PRAuth* authentication.

You can apply the following security measures to respond to failed login attempts:

Excessive login attempts

When the number of login attempts exceeds a specific number of attempts, the system presents the user with an error page. The default value is 3, and you can adjust this value by editing the *authentication/maxLoginAttemptsCount* dynamic system setting. If a user tries to log in three times in succession, regardless of the user name that they enter, the excessive login attempt failure protocol is enforced.**Note:** The excessive login attempt count is maintained on a session (or requestor) basis. If the user fails to log in on the first try, and closes and reopens their browser, then the next login attempt is considered the first attempt because the system creates a new session after a browser restart.

Lockout policies

When you enable these settings, after a number of login failures that you define, lockout policies cause a response delay. Additionally, the delay increases with each successive failed login. The lockout penalty (response delay) occurs when the lockout limit (number of failed attempts) has been exceeded. **Note:** This policy is linked to a specific user name, not the session, as is the case with the excessive login attempts algorithm. Even after exceeding the number of login attempts that is specified in this policy, if a different user name is used in each attempt, no delay occurs. You configure these settings on the **Security policies** landing page.

For example: You can choose to not set the dynamic system setting for excessive login attempts, and use the default of three attempts instead. Additionally, you can configure the lockout policies to run after 10 failed login attempts. As a result, a failed login attempt refreshes the page and displays the login page along with a generic error message.

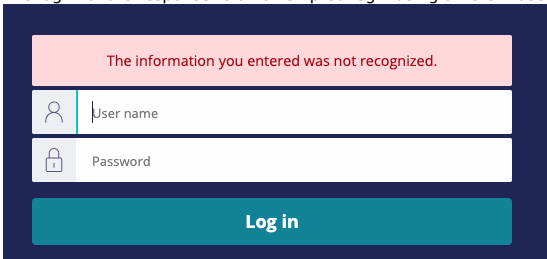
Login failure responses

Login failures use either different or the same user names on each attempt.**Note:** The validity of the user name is irrelevant to the behavior of authentication security policies.

If the user enters a different user name on each attempt, no delay occurs, and after any number of failed attempts, the login failure response is displayed, as shown in the following figure:

Login failure response to attempts with different user names

The login failure response to an attempted login using different user names



If the user enters the same user name on each attempt, the following behavior occurs:

Login failure responses to attempts with the same user name

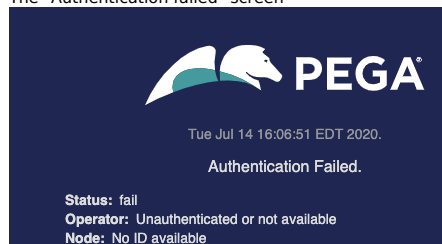
Action	Result

3 failed attempts	Error page
6 failed attempts	Error page
9 failed attempts	Error page
10 failed attempts	Initial delay on the eleventh attempt
12 failed attempts	Incremental delay, followed by error page
13 failed attempts	Delay (even though new session is established)

The following figure shows the error page response:

Authentication error page

The "Authentication failed" screen



Security tab of the Application Definition

Use this tab to define security settings in your application, map authentication services, and enable content, mashups, and digital messaging security.

Application Definition

To view the Application Definition, in the header of Dev Studio, click your application name, and then click **Definition**.

Security Tab

The Application Definition contains seven tabs, including the Security tab. To view the **Security** tab, in the header of Dev Studio, click your application name, and then click **Definition Security**.

The **Security** tab contains the following sections: **Application security**, **Authentication**, **Content security**, **Mashup security** and **Digital Messaging security**.

Application security

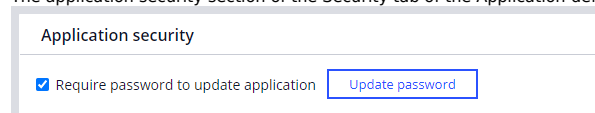
Application security has one setting: the **Require password to update application** check box, which should be selected when creating an application. The **Require password to update application** check box should remain selected.

Select the **Require password to update application** check box if you want to change or update the password that users must enter when updating the application, and then click **Update password** to set the password.

For more information, see [Setting your application password](#).

The following figure shows the **Application security** section:

The application security section of the Security tab of the Application definition



Authentication

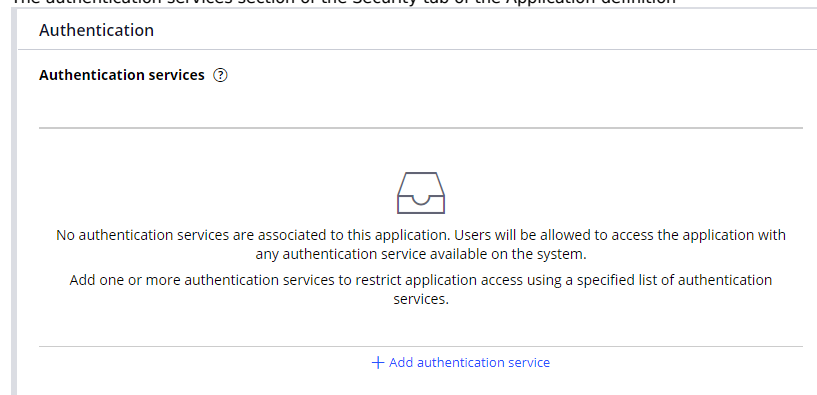
Authentication services are new to Pega Platform. Users with the *pzAdvancedSecurityUser* privilege can map authentication services to an application. By default, the *PegaRULES:SecurityAdministrator* includes the *pzAdvancedSecurityUser* privilege.

By mapping an authentication service, a security administrator can define the authentication service mechanism that users use to log in to an application. Administrators can also create a default login mechanism and interactive screen the user selects an authentication service to login.

The Authentication section changes depending on whether you have already mapped authentication services.

The following figure shows the Authentication section for a system that does not have any mapped authentication services:

The authentication services section of the Security tab of the Application definition



The following figure shows the Authentication section for a system that already has mapped authentication services:

A populated authentication services section of the Security tab of the Application definition

Authentication

Authentication services ⓘ

Name	Alias	Type	Status	
solomGoogle ⓘ	oidc-Google	OIDC	ENABLED	⋮
Okta-SAML ⓘ	saml-okta	SAML2	ENABLED	⋮
Custom ⓘ	Basic-PhantomOperator	Basic	ENABLED	⋮

+ Add authentication service

Login page configuration ⓘ

☒ Show all authentication services on the login page
☐ Log in by using a default authentication service

For more information, see [Mapping authentication services in Dev Studio](#).

You can also map authentication services in App Studio, without using the Application Definition. For more information, see [Mapping authentication services in App Studio](#).

Content security

The content security policy (CSP) is a set of directives that inform the user's browser of locations from which an application is allowed to load resources, such as fonts, images, and style sheets.

Use the **Policy name** field to select the content security policy. You can configure this content security policy directly from this page by clicking the configure icon after you select the policy.

Use the **Mode** setting to specify what the browser does when a policy is violated:

Mode	System behavior
Reject and Report	Enforce the policy and report the violation. Note: The content source must be in the correct list.
Report Only	Report the violation, but do not enforce the policy.

The following figure shows the **Content security** section:

The content security section of the Security tab of the Application definition

Content security

Policy name

Mode

☐ Reject and report
☒ Report only

For more information, see:

- [Content security policies](#)
- [Creating a content security policy](#)

Mashup security

Mashup security is used to define the external URLs that are allowed to access Pega Platform so that the host page can communicate with the mashup gadget, if you use the mashup feature to embed Pega Platform content in an external application. This feature functions, in part, as an allow list.

The following figure shows the **Mashup security** section:

The mashup security section of the Security tab of the Application definition

Mashup security

List only the origins you trust to use Pega Web Mashup gadgets from this application. Specify the host, port and protocol for each of your websites. For example: https://www.example.com:443 (note: port # is optional for well-known ports).

Trusted origin	Note
No items	

+

For more information, see [Securing your application for mashup communication](#).

Digital Messaging security

This security feature is used to set up messaging platforms for Pega Intelligent Virtual Assistant (IVA) for Digital Messaging. To start using Digital Messaging, you must configure and define the security settings for the channel and the system. Examples of messaging platforms that are useable by the IVA include Apple Business Chat, Facebook Messenger, MMS/SMS (Twilio), Twitter, and WhatsApp Messenger.

The following figure shows the **Digital Messaging security** section:

The Digital messaging security section of the Security tab of the Application definition

Digital Messaging security

Manager ID

Manager Key

For more information, see [Configuring channel security](#).

Web Service Security profile

Use a Web Service Security profile (WS-Security) to determine how to authenticate a web service message.

In the WS-Security profile, you can define a SOAP message that verifies whether the message comes from a known sender and that the message was not modified during transmission. This configuration provides content integrity and confidentiality for SOAP requests and responses.

Enable Web Services Security on a SOAP connection to securely send messages to and from your application.

- [Creating a WS-Security profile data instance](#)

Create a WS-Security profile data instance to securely exchange messages between your application and a web service. By configuring a SOAP message in your WS-Security profile, you can authenticate a web server by validating message senders and the integrity of messages in your application.

- [Defining keystores for WS-Security profile data instances](#)

Create a WS-Security profile to securely exchange messages between your application and a web service. Use the Keystore tab to specify the keystore data instance to use with the security profile.

- [Defining inbound SOAP messages for WS-Security profile data instances](#)

Create a WS-Security profile to securely exchange messages between your application and a web service. Use the In Flow tab to configure WS-Security on inbound SOAP messages.

- [Defining outbound SOAP messages for WS-Security profile data instances](#)

Create a WS-Security profile to securely exchange messages between your application and a web service. Use the Out Flow tab to configure WS-Security on outbound SOAP messages.

Creating a WS-Security profile data instance

Create a WS-Security profile data instance to securely exchange messages between your application and a web service. By configuring a SOAP message in your WS-Security profile, you can authenticate a web server by validating message senders and the integrity of messages in your application.

1. In the header of Dev Studio, click **Create Security WS Security Profile**.
2. In the **Short description** field, enter text that describes the data instance.
3. In the **Web Service security profile name** field, enter a name for the data instance.
4. Click **Create and open**.

- [Web Service Security profile](#)

Defining keystores for WS-Security profile data instances

Create a WS-Security profile to securely exchange messages between your application and a web service. Use the Keystore tab to specify the keystore data instance to use with the security profile.

1. Click the **Keystore** tab.
2. In the **Keystore** field, enter the keystore record that contains the Pega Platform private/public key pair for SSL security.
3. In the **Truststore** field, enter the keystore record that contains the SSL certificates that are trusted by Pega Platform.
4. Click **Save**.

- [Web Service Security profile](#)
- [Creating a WS-Security profile data instance](#)

Defining inbound SOAP messages for WS-Security profile data instances

Create a WS-Security profile to securely exchange messages between your application and a web service. Use the In Flow tab to configure WS-Security on inbound SOAP messages.

Note: The order of the configuration type is important. For example, if your outbound message is first signed and then encrypted, the inbound message must first decrypt the message and then check the signature.

1. On the **In Flow** tab, click the **Add new configuration** icon.
2. In the **Configuration type** list, select one of the following inbound SOAP message types.
 - **Decryption** – Enables the decryption configuration on the inbound SOAP message.
 1. In the **Encryption key identifier** list, select the encryption key to use in the SOAP message.
 2. Click **Change decryption password** and then enter the new password to change the private key password.
 3. To use symmetric key encryption, where the user and the service have a shared binary key, in the **Embedded key** field, enter the Base64 value of a binary shared key.
 4. In the **Embedded key name** field, enter the name of the shared embedded key.
 5. In the **Encryption sym algorithm** list, select the algorithm to encrypt the symmetric key.
 6. In the **Key transport algorithm** list, select the algorithm used for encrypting and decrypting the encryption key.
 - **Signature** – Enables the signature configuration type on an inbound SOAP message.
 1. In the **Signature algorithm** list, select the digital signature algorithm to use for encryption.
 2. In the **Digest algorithm** list, select a hash code that verifies that the signature came from the claimed source.
 3. In the **Signature key identifier** list, select the key identifier type to use to identify the signature token.
 - **Timestamp** – Enables the time stamp configuration type on an inbound SOAP message.
 - **Username** – Enables the user name configuration type on an inbound SOAP message.
 1. In the **User name** field, enter a user name for authentication.
 2. Click **Change password** to change or add a password associated with the specified user name.
 3. In the **Password type** list, select the type of the password to use with the connection.
 - **Text** – The password is sent as plain text in the SOAP message.
 - **Digest** – The password is sent as a Base64-encoded SHA1 hash of the original value.
 4. To change the SOAP message to a randomly generated Base64 string, select the **Add nonce value** check box.
 5. To indicate the creation time of the message by including a time stamp in the SOAP message, select the **Add created timestamp** check box.
 - **SAML** – Enables the SAML configuration type on an inbound SOAP message.
 1. In the **SAML version** list, select the SAML version to use in the SOAP message.
 2. In the **Clock skew** field, enter the time difference (in seconds) between two different servers that are out of sync.
3. Repeat steps 1 and 2 to add more configurations.
4. Click **Save**.

- [Web Service Security profile](#)
- [Creating a WS-Security profile data instance](#)

Defining outbound SOAP messages for WS-Security profile data instances

Create a WS-Security profile to securely exchange messages between your application and a web service. Use the Out Flow tab to configure WS-Security on outbound SOAP messages.

Note: You can add as many configuration types for the outbound SOAP message as you need.

1. On the **Out Flow** tab, click the **Add new configuration** icon.
2. In the **Configuration type** list, select one of the following outbound SOAP message types.
 - **Encryption** – Enables the encryption configuration on the outbound SOAP message.
 1. In the **Encryption parts** field, enter a semicolon separated list of element names to encrypt. Element names must be in the format: *{Element}{Namespace URI}ElementName*. For example, to encrypt and/or digitally sign the WS-Security *UsernameToken* element, the value would look like this: *{Element}{http://schemas.xmlsoap.org/ws/2002/07/secext}UsernameToken*
 2. In the **Encryption key identifier** list, select the encryption key to use in the SOAP message.
 3. In the **Encryption user** field, enter a certificate alias that is specified in the **Keystore** field on the **Keystore** tab in this rule form.
 4. To use symmetric key encryption, where the user and the service have a shared binary key, in the **Embedded key** field, enter the Base64 value of a binary shared key.
 5. In the **Embedded key name** field, enter the name of the shared embedded key.
 6. In the **Encryption sym algorithm** list, select an algorithm to encrypt the symmetric key.
 7. In the **Key transport algorithm** list, select an algorithm to encrypt and decrypt the encryption key.
 - **Signature** – Enables the signature configuration type on an outbound SOAP message.
 1. In the **Signature algorithm** list, select the digital signature algorithm to use for encryption.
 2. In the **Signature key identifier** list, select the key identifier type to use to identify the signature token. As a best practice, select **Issuer Name and Serial**. When you select this option, only the user name and serial number of the certificate are sent in the message; the certificate is not sent in the security header.
 3. Click **Change signature password** to change or add a password that is associated with the signature.
 4. In the **Signature user** field, enter the name of the alias listed in the **Keystore** field on the **Keystore** tab in this rule form.
 5. In the **Signature parts** field, enter a semicolon-separated list of element names to sign. Element names must be in the format: *{Element}{Namespace URI}ElementName*. For example, to encrypt and/or digitally sign the WS-Security *UsernameToken* element, the value would look like this: *{Element}{http://schemas.xmlsoap.org/ws/2002/07/secext}UsernameToken*
 - **Timestamp** – Enables the time stamp configuration type on an outbound SOAP message.
 - In the **Time to live** field, enter the amount of time in seconds, for which the SOAP message is valid.
 - **Username** – Enables the user name configuration type on an outbound SOAP message.
 1. In the **User name** field, enter a user name for authentication.
 2. Click **Change password** to change or add a password that is associated with the specified user name.
 3. In the **Password type** list, select the type of password to use for the SOAP message.
 - **Text** – Sends the password as a plain text in the SOAP message.
 - **Digest** – Sends the password as a Base64-encoded SHA1 has of value of the original value.
 4. To change the SOAP message to a randomly generated Base64 string, select the **Add nonce value** check box.
 5. To indicate the creation time of the message by including a timestamp in the SOAP message, select the **Add created timestamp** check box.
3. Repeat steps 1 and 2 to add more configurations.
4. Click **Save**.

- [Web Service Security profile](#)
- [Creating a WS-Security profile data instance](#)

Authorization

Authorization ensures that after logging in, users have access to only the features and data that they need for their work. Pega Platform offers three types of authorization: role-based access control, attribute-based access control, and client-based access control. You can use these authorization features together to provide the strictest level of control.

Authorization, or access control, is a security mechanism that determines access rights and privileges that are related to system resources, including data and application features. Authorization is normally preceded by authentication for user identity verification.

You can use authorization features in Pega Platform to determine the access rights and privileges to restrict or increase the scope of user actions.

Pega Platform offers the following basic authorization models that provide different yet complementary functionality:

- Role-based access control (RBAC)
- Attribute-based access control (ABAC)
- Client-based access control (CBAC)

Role-based access control

Role-based access control typically implements the first two sets of requirements, where based on the defined roles and privileges that are derived from the user's access group, the access level for each user might be restricted to only certain UI elements, certain actions in the UI, or a class.

Attribute-based access control

Attribute-based access control restricts access to specific instances of classes or to specific properties within instances. Access control policies enforce these restrictions by specifying the type of access that is allowed. These policies define a set of policy conditions that compare user properties or other information on the clipboard to properties in the restricted class.

For example, an access control policy for a sales automation application might restrict access to a sales account to the user who owns the account or to a user who is included in an exception list of non-owners who have access. You can express this policy condition as one of the following values:

- `OperatorID.pyUserIdentifier = Account.Owner`
- `OperatorID.pyUserIdentifier OneOf Account.AccessExceptionList`

Access control policies have the following key characteristics:

- Policies can be defined only for *Assign*-, *Data*-, and *Work*- classes that are in the database.
- Attribute-based access control policies use the full inheritance functionality. Access policy rules can be inherited from multiple classes, in which case the policy conditions from the relevant policies are combined and access is allowed only when all such policy conditions are satisfied.
- All policy conditions are applied to the operator. As a result, the access is granted only when RBAC and ABAC are both true.
- Pega Platform enforces access control policies everywhere, not just within the UI, including in these areas:
 - All reporting rules (list views, summary views, report definitions)
 - Search
 - Custom SQL that is written by developers

Client-based access control

Client-based access control helps you to satisfy the data privacy requirements of the European Union (EU) General Data Protection Regulation (GDPR) and similar regulations. CBAC rules define where to store and how to access personal data. Personal data is associated with an actual person, not with an abstract entity such as a business.

For more information about the CBAC process, see *Building a client-based access control environment*.

Key rule and data classes in RBAC and ABAC authorization models

You can use role-based access control or attribute-based access control to restrict the following types of user access and actions:

- Accessing portions of the UI (for example, harnesses and controls) and performing actions by using those components

For example, you can use a privilege to control access to a UI control such as a command button. A user might be denied access to the audit trail (**History and Attachments**) section of a case, but a manager might be granted this access.

- Creating, reading, updating, or deleting instances of any class
- Reading, updating, or deleting a specific instance of a class (instance-level or row-level security) or specific properties within that instance (property-level or column-level security)

Key security classes and relationships

Role-based access control uses the following key classes:

- Access group (*Data-Admin-Operator-AccessGroup*)
- Roles (*Rule-Access-Rolename*)
- Access of roles to objects (*Rule-Access-Role-Obj*)
- Deny rules (*Rule-Access-Deny-Obj*)
- Privileges (*Rule-Access-Privilege*)

Policy-based access control uses the following key classes:

- Access control policies (*Rule-Access-Policy*)
- Access control policy conditions (*Rule-Access-PolicyCondition*)
- [Role-based access control](#)

Use role-based access control (RBAC) to restrict users from having access to certain UI elements, to performing only certain actions in the UI, or to having any access to a class, based on defined roles and privileges that are derived from the user's access group. Access groups define the actions that groups of users can do in an application. For example, you can configure a case manager access group so that case managers can approve important cases that are not permitted for other case workers.

- [Attribute-based access control](#)

You can restrict the ability of a user to view, modify, and delete instances of classes, or properties within classes. Use attribute-based access control (ABAC) to enforce row-level and column-level security in your application. Restrict access to cases and properties by using attribute-based access control (ABAC).

- [Client-based access control](#)

If your application stores data that might be used to identify a person and you are subject to GDPR or similar regulations, use client-based access control (CBAC) to track and process requests to view, change, or remove the data.

- [Authentication](#)
- [Auditing](#)
- [Security operations](#)
- [Security](#)

Role-based access control

Use role-based access control (RBAC) to restrict users from having access to certain UI elements, to performing only certain actions in the UI, or to having any access to a class, based on defined roles and privileges that are derived from the user's access group. Access groups define the actions that groups of users can do in an application. For example, you can configure a case manager access group so that case managers can approve important cases that are not permitted for other case workers.

The basic components of RBAC are operators, access groups, and access roles. An operator ID represents a user who is permitted to log in to a Pega Platform application.

An access group is a group of permissions within an application. An operator belongs to one or more access groups, depending on what applications and functions the user can access. At any given time, one access group is in effect for a logged-in user.

An access group includes one or more access roles which define what the group can do. The same role can be used in multiple access groups.

- [Operators](#)
- [Learning about access groups](#)
- [Managing access roles](#)
- [Understanding role permissions and privileges in Access Manager](#)

Learning about access groups

An access group is a group of permissions within an application. Pega Platform uses these permissions for operators, external system access, and background processes. You define an access group for operators who have similar responsibilities. For example, most applications allow case managers to do actions that are different from the actions of regular operators, so case managers and regular operators belong to different access groups.

Access group names have the format *application name:access group name*. For example, for the MyApp application, you can define the MyApp:Administrators access group for administrators and the MyApp:Users access group for regular operators.

Operators can belong to multiple access groups. You select one of the access groups as the default, which is used when the operator initially logs in. If an operator belongs to multiple access groups, the operator can switch between groups. Only one access group is in effect at any given time during a session.

When you create an access group, you define permissions and settings that are used for operators who belong to that access group and who use the application defined for that access group. These permissions and settings include the following:

- Access roles and privileges
- The portal layout
- The work pools that are available
- The types of work items that operators can work on
- The rulesets that are displayed at the top of the ruleset list
- Details of rule caching for performance
- For developers, the initially displayed ruleset and version for rules that they create

Access groups and ruleset lists

When an operator logs in, Pega Platform looks for an access group in the following order until an access group is found, and uses that access group to assemble the operator's ruleset list:

1. The default access group defined on the **Profile** tab of the **Operator ID** form
2. The default access group for the Org Division that is identified on the **Work** tab of the **Operator ID** form
3. The default access group for the Org that is identified on the **Work** tab of the **Operator ID** form
4. The default access group for the appropriate requestor type

Access groups and external systems

An access group determines the ruleset list that is available to an external system that requests services. The following data instances and rules reference access groups directly, or indirectly by specifying an operator:

- Listener data instances
- Service package data instances
- Agent rules

- Agent schedule data instances

When effective

When you save an access group, active requestor sessions on the current node that are associated with that access group are immediately updated. Requestors at other nodes in a cluster are updated when the next system pulse occurs on their nodes.

Facilities provided to unauthenticated (guest) requestors

Guest users, or unauthenticated requestors, typically have access to only the rules in the rulesets in the *PRPC:Unauthenticated* access group, as referenced in the requestor type instance named *pega Browser*.

Caution: You should review **Security** tab of each activity in the rulesets, to verify that the **Require authentication to run** check box is not selected if you update the:

- *pega Browser* requestor type to reference a different access group, or
- *PRPC:Unauthenticated* access group to make additional rulesets available to unauthenticated users.

The clipboard for a guest requestor does not include pages for the operator ID, organization, division, or organization unit.

- [Managing access roles](#)
- [Fields for operator contact information and application access](#)
- [Understanding Requestor Type data instances](#)

Viewing access groups and operators

You can view all the access groups or view only the groups that reference an application. You can see all the access groups across the Pega Platform applications and the operators who have access to those applications.

- To view all the access groups in Pega Platform, complete the following steps.
You can also view the operators in a group.
 - In the header of Dev Studio, click **Configure Org & Security Groups & Roles Access Groups**.
 - The fields on this page allow you to learn more about access groups and operators for the current application.
 - **Search text** - Filter the list to display only those access groups with a specific text value in the name or description.
 - **View in Excel** - Click to export access group information and operator counts to Microsoft Excel file.
 - **Name** - Click the access group name to open the access group instance.
 - **Description** - A short description of the access group.
 - **Operators** - The number of operators associated with this access group as defined in the operator ID instance. Click the number to display the list of operators.
- To view only the groups that reference the current application, click **Configure Application Structure Access Groups and Users**. The names of the operators in each group are also displayed.
- To view the operators who belong to a selected access group, complete the following steps.
 - Open an existing access group from the navigation panel by clicking **Records Security Access Group** and selecting an instance.
 - On the **Operators** tab, review the operator ID instances that reference this access group. You can click a row to open an operator ID record.
The following information is displayed for each operator:
 - Full name
 - Operator ID
 - Job position
 - Date and time of last sign-on
- To view the operators who are assigned to a particular organization, division, or unit, complete the following steps.
 - Display the organizational chart by doing one of the following steps.
 - Click **Configure Org & Security Organization Organizational Chart**.
 - Open the rule form for the organization, division, or unit, and go to the **Chart** tab.
 - Select any node within the organization.
 - Right-click to open the pop-up panel and click **View Operators**.
- To view all operators, do one of the following.
 - Click **Configure Org & Security Organization Operators**.
 - In the navigation panel, click **Records Organization Operator ID**.

- [Operators](#)
- [Fields for operator contact information and application access](#)

Creating an access group

An access group is a group of application permissions that are used by an operator, external system, or background process. Create an access group to define the actions that are allowed when such an entity uses an application.

Before you begin: To create an access group, you must have the *pzCanAlterRoles* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.

- In the Dev Studio header, click **Create Security Access Group**.
- Enter a short description for this access group data instance.
- In the **Access Group Name** field, enter a name. As a best practice, use the format of an application name followed by a single colon and a description of the job or user function of the users or other requestors who belong to this access group.
Do not use an asterisk character(*), parentheses (), or spaces in an access group name. Begin the name with a letter and use only letters, digits, dash, ampersand, period, and colon characters.
For example: ConsumerLoans:Supervisor
- Click **Create and open**. **Result:** The access group is created in a ruleset for the application that you are currently logged in to. You can change the ruleset by clicking **Edit** next to the ruleset name.
- In the **Application** section, in the **Name** field, press the Down Arrow key and select the name of the application that this access group can access.
- In the **Application** section, in the **Version** field, press the Down Arrow key and select the version of the application that this access group can access.
When a user logs in, the ruleset names and ruleset versions identified in the application rule are added to the user's ruleset list.
- [Grant portal access to the access group](#).
- [Add roles to the access group](#).

- [Learning about access groups](#)

Assigning work pools to an access group

Work pools are the case types in which users in an access group are allowed to create cases. You specify the work pools that are available to an access group.

Before you begin: Complete the following task before assigning work pools to an access group: [Creating an access group](#).

- Create an access group, or open an existing instance from the navigation panel by clicking **Records Security Access Group** and selecting an instance.
- On the **Advanced** tab, in the **Name** field of the **Work Pools** section, list the work pools that are accessible to this access group.
 - In the **Name** field, press the Down Arrow key and select a work pool.
 - To add more work pools to the access group, click **Add item** and select a work pool.
 - To select the user's default work pool, click the button next to the work pool name.
As a best practice, select the work pool that contains the case types in which users that belong to this access group most often create cases. This selection determines the default work pool that appears on the top of the Application Explorer tree when you open an application.

- [Adding personas to organize users](#)

Work pools and access groups

Work pools are the case types in which users in an access group are allowed to create cases. The work pools that you add to an access group affect the user experience at run time.

- Work pools are classes that are marked as **is a class group** on the **General** tab of the class rule form. A work pool is a named collection of case types in which users that belong to the access group associated with the work pool are permitted to create cases.
 - When you define the access group, in the access group's work pool list, you can enter any class group for which the container class (the *Rule-Obj-Class* instance with the same name as the class group) belongs to a ruleset listed in the **Production Rulesets** list for the access group or that is available to users through application rules. This restriction ensures that the rules of the application are available to users associated with the access group.
 - At run time, the work pools that you add to an access group appear in the Application menu **Switch Work Pool** list of the users who belong to the access group. Work pool names appear in the order in which you list the work pools for the access group. **Tip:** If this list contains many work pools, consider reordering them to display the work pool names alphabetically or in another order that is meaningful to users.
 - Leave a work pool list empty if users who create new cases and belong to the access group associated with the work pool use a composite portal that includes the standard section *@baseclass.NewWork*. The cases that such users can create appear on the **Cases and Data** tab of the Application rule, not in the access group.
 - At run time, if you leave a work pool list empty and the Application rule is not used for work pools, the system behaves in the following manner:
 - When users that belong to the access group associated with the work pool open an application, the Application Explorer class autocomplete field is empty, and no classes appear in the tree.
 - Users that belong to the access group associated with the work pool cannot create cases. The work pool list does not affect which cases the user can update. The list determines only the types of cases that the user can create. The assignments that the user can open and update are determined by access roles and ruleset lists, not by work pools.
- [Assigning work pools to an access group](#)
 - [Learning about access groups](#)
 - [Working with class groups](#)

Granting portal access to an access group

Associate a portal with an access group to control which workspaces or web channels are available to users while they work in your application.

1. Open an access group by searching for it or by using the Records Explorer.
Tip: To open your current access group, select the **Access group** option from the **Profile** menu.
2. Click the **Definition** tab.
3. In the **Available portals** section, click **+ Add portal**.
4. In the **Name** field, press the Down Arrow key, and then select a portal.

For example, select *pxPredictionStudio* to give the data scientists on your team access to the Prediction Studio workspace.

5. **Optional:** To make this portal the default for all operators in the access group, select **Default** next to the portal name.
6. Click **Save**.
7. Log off and then log in to Dev Studio.

Result: Users who belong to the access group can switch from their current workspace or web channel to the portal that you provide.

- [Configuring portals](#)
- [Changing your workspace](#)
- [Adding a role to an access group](#)
- [Learning about access groups](#)

Configuring case type access

The Access Manager **Work & Process** tab provides information about the authorizations to operations in the application that users have, based on case types (work classes) in the application. Use Access Manager to configure the actions that access groups can do with case types.

You can modify authorization for the following user operations on instances of the expanded case type (work class):

- **Open** – View and open cases.
- **Run Reports** – Run reports that reference cases.
- **Modify** – Edit and save cases.
- **Delete** – Delete cases.
- **View History** – View case history class types.
- **Perform** – Access and work on cases of this type that are assigned to other operators.
- **Process Flows** – Run the flow, if defined for the case type.
- **Flow Actions** – View and click items in the **Action** menu.

- [Viewing authorizations for case type items in a single access group](#)

From the Access Manager you can view settings for access groups other than your own. This allows you to review the authorization settings for access groups and case type items.

- [Editing authorizations for case type items in a single access group](#)

In Access Manager you can edit and change the authorization for case type items in a single access group.

- [Viewing authorizations for case type items in all access groups](#)

From the Access Manager allows you to view the authorization settings for case type items. You cannot modify authorizations when viewing all access groups.

- [Viewing authorizations for case type flows and flow actions in a single access group](#)

Access Manager displays all flows and flow actions defined for each case type in the selected application or applications. This view allows you to review the authorization settings for flows and flow actions.

- [Editing authorizations for case type flows and flow actions in a single access group](#)

In Access Manager you can edit and change the authorization for case type flows and flow actions in a single access group.

- [Access Manager](#)

Viewing authorizations for case type items in a single access group

From the Access Manager you can view settings for access groups other than your own. This allows you to review the authorization settings for access groups and case type items.

Before you begin: To manage case types on the Work & Process tab of Access Manager, you must be logged in as an operator associated with an access group that contains the *PegaRULES:SecurityAdministrator* or *PegaRULES:BasicSecurityAdministrator* role.

You can see the following information:

- Operators in this access group who have full access to this action, flow, or flow action in the case type.
- Operators in this access group who have no access to this action, flow, or flow action in the case type.
- A combination of conditional access and possibly no access to this action, flow, or flow action among operators in the access group.

To view the authorizations for items in a case type in a single access group, complete the following steps:

1. In the header of Dev Studio, click **Configure Org & Security Access Manager Work & Process**.
2. Click **Application**.

3. Select or clear the applications that you want to see.
4. Click **Apply**.
5. Select an access group from the **Access Group** list.
6. View or edit the case type.

The page displays the case types and their access status for the access group.

 - a. Click an individual case type or role to edit it.
 - b. Expand a case type category to see authorization for each user operation in the case type.
7. To generate a PDF file that shows all the authorizations for each user operation in the case type, click **Export authorizations**.

- [Access Manager](#)
- [Configuring case type access](#)

Editing authorizations for case type items in a single access group

In Access Manager you can edit and change the authorization for case type items in a single access group.

Before you begin: To manage case types on the Work & Process tab of Access Manager, you must be logged in as an operator associated with an access group that contains the *PegaRULES:SecurityAdministrator* or *PegaRULES:BasicSecurityAdministrator* role. **Note:** You cannot edit standard Pega Platform roles.

1. In the header of Dev Studio, click **Configure Org & Security Access Manager Work & Process**.
2. Click **Application**.
3. Select or clear any applications that you want to edit.
4. Click **Apply**.
5. Select an access group from the **Access Group** list. **Result:** The page displays the case type and the access status for the access group.
6. Expand a case type to see the authorization for each user operation in the case type.
7. In the column for the access role to be authorized, click the **Access** icon.
8. Select one of the following access types.
 - **Full Access** – Grant operators with this role full access to the item.
 - **No Access** – Deny operators with this role access to the item.
 - **Conditional** – Grant access based on an Access When condition. Select an Access When condition under which an operator in the access group can access the item.
9. Click **OK**.

- [Access Manager](#)
- [Configuring case type access](#)

Viewing authorizations for case type items in all access groups

From the Access Manager allows you to view the authorization settings for case type items. You cannot modify authorizations when viewing all access groups.

Before you begin: To manage case types on the Work & Process tab of Access Manager, you must be logged in as an operator associated with an access group that contains the *PegaRULES:SecurityAdministrator* or *PegaRULES:BasicSecurityAdministrator* role.

The view shows you only the following information:

- Operators in this access group have full access to this action in the case type.
- Operators in this access group have no access to this action in the case type.
- Operators in this access group who have a combination of conditional access and possibly no access to this case type.

To view the authorizations for items in a case type in all access groups, complete the following steps:

1. In the header of Dev Studio, click **Configure Org & Security Access Manager Work & Process**.
2. Click **Application**.
3. Select or clear the applications that you want to see.
4. Click **Apply**.
5. In the **Access Group** field, select **All Access Groups**.
6. View or edit the case type.

The page displays the access groups for the application and their access status for each.

 - a. Click an individual case type or role to edit it.
 - b. Expand a case type to see authorization for each user operation in the access groups.
7. To generate a PDF file that shows all the authorizations for each user operation in the case type, click **Export authorizations**.

- [Access Manager](#)
- [Configuring case type access](#)

Viewing authorizations for case type flows and flow actions in a single access group

Access Manager displays all flows and flow actions defined for each case type in the selected application or applications. This view allows you to review the authorization settings for flows and flow actions.

Before you begin: To manage case types on the Work & Process tab of Access Manager, you must be logged in as an operator associated with an access group that contains the *PegaRULES:SecurityAdministrator* or *PegaRULES:BasicSecurityAdministrator* role. You can see the following information:

- Operators in this role have full access to this process flow, or flow action in the case type.
- Operators in this role have no access to this process flow, or flow action in the case type.
- Some combination of conditional and possibly no access to this process flow, or flow action among operators in the access group.

To view the authorizations for flows and flow actions in a case type in a single access group, complete the following steps:

1. In the header of Dev Studio, click **Configure Org & Security Access Manager Work & Process**.
2. Click **Application**.
3. Select or clear any applications that you want to view.
4. Click **Apply**.
5. Select an access group from the **Access Group** list.
6. Expand a case type to view authorizations for each user operation in the case type.
7. Click **Process Flows** or **Flow Actions** to expand the list.
8. To generate a PDF file that shows all the authorizations for each user operation in the case type, click **Export authorizations**.

- [Access Manager](#)
- [Configuring case type access](#)

Editing authorizations for case type flows and flow actions in a single access group

In Access Manager you can edit and change the authorization for case type flows and flow actions in a single access group.

Before you begin: To manage case types on the Work & Process tab of Access Manager, you must be logged in as an operator associated with an access group that contains the *PegaRULES:SecurityAdministrator* or *PegaRULES:BasicSecurityAdministrator* role. **Note:** To edit flows and flow actions, be sure that their containing ruleset is unlocked.

1. In the header of Dev Studio, click **Configure Org & Security Access Manager Work & Process**.
2. Click **Application**.
3. Select or clear any applications that you want to edit.
4. Click **Apply**.
5. Select an access group from the **Access Group** list.
6. Expand a case type to see authorizations for each user operation in the case type.

7. Click **Process Flows** or **Flow Actions** to expand the list.
8. In the column for the access role to be authorized, click the **Access** icon.
9. Select one of the following access types:
 - **Full Access** – Grant operators with this role (as well as all other operators in the selected access group) full access to the item.
 - **No Access** – Deny operators with this role access to the item.
 - **Conditional** – Grant access based on an Access When condition. Select an Access When condition under which an operator in the access group can access the item.
10. Click **OK**.

- [Access Manager](#)
- [Configuring case type access](#)

Access Manager

Use Access Manager to view and determine what authorization operators have for accessing case types, data, and tools in an application.

Access Manager allows you view and edit authorizations for users in an access group, or for all access groups of the current operator. You can display all flows and flow actions defined for each case type in the selected application or applications. Access Manager can alert you to areas in your business processes that need tighter restrictions.

To use Access Manager, you must be logged in as an operator associated with an access group containing the *PegaRULES:SecurityAdministrator* or *PegaRULES:BasicSecurityAdministrator* role.

Note: If you have the *PegaRULES:BasicSecurityAdministrator* role, you do not see the **Tools** tab.

The following tabs are available on the Access Manager landing page. Click **Configure Org & Security Access Manager** and then click one of the following:

- **Work & Process** – On the **Work & Process** tab, you can see and manage the authorization in access groups for case types, process flows, flow actions, and the actions that users can perform on them.
- **Tools** – On the **Tools** tab, you can see and manage the authorization in access groups to access built-in and custom application tools.
- **Privileges** – On the **Privileges** tab, you can create and manage privilege-based authorization for case and data types.

The **Access Manager** indicates the type of authorization for an item:

- Green check mark icon – Indicates permission is granted.
- Red X icon – Indicates permission is denied.
- Yellow circle icon – Indicates a run-time decision that is based on the logic in an access when rule.

- [Configuring case type access](#)
- [Configuring tools access](#)
- [Granting privileges to an access role](#)

Configuring tools access

The Access Manager **Tools** tab provides information about the authorizations that users have to the tools you can secure in your application. Use Access Manager to configure the actions that access groups can do with tools.

You can view all access groups in the application and see a summary of authorizations for each access group.

You can also customize the tools that appear on the **Tools** tab. For more information, see [Adding custom tools to Access Manager](#).

When you expand a tool category, icons indicate whether full access, no access, or conditional access is granted to members of the access group.

Summarized views can alert you to areas in your business processes that need tighter restrictions. You can expand the view to edit the authorizations of a role.

- [Viewing tools authorization for a single access group](#)

Access Manager allows you to view settings for access groups other than your own. You can see the following information:

- [Viewing tools authorizations in all access groups](#)

You cannot modify authorizations when viewing all access groups. The view shows you only whether:

- [Editing tools authorization for a single access group](#)

You can change the access status of a role for all the operators in the same access group.

- [Access Manager](#)
- [Application tools](#)

Application tools

The standard application tools include tools for users and managers and for administrative and development activities.

In the Access Manager, the **Tools** tab is populated by the data transform rule *pxPopulateTools*. However, this rule can apply to another data transform rule, *pyPopulateTools*, which appends other tools that you define to the bottom of the **Tools** tab.

For more information, see [Adding custom tools to Access Manager](#).

The following table provides a list of standard application tools available in the system.

Tool category	Tool name	Description or Link
User & manager		Tools intended for end users. These tools do not affect rule definitions.
Dictionary	Add words	Add words to the spell checker dictionary. When authorized, an operator can right-click a word flagged by the spell checker and add it to the dictionary. Spell checker
Reports	Provide criteria on reports	Provide selection criteria on reports. When authorized, an operator can view the Available Values tab on the Pick Values pop-up window when customizing criteria for a report.
Administrative & development		Tools intended for system architects and developers. They can affect the run-time system display and behavior for end users.
Application distribution	Download SQL during import	View and download SQL during application import for database administration. Viewing and applying schema changes
	Enable automatic schema changes	Enable automatic schema changes during application import. Viewing and applying schema changes
	Export applications	Export applications.

Tool category	Tool name	Exporting an application, product rule, or ruleset to an archive or repository by using the Export wizard Description or Link
	Lock and roll other applications	Lock and roll other applications. Other Applications tab
	Preassemble an application	Run the static assembler on demand to preassemble an application. <i>Improving response time by caching rules with the Static Assembler</i>
	Skim a ruleset	Save the highest version of a rule into a new, higher ruleset version. Rule skimming for higher ruleset versions
Clipboard	Execute activity on page	Execute an activity on a page on the clipboard. Using the Clipboard tool
Feedback gadget	Use the feedback gadget	Use the feedback gadget. Requesting feature enhancements
Message queue management	Modify JMS queue	View, add, and remove messages on a JMS queue. Understanding agent queues
	View JMS queue	View JMS message queue. For more information, see Admin Studio help.
	Modify MQ queue	View, add, and remove messages on an MQ queue. Understanding agent queues
	View MQ queue	View MQ message queue. For more information, see Admin Studio Help.
Reports	Optimize property	Use the one-click property optimization wizard. Exposing properties and populating database columns
	Optimize schema	Use the schema optimization wizard. Optimizing a schema
Rules & rulesets	Private check-out	Perform private checkout or check-in of rules. Performing a private edit
Security	Configure system security policy settings	Adjust the security policy rules for all operator IDs and passwords, mitigate brute-force attack with CAPTCHAs and audit levels. Using the login policies settings
	Rule security analyzer	Use the rule security analyzer to search through non-autogenerated rules, to find specific coding patterns that may indicate a security vulnerability. Implementing security guidelines for custom HTML
Tracer	Advanced tracer options	Use tracer breakpoints and watch variables. Configuring trace conditions
	Basic tracer options	Use settings, configure tracer output, use play and pause, and view details for any row. Tracing and capturing events
	Use remote tracer	Use remote tracer to connect to another requestor session. Selecting a session to trace
User interface	Toolbar: Preview button	An authorized operator can click the Preview button in Dev Studio. <i>Previewing rules</i>
	Toolbar: Run button	An authorized operator can click the Run button in Dev Studio. Unit testing individual rules
	Toolbar: Save As button	An authorized operator can click the Save As button in Dev Studio. Copying a rule or data instance
	Live UI tool	An authorized operator can click the Live UI tool. Live UI

- [Access Manager](#)
- [Configuring tools access](#)

Viewing tools authorization for a single access group

Access Manager allows you to view settings for access groups other than your own. You can see the following information:

Before you begin: To use the tools tab in Access Manager, you must be logged in as an operator associated with an access group that contains the *PegaRULES:SecurityAdministrator* role.

- Operators in this access group have full access to the tool.

- Operators in this access group have no access to the tool. The menu item or UI element is disabled.
- A combination of conditional access and possibly no access to this tool among operators in the access group.

1. In the header of Dev Studio, click **Configure Org & Security Access Manager Tools**.
2. Click **Application**.
3. Select or deselect the applications you want to process.
4. Click **Apply**.
5. Select an access group from the **Access Group** list. **Result:** The page displays the tools and their access status for the access group.
6. View or edit the tool category.
 - a. Click an individual access group to edit.
 - b. Expand a tool category to see the authorization for each tool in the category.

- [Access Manager](#)
- [Application tools](#)

Viewing tools authorizations in all access groups

You cannot modify authorizations when viewing all access groups. The view shows you only whether:

Before you begin: To use the tools tab in Access Manager, you must be logged in as an operator associated with an access group that contains the *PegaRULES:SecurityAdministrator* role.

- All operators in an access group are authorized to use all the tools listed in a tool category.
- No operators in an access group are authorized to use any tools listed in a tool category.
- A combination of conditional access and possibly no access to this tool among operators in the access group.

1. In the header of Dev Studio, click **Configure Org & Security Access Manager Tools**.
2. Click **Application**.
3. Select or deselect the applications you want to modify.
4. Click **Apply**.
5. In the **Access Group** field, select **All Access Groups**.
6. The page displays the access groups for the application and their access status for each tool. Expand a tool category to see the authorization for each tool in the category.

- [Access Manager](#)
- [Application tools](#)

Editing tools authorization for a single access group

You can change the access status of a role for all the operators in the same access group.

Before you begin: To use the tools tab in Access Manager, you must be logged in as an operator associated with an access group that contains the *PegaRULES:SecurityAdministrator* role.

1. In the header of Dev Studio, click **Configure Org & Security Access Manager Tools**.
2. Click **Application**.
3. Select or deselect applications that you want to process.
4. Click **Apply**.
5. Select an access group from the **Access Group** list.
6. Expand a tool category to see the authorization for each tool in the category.
7. In the column for the access role to be authorized, click the icon.
8. Select the access type.
 - **Full Access** – Grant operators with this role full access to the item.
 - **No Access** – Deny operators with this role access to the item. At run time, the system can hide or deactivate the item.
 - **Conditional** – Grant access based on an Access When condition. Select an Access When condition under which an operator in the access group can access the item.
9. Click **OK**.

- [Access Manager](#)
- [Application tools](#)

Configuring advanced settings for access groups

You can define settings for an access group that are related to authentication, security, connectivity, and accessibility for users and other requestors who reference the access group.

- [Configuring security settings for an access group](#)

You can define settings for an access group that are related to authentication, security, connectivity, and accessibility for users and other requestors who reference the access group.

- [Defining a run-time configuration for an access group](#)

You can define settings for an access group that are related to run-time rulesets and accessibility for users and other requestors who belong to an access group.

- [Defining a design-time configuration for an access group](#)

You can define the default destination ruleset for users who belong to an access group.

- [Learning about access groups](#)
- [Creating an access group](#)

Configuring security settings for an access group

You can define settings for an access group that are related to authentication, security, connectivity, and accessibility for users and other requestors who reference the access group.

Before you begin: You must complete one of the following tasks before defining access control for an access group:

- Create an access group as described in [Creating an access group](#).
- Open an existing instance from the navigation panel by clicking **Records Security Access Group** and selecting an instance.

To configure access control for an access group, click the **Advanced** tab and navigate to the **Access Control** section. Choose any of the following options.

- In the **Authentication timeout** field, describe the number of seconds after which the system challenges idle browser sessions by asking users to re-enter their operator ID and password.
This time-out event does not cause session context or clipboard contents to be lost.

If users respond to the challenge and are reauthenticated, they can usually continue processing where they left off, unless the system released locks that they held in the meantime or the system was stopped and restarted.

This setting is also taken into account in an offline-enabled application that is either online or offline. When the time-out value is reached, users are automatically logged out when they perform any action and the login screen is displayed.

Note: This authentication time-out is not related to the `timeout/browser` value in the `prconfig.xml` file or dynamic system settings, which control the time-out for passivation to occur.

- In the **HTTP/HTTPS home directory** field, accept the default of `/webwb`. Directories within this directory hold important static XML forms, JavaScript, style sheets, and images. If you changed the name of this directory, enter the new directory name in this field.
- In the **Rule security mode** field, select whether and how the system executes rules accessed. In Pega Platform, rules can be made to require privileges in two ways:
 - By an administrator explicitly editing the rule's **Security** tab (where such a tab is available).
 - By a Pega Platform activity, *pyRuleExecutionMessagesLogged*, that automatically generates privileges and adds them to roles that access rules that require privileges.
 - **Allow** [default] – The system allows users that belong to the access group to execute a rule that has no privilege defined, or to execute a privileged rule for which the user has the appropriate privilege. This includes privileges predefined for standard roles, set by an administrator, or generated by Pega Platform. This is the default and recommended setting.

A best practice is to use this mode, and to use the Access Manager to configure authorization for rules. When more specific security is needed for an individual rule, specify a privilege for the rule. For more information, see [Access of Role to Object form - Completing the Privileges tab](#). If you want to specify privileges for all rules and users and you have sufficient time and resources to perform a system-wide exercise including all expected users, see [Completing the Security tab for Access Deny rules](#).

- **Deny** – The system checks for either of these conditions:
 - The user has an explicit privilege or a generated privilege for the rule. If the user has either privilege, the system executes the rule. If the user has neither privilege, the system denies execution of the rule and logs an error message in the PegaRULES log.
 - The user has the generated privilege. If the user lacks the generated privilege, the system denies execution and writes an error message to the PegaRULES log.
- **Warn** – The system performs the same checking as in **Deny** mode, but performs logging only when no privilege has been specified for the rule or the user role:
 - If a privilege has been specified for the rule, the system checks whether the user has an explicit or a generated privilege for the rule. If the user has neither privilege, the system denies execution of the rule.
 - If no privilege has been specified for the rule, the system checks whether the user has a generated privilege. If the user lacks the privilege, the system allows execution and writes a warning message to the PegaRULES log. Warning messages are used to generate privileges with *pyRuleExecutionMessagesLogged*.

Use the **Warn** mode only when you want to identify rules for which users need generated privileges to execute. You can then use the Pega Platform activity to generate missing privileges and add them to roles where needed. For more information, see [Completing the Security tab for Access Deny rules](#).

- [Learning about access groups](#)
- [Creating an access group](#)

Defining a run-time configuration for an access group

You can define settings for an access group that are related to run-time rulesets and accessibility for users and other requestors who belong to an access group.

Before you begin: You must complete one of the following tasks before defining a run-time configuration for an access group:

- Create an access group as described in [Creating an access group](#).
- Open an existing instance from the navigation panel by clicking **Records Security Access Group** and selecting an instance.

To define a run-time configuration for an access group, click the **Advanced** tab and navigate to the **Run time configuration** section. Choose any of the following options.

- To provide users with more accessible versions of several standard harness, section, and flow actions, select **Enable accessibility add-on**. To provide maximum accessibility support, include the *PegaWAI* ruleset in the Production Rulesets list for such users. For more information, see [Understanding accessibility](#).
- For users who belong to the access group and who have design-time permission in a production setting, provide the names of one or more **Production Rulesets** that contain the updated rules. For example, in a production setting, you can identify one ruleset and version that remains unlocked and holds only rules that are expected to change often. Such rules can be delegated to management. A ruleset with this purpose is sometimes called a local-only or *production ruleset*. **Caution:** Leave this blank except for developers and other users who modify rules. While your profile includes the ruleset versions listed here, they are not considered part of the current application. Rules in the ruleset versions listed here might not be visible in the Application Explorer, Guardrails tool, or Document wizard facilities.

As a best practice for good security, and to avoid a warning when you save the Access Group form, select from the ruleset versions that appear in the **Production Rulesets (customization)** array on the **Definition** tab of the Application rule.

The system uses this information at log-on time to assemble the *ruleset list* for this user. The order of your entries in this array affects rule resolution. At login, the system appends these entries to the top of your ruleset list, but starting at the bottom of this array. The order of rows in this array becomes the order in which they appear in the ruleset list.

Except for access groups that have the *PegaRULES:WorkUser4* role, include at least one ruleset version that is not locked. If all ruleset versions that a user can access are locked, the user cannot create new rules. Typically, managers have access to a local customized ruleset for storing only those rules that are personal, customized versions of reports.

Note: List distinct rulesets here. A user or other requestor can access rules in only one major version of a ruleset. For example, access to version 04-10-15 includes access to 04-10-14 and 04-04-11, but not to 03-01-01 or 02-15-07.

For example:

If, during sign-on, the access group is accessed when the (partial) ruleset list contains Alpha:01, Beta:02, and Gamma:03 (in that order), and this array contains Red:07, Blue:08, and Green:09 (in that order), the result is Red:07, Blue:08, Green:09, Alpha:01, Beta:02, Gamma:03.

You can include a full version number or an initial portion of a version number. Separate the ruleset name from the version (or partial version) with a colon, as in these examples:

- MORTGAGE:02-07 – Initial portion (major and minor version)
- MORTGAGE:02-07-20 – Full version number

- [Learning about access groups](#)
- [Creating an access group](#)

Defining a design-time configuration for an access group

You can define the default destination ruleset for users who belong to an access group.

Before you begin: You must complete one of the following tasks before defining a design-time configuration for an access group:

- Create an access group as described in [Creating an access group](#).
- Open an existing instance from the navigation panel by clicking **Records Security Access Group** and selecting an instance.

To define a design-time configuration for an access group, click the **Advanced** tab and navigate to the **Design time configuration** section. Choose any of the following options.

Note: Leave these fields blank for an access group that supports logging on to Pega Platform from external systems, or that supports workers or managers who never create rules.

- In the **Default destination ruleset** field, press the Down Arrow key and select a default ruleset. This field contains a ruleset from those listed in the **Production Rulesets** array or from those listed in the application that the system suggests as a destination ruleset when users that belong to an access group create a rule.
 - If the access group is to support application developers, select a ruleset that the developers usually work in and add rules to.
 - For managers who create and customize reports, select a ruleset that is not intended to be moved to other Pega Platform systems.
 - For language specialists involved in internationalization, select a language-specific ruleset.

When a user that belongs to the access group creates a rule by clicking the **New** or **Save As** buttons, the system suggests this ruleset for the new rule.

If you leave this field blank, the Save As form uses the ruleset of the existing rule as the default ruleset for the new rule, if an unlocked version is available.

- In the **Version** field, press the Down Arrow key and select a ruleset version.
This field contains the ruleset version (for the ruleset identified in the **Default destination ruleset** field) that the requestor who belongs to an access group usually adds rules to. During **New** and some **Save As** operations, this version appears as the default ruleset version.

- [Learning about access groups](#)
- [Creating an access group](#)

Dynamically adding roles during user authentication

This articles describes how to configure a single access group and activity so that the correct role combination is dynamically added to the operator's clipboard page when they log in.

An access group defines a static relationship between an operator and a set of roles. During the standard login and authentication process, users acquire their access roles as defined in the Roles list on the Access Group rule form's Access tab. The system adds the value for each role to the *pxUserRoles* value list within the *pxThread.pxSecuritySnapshot* page.

Some organizations maintain user roles in a system of record that is external to Pega Platform. In such cases, the best approach is to reference that system of record during login and dynamically add the roles. This approach for associating roles with users eliminates the need to separately maintain access groups and 'operator ID to access group' mappings.

If you are using a custom authentication activity to retrieve user authentication information from LDAP, consider retrieving all of the user's information in a single LDAP request. Your authentication activity can create the **RolePage**, and populate the **ListOfRoles** property, which can then be used by the *ApplicationProfileSetup* activity.

The following example dynamically adds the *PegaRULES:SysArch4* role to an operator whose access group provides only the *PegaRULES:SysAdm4* role. **Note:** For the sake of illustration, this example uses standard Pega Platform roles. Typically, these are replaced by project-specific roles.

1. Create the roles, privileges, and *Rule-Access-Role-Obj* instances that are required for authorization within your application.
The role names should either identically match the roles stored in the external system of record, or enable a one-for-one mapping to be defined between the external and Pega Platform role names.
2. Create a separate role and *Rule-Access-Role-Obj* instances for a base role that all users will share. At a minimum, this role must be configured to permit activity processing in the *Code-Security* class.
In the example, *PegaRULES:SysAdm4* is the base role. In typical applications, this would be an application-specific role.
3. Create a base access group, which contains the following:
 - application, work pools, and settings needed to execute the application
 - the base role that you created in step 2.
 - (optional) any roles that are static (shared by all operators who belong to this access group)
4. Add the base access group to all the operator ID records for your application, including any template operator IDs that support single sign-on.
In the example, the operator is *DynamicRolesSysAdmin*.
5. The standard activity *Code-Security.ApplicationProfileSetup* is an extension point — an empty activity executed during sign-on, intended for you to override with organization-specific features. Copy the standard *ApplicationProfileSetup* activity in the *Code-Security* class into an application ruleset at a level in the ruleset hierarchy that reflects how roles are managed (for instance, enterprise-wide, or only one application).
6. Create a *Value List* text property (named *ListOfRoles* in the example) in the *Code-Security* class.
This property holds the role names that can be added during login.
7. Modify the *ApplicationProfileSetup* activity as follows:
 - a. Enter a precondition on the first line so that the activity exits if an operator logging in should be excluded from dynamic role processing. During initial implementation, test by excluding one or more development operator IDs. Adjust this precondition to meet your requirements.
 - b. Create the **RolePage** page, of class *Code-Security*.
 - c. Enter the roles that you want to add to the **ListOfRoles** property.
In the example, *PegaRULES:SysArch4* is the property value. In most projects, your application queries the external system of record to retrieve the roles. If they are not identical to the Pega Platform role names, create a map value rule that converts the external name into a Pega Platform role.
 - d. With **RolePage** as the step page, create the following java step:

```
boolean myResult=  
tools.getAuthorizationHandle().setRoles(tools, PRAuthorization.UPDATE_APPEND, myStepPage.getProperty("ListOfRoles"));
```

- e. Remove the **RolePage** page.
8. Save the rule.
Note: The activity **appends** the new roles to the existing access group's list of roles. Each operator gets these roles **and** any roles defined on the external system of record. The roles listed in the shared access group are always granted and cannot be overridden by adding other roles. Therefore, the roles from the external source of record can only maintain or increase access.

What to do next:

Test the activity: Log in as *DynamicRolesSysAdmin*, open the clipboard, and verify that the additional role is added to the *pxUserRoles* list.

[Use access roles and privileges to limit access to rules and data](#)

Managing access roles

An access role rule defines a name for a role, and represents a set of capabilities. To deliver these capabilities to users, you reference the access role name in other rule types to assign the access role to users and to provide, or restrict, access to certain classes.

An access role identifies a job position or responsibility defined for an application. For example, an access role can define the capabilities of *LoanOfficer* or *CallCenterSupervisor*. The system grants users specified capabilities, such as the capability to modify instances of a certain class, based on the access roles they acquire at sign on.

An access group includes one or more access roles which define what the group can do. The same role can be used in multiple access groups.

When you create an access role, you can configure dependencies from other roles to determine access rights. Role dependencies are relationships between roles that can mirror an organizational hierarchy or more complex relationship between groups of operators, roles, or functional areas. For example, a manager should have all or almost all the access rights that a user has, plus some access rights that only managers have. An example of a more complex access relationship is an HR application. An operator can be allowed to create or modify a job position, set a salary, enter an interview evaluation, make a job offer, or run reports. The access rights in all these cases depend on the operator's department and level in the organization.

If you create access roles, be sure to create a last-resort Access of Role to Object rule at *@baseclass* for that access role, so that the class inheritance search always ends successfully.

Access role names have the format *application name:role name*, where application name is the name of your application and role name is the name of a role that uses the application.

- [Viewing access roles](#)
- [Understanding Access of Role to Object rules](#)
- [Learning about access groups](#)

Standard access roles

Pega Platform provides several standard access role name rules. These role names are referenced in standard Access of Role to Object rules to control the classes that users with these access role names can see and update.

Name	Description
<i>PegaRULES:AutoTest</i>	Allows access to use automated unit testing.
<i>PegaRULES:Batch</i>	For background requestors, including agents.
<i>PegaRULES:Guest</i>	Allows unauthenticated access with minimal access.

<i>PegaRULES:Name</i>	Allows unauthenticated access with more capabilities, including the ability to open and update the user's own work items.
<i>PegaRULES:PegaAPI</i>	Allows a user to use the Pega API.
<i>PegaRULES:SysAdm4</i>	Allows a developer to have full capabilities.
<i>PegaRULES:SysOpsAdministrator</i>	Allows an administrator to have full use of the Admin Studio portal (<i>pxAdminStudio</i>). There is also a standard access group named <i>SysOpsAdministrator</i> .
<i>PegaRULES:SysOpsObserver</i>	Allows a user to have read-only use of the Admin Studio portal (<i>pxAdminStudio</i>). There is also a standard access group named <i>SysOpsObserver</i> .
<i>PegaRULES:User1</i>	Application user with limited capabilities. Cannot perform assignments on worklists other than the user's own worklist. Uses Get Most Urgent to obtain the next assignment.
<i>PegaRULES:User4</i>	For users with broader capabilities. Can open any work object in the application. Can perform assignments only on the user's own worklists.
<i>PegaRULES:WorkMgr4</i>	Allows a work manager to have full capabilities, including viewing and updating delegated rules.
<i>PegaRULES:SysArch4</i>	For a business analyst or a system architect who defines processes, classes, and properties (Data Model category), and can develop activities.
<i>PegaRULES:ProArch4</i>	For a process architect who focuses on flows and other rules in the Process and User Interface categories.
<i>PegaRULES:EditorCollaborator</i>	For collaborators who participate in development of the application as editors. Can view and change application assets.
<i>PegaRULES:ViewerCollaborator</i>	For collaborators who participate in development of the application as reviewers. Can view application assets, but cannot change them.

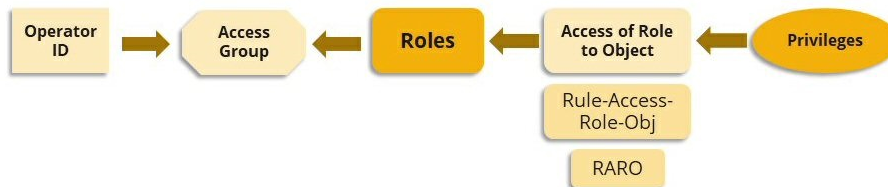
Security of access roles

An access role rule defines a name for a role, and represents a set of capabilities. As Access roles are a fundamental part of the security in Pega Platform, there are a number of rules that affect access roles.

Explanation of access roles

The access roles are described in text below, however, a summary image is shown below.

A flow diagram explaining the flow of access roles



Each authorized user has an *Operator ID*.

The circled operator ID in the access group

Edit Operator ID: User.InlineView
 ID: User.InlineView RS: InlineView [Edit]

Profile Work Security History

Contact Information

Choose File No file chosen
 Upload Image

Title
 First name
 Last name
 Full name ★
 User.InlineView
 Position / job title

Application Access

Use the radio button to select a default access group

Access Group
☒ PRPC:WorkUsers

Application
☐ PegaRULES 8

Localization

Default locale
 Time zone
 America/New_York

The *Operator ID* contains at least one *Access Group*. An *Operator ID* must have at least one access group, but can have multiple access groups.

The *Access Group* contains at least one *Role*. An *Access Group* must contain at least one role, but can have multiple roles.

In the Available roles section of the Definition Tab of the Access group

Edit Access Group: PRPC:WorkUsers
ID: PRPC:WorkUsers RS: Pega-ProcessCommander [Edit]

Definition
Advanced
Operators
History

Application

Name★ PegaRULES Version★ 8

Available portals

Default Name

pyCaseManager7

+ Add portal

Available roles

☐ Stop access checking once a relevant Access of Role to Object instance explicitly denies or grants access

PegaRULES:User4

+ Add role

The Roles contain an Access Role name. This Access Role name makes a call to the Rule-Access-role-Obj, which are also called RARO. The RARO is contains Privileges.

The Access Role name is assigned based on the classes in the application. For each class, there can be an Access of role to Object rule, which assigns the privileges to the role for that class.

The circled Privilege or access role name

Access Role Name: User Level 4 (Pega) access to assignments/worklists [Available]
ID: PegaRULES:User4 RS: Pega-RULES:08-01-05
Save as

Access Class	Read instances	Write instances	Delete instances	Read rules	Write rules	Delete rules	Execute reports	Execute activities	Privileges
@baseclass	0			5			5	5	AllFlows (5) AllFlowActions (5)
Assign-	canPerform	5	5	5			5	5	None
Code-								5	None
Data-	5			5			5	5	AllFlows (5) AllFlowActions (5)
Data-AdHocNotes	5	5	5	5			5	5	None
Data-Admin-Calendar	5			5			5	5	None
Data-Admin-Operator-ID	5	pzIsSocialContext		5			5	5	None
Data-Admin-StartingFlows	5	5	5	5			5	5	AllFlows (5) AllFlowActions (5)
Data-Admin-Toggle	5	0	0	5	5	5	5	5	None
Data-Admin-WorkBasket	5	pxIsTeamManager		5			5	5	AllFlowActions (5) AllFlows (5)
Data-Admin-WorkGroup	5	pxIsTeamManager	pxIsTeamManager	5			5	5	AllFlowActions (5) AllFlows (5)
Data-Application-Feature	5	5		5			5	5	None
Data-Content-Image	5	5	5	5	5	5	5	5	None
Data-Corr-	5	5		5			5	5	None
Data-Decision-IH-	5	5	5						None
Data-EmailAccount	5	5		5			5	5	None
Data-Mobile-APNSToken	5	5	5	5			5	5	None
Data-Mobile-Certificate	5	5	5	5			5	5	None
Data-Mobile-GCMToken	5	5	5	5			5	5	None

The privileges shown in the image below are explicit privileges and are assigned on the rules.

The privileges assigned to a certain access role

Access of Role to Object: @baseclass
ID: PegaRULES:User4 **RS: Pega-RULES**

Security **Privileges** Settings History

	Privilege	Level
1	AllFlows	5
2	AllFlowActions	5

+

The privilege can also be seen on the **Security** tab of the activity.

The privilege on the Security Tab

Flow Action: Add Attachments [Available]
CL: Work- **ID: AddAttachments** **RS: Pega-ProCom:08-01-01**

Layout Validation Action Help setup **Security** HTML Pages & Classes History

Required privileges

	Privilege class	Privilege name
1	Work-	ActionAddAttachments
2		AllFlowActions

+ Add privilege

Dependent roles

For Pega Platform versions prior to 8.x, the original recommendation for clients who were implementing applications was to create their own roles by copying or “cloning” the roles that Pega shipped with the application, and then customize them to meet the client’s security model.

Role dependencies

When Pega added new functionality to an application and included a new privilege to provide appropriate security for that new feature, the clients’ custom roles would not automatically get this new privilege, but would have to add it manually.

To avoid this manual work, Pega redesigned the role process, and added role dependencies.

The Manage dependent roles button in the Access Role Name

Edit Access Role Name: PEBranches:Manager [Available] Delete Actions Save

ID: PEBranches:Manager **RS: PEBranches:01-01-01**

Role Specifications History

Clone from Clone **Manage dependent roles**

☐ Inherit privileges within class hierarchy

Access Class	Read instances	Write instances	Delete instances	Read rules	Write rules	Delete rules	Execute reports	Execute activities	Privileges	Settings
No items										

+

The concept behind a role dependency is that instead of a client “cloning” a role and then customizing it, the client creates a new role which is based on – dependent on – a Pega role which is shipped with the product. The client’s new role will not only have the privileges provided by their customized role, but will also inherit the privileges provided by the dependent role.

If Pega releases a new version of the application which requires a new privilege, that new privilege will be added to the appropriate Pega role(s) in the new release.

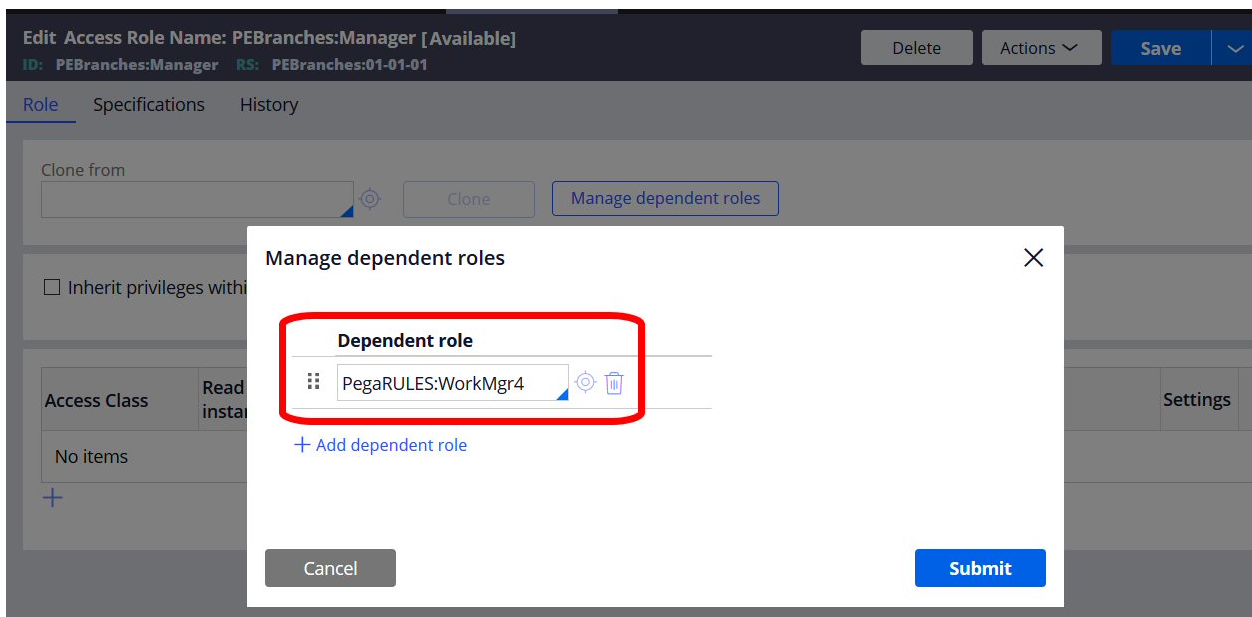
During the upgrade process, this Pega role will be updated, just as all the other changed rules are. If the client has a role which is dependent upon the updated Pega role, then they will **automatically** get this new privilege when they upgrade to that new version (rather than having to manually add this new privilege to their custom roles).

Example of Dependent roles

In this example, the application contains the *PEBranches:Manager* role.

There are no privileges specified in this record, but if you click on the **Manage dependent roles** button, the **Manage Dependent Roles** dialog will appear:

The Manage dependent roles dialog box



This information indicates that the *PEBranches:Manager* role is **dependent** upon the shipped *PegaRULES:WorkMgr4* role. That means that *PEBranches:Manager* **inherits** all the privileges from *PegaRULES:WorkMgr4*.

The access role name and the inherited privileges to those access roles

Access Role Name: Work Manager Level 4 (Pega) access [Available]									
Access Class	Read instances	Write instances	Delete instances	Read rules	Write rules	Delete rules	Execute reports	Execute activities	Privileges
Assign-	canPerform	5	5	5	InMyRuleSetList	InMyRuleSetList	5	5	None
@baseclass	5			5			5	5	UpdateLimitedForm (5) AllFlows (5) AllFlowActions (5) ReportWizard (5) ShowStoredValues (5) TraceOptionsBasic (5) pxEditTeamManagement (5) zipMoveExport (5)
Code-								5	None
Code-Automation-Errors				5				5	None
Data-	5			5			5	5	AllFlows (5) AllFlowActions (5)
Data-AdHocNotes	5	5	5	5			5	5	None
Data-Admin-Calendar	5	5	5	5			5	5	None
Data-Admin-Operator-ID	5	pzIsSocialContext		5			5	5	UpdateWorkGroupAvailability (5) UpdateWorkGroupSkills (5)
Data-Admin-OrgDivision	5			5			5	5	None
Data-Admin-OrgUnit	5			5			5	5	None
Data-Admin-StartingFlows	5	5	5	5			5	5	AllFlows (5) AllFlowActions (5)

Viewing access roles

You can view all the access roles or view only the roles that reference an application.

1. To view all the access roles in Pega Platform, click **Records Security Access Role Name**.
 2. To view only the roles that reference the current application, click **Configure Org & Security Groups & Roles Access Roles**.
 3. To view the roles for a specified access group, do one of the following steps.
 - Click **Records Security Access Group**, click an access group name, and look in the **Available roles** section.
 - Click **Configure Org & Security Groups & Roles Access Groups**, click an access group name, and look in the **Available roles** section.
- [Managing access roles](#)

Creating an access role by using the rule form

To define a set of permissions within an application, create and configure an access role. This topic shows how to create an access role directly from the rule form. You can also create an access role by using the access role dialog.

1. In the Dev Studio header, click **Create Security Access Role Name**.
 2. Enter a short description.
 3. Click **Create and open**. **Result:** The **Role** tab opens with a role name composed of your application name, followed by a colon, followed by the short description that you entered. If your short description contains spaces or special characters, they are removed. For example, if your application is *MyApp* and the role short description is *My Role*, the **ID** on the **Role** tab is *MyApp:MyRole*.
 4. Configure the access role as described in [Configuring an access role](#).
- [Managing access roles](#)
 - [Creating an access role by using the access role dialog box](#)

Creating an access role by using the access role dialog box

You can add *Rule-Access-Role-Name* rules and associated *Rule-Access-Role-Obj* rules to the current application rule set to define the classes that a user can view, update, and delete through the *Access of Role to Object* and *Access Deny* rule types. This topic shows how to create an access role by using the access role dialog. You can also create an

access role by using the rule form.

1. In Dev Studio, click **Configure Org & Security Groups & Roles Access Roles**.
The tab displays the access roles for the current application. If there is a work queue with the same name as the second portion of the Role Name it is also displayed.
2. Click the **Add** icon to open the **Add a New Role** dialog box.
3. Enter all the required information in the dialog box. You can add an access role and corresponding *Rule-Access-Role-Obj* rules based on your current access role, or clone them from another specified role.
 - a. **Role Name** – Enter an access role name in the form AppName:RoleName. Choose a name that is unique system-wide.
 - b. **Optional: Description** – Enter a short text description for the access role.
 - c. **Optional: Clone From** – Create the *Rule-Access-Role-Obj* rules for the new role by copying those associated with a specific existing access role, identify that source access role here. If you leave this blank, the **Submit** button uses your current access role as the source.
 - d. **Create Work queue** – Select to create a work queue of the same name as the Role Name.
 - e. **RuleSet** – Chose a ruleset from the list, or click the add icon to enter a new ruleset version.
 - f. **RuleSet Version** – Select the version of the ruleset that you entered in the **RuleSet** field.
4. Click **Submit**.
5. To view all the access roles in the application with rulesets and ruleset versions associated with them, click **Configure Org & Security Tools Security Role Names**.
6. **Optional:** To delete an access role, click the **Delete** icon in the table row. Any corresponding *Rule-Access-Role-Obj* instances are also deleted. Optionally, you can delete the corresponding work queue instance, if it exists.

- [Creating an access role by using the rule form](#)

Adding a role to an access group

You can assign access roles to an access group so that users who belong to the access group have a consistent set of functions available to them.

Before you begin:

- To add or remove roles from an access group, you must have the *pzCanAlterRoles* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.
- Before adding a role to an access group, complete the following task: [Creating an access group](#).

1. Create an access group or open an existing instance from the navigation panel by clicking **Records Security Access Group** and selecting an instance.
 2. Click the **Definition** tab.
 3. **Optional:** Select **Stop access checking once a relevant Access of Role to Object instance explicitly denies or grants access** if you want the system to stop searching as soon as an access role is found with a relevant Access of Role to Object rule. Clear this check box to use all roles.
 4. In the **Available roles** section, enter the access roles that apply to operators and other requestors that use this access group.
 - a. In the role field, press the Down Arrow key and select a role.
Note:
 - If you have not selected **Stop access checking once a relevant Access of Role to Object instance explicitly denies or grants access**, the order is not relevant; the access roles available to a user act as a collection of capabilities granted, and not as a hierarchy.
 - Enter access roles that are consistent with the values that you enter in the **Portals** field.
 - For non-developer workers and work managers using an application, a best practice is to create and use custom access roles that define the capabilities of the role by using *Access of Role to Object* rules. Pega Platform comes with one standard role for users and one for managers, but your application will probably have multiple different roles for users, managers, and others.
 - The *PegaRULES:ProArch4* role is supported but deprecated. Do not use this role for new development.
- For example:** The following examples show the Pega-supplied roles. In your environment, use the roles that you have created.
- For workers, use *PegaRULES:User4*.
 - For work managers, use *PegaRULES:WorkMgr4*.
 - For business analysts, use *PegaRULES:SysArch4*.
 - For developers and designers, use *PegaRULES:SysAdm4*.
- b. To add more access roles, click **Add role** and select a role.
5. Click **Save**.

- [Learning about access groups](#)
- [Granting portal access to an access group](#)
- [Controlling role evaluation for access groups](#)
- [Managing access roles](#)
- [Understanding Access of Role to Object rules](#)

Configuring an access role

To define a set of permissions within an application, configure an access role.

Before you begin: You must complete the following task before you can configure an access role: [Creating an access role by using the rule form](#)

1. Create an access role, or open an existing access role by doing one of the following steps:
 - In the navigation panel, click **Records Security Access Role Name** and choose an access role from the instance list.
 - In the Dev Studio header, click **Configure Org & Security Tools Security Role Names** and double-click a role name.
2. **Optional:** To copy permissions from an existing access role into your access role, do the following steps:
This step removes Access of Role to Object rules that were previously defined for your access role.
 - a. In the **Clone from** field, press the Down Arrow key and select an access role to copy.
 - b. Click **Clone. Result:** The *Access of Role to Object* rules for the **Clone from** role are copied to your access role (overriding any that you already defined). The new values appear in the **Access Class** table.
3. **Optional:** To inherit permissions from existing access roles without copying them, do the following steps:
 - a. Click **Manage dependent roles**.
 - b. To add a role dependency, click **Add dependent role**, press the Down Arrow key, and select an access role.
 - c. To remove a role dependency, click the **Delete this row** icon.
 - d. Click **Submit**.
4. To simplify the process of granting operator access to a feature protected by privileges, select **Inherit privileges within class hierarchy**.
When this is selected, at run time, the system searches the class hierarchy for Access of Role to Object instances. For more information, see [Privilege inheritance for access roles](#).
5. To define the permissions that this access role has for various classes, you add, update, or delete Access of Role to Object rules in the **Access Class** table. When you click an **Access Class** name or the **Add a row** icon, a dialog is displayed where you configure the Access of Role to Object rule.
Standard access roles such as *PegaRULES:SysArch4* and *PegaRULES:User4* include corresponding standard Access of Role to Object rules, including a rule for *@baseclass*. Be sure to create a last-resort Access of Role to Object rule at *@baseclass* so that the class inheritance search always ends successfully.
 - a. For a new Access of Role to Object rule, in the **Class** field, press the Down Arrow key and select a class.
 - b. In the listed fields, enter a *production level* or Access When rule name. At run time, the system evaluates the value to determine whether access is granted.
For more information about these fields, see [Defining permissions by using Access of Role to Object rules](#)
 - c. **Optional:** In the **Privileges** table, enter one or more privileges. For each privilege, enter a *production level* or Access When rule name.
For more information about assigning privileges, see [Specifying privileges for an Access of Role to Object rule](#)
 - d. **Optional:** In the **Access** table, enter one or more settings. For each setting, enter a value.
For more information about assigning settings, see [Defining access settings for an Access of Role to Object rule](#).
 - e. Click **Submit**.
6. Click **Save**.

- [Understanding Access of Role to Object rules](#)
- [Understanding role permissions and privileges in Access Manager](#)
- [Turning on privilege inheritance for access roles](#)
- [Privilege inheritance for access roles](#)
- [Viewing the roles that have a privilege](#)

Granting portal access to an access group

Associate a portal with an access group to control which workspaces or web channels are available to users while they work in your application.

1. Open an access group by searching for it or by using the Records Explorer.
Tip: To open your current access group, select the **Access group** option from the **Profile** menu.
2. Click the **Definition** tab.
3. In the **Available portals** section, click **+ Add portal**.
4. In the **Name** field, press the Down Arrow key, and then select a portal.

For example, select *pxPredictionStudio* to give the data scientists on your team access to the Prediction Studio workspace.

5. **Optional:** To make this portal the default for all operators in the access group, select **Default** next to the portal name.
6. Click **Save**.
7. Log off and then log in to Dev Studio.

Result: Users who belong to the access group can switch from their current workspace or web channel to the portal that you provide.

- [Configuring portals](#)
- [Changing your workspace](#)
- [Adding a role to an access group](#)
- [Learning about access groups](#)

Updating access groups by submitting a request to an active instance

Use the `prpcServiceUtils` command-line tool to update access groups to provide access to a new application version as part of an automated deployment.

You must have a running Pega Platform instance to use the `prpcServiceUtils` tool.

Follow these steps to update an access group.

1. Configure the common properties of the `prpcServiceUtils.properties` file.

Property name	Value	Required for Pega Cloud clients?
<code>pega.rest.server.url</code>	The URL for the REST service, specified in the following format: <code>http:// <hostname> : <port> / <context> /PRRestService/</code> For example: <code>http://myhost:8080/prweb/PRRestService</code>	Yes
<code>pega.rest.username</code>	The operator name on the target system with access to REST services.	Yes
<code>pega.rest.password</code>	The password of the specified operator.	Yes
<code>pega.rest.proxy.host</code>	The host name of the REST proxy server. Do not use localhost as the host name.	Required only when using a proxy for a URL.
<code>pega.rest.proxy.port</code>	The port for the REST proxy server.	Required only when using a proxy for a URL.
<code>pega.rest.proxy.username</code>	The operator name on the REST proxy server with import and export access.	Required only when using a proxy for the URL
<code>pega.rest.proxy.password</code>	The password of the REST proxy operator.	Required only when using a proxy for the URL
<code>pega.rest.proxy.domain</code>	The domain of the REST proxy server.	Required only when using a proxy for the URL
<code>pega.rest.proxy.workstation</code>	The workstation ID for the REST proxy server.	Required only when using a proxy for the URL
<code>pega.rest.response.type</code>	The REST response type, either <code>xml</code> or <code>json</code> . The default value is <code>json</code> . Rollback, restore point, and update access group operations support only <code>json</code> .	Not required
<code>user.temp.dir</code>	Enter the full path to the temporary directory. Leave this blank to use the default temporary directory. For more information about temporary directories, see Temporary files and temporary files directories .	Not required

2. **Optional:** Specify the names of the access groups to update.
If you do not enter a list of access groups, the system updates all access groups that access the application.

```
updateAccessGroup.groupList=AccessGroup1, AccessGroup2, AccessGroup3
```

3. Specify the application name.

```
updateAccessGroup.applicationName=your-application-name
```

4. Specify the new application version.

```
updateAccessGroup.applicationVersion=X.X.X
```

5. Save and close the `prpcServiceUtils.properties` file.

6. Run the `prpcServiceUtils.bat` or `prpcServiceUtils.sh` script with the `updateaccessgroup` option, for example:

```
prpcServiceUtils.bat updateaccessgroup
```

Optional: Pass one or more arguments.

prpcServiceUtils script argument	Action
<code>connPropFile</code>	Enter the full path to the <code>connection.properties</code> file that includes information for multiple targets.
<code>poolSize</code>	Enter the thread pool size. Default is 5.
<code>requestTimeOut</code>	Specify how long the system waits for a response before failing with a time-out error. Default is 300 seconds.

- [Service-enabled system management log files](#)
- [Using service-enabled scripting and the prpcServiceUtils tool](#)

Controlling role evaluation for access groups

Typically, when the system determines a user's access rights to a class, it searches Access of Role to Object (*Rule-Access-Role-Obj*) rules for all of the access roles listed in the operator's access group. Access is granted if any of these access roles permit it. You can, instead, control how many access roles are searched and the order in which they are

searched.

The order in which access roles are searched is determined by the order in which they are listed in the operator’s access group. An option on the access group lets you specify that the search process stops as soon as an access role is found with a relevant Access of Role to Object rule that either grants or denies access.

1. In the header of Dev Studio, click **Configure Org & Security Groups & Roles Access Groups** .
2. Click an access group name to open the Access Group rule form.
3. On the **Definition** tab, select the **Stop access checking once a relevant Access of Role to Object instance explicitly denies or grants access** check box.

- [Privilege inheritance for access roles](#)
- [Turning on privilege inheritance for access roles](#)
- [Understanding Access of Role to Object rules](#)

Viewing operations available to access role assignees

To review all the access roles and the access classes that are linked to them in the application, use the Allow Access to Class landing page. The page lists Access of Role to Object rules in the system. These rules link an access role to a class, and determine what operations are available to users with these access roles.

To create the **Allow Access to Class** list, complete the following steps.

1. In the header of Dev Studio, click **Configure Org & Security Tools Security Allow Access to a Class** .
2. Select a row from the list to view Access of Role to Objects instances.

- [Standard access roles](#)
- [Creating an access role by using the access role dialog box](#)

Viewing activities available to access role assignees

To review all the activities available for specific access roles in your application use the Allow Access to an Activity landing page. The page lists activities in the system with a specific Applies To key part.

To create the **Allow Access to an Activity** list, complete the following steps.

1. In the header of Dev Studio, click **Configure Org & Security Tools Security Allow Access to an Activity** .
2. In the **Value** field, enter a value or select one from the list.
3. Click **Submit**.

- [Standard access roles](#)
- [Creating an access role by using the access role dialog box](#)

Security tab of the application ID form

In Pega Platform, you can set the default access group and access information on lag time and failed login counts on the **Security** tab of the **Application ID** form.

To access the **Application ID** form in Dev Studio, in the navigation pane click **Records Security Application ID** . Select the application that you want to edit from the **Application ID** table. When the form launches, it opens on the **Security** tab by default.

The **Security** tab of the **Application ID** form has two sections: **Access** and **System**.

Access

The **Access** section of the **System** tab contains information on the access group and password:

Field	Description
Group	Enter or select an access group. This access group is used only until the operator is identified.
Password	Enter the password for users who that this application ID represents. At run time, the system matches this password against the password used to create the security token that was submitted in a single sign-on implementation.

System

The **System** section of the **Security** tab contains information on lag time and the failed login count:

Field	Description
Lag time	Each single sign-on URL contains a date and time specified as a <i>SenderTime</i> value. Enter a value to specify the number of seconds allowed between the <i>SenderTime</i> in the URL and the server time when the security token is evaluated. If the <i>SenderTime</i> is older than the <i>LagTime</i> value subtracted from the current server time, the system rejects the attempted login as stale.
Failed logon count	Leave this field blank. This option is not implemented. Each login failure is recorded in a <i>Log-</i> class.

- [Application ID data instances](#)
- [Creating an application ID data instance](#)

Application ID data instances

Pega Platform supports single sign-on. A person or external system that is authenticated by an external application can be authenticated without providing a second user ID and password. The single sign-on feature is implemented through a security token, which is referenced by an application ID data instance.

Note: This is a deprecated feature that is not used in Pega Platform.

An Application ID instance contributes information to the construction and evaluation of security tokens. It provides one password for a group of users who are authenticated externally. It also determines the amount of time allowed between the time a security token is created and the time that the token is evaluated (known as lag time).

Application ID data instances, which support authentication, are different from application rules, which provide the rulesets and other characteristics of a single application.

Category

The *Data-Admin-AppID* class contains Application ID data instances. They are part of the Security category.

- [Creating an application ID data instance](#)
- [Security tab of the application ID form](#)

Advanced settings for access roles

Access roles define the classes that a user can view, update, and delete through the *Access of Role to Object* and *Access Deny* rule types. Access roles are critical for ensuring valid authentication in an application. The access roles that a user holds defines what they can and cannot due in an application. For example, a manager access role allows

managers to reopen closed cases, whereas those with the case manager role cannot reopen cases.

An access role is different from a work party role, which defines the types of parties that can appear in work items.

Use an access role name to grant permissions to users. Requestor type instances and access group instances can reference access roles. Pega Platform provides built-in access roles, with names starting with *PegaRULES*;, for a variety of users:

- Guests
- Authenticated work users
- Administrators
- Developers

An access role is an instance of the *Rule-Access-Role-Name* rule type.

At user log in, the system assembles a set of roles for the user based on information in the user's requestor instance (Operator ID instance), and the associated access group instance. Access roles associated with a user are important because an access role allows access to features and determines the user's capabilities.

To determine whether the current user has a role, call the standard Boolean function *HaveRole()*:

```
Lib(Pega-RULES:Default).HaveRole(this, "rolename");
```

Where referenced

Access role names form the first key part of Access of Role to Object rules (*Rule-Access-Role-Obj* rule type), and the Access Deny Obj rules (*Rule-Access-Deny-Obj* rule type). You convey access roles to users through access groups.

Access

Use the **Access Roles** tab on the Security landing page to view or modify a list of the access roles in your application.

You can access the **Access Roles** landing page tab in the header of **Dev Studio**, by clicking **Configure Org & Security Security Access Roles** .

Use the Records Explorer to list all the access roles that are available to you.

Category

Access Role Name rules are instances of the *Rule-Access-Role-Name* class and are part of the Security category.

- [Understanding Access of Role to Object rules](#)
- [Understanding Access Deny rules](#)
- [Understanding Access When rules](#)
- [Using setting rules](#)
- [Managing access roles](#)

Understanding Access When rules

An Access When rule defines a test that the system performs to allow, or disallow, a user from performing an operation or accessing information (instances of a specific class) based on security requirements.

Important: Don't confuse the *Rule-Access-When* rule type — referenced only to control user access to a class — with the more widely used *Rule-Obj-When* rule type. The forms are similar, but the purpose and references are different.

Where referenced

Access When rules are referenced in the Access Manager, Access of Role to Object rules and Access Deny rules. In the Access Manager, use Access When rules to conditionally authorize access by case type to cases (class instances), assignments, flows and flow actions, and to functions on the Tools tab. Use of the Access Manager is recommended instead of directly modifying Access of Role to Object and Access Deny rules.

For a property that has a Type of TextEncrypted, an Access When rule controls when the system decrypts the encrypted value. If your system uses the TextEncrypted type, limit the ability to create, update or delete rules of this type, as such rules can enable access to the decrypted values.

Access

Use the Application Explorer to list Access When rules in your application. Use the Records Explorer to list all the Access When rules that are available to you.

Category

Access When rules are instances of the *Rule-Access-When* class. They belong to the **Security** category.

- *Rules development*

Completing the New or Save As form for Access When rules

Records can be created in various ways. You can add a new record to your application or copy an existing one. You can specialize existing rules by creating a copy in a specific ruleset, against a different class or (in some cases) with a set of circumstance definitions. You can copy data instances but they do not support specialization because they are not versioned.

Create an Access When rule by selecting *Access When* from the *Security* category.

An Access When rule has two key parts:

Field	Description
Apply to	Select the class to which this Access When condition applies. For Access When rules that restrict the opening, modification, or deletion of rules, use Rule- as the Apply to class.
When Name	Enter a name for this rule. Begin the name with a letter and use only letters, numbers, the ampersand character, and hyphens. As a best practice, start the name with a capital letter. Do not enter either of the keywords True or False as the When Name.

Additional creation options

Field	Description
When expression	Enter an expression that evaluates to true or false. Type a single period and use the SmartPrompt to access a list of properties. Note: The string expression you define here is used to allow or disallow, a user from performing an operation or accessing information (instances of a specific class).

- [Understanding Access When rules](#)
- [More about report definition rules](#)

Completing the Conditions tab for Access When rules

Use the **Conditions** tab to enter or revise an Access When condition rule that can be expressed as a single Boolean expression, or the conjunction (AND or OR) of one or more Boolean expressions.

Each node on the conditions tree defines a single comparison or other condition that evaluates to `True` or `False`. The Access When condition rule evaluates to `True` only if all nodes evaluate to true. While the order of nodes may affect computational performance, order does not affect the outcome of the Access When condition rule evaluation. Use these options:

The conditions and logical operators you enter on this tab also appear on the **Advanced** tab's **Condition** array and **Logic String** field.

- To create the first expression, perform the following actions:
 - a. Double-click the link below the When node.
 - b. The **Condition** dialog appears, and contains the default condition, which compares two values.
 - c. Enter properties, literal constants, or function calls and parameters in the fields, and select a relational operator.
 - d. **Optional:** Select the down arrow at the end of the row for a list of standard conditions (derived from function alias rules). Open it by clicking the **Open** icon. If you select **[expression evaluates to true]**, use the Expression Builder for guided assistance. See [About Expressions](#).
 - e. **Optional:** Click **OK** to close the dialog and display the condition on the tree.
- To add, edit, or delete conditions, or to create groups of nested conditions:
 - Open the **Actions** menu next to the node.
 - Right-click the node.

You can use the following options:

Edit

Update the condition.

Insert Condition

Add a condition beneath the selected node. When you close the dialog, an AND or OR logical operator appears to the left of the new node. You can toggle between them by clicking the operator. Changing one value changes all of them at the same node level.

Insert Group

Create a group of nested conditions. Enter a name in the **Group Label** field to help identify the group on the tree. When you close the dialog, the condition is indented beneath the node above it. Use the expand and collapse icons to display or hide the nodes within the group. When collapsed, only the label is displayed. To edit the label, select it on the tree and choose the **Edit** menu option. You can nest groups at multiple levels; for example: A AND B AND (C OR (E AND F)) .

Delete

Delete the expression node. Node groups nested beneath it are also deleted.

- [Understanding Access When rules](#)
- [Completing the New or Save As form for Access When rules](#)
- [More about report definition rules](#)

Implementing advanced settings for Access Deny rules

For complex conditions, use this tab to enter conditions and a logic statement that combines them.

As a best practice, use the **Conditions** tab to configure and update the rule. Your configuration populates the conditions array and the Logic String field on this tab. If you add, change, or delete rows, or edit the Logic String field, you can no longer use the Conditions tab. Updating values in existing conditions or using the Options area does not disable that tab.

Conditions

In each row, enter a condition and label you want to include in the Access When evaluation.

label

Enter a unique number or text value in the row header that serves as an identifier for this row. These are used in the Logic String elements. Do not choose `and`, `or`, or `not` as labels. You can enter `//` as a label to mark this row as a comment; a condition that is not evaluated.

By default, the system uses A, B, C, and so on.

condition

Use the fields to enter property references, literal constants, or calls and functions. By default, the first condition compares two values.

Optionally, click the **Down Arrow** icon at the end of the row for a list of standard conditions (derived from function alias rules).

Logic String

Enter the Boolean logic operations performed on the Conditions array that will compute to a True or False result at runtime.

If you created the conditions on the **Conditions** tab, the system populates this field using the AND and OR operators, and the group hierarchies defined on the condition tree.

You can use `and`, `or`, and `not` in the statement. You can use parentheses to control the order of evaluation.

For example, if the table contains four labels A, B, C, and D, you can enter: (A and B) or (C and not D)

When you save the rule, if the form contains only one test row, the system inserts the label for that test row here. If you have more than one row, the system defaults to requiring all rows to be true. If that's not what you want, revise the statement and save again.

- [Understanding Access When rules](#)
- [Completing the New or Save As form for Access When rules](#)
- [More about report definition rules](#)

Understanding Access of Role to Object rules

Access of Role to Object rules specify permissions that are granted to a role and access class. These permissions restrict what developers and operators can do with rule and data instances. An Access of Rule to Object rule applies to all instances of its access class.

Access Manager simplifies the process of granting authorization. As a best practice, use Access Manager instead of working directly with the Access of Role to Object rule form. For more information, see [Access Manager landing page](#).

You can use the rule form to do the following tasks:

- Create or delete an Access of Role to Object rule.
- Specify settings for an Access of Role to Object rule. For more information, see [Defining access settings for an Access of Role to Object rule](#).
- Specify production levels 2, 3, or 4 for an access control. For more information, see [Defining permissions by using Access of Role to Object rules](#).

Access of Role to Object rules can also be created from the Access Role Name rule form.

Typically, an access group contains multiple roles. At run time, the Access of Role to Object rules for all of the user's roles are evaluated. The rule with the most permissive setting takes precedence. For example, if the user's access group has roles A and B, and role A does not have access, but role B has access, then the user has access.

- [Rules development](#)
- [Managing access roles](#)
- [Access Manager](#)

Creating an Access of Role to Object rule

Create an Access of Role to Object rule to associate an access role with a class.

1. In the header of Dev Studio, click **Create Security Access of Role to Object**.
2. On the Create form, define the context of the rule.
 - a. In the **Access class** field, press the Down Arrow key and select the class that defines the abstract or concrete class that users who hold this access role need to access in some way. To provide uniform access to both a parent class and all its subclasses, enter the parent class name.
 - b. In the **Add to ruleset** field, select the name and unlocked version of a ruleset that stores the privilege. As a best practice, to avoid confusion and difficult-to-debug security configurations, use the same ruleset as the ruleset of the Access class that you selected in [2.a](#).
3. Click **Create and open** to open the Access of Role to Object form.
4. Click **Save**.

1. Defining permissions by using Access of Role to Object rules

Use the Access of Role to Object rule form to define the permissions that an access role has for a class.

2. Specifying privileges for an Access of Role to Object rule

To more precisely define access to instances of a class, you update the Access of Role to Object rule to grant or revoke privileges for a role and access class. Specifying privileges is optional.

3. Defining access settings for an Access of Role to Object rule

To more precisely define access to instances of a class, you associate one or more access settings with an access role by using the Settings tab of the Access of Role to Object rule form. Specifying access settings is optional.

- [Understanding Access of Role to Object rules](#)
- [Copying a rule or data instance](#)
- [Creating a rule specialized by circumstance](#)
- [Managing access roles](#)

Defining permissions by using Access of Role to Object rules

Use the Access of Role to Object rule form to define the permissions that an access role has for a class.

When you define an Access of Role to Object rule, you define access controls for various core functions by role and access class. You can define access controls by using either Access Manager or the Access of Role to Object rule form. The rule form enables somewhat more precise control when entering production levels. For more information about Access Manager, see [Access Manager landing page](#).

You specify access controls by entering either a production level or an Access When rule name. At run time, the system evaluates the value to determine whether access is granted.

- If you enter a When rule name, the system uses the access class and class inheritance to find the Access When rule, and then evaluates the When rule to see if access is granted.
- If you enter a *production level*, the system compares this level with the production level of the current system. The privilege is granted only if the access control's production level is greater than or equal to the system's production level. Enter 0 to provide no access. Enter 5 to allow access to all systems. When privilege inheritance and role dependency are in effect, a value of blank is undefined (neither true nor false); otherwise, a value of blank indicates no access. For more information on privilege inheritance and role dependency, see [Privilege inheritance for access roles](#) and [Configuring an access role](#).

1. In the navigation panel, click **Records Security Access of Role to Object**, open the rule you want to change, and click the **Security** tab.
2. You can define the following access controls. For each field, enter a production level or When rule name.

Open instances

Controls whether users with this access role can open instances of this access class.

Modify instances

Controls whether users with this access role can save new or modified instances of this access class.

Delete instances

Controls whether users with this access role can delete instances of this access class.

Run reports

Controls whether users with this access role can run reports against instances of this access class.

Execute activities

Controls whether users with this access role can execute activities with an Applies to class that is equal to this access class.

3. You can define the following Open, Modify, and Delete rules.

The following Open, Modify, and Delete rules controls are typically needed only by application developers. If you enter a When rule, then at run time, the system uses the class of the primary page to locate the When rule. As a best practice, create the When rule in the Rule- base class.

Open rules

Controls whether users with this access role can open rules having this access class as a key part.

Modify rules

Controls whether users with this access role can create or modify rules having this access class as a key part.

Delete rules

Controls whether users with this access role can delete rules having this access class as a key part.

- [Understanding Access of Role to Object rules](#)
- [Understanding Access When rules](#)

Specifying privileges for an Access of Role to Object rule

To more precisely define access to instances of a class, you update the Access of Role to Object rule to grant or revoke privileges for a role and access class. Specifying privileges is optional.

You can make these updates by using Access Manager or the Access of Role to Object rule form. The *production level* values that you can specify are different on these two forms. For more information on Access Manager, see [Access Manager landing page](#).

A privilege associated with an Access of Role to Object rule is granted to users who have the access role, when they work with instances of the rule's access class (or child class), on a system that has a specific production level (or lower).

1. In the navigation panel, click **Records Security Access of Role to Object**, open the rule you want to change, and click the **Privileges** tab.
2. In the **Privilege** field, enter the name of the privilege to grant to this role. The privilege must already be defined for this access class.

You can add more privileges by clicking the **Add a row** icon.

You can remove privileges by clicking the **Delete row** icon, or updating the **Level** to 0 or blank.
3. In the **Level** field, enter a When rule name or a *production level*. At run time, the system evaluates this value in the following ways:
 - If you enter a When rule name, the system uses the access class and class inheritance to find the Access When rule, and then evaluates the When rule to see if access is granted.
 - If you enter a production level, the system compares this level with the production level of the current system. The privilege is granted only if the privilege's production level is greater than or equal to the system's production level. Enter 0 or leave blank to provide no access. A value of 5 allows access to all systems.
4. Click **Save. Result:** Active requestor sessions on the current node are immediately updated. Requestors on other nodes in a cluster are updated when the next system pulse occurs on their nodes.

- [Understanding Access of Role to Object rules](#)
- [Understanding role permissions and privileges in Access Manager](#)

Defining access settings for an Access of Role to Object rule

To more precisely define access to instances of a class, you associate one or more access settings with an access role by using the **Settings** tab of the Access of Role to Object rule form. Specifying access settings is optional.

1. In the navigation panel, click **Records Security Access of Role to Object** , open the rule you want to change, and click the **Settings** tab.
2. In the **Setting** field, enter the name of the setting to grant to this role.
The setting must have already been defined for this access class.
3. In the **Value** field, enter the setting value that applies to users with this role.
4. Click **Save**.

Result: After you save the Access of Role to Object rule form, active requestor sessions on the current node are immediately updated. Requestors on other nodes in a cluster are updated when the next system pulse occurs on their nodes.

- [Understanding Access of Role to Object rules](#)
- [Using setting rules](#)

Listing Access of Role to Object rules

To see how your system's permissions are configured, you can list all of the Access of Role to Object rules, or just the rules in your ruleset list.

- To list all the Access of Role to Object rules, use the Access Definitions by Class report. To generate the report, click **Configure Org & Security Tools Security Allow Access to a Class** .
- To list only the Access of Role to Object rules in your ruleset list, from the navigation panel, click **Records Security Access of Role to Object** .
- [Understanding Access of Role to Object rules](#)

Standard Access of Role to Object rules

Pega Platform contains over 240 standard Access of Role to Object rules (*Rule-Access-Role-Obj* rule type). Each of these associates a standard access role name such as *PegaRULES:SysAdm4* with an Applies To class. Each rule determines the capabilities that users with the access role name can do with instances of that class (and with instances of all the classes derived from that class).

Standard Access of Role to Object rule names start with the prefix *PegaRULES:rolename* and belong to the *Pega-RULES* ruleset.

For example, the standard Access of Role to Object rule named *PegaRULES:SysAdm4.PegaSample-Task* defines the capabilities of developers for the PegaSample sample application.

You cannot change the standard rules, but you can define other Access of Role to Objects rules to supplement or override them.

- [Understanding Access of Role to Object rules](#)

Understanding Access Deny rules

Access Manager simplifies the process of granting authorization and as a best practice should be used instead of working directly with Access Deny rule forms. In the Pega Platform, select Dev Studio > *Org & Security > Access Manager*.

Use an Access Deny rule to restrict users who have a specified access role from accessing instances of specific classes under certain conditions. Denial of access to the class can depend on the production level value (1 to 5) of your system or whether certain Access When rules evaluate to true.

Conversely, a value of zero or blank allows access (access is not denied).

By default, all access to a class is denied except when explicitly granted using Access Manager (or editing an Access of Role to Object rule). However, government, or company regulations and policies sometimes require explicit denial of access to specific capabilities. In these cases, use Access Deny rules to explicitly deny access to an access role and class combination.

Access

Use the Application Explorer or Records Explorer to list access deny rules available to you.

Category

Access Deny rules are instances of the *Rule-Access-Deny-Obj* rule type. They belong to the Security category.

- [Completing the New or Save As form for Access Deny rules](#)
- [Security tab of the application ID form](#)
- [More About Access Deny rules](#)

Completing the New or Save As form for Access Deny rules

Access Deny rules - Completing the Create or Save As form

Records can be created in various ways. You can add a new record to your application or copy an existing one. You can specialize existing rules by creating a copy in a specific ruleset, against a different class or (in some cases) with a set of circumstance definitions. You can copy data instances but they do not support specialization because they are not versioned.

Create an Access Deny rule by selecting *Access Deny* from the *Security* category.

Key parts

An Access Deny rule has two key parts. The key links an access role with a class:

Field	Description
Role Name	Enter an access role name (<i>Rule-Access-Role-Name</i> rule type).
Applies To	Identify an abstract or concrete class to empower users who hold this access role to have some access to that class.

Specify a RuleSet Name. The RuleSet Name is used only by the Archive tools when creating or importing a ZIP file.

Rule resolution

Rule resolution does not apply to Access Deny rules. Your system can contain at most one Access Deny rule for each Applies To class and Role Name combination.

Class inheritance does not apply. Create a rule for each affected class.

[About Access Deny rules](#)

- [Understanding Access Deny rules](#)
- [More About Access Deny rules](#)

Completing the Security tab for Access Deny rules

Best practice: Use Access Manager to deny authorization instead of working directly with the fields in this tab. Access Manager simplifies the process and updates your Access Deny rules.

Select Dev Studio > *Org & Security* > *Access Manager*.

For each of the user actions that you want to deny, you can enter a numeric value between 1 and 5, or reference an Access When rule. Access is denied when the Access Control value is greater than or equal to the production level of this system. When an Access When rule is used, the system evaluates the rule and denies access when the result of the Access When rule is True.

If a field contains 0 or is blank, access is permitted (not denied).

Users may need the first six types of access to operate on instances. The last three types are usually needed only by application developers.

The production level of the system is visible on the System form.

Field	Description
Open Instances	Optional. Enter the When Name key part of an Access When rule, or a level value between 1 and 5. If you enter a name, the system uses the Access Class key part of the rule to open, and class inheritance, to find the Access When rule. This determines whether holders of the access role identified as the first key part of this rule are denied the ability to open existing instances of the class identified in the second key part of this Access Deny rule.
Modify Instances	Optional. Enter the When Name key part of an Access When rule, or a level value between 1 and 5. If you enter a name, the system uses the Access Class key part of the rule to be modified, and class inheritance, to find the Access When rule. This determines whether holders of the access role identified as the first key part of this rule are denied the ability to save new or modified instances of the class identified as the second key part of this rule.
Delete Instances	Optional. Enter the When Name key part of an Access When rule, or a level value between 1 and 5. If you enter a name, the system uses the Access Class key part of the page passed in to the Delete method. This is usually, but not necessarily, the entire page. It is possible to pass to the Delete method a page containing only the keys of the instance to be deleted. This determines whether holders of the access role identified as the first key part of this rule are denied the ability to delete instances of the class identified as the second key part of this rule.
Run Reports	Optional. This determines whether holders of the access role identified as the first key part of this rule can run reports against the class being reported on or listed. Enter the When Name key part of an Access When rule, or a level value between 1 and 5. The message: You are not authorized to run this view. indicates that a user lacks the capability provided by this field.
Execute Activities	Optional. Enter the When Name key part of an Access When rule, or a level value between 1 and 5. If you enter a name, the system uses the Access Class key part of this Access of Role to Object rule and class inheritance to find the Access When rule. This determines whether holders of the access role identified as the first key part of this rule are denied the ability to execute activities that belong to the class identified as the second key part of this rule.
Open Rules	Optional. Enter the When Name key part of an Access When rule, or a level value between 1 and 5. If you enter a name, the system uses the class of the primary page at runtime to locate an Access When rule. As a leading practice, create the Access When rule in the Rule- base class. That is, set the Applies To key part of the Access When rule to Rule-. This determines whether holders of the access role identified as the first key part of this rule are denied the ability to open rules with the class as a key part.
Modify Rules	Optional. Enter the When Name key part of an Access When rule, or a level value between 1 and 5. If you enter a name, the system uses the class of the primary page at runtime to locate an Access When rule. As a leading practice, create the Access When rule in the Rule- base class. That is, set the Applies To key part of the Access When rule to Rule-. This determines whether holders of the access role are denied the ability to save new or modified rules with the class as a key part.
Delete Rules	Optional. Enter the When Name key part of an Access When rule, or a level value between 1 and 5. If you enter a name, the system uses the class of the primary page at runtime to locate an Access When rule. As a leading practice, create the Access When rule in the Rule- base class. That is, set the Applies To key part of the Access When rule to Rule-. This determines whether holders of the access role are denied the ability to delete rules with the class as a key part.

[About Access Deny rules](#)

- [Understanding Access Deny rules](#)
- [Completing the New or Save As form for Access Deny rules](#)
- [More About Access Deny rules](#)

More About Access Deny rules

Learn about when access deny rules take effect and see an example.

When effective

After you save an Access Deny form, active requestor sessions on the current node that are associated with that rule are immediately updated. Requestors at other nodes in a cluster are updated when the next system pulse occurs on their node.

Example

Assume an application includes classes named Data-Refund-Basic, Data-Refund-Silver, and Data-Refund-Gold, all derived from the Data-Refund- abstract class, and two access roles named Refund:Worker and Refund:Super.

You want both types of users to be able to work on all three classes, except that users with the Refund:Worker role may not delete Data-Refund-Gold data instances.

You can accomplish this with three rules:

Rule type	Key class	Access role	Description
Access of Role to Object	Data-Refund-	Refund:Worker	Grant full access to all three classes.
Access of Role to Object	Data-Refund-	Refund:Super	Grant full access to all three classes.
Access Deny	Data-Refund-Gold	Refund:Worker	Deny Delete access to this class for workers; no change for others.

You can achieve the same results without any Access Deny rules, by granting needed Data-Refund-Gold access capabilities one-by-one. However, using one Access Deny rule in this situation is simpler and easier to understand and maintain.

- [Understanding Access Deny rules](#)
- [Completing the New or Save As form for Access Deny rules](#)

Using setting rules

Use a setting rule to define a name and data type for a setting that your application can use for finely tuned access control. You can define access for different access roles based on the value of a setting. You associate the setting name and value on the Access of Role to Object rules for various class/role combinations.

Settings and values become associated with individual users indirectly. Each user has an associated access group, referenced in the Operator ID data instance. The access group identifies one or more access roles. Access of Role to Object rules with these roles as a first key part define many aspects of user access control. These Access of Role to Object rules might reference settings rules and define values for each setting.

Setting rules have no relation to application settings rules (*Rule-Admin-System-Settings* rule type) or dynamic system settings (*Data-Admin-System-Settings* class).

Access

Use the Application Explorer to list setting rules in your application. Use the Records Explorer to list all the setting rules that are available to you.

Example

For example, an application includes two access roles named Underwriter and SeniorUnderwriter. You create a setting rule named *PolicyLimit* so that the application can assign work to operators of each role based on the amount of a policy.

In the appropriate Access of Role to Object rules, you associate a different policy limit amount with each access role. In processing, policies with a value of \$50,000 or less can be routed to an Underwriter and policies worth more than \$50,000 can be routed to a Senior Underwriter.

Functions that evaluate settings

Activities in your application can use three standard functions in the *Pega-RULES:Settings* library to evaluate settings at run time:

getSetting(AppliesTo, SettingName)

Return the setting value.

getSettingType(AppliesTo, SettingName)

Return a Java type for the setting, such as string, int, double, datetime, or null (not found).

getNumericSetting(AppliesTo, SettingName)

Return the value as a Decimal property value.

- [Defining access settings for an Access of Role to Object rule](#)
- [Understanding Access of Role to Object rules](#)
- [Learning about access groups](#)

Creating a setting

Create a setting rule to define when an Access of Role to Object rule takes effect. When an Access of Role to Object rule is not associated with a setting, it is applied unconditionally.

1. In the header of Dev Studio, click **Create Security Setting**.
2. Enter a short description and a setting name. Begin the name with a letter, and use only alphabetic and numeric characters.
3. Enter the applies-to class.
The applies-to class equals the access class on the Access of Role to Object instance that uses this setting.
4. Click **Create and open**.
5. Configure the setting as described in [Configuring a setting](#).

- [Using setting rules](#)

Configuring a setting

Configure the data type and other aspects of the setting rule to define when an Access of Role to Object rule takes effect.

Before you begin: You must complete the following task before you can configure a setting: [Creating a setting](#).

1. Create a setting, or open an existing instance from the navigation panel by clicking **Records Security Setting** and selecting an instance.
2. In the **Data Type** list, select the data type of the property to be evaluated for this setting.
 - **String**
 - **Integer**
 - **Decimal**
 - **Date Time**
3. In the **Multiple Setting Behavior** list, select how to evaluate the setting when a user holds two or more access roles that each have a value for this setting.
 - To use the minimum value, select **Smallest**.
 - To use the maximum value, select **Largest**.
4. **Optional:** In the **Setting Edit** field, press the Down Arrow key and select an edit input rule to apply to the setting value when a developer saves an Access of Role to Object rule containing this setting.
5. **Optional:** To sort values for this setting in a different order than the default sort implied by the data type, in the **Sort Function** field, press the Down Arrow key and select a function. **For example:** The three setting values are "Gold", "Silver", and "Platinum", which you want to appear in that order. Your function can define an order that makes "Platinum" the maximum value rather than the middle value.
6. To identify product rules for this setting rule, in the **Product List** section, press the **Add a row** icon and enter the following values:
 - a. In the **Product Name** field, press the Down Arrow key and select the product name.
 - b. In the **Product Version** field, press the Down Arrow key and select the product version.
7. Click **Save**.

- [Using setting rules](#)
- [Defining access settings for an Access of Role to Object rule](#)

Understanding role permissions and privileges in Access Manager

Privileges complement the security and access control features provided by access roles by restricting access to specific rules rather than to entire classes. A privilege associates an access role with a rule that needs to be secured. Create privileges to more precisely define the access control features that are provided by *Access of Role to Object* rules. A privilege is identified by its name and **Applies to class**.

Privilege setup involves two elements that link the rule, access role, and class:

- A rule that requires a privilege before it can be used or accessed.
- Access roles that are evaluated at run time to determine whether the privilege has been granted for a requestor.

At run time, the system compares the set of privileges that are associated with the requestor's access roles with the set of privileges that are required by a rule. If the requestor holds any of the required privileges, the requestor can, for example, run the activity, use the correspondence, or generate the report.

Circumstances and time-based qualifications are not available for privilege rules.

- [Specifying privileges for an Access of Role to Object rule](#)
- [Granting privileges to an access role](#)
- [Privilege inheritance for access roles](#)
- [Property security rules](#)

Creating a privilege

You can associate many rule types with privileges to more precisely define the access control features that are provided by Access of Role to Object rules. You create a privilege by using the Privilege rule form.

1. In the header of Dev Studio, click **Create Security Privilege**.
2. On the **Create** form, enter values in the fields to define the context of the privilege.
 - a. In the **Label** field, enter text that describes the purpose of the privilege. Begin the name with a letter and use only alphanumeric, ampersand, and dash characters. As a best practice, start the name with a capital letter. In compound names such as *ReconcileProblemWork*, capitalize the first letter of each word.
 - b. In the **Apply to** field, press the Down Arrow key and select the class that defines the scope of the privilege. The class controls which *Access of Role to Object* rules you can link to the privilege.
 - c. In the **Add to ruleset** field, select the name and unlocked version of a ruleset that stores the privilege.
 - d. **Optional:** To change the default identifier for the privilege, click **Edit**, and then provide a unique value in the **Identifier** field.
3. Click **Create and open** to open the **Privilege** form.
4. Click **Save**.

- [Understanding role permissions and privileges in Access Manager](#)
- [Requiring a privilege for a rule](#)
- [Viewing the roles that have a privilege](#)
- [Specifying privileges for an Access of Role to Object rule](#)

Granting privileges to an access role

On the **Privileges** tab of the Access Manager landing page, you can create, review, and modify privileges for users with different roles to access specific objects of a case and data types.

A privilege rule defines only a name. This rule does not convey any capabilities to any user until you associate it with an object (flow, flow action, report definition, and so on) and an access role.

The source for the list of privileges is the data page *D_PrivilegeList*. By default, the data page uses a report definition to return the list of privileges in the current application.

- [Granting privileges by using Access Manager](#)

You can use Access Manager grant privileges to users with different roles to access specific case and data types.

- [Reviewing user privileges for a role by using Access Manager](#)

In Access Manager you can see all the associated privileges for a selected role in a selected class. You can also generate a report that shows all the privileges granted to each case type.

- [Customizing the Access Manager privileges tab](#)

You can customize the content of the lists for the selectable options that appear for many fields on the Access Manager Privileges tab.

- [Access Manager](#)

Standard privileges

Pega Platform includes about 300 standard privileges. Many of the standard privileges provide the ability to run specific individual standard flow actions and individual standard flows. Alternatively, if you have the *AllFlows* and *AllFlowActions* privileges, you can access all the standard flows and all the standard flow actions respectively.

- [Access Manager](#)
- [Granting privileges to an access role](#)

Granting privileges by using Access Manager

You can use Access Manager grant privileges to users with different roles to access specific case and data types.

Before you begin: To use the privileges tab of Access Manager, you must be logged in as an operator associated with an access group that contains the *PegaRULES:SecurityAdministrator* or *PegaRULES:BasicSecurityAdministrator* role.

1. In the header of Dev Studio, click **Configure Org & Security Access Manager Privileges**.
2. Select **Case Type** or **Data Type**.
3. In the **Role** field, press the Down Arrow key and select a user role to update.
4. In the **Case Type** or **Data Type** field (depending on the class type that you chose), press the Down Arrow key and select a case or data type from the list.
5. Click **Show inherited privileges**.
6. Click the **Add** icon.
7. In the **Privilege** field, press the Down Arrow key and select a privilege to grant to the role.
8. Specify the allowed access.
 - **Full Access**
 - **No Access**
 - **Conditional**
9. Click **OK**.

- [Access Manager](#)
- [Granting privileges to an access role](#)
- [Standard privileges](#)
- [Understanding role permissions and privileges in Access Manager](#)

Modifying operator permissions with Access Manager

From Access Manager, you can view and edit operator permissions in an access group or for all access groups of the current operator. You can view all flows and flow actions defined for each case type in the selected application or applications.

You do not have to save modifications made in Access Manager because they are immediately implemented.

To customize role-based security for applications:

1. In the header of Dev Studio, click **Configure Org & Security Access Manager Work & Process**.
2. In the **Access Group** list, select the access group you want to modify.
3. In the **Case Types** column, click the arrow next to the case type that you want to modify. **Result:** This expands the operator permissions section.
4. Click the permission icon to change the operator permission in the **Role** dialog box.
 - Select **Full Access** to grant the operator full permission.
 - Select **No Access** to deny the operator access.
 - Select **Conditional** to grant conditional access, and enter or select a condition that, when true, grants the operator permission.
5. In the Role dialog box, click **OK** to apply the changes.

Reviewing user privileges for a role by using Access Manager

In Access Manager you can see all the associated privileges for a selected role in a selected class. You can also generate a report that shows all the privileges granted to each case type.

Before you begin: To use the privileges tab of Access Manager, you must be logged in as an operator associated with an access group that contains the *PegaRULES:SecurityAdministrator* or *PegaRULES:BasicSecurityAdministrator* role.

1. In the header of Dev Studio, click **Configure Org & Security Access Manager Privileges**.
2. Select **Case Type** or **Data Type**.
3. In the **Role** field, select a user role from the list.
4. In the **Case Type** or **Data Type** field, press the Down Arrow key and select a case or data type from the list.
5. Click **Show inherited privileges**.

The table displays the following information:

 - **Access** – The icons indicate the allowed access
 - **Privilege Name** – The name of the privilege
 - **Description** – What the privilege does
 - **Usage** – What a user with this privilege can do
6. To generate a report in a PDF file that displays all privileges for each case type, click **Export authorizations**.

- [Access Manager](#)
- [Granting privileges to an access role](#)
- [Viewing the roles that have a privilege](#)
- [Standard privileges](#)

Customizing the Access Manager privileges tab

You can customize the content of the lists for the selectable options that appear for many fields on the **Access Manager Privileges** tab.

You can customize the content of the lists for the following data categories:

- **Roles** – By default the data page retrieves all roles defined in the current application rulesets.
- **Case Types** – By default the data page uses a report definition to return the list of case types in the current application.
- **Data Types** – By default the data page uses a report definition to return the list of data types in the current application and industry applications.
- **Privileges** – By default the data page uses a report definition to return the list of privileges in the current application.
- **Access When** – By default the data page uses a report definition to return the list of Access When rules in the current application.

In the table are important information needed for customization.

Data category	Source data page	Class name	Properties
<i>Roles</i>	<i>D_AccessRoleList</i>	<i>Rule-Access-Role-Name</i>	<i>pyAccessRole</i> <i>pyLabel</i> <i>pyRuleSet</i>
<i>Case Types</i>	<i>D_CaseTypeList</i>	<i>Rule-Obj-Class</i>	<i>pyClassName</i> <i>pyLabel</i>
<i>Data Types</i>	<i>D_DataTypeList</i>	<i>Rule-Obj-Class</i>	<i>pyClassName</i> <i>pyLabel</i>
<i>Privileges</i>	<i>D_PrivilegeList</i>	<i>Rule-Access-Privilege</i>	<i>pyPrivilegeName</i> <i>pyLabel</i>
<i>Access When</i>	<i>D_AccessWhenList</i>	<i>Rule-Access-When</i>	<i>pBlockName</i> <i>pyLabel</i>

To customize the privilege tab, do the following steps.

1. Create a report definition, data transform, or custom activity which returns the instances of Class name that you want, providing each Property.
 2. Save a copy of the Source data page into your application's ruleset, to override the default instance.
 3. Change the source of your copy of the Source data page to reference the report definition, data transform, or custom activity that you created in step 1.
- [Access Manager](#)
 - [Granting privileges to an access role](#)
 - [Standard privileges](#)

Enabling a user to perform other users' assignments

Access Manager makes it easy to authorize users to perform work on other users' assignments of a particular case type. Pega Platform includes predefined **Access When** rules (conditions) that take into account the operator's organization and extend authorizations based on organizational structure. For example, you can enable an operator to work on a case assigned to any other operator in an organization, or assigned to a manager in the organization.

An operator may not perform an action in Pega Platform unless explicitly granted authorization by a privilege. (Some standard roles are delivered with privileges assigned.) Access Manager lets you assign a predefined **Access When** condition to a user in an access group specifically for assignments of a particular case type. For assignments, case types derive from the *Work*-class.

In Access Manager, you set authorizations for assignments by modifying the **Perform** row for a given case type and role. This adds a *Perform* privilege to the role for cases of that class (the case type you are authorizing). Operators in the access group are also granted the privilege. You can grant access based on a standard **Access When** condition.

- [Granting conditional access to all users in an access group](#)
- [Granting conditional access to one user](#)
- [Standard organization-based Access When conditions](#)

Granting conditional access to all users in an access group

In Access Manager, you can authorize all the users in an access group to perform work on another user's assignments.

In this example, you are a department manager working with a Purchasing application that includes a case type called *Purchase Request*. Operators in the *ShippingReceiving* access group can perform work only on Purchase Request cases assigned to them. However, you are going on vacation for two weeks, and want to authorize all the users in the access group to work on your purchase request assignments in your absence.

1. Navigate to Access Manager: click **Configure Org & Security Access Manager Work & Process**.
2. Expand the **Purchase Request** case type and click an icon in the **Perform** row for a role. You can select any role.
3. Select the appropriate **Access When** condition. In this example: *pyAssignedToMyManager*.

Result: This applies the **Access When** condition to all users in the *ShippingReceiving* access group. As long as the modified role is included in the access group (and the user is defined as reporting to you), each can perform work assignments of case type Purchase Request assigned to you.

- [Standard organization-based Access When conditions](#)

Granting conditional access to one user

In Access Manager, you can authorize a single user in an access group to perform work on another user's assignments.

In this example, you are a department manager working with a Purchasing application that includes a case type called *Purchase Request*. Operators in the *ShippingReceiving* access group can perform work only on Purchase Request cases assigned to them. However, you are going on vacation for two weeks, and want to authorize a single user in the access group to work on your purchase request assignments in your absence.

1. Create a new access group named *ShipRecPlusManager* as a copy of the *ShippingReceiving* access group.
2. Create a new role as a clone of the **Shipping & Receiving** role. In this example: *ShipRecPlusMgrRole*.
3. Add the new role to the new access group.
4. Add the new access group to the operator and set it as the default for the operator.
5. In Access Manager, edit this new role and access group and conditionally grant the perform privilege using the **Access When** rule.

Result: When the privilege is granted, the user assigned *ShipRecPlusManager* can view and work on Purchase Request cases assigned to you, as long as the **Access When** condition is met (you are the manager) and as long as the user is assigned to the **ShipRecPlusManager** access group.

- [Standard organization-based Access When conditions](#)

Standard organization-based Access When conditions

Access Manager includes the standard rules for granting authorization to a role, based on the organizational structure.

In Access Manager, selecting this Access When condition	Enables operators with this role to perform assignments when the work item is assigned to ...
<i>Work-.pxAssignedToMyUnit</i>	the current operator's unit.
<i>Work-.pxAssignedToMyUnitStaffManager</i>	the current operator's unit Staff Manager.
<i>Work-.pxAssignedToMyUnitCostCenterManager</i>	the current operator's unit Cost Center Manager.
<i>Work-.pxAssignedToMyDivision</i>	the current operator's division.
<i>Work-.pxAssignedToMyDivisionCostCenterManager</i>	the current operator's division Cost Center Manager.
<i>Work-.pxAssignedToMyOrganization</i>	the current operator's organization.
<i>Work-.pxAssignedToMyOrganizationCFO</i>	the current operator's organization CFO.
<i>Work-.pxAssignedToMyDirectStaff</i>	an operator that reports to the current operator.
<i>Work-.pxAssignedToMyManager</i>	the current operator's manager.
<i>Work-.pxAssignedToMyWorkGroupManager</i>	the current operator's work group manager.
<i>Work-.pxAssignedToMyWorkGroup</i>	the current operator's work group.
<i>Work-.pxAssignedToMyStaff</i>	an operator under the reporting chain of the current operator.

- [Enabling a user to perform other users' assignments](#)

Turning on privilege inheritance for access roles

You can turn on privilege inheritance for access roles to simplify how you define privileges and access settings that are relevant in multiple classes.

1. In the header of Dev Studio, click **Configure Org & Security Tools Security Role Names**.
2. Click an access role name and open for editing.
3. On the **Role** tab, select the **Inherit privileges within class hierarchy** check box.
4. Click **Save**.

- [Privilege inheritance for access roles](#)
- [Controlling role evaluation for access groups](#)
- [Configuring an access role](#)

Privilege inheritance for access roles

Privilege inheritance simplifies the process of defining privileges and access settings that are relevant in multiple classes.

When determining a user's access rights to a class, Pega Platform searches for Access of Role to Object (*Rule-Access-Role-Obj*) rules that are relevant to the target class and to the access roles listed in the user's access group, and considers the privileges and access settings granted or denied in those rules.

Privilege inheritance lets you define access rights within a class hierarchy more easily and economically. When privilege inheritance is enabled within an access role, the search for relevant Access of Role to Object rules begins with the target class where access is requested. If no relevant rule is found that grants or denies access for the role, the search continues for relevant Access of Role to Object rules in the parent class, and continues up the class hierarchy until a relevant rule is found.

Privilege inheritance lets you avoid having to define Access of Role to Object rules at multiple levels of a class hierarchy when the privileges and access settings for a class are the same at multiple levels.

The **Inherit privileges within class hierarchy** option on a role determines whether privilege inheritance is enabled for that role.

Example

As a security administrator, you need to restrict user access to a feature called *NewJob*.

To simplify this process, you can:

1. Set up the *Work-HRApps-NewJob* rule so that it is protected by a privilege.
2. For the HRApps:Users access role, enable the **Inherit privileges within class hierarchy** option on the Access Role rule form.

When a user attempts to create a New Job case, the system begins by checking the current class for a valid value for that user, for that privilege (in other words, the system seeks a valid instance of the Access of Role to Object rule for the privilege).

A valid Access of Role to Object rule instance specifies the privilege with a non-blank access value. The access value determines whether or not the user is granted permission.

If a valid instance is not found in the current class, the system continues searching the class hierarchy until it finds a valid instance. If the search exhausts all possibilities without finding a valid instance, the user is not granted permission.

- [Turning on privilege inheritance for access roles](#)
- [Controlling role evaluation for access groups](#)

Requiring a privilege for a rule

You can update a rule to specify that a privilege is required to access or use that rule. Only certain rule forms include the **Privilege** field, and for the rules that do, entering privileges is optional.

The following table lists the rules that require privileges and the function of the privilege for each of the rules:

Rule	Privilege function
<i>Activities</i>	Only users with the privilege can execute the activity.
<i>Attachment category rules</i>	Only users with the privilege can perform certain operations on an attachment.
<i>Correspondence</i>	Only users with the privilege can use the rule.
<i>Flows</i>	Only users with the privilege can start the flow.
<i>Flow actions</i>	Only users with the privilege can select certain actions.
<i>Parse structured rules</i>	Only users with the privilege can execute rules.
<i>Report definition rules</i>	Only users with the privilege can produce the report or chart.

Note: There are two ways to require a privilege on a rule:

- Certain rules always require a privilege.
 - For a specific use case, you can require a privilege.
1. Open the rule form for the rule that requires the privilege.
 2. On the **Security** tab, in the **Privilege name** field, enter the privilege name.
 3. Click **Save As**.
 4. Click **Create and open**.

For example: For example, to add the *AllFlowActions* privilege to the *ShowDescription* flow action, complete the following steps:

1. In the navigation pane of Dev Studio, click **Records Process Flow Action**.
2. In the **Flow Action** tab, in the **Action Name** column, click **Show Description**.
3. Click the **Security** tab.
4. In the **Privilege name** field, choose **AllFlowActions**.
5. Click **Save As**.
6. Click **Create and open**.

- [Understanding role permissions and privileges in Access Manager](#)
- [Creating a privilege](#)

Viewing the roles that have a privilege

To understand who has various privileges, you can view the roles that have a privilege. You can do this in various ways. The way you view roles and privileges depends on your familiarity with role and privilege names.

- If you know the privilege name, complete the following steps to view all roles that have the privilege.
 - a. In the navigation panel, click **Records Security Privilege** and open the privilege you want to view.
 - b. On the **Roles** tab, view the roles that have the privilege.
- If you do not know the privilege name, use Access Manager to view privileges for the roles in your application.

- [Reviewing user privileges for a role by using Access Manager](#)
- [Granting privileges by using Access Manager](#)
- [Understanding role permissions and privileges in Access Manager](#)
- [Creating a privilege](#)
- [Configuring an access role](#)

Creating a property security rule

Create a property security rule so that operators who attempt to access the property in the Report Browser must have at least one of the privileges specified on the rule.

1. In the header of Dev Studio, click **Create Security Property Security**.
2. Enter a short description.
3. Enter the property name and applies-to class of the property to be secured.
4. Click **Create and open**.
5. On the **Privilege** tab, click the **Add a privilege** icon, and in the **Privilege** field, press the Down Arrow key and select a privilege.
The applies-to class for the privilege must be valid for the applies-to class of the property security rule.
6. **Optional:** Repeat step 5 to add more privileges.
7. Click **Save**.

- [Property security rules](#)
- [Understanding role permissions and privileges in Access Manager](#)

Property security rules

Property security rules enable you to restrict operator access to properties by privilege. To view a report that references the secured property, operators must have at least one of the privileges specified on the property security rule.

The Report Browser prevents users who lack all of the listed privileges from viewing or editing reports that reference a property that is secured with a property security rule. This includes properties that are referenced through inheritance.

At run time, the system finds the relevant property security rules by doing the following steps:

- Filter the candidate rules based on the requestor's ruleset list.

- If no matching rule is found in the starting class, search for candidates in ancestor classes in the class hierarchy.

Circumstances and time-based qualifications are not available for property security rules.

- [Standard privileges](#)
- [Report definitions](#)
- [Managing access roles](#)
- [Privilege-Check method](#)

Configuring a property security rule

Enter the privilege an operator needs in order to access the property secured by this rule. Operators who attempt to access the property in the Report Browser must have at least one of the privileges specified on the property security rule.

Before you begin: You must complete the following task before you can configure a setting: [Creating a property security rule](#).

1. Create a property security rule, or open an existing instance from the navigation panel by clicking **Records Security Property Security** and selecting an instance.
2. Click the **Add a privilege** icon, and in the **Privilege** field, press the Down Arrow key and select a privilege.
The applies-to class for the privilege must be valid for the applies-to class of the property security rule.
3. **Optional:** Repeat step 2 to add more privileges.
4. Click **Save**.

- [Property security rules](#)

Generating "Work & Process" application authorization settings documentation

Access Manager makes it easy to generate a report of your application security model.

You can select one access group, or all access groups, and generate a file showing an expanded view of all items in the Work & Process tab. The document indicates the level of access — full, none, or conditional — to all case types, process flows, flow actions, and assignments permissions for your application.

Each case type represents a work class. The System Authorization report shows the case type short description followed by the class name. Rows beneath it are operations on instances of the case type that can be secured with Access Manager: Open, Run Reports, Modify, Delete, View History, Perform (other operators' assignments), plus any operations performed within process flows or flow actions.

The layout is similar to the display of Access Manager itself.

You can generate the following reports:

- [Single Access Group report](#)
- [All Access Groups report](#)

Single Access Group report

Access Manager makes it easy to generate a report of your application security model. The Single Access Group report provides data for just one access group

The single access group System Authorization report displays all work and process items expanded. This example shows one access group consisting of seven roles:

Example: Single Access Group report

Example: Single access group report

SYSTEM AUTHORIZATION									
CASETYPE	ACCESS GROUP	ROLES							
		PurchaseFW Administrator	System Administrator Level 4 (Pega)	Security Admin	Purchase Requestor Role	Purchase Manager Role	Purchasing Role	Shipping & Receiving Role	Accounts Payable Role
Program Fund									
PurchaseFW/PurchaseFW/Work/Purchase/ProgramFunding									
Open									
Run Reports									
Modify									
Delete									
View History									
Perform									
Process Flows									
Approvals									
CloseProgramFunding									
Disbursement									
Get Cascading Approval									
Program Fund									
ProgramFundEntry									
Flow Actions									
Enter Program Fund									
Review And Submit For Approval									
Approve									
Reject									
Continue Process									

The second column of the report, Access Group, displays icons showing the access group's aggregate authorization value. In this example, all members of the access group PurchaseFW:Administrators have full access to all operations on instances of the Program Fund class.

An example single access group report with callouts

All users in this access group are authorized to **open**, **modify**, etc. instances of this case type.

Full access granted to the System Administrator role, the most permissive setting, aggregates to full access for all members of the access group.

CASETYPE	ACCESS GROUP	ROLES			
		PurchaseFW Administrators	System Administrator Level 4 (Pega)	Security Admin	Purchase Requestor Role
Program Fund					
PurchaseFW/PurchaseFW/Work/Purchase/ProgramFunding					
Open					
Run Reports					
Modify					
Delete					
View History					
Perform					
Process Flows					
Approvals					
CloseProgramFunding					
Disbursement					
Get Cascading Approval					
Program Fund					
ProgramFundEntry					
Flow Actions					
Enter Program Fund					
Review And Submit For Approval					
Approve					
Reject					
Continue Process					

Top-row icons indicate permissions for the role are all full access, no access, or a combination of full, conditional, and none.

The aggregate level of access is equivalent to the most permissive authorization granted to a role in the access group: in this example, System Administrator Level 4.

Use the access group aggregate values to determine whether authorizations for roles in the access group should be made more or less permissive.

Aggregate values for an access group indicate the following:

- — No operators in the access group can access the item.
- — All operators in the access group have full access to the item.
- — Combination of conditional and possibly "no access" among operators in the access group.

When one access group is selected, Access Manager displays this aggregate value to the left of each work and process item:

Program Fund

	Open
	Run Reports
	Modify
	Delete
	View History
	Perform
Process Flows	
	Approvals
	CloseProgramFunding

For more information on aggregate values, see [Access Manager — Getting Started](#).

Generating a single access group report

- Navigate to Access Manager: click **Configure Org & Security Access Manager Work & Process** .
- Select an access group from the Access Group menu.
- Click **Export Authorizations**.

Note: If the selected application access group(s) has a large number of case types and/or roles defined it may take a few minutes for the report to be generated.
- Click the new browser tab opened by Pega Platform.
- On the File Download dialog box, click **Open** or **Save** to view the report, or **Cancel** to discard it.

All Access Groups report

The all access groups System Authorization report displays all work and process items expanded. Each column shows the aggregate value of authorization for the access group for each case type item.

As in the single access group report, the Case Type column displays the case type's short description followed by the class name.

Beneath the class name header are rows for each of the operations users can perform on instances of the case type, and by column, the aggregate level of access for all users in each access group.

This example shows a section of a report for an application with three access groups:

A report for an application with three access groups

All users in this access group are authorized to open, modify, etc. instances of this case type

No users in this access group are authorized to delete instances of this case type

No users in this access group are authorized to act on any instances of this case type

CASETYPE	ACCESS GROUPS		
	AES Remote Access Group	PRPC Reference Access Group For Work Managers	PRPC Reference Access Group For WorkUsers
Application Express PegaAccel-Task-GenerateApp-ExpressApp			
Open			
Run Reports			
Modify			
Delete			
View History			
Perform			
Process Flows			
Generate an Application (screenflow)			
Flow Actions			
CreateObjects step for Application Express			
CreateProcesses			
ppUserCamtGenerateScreen			

Top-row icons indicate users in the access group all have full access, no access, or a combination of full and none.

The top row indicates only whether the aggregate level of access for the users in the access group is consistently full access or no access for all operations on case type instances, or whether there is a combination of access levels.

Generating the all access groups report

- Navigate to Access Manager: click **Configure Org & Security Access Manager Work & Process** .
- Select **All Access Groups** from the Access Group menu.
- Click **Export Authorizations**.

Note: If the selected application access groups have a large number of case types and/or roles defined it may take a few minutes for the report to be generated.
- If a **File Download** dialog box does not appear, click the new browser tab opened by Pega Platform.
- On the File Download dialog box, click **Open** or **Save** to view the report, or **Cancel** to discard it.

Adding custom tools to Access Manager

You can add custom tools that are created by your developers to Access Manager.

This article describes an example of adding a hypothetical custom tool, Worklist Counter, to Access Manager. The tool queries a database to determine the number of items in a worklist, and the counter's results can be displayed in a manager's dashboard.

In this example, the customizable data transform rule *pyPopulateTools* is edited to add Worklist Counter so that security administrators can set user authorizations to it in Access Manager.

The standard **Tools** tab contains a system-defined data transform rule that applies *pyPopulateTools* to display the tools specified there just below the standard tools.

Note: All Pega Platform tools require a privilege. This example assumes that the Worklist Counter tool requires the *CanUseWorklistCounter* privilege that is provided by Pega Platform.

Adding a Worklist Counter tool to Access Manager

This example shows how to add a custom tool by modifying the data transform rule *pyPopulateTools* to do the following tasks:

- Create an expandable section for all custom tools.
- Create a section for a subcategory Worklist tools.
- Create an entry for the tool that specifies its class, privilege, and display features.

Data transform hierarchy

Sections and tools in the Access Manager UI reflect page lists that are nested in the data transform. The hierarchy for your custom tools follows that of the system-defined data transform that generates the standard Tools display, as shown in the following figure:

Hierarchy tools display

Top-level section

Sub-category

Tool

Tool

Comment

@***** Administrative & development *****

Append and Map to

pyRulesAndToolsList

a new page

Set

pyLabel

equal to

Administrative & development

Set

pyExpanded

equal to

True

Comment

@-----"Application distribution"-----

Append and Map to

pyRulesAndToolsList

a new page

Set

pyLabel

equal to

Application distribution

Append and Map to

pyRulesAndToolsList

a new page

Set

pyNote

equal to

View & download SQL during ap

Set

pyPrivilege

equal to

ViewSchemaOnImport

Set

pyClassName

equal to

@baseclass

Set

pyLabel

equal to

Download SQL during import

Append and Map to

pyRulesAndToolsList

a new page

Set

pyNote

equal to

Enable automatic schema chang

Set

pyPrivilege

equal to

SchemaImport

Set

pyClassName

equal to

@baseclass

Set

pyLabel

equal to

Enable automatic schema chan

Append and Map to

pyRulesAndToolsList

a new page

For

Base section

Note: When working with a data transform, use the right-click menu for commands such as **Add Sibling** or **Add Child**. You can also drag a row's handle (the blue dot in the **Action** column) to nest items or move them up or down.

The sections in the data transform shown above render the following section in Access Manager:

Administrative & development section of Access Manager

Administrative & development	
Application distribution	
Download SQL during import	
Enable automatic schema changes	

- [Creating an expandable section for custom tools](#)
- [Adding a subsection for similar tools](#)
- [Adding a custom tool](#)

Creating an expandable section for custom tools

This procedure uses the example scenario from [Adding custom tools to Access Manager](#). In *pyPopulateTools*, follow the hierarchy displayed in [Adding custom tools to Access Manager](#), starting with the top-level section.

1. Open the *pyPopulateTools* data transform.
2. Add a **Comment** step.
 - a. In the **Action** column, select **Comment**.
 - b. Enter your text. **For example:** @***** Custom Tools *****@
3. Add a new page row.
 - a. In the **Action** column, select **Append and Map to**.
 - b. In the **Target** column, enter .pyRulesAndToolsList.
 - c. In the **Relation** column, select **a new page**.
4. Complete two **Set** rows for the top-level expandable section. In the **Action** column, select **Set** and then enter the following:

Row	Target column value	Source column value
1	.pyLabel	The section name to be displayed in Access Manager. For example: "Custom Tools"
2	.pyExpanded	"True"

For example:

An example of an expandable top-level section for custom tools might look like the following figure:

Expandable top-level section

ACTION	TARGET	RELATION	SOURCE
Comment	@*****Specify additional tools and functions here for display on the Tools tab in Access Manager*****@		
Comment	@***** Custom Tools*****@		
Append and Map to	.pyRulesAndToolsList	a new page	
Set	.pyLabel	equal to	"Custom Tools"
Set	.pyExpanded	equal to	"true"

- [Adding custom tools to Access Manager](#)
- [Adding a subsection for similar tools](#)
- [Adding a custom tool](#)

Adding a subsection for similar tools

This procedure uses the example scenario from [Creating an expandable section for custom tools](#).

You can group similar tools (for example, Worklist tools) in an expandable section.

To add a nested page and label, perform the following steps:

1. Under the **Set** row that specifies *pyExpanded* equals "true", add a **Comment** step.
 - a. In the **Action** column, select **Comment**.
 - b. Enter your text. **For example:** @***** Worklist Tools *****@
2. Add a new page row.
 - a. In the **Action** column, select **Append and Map to**.
 - b. In the **Target** column, enter .pyRulesAndToolsList.
 - c. In the **Relation** column, select **a new page**.
3. Add a new page row.
 - a. In the **Action** column, select **Set**.
 - b. In the **Target** column, enter .pyLabel.
 - c. In the **Source** column, enter the subcategory name. Be sure to include the quotation marks. **For example:** "Worklist tools"

- [Adding custom tools to Access Manager](#)
- [Creating an expandable section for custom tools](#)

- [Adding a custom tool](#)

Adding a custom tool

This procedure uses the example scenario from [Adding a subsection for similar tools](#).

1. Right-click the handle (blue dot) of the **Comment** row for the section that you want to add tools to and click **Add Sibling Below**.
2. Add a new page row.
 - a. In the **Action** column, select **Append and Map to**.
 - b. In the **Target** column, enter `.pyRulesAndToolsList`.
 - c. In the **Relation** column, select **a new page**.
3. Complete four **Set** rows for the top-level expandable section. In the **Action** column, select **Set** and then enter the following:

Row	Target column value	Source column value Note: Be sure to include the quotation marks
1	<code>.pyNote</code>	The tooltip text to be displayed in Access Manager; a short description of the tool function. For example: "Retrieve number of items in workload."
2	<code>.pyPrivilege</code>	The privilege name. For example: "CanUseWorklistCounter"
3	<code>.pyClassName</code>	The privilege class name. For example: "@baseclass"
4	<code>.pyLabel</code>	The tool function name to be displayed on Access Manager. For example: "Worklist items counter"

4. Save the data transform.

For example:

The preceding example creates a custom tools section with a subsection for Worklist tools, as shown in the following figure:

Custom tools section with Worklist tools subsection

ACTION	TARGET	RELATION	SOURCE
• Comment	@*****Specify additional tools and functions here for display on the Tools tab in Access Manager		
• Comment	@*****Custom Tools*****		
▼ • Append and Map to	<code>.pyRulesAndToolsList</code>	a new page	
• Set	<code>.pyLabel</code>	equal to	"Custom Tools"
• Set	<code>.pyExpanded</code>	equal to	"true"
• Comment	@-----Worklist tools-----		
▼ • Append and Map to	<code>.pyRulesAndToolsList</code>	a new page	
• Set	<code>.pyLabel</code>	equal to	"Worklist tools"
▼ • Append and Map to	<code>.pyRulesAndToolsList</code>	a new page	
• Set	<code>.pyNote</code>	equal to	"Retrieve number of item"
• Set	<code>.pyPrivilege</code>	equal to	"CanUseWorklistCounter"
• Set	<code>.pyClassName</code>	equal to	"@baseclass"
• Set	<code>.pyLabel</code>	equal to	"Worklist items counter"

This data transform renders a section in Access Manager as shown in the following figure. (Refresh the display to see the changes.)

Access Manager section with Worklist tools subsection

▼ Custom Tools	
▼ Worklist tools	✖
✖ Worklist items counter	✖

What to do next: In Access Manager, secure the tool. For more information, see [Editing tools authorization for a single access group](#).

- [Adding custom tools to Access Manager](#)
- [Creating an expandable section for custom tools](#)
- [Adding a subsection for similar tools](#)

Attribute-based access control

You can restrict the ability of a user to view, modify, and delete instances of classes, or properties within classes. Use attribute-based access control (ABAC) to enforce row-level and column-level security in your application. Restrict access to cases and properties by using attribute-based access control (ABAC).

Access restrictions are enforced by defining access control policies. Conditions used in access control policies compare attributes in class instances to other information (typically, information about user's identity, organizational reporting relationships, or other security credentials that might be case-specific).

Access is permitted only when all relevant policy conditions are satisfied.

Encryption policies under ABAC do not have associated conditions. These encryption policies are used to unconditionally encrypt sensitive property values, and can be used together with other access control policies to conditionally obfuscate or mask these values within application user interfaces.

Attribute-based access control policies do not restrict the processing of client-based access control requests.

Attribute-based access control in Pega Platform

Two rule types (*Access Control Policy* and *Access Control Policy Condition*) are used to define policies for different types of actions (Read, Update, Delete, Discover, PropertyRead, PropertyEncrypt). The rule types compare property values in class instances to clipboard property values.

When multiple policies are defined or inherited for a specific class, the conditions for those policies are aggregated by combining the filter logic strings for the conditions and the AND operator. Access is permitted only if all conditions are satisfied. This type of access differs from how role-based access is determined, where a user with multiple roles is granted access if any of those roles permit it.

Access control policies are enforced in all Pega Platform features that access and manipulate data from the Pega Platform database or from Pega Platform search indexes. These features include all report rules, searches, operations on individual cases such as opening cases, custom SQL, and so on.

Access control policy enforcement exceptions

Access control policies specify conditions that must be satisfied for an operator or user to view any data for a class instance. To prevent these conditions from being circumvented by end users, the following exceptions are made:

- Access control policies can be defined only for instances of *Assign*-, *Data*-, and *Work*- classes.
- Access control policies defined on *Data*- classes are not enforced in search queries.
- Only read policies are enforced in custom SQL.
- Advanced search queries (for example, search queries that reference specific properties such as *pxObjClass:Work-MyProperty AND CustomerName:MyCorp*) are not allowed when access control policies are defined on any *Assign*-, *Data*-, and *Work*- classes.

Special considerations apply when access control policies are enforced in certain features that retrieve data for potential use by multiple end users who might have different credentials, such as node-scoped data pages and scheduled reports.

- [Access Control Policy rule](#)
- [Access Control Policy Condition rule](#)
- [Client-based access control](#)

Creating an access control policy

In the access control policy rule form, you define a policy that grants access to an object by evaluating selected conditions. For each rule, you can set one level of access, such as read, update, or delete, and the condition that defines whether the access is granted.

Before you begin:

- You must have the *pzCanManageSecurityPolicies* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.

Note: Access control policies only work when a rule is checked in. Therefore, any changes in a checked out rule will be ignored. When changing or modifying any ABAC policy rules, you must Check out the rule, make changes, Save those changes, then Check in the new rule for the changes to take effect. **Note:** You can create access control policies only for *Assign*-, *Data*-, *Index*- and *Work*- classes.

1. In the navigation panel, click **Records Security Access Control Policy** , and then click **Create**.
2. In the **Label** field, enter the policy name.
3. In the **Action** list, click the action.
 - **Read** – The user can open a case that meets the policy conditions or view data for the case in lists, reports, searches, and so on.
 - **Update** – The user can create a case that meets the policy conditions or update data for such a case.
 - **Discover** – The user can see limited information (defined by a developer) about a case that does not meet Read policy conditions, but does satisfy the Discover policy conditions.
 - **Delete** – The user can delete a case that meets the policy conditions.
 - **PropertyRead** – The user has restricted visibility to property values, including property values with read and update access.
 - **PropertyEncrypt** – The property is encrypted in the database, clipboard, logs, and search indexes. If no PropertyRead policy obfuscates the property, then the decrypted property value is visible to the user in a UI control. In report definitions, the property can be displayed in report results and can also be referenced on the left side of filter conditions that use the **Is equal** and **Is not equal** operators. It cannot otherwise be referenced in report definitions (for example, to sort, rank, or group results in SQL functions, and so forth). **Note:**
 - Properties specified in a **PropertyEncrypt** policy are encrypted unconditionally. Access control policy conditions are not used for **PropertyEncrypt**. Define **PropertyRead** policies to obfuscate or mask these values depending on who is viewing them. To get the cleartext value outside of UI controls (for example, in the background processing of cases), use the *@decryptPW* function.
 - You can define **PropertyEncrypt** access control policies for properties that are optimized for reporting only if the property type is equal to Text. To define a report filter for an encrypted property that is not Text, convert the values to text and store them in a Text property that is encrypted and optimized.
 - When you want to configure property encryption for embedded properties, then you need to specify a **PropertyEncrypt** policy at both the class level and at the property level for the Embedded page properties. For example, for the property *PageProp1.PageProp2.Property1* you need to define a **PropertyEncrypt** policy on the *pageProp1* class and at the *Property1* class.
 - When you create a property encrypt policy, you need to configure an encryption mechanism in order to encrypt your data. You can do this using a custom cipher or a key management system. For more information see:
 - [Creating a custom cipher](#).
 - [Key management system for application data encryption](#).
- Caution:** If you define a **PropertyEncrypt** policy for a property, make the column size greater than the number of characters of your longest cleartext value to avoid truncation of the exposed values. The required size of a **PropertyEncrypt** property depends on your cipher. In most cases, for 64 characters of cleartext, 255 characters are adequate for the encrypted value.
4. In the **Context** section in the **Apply to** field, enter a class.
5. In the **Add to ruleset** field, select a ruleset.
6. Click **Create and open**.
7. **Optional:** To prevent overriding the policy in a descendant class, on the **Definition** tab, select the **Disallow creation of a policy with the same name as a descendant class** check box.
8. If the action is not PropertyEncrypt, in the **Permit access if** field, enter the access control policy condition rule name.
9. If the action is PropertyRead or PropertyEncrypt, do the following steps.
 - Click **Add property** and select a property name that exists on the case type target.
 - If the action is PropertyRead, specify the masking method (Full Mask, Mask all but last 'N', or Mask all but first 'N').
10. Click **Save**.

- [Attribute-based access control](#)
- [Masking property visibility for users](#)
- [Access Control Policy rule](#)

Access Control Policy rule

You use access control policies to restrict user actions. In the Access Control Policy form, you define a policy that grants access to an object by evaluating the conditions that you specify. You can set one of four levels of access: read, update, discover, or delete.

For example, an access control policy for a sales automation application might restrict access to a sales account to the user who owns the account or to a user who is included in an exception list of non-owners who have access.

Note: You can create access control policies only for *Assign*-, *Data*-, *Index*- and *Work*- classes.

Access control policies are enforced everywhere in Pega Platform, not just within the UI, including all reporting rules, search, and custom SQL written by developers.

Rule resolution

Unlike role-based access controls, attribute-based access control policies use the system's full inheritance functionality. Access policy rules can be inherited from multiple classes, in which case the relevant policies are combined and access is allowed only when all such policy conditions are satisfied.

- [Attribute-based access control](#)
- [Using security attributes markings](#)
- [Creating an access control policy](#)
- [Excluding properties with access control policies from search results](#)

Masking property visibility for users

You can restrict access to values of one or more properties by using a property-level access control policy. By using various masking options in the access control policy, you can display partial information about a value to users who are not allowed to see the full value.

Before you begin:

- You must have the *pzCanManageSecurityPolicies* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.

Property-level policies can be enforced only on optimized properties. Also, the policies cannot be enforced in some Pega Platform features. For example, policies cannot be enforced in features that retrieve data for potential sharing across multiple users whose credentials are not available at the time of retrieval, and whose credentials might vary and might change following retrieval, such as node-scoped and cluster-scoped data pages and scheduled reports. These same limitations also apply to row-level policies.

1. In the navigation panel, click **Records Security Access Control Policy**, and then click **Create**.
2. In the **Label** field, enter the policy name.
3. In the **Action** list, click **PropertyRead**.
4. In the **Context** section in the **Apply to** field, enter a class.
5. In the **Add to ruleset** field, select a ruleset.
6. Click **Create and open**.
7. **Optional:** To prevent overriding the policy in a descendant class, on the **Definition** tab, select the **Disallow creation of a policy with the same name at a descendant class** check box.
8. In the **Permit access if** field, enter the condition rule name under which the access is permitted.
9. Click **Add property**.
10. In the **Property** field, select the property to mask.
You can mask DateTime, Integer, and Text property types.
11. In the **Restriction Method** field, select one of the following masking options for the property.

DateTime

- **Mask entire Date** – All the date information is replaced.
- **Mask Year** – Only the year information is replaced.
- **Mask Day and Month** – Only the day and month information is replaced.

Integer

- **Mask with N digits** – The whole value is replaced with a defined number of characters.

Text

- **Full Mask** – The whole text is replaced with one character.
- **Mask all but last 'N'** – The whole value is replaced, except for the last N characters.
- **Mask all but first 'N'** – The whole value is replaced, except for the first N characters.

12. Click the **Switch to edit mode** icon.
13. In the **Masking and Formatting Options** form, fill out the required fields.
Note: When the value for a restricted property is NULL for a case, the value looks as though it is not set.

DateTime property type

1. Depending on the selected masking option, in the **Masking values** section, in the **Month**, **Day**, or **Year** field, select or enter the value to replace.
2. Click **Submit**.

Integer property type

1. In the **Masking digit** field, enter a digit, letter, or symbol to replace the property value.
2. In the **Number of digits** field, enter the number that is the number of times the digit, letter, or symbol appears in the property values.
3. Click **Submit**.

Text – Full Mask

1. In the **Masking character** field, enter a digit, letter or symbol that should be used to replace the property value.
2. Select the **Display length is fixed** check box or **Display length matches value** check box, to specify the length of the replaced property.
3. If you selected **Display length is fixed** check box, enter a digit in the **Display characters length** field to specify the length of the replaced property.
4. Click **Submit**.

Text – Mask all but last 'N' and Mask all but first 'N'

1. In the **Masking character** field, enter a digit, letter, or symbol to replace the property value.
2. In the **Number of unmasked character** field, enter a digit to specify the length of the characters that are not replaced in a property.
3. Select the **Display length is fixed** check box or the **Display length matches value** check box to specify the length of the replaced property.
4. Click **Submit**.

14. Click **Save**.

- [Creating an access control policy](#)
- [Creating an access control policy condition](#)

Configuring encryption for properties in embedded classes

To better secure your data, you could encrypt individual properties. This means you would need to create an ABAC policy type for property encryption, and encrypt the property in the embedded class.

For example, you might want to encrypt clients addresses and phone numbers to ensure that only certain people in the organization can access this information. When you configure property encrypt policies, you create an access control policy on both classes in which the property appears.

For example, if you want to encrypt a postal code, you encrypt the *embed class* and the *data-admin-operator-ID* class that correlate to those properties. In this case, you encrypt *embed-address.postalcode* and *data-admin-operator-id.address.postalcode*. **Note:**

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Security** category, and then click **Access Control Policy**.
3. In the list of policies, click **propertyencrypt**.
4. Click **Actions Create**.
5. In the **Label** field, enter a meaningful name for the new policy.
6. In the **Action** list, select **propertyencrypt**.
7. In the **Applies to** field, enter the class that you want to encrypt.
8. In the **Property** field, select the properties that you want to encrypt.
9. Click **Save**.

Creating an access control policy condition

You can define a set of conditions and comparison logic that you want to evaluate to grant access to an object.

Using the **Data Access** tab, you can build complex authorization models in which access restrictions for a class depend on the attributes that are present in the associated and indexed classes, along with the attributes in the current class. The **Data Access** tab is read-only, and any information that is displayed on the tab is input into the **Column source** field.

Before you begin:

You must have the *pzCanManageSecurityPolicies* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.

1. In the navigation pane of Dev Studio, click **Records**.
2. Expand the **Security** category, and then click **Access Control Policy Condition**.
3. Click **Create**.
4. In the **Label** field, enter the policy condition name.
5. In the **Context** section, in the **Apply to (class)** field, press the Down arrow key, and then select the rule to which the policy condition applies.
6. In the **Add to ruleset** field, select a ruleset.
7. Click **Create and open**.
8. **Optional:** To configure a filter logic string for the condition, click **Add conditional logic**, and then define the logic:
 - a. On the **Definition** tab, in the **Conditional logic** section, click **Add conditional logic** for situations where you need to apply different logic.
 - b. In the **WHEN** field, enter an Access When rule that evaluates whether the conditional logic should be used.
 - c. **Optional:** To enforce a policy condition, in the adjacent field, enter a filter logic string to apply when the Access When rule evaluates to true.
When the set of filters to be applied in an Access Control Policy Condition rule is determined conditionally by using Access When rules, leave the filter logic entry blank if you want to enforce no policy condition at all, for example, for certain highly privileged users.
 - d. In the **OTHERWISE** field, enter the filter logic string that is used when all the when rules evaluate to false.
9. On the **Definition** tab, in the **Policy Conditions** section, in the **Condition** field, enter a condition name.
10. In the **Column source** field, press the Down Arrow key and select a property from the *Apply To* class from the list.

Use the **Column source** field to add content from the *Applies to*, *associations* and *declarative index* classes in your policy conditions. When you select *Applies to*, *associations* and *declarative index* classes, this information auto-populates on the **Data Access** tab.

- In the **Relationship** list, click the comparison logic appropriate for the evaluated attribute type.

For Numeric attributes:

Attribute	Behavior
Is equal	The <i>Apply To</i> property value and comparison value are equal.
Is not equal	The <i>Apply To</i> property value and comparison value are not equal.
Is greater than	The <i>Apply To</i> property value is greater than the comparison value.
Is greater than or equal to	The <i>Apply To</i> property value is greater than or equal to the comparison value.
Is less than	The <i>Apply To</i> property value is lower than the comparison value.
Is less than or equal to	The <i>Apply To</i> property value is lower than or equal to the comparison value.

For String attributes:

Attribute	Behavior
Is equal	The <i>Apply To</i> property value and comparison values are equal. The comparison value can be a single value or a comma-delimited list.
Is not equal	The <i>Apply To</i> property value and comparison value are not equal.
All of	Both the <i>Apply To</i> property value and the comparison value are strings that consist of a comma-delimited list. The list does not contain any spaces within the string (except for spaces within a value), and all elements in the list are capitalized, for example BRAZIL,CANADA,FRANCE,GERMANY,SOUTH AFRICA,UK,USA. The condition is satisfied if every element of the list within the <i>Apply To</i> property value is also an element in the list within the comparison value.
One of	Both the <i>Apply To</i> property value and the comparison value are strings that consist of a comma-delimited list. There should be no spaces within the string (except for spaces within a value), and all elements in the list must be capitalized, for example: "BRAZIL,CANADA,FRANCE,GERMANY,SOUTH AFRICA,UK,USA". The condition is satisfied if at least one element of the list within the <i>Apply To</i> property value is also an element in the list within the comparison value.

For all attributes:

Is null	The <i>Apply To</i> property value is null.
Is not null	The <i>Apply To</i> property value is not null.

Note: If you select **Is null** or **Is not null** in the **Relationship** field, the **Treat Empty As Null** check box is automatically selected, which means that even empty values are considered null.

- In the **Value** field, enter the comparison values that you want the condition to check.
If you select **Is null** or **Is not null** in the **Relationship** field, the **Value** field is not active.
- Optional:** To define additional conditions, click **Add Condition** and repeat steps 7 through 10.
- Optional:** For multiple conditions, to define more complex Boolean operations, complete the **Conditional Logic** field.
By default, multiple conditions are combined by using the AND operator.
- Click **Save**.

- [Access Control Policy Condition rule](#)
- [Attribute-based access control](#)
- [Using security attributes markings](#)
- [Storing of operator security attributes](#)
- [Using One Of and All Of conditions](#)
- [Managing access control policy condition performance](#)
- [Managing hierarchical attributes](#)

Access Control Policy Condition rule

An Access Control Policy Condition rule defines a set of filters, and the filter logic combining them, for an access control policy. They describe the conditions under which the access type is granted to a property.

Each filter compares a column source (a property of the policy's class) to a target value. An example filter is *Case.RequiredClearance <= UserInfo.SecurityLevel*. Each set of filters compares a case attribute (property value) to any clipboard property value that you want. This comparison value typically represents information about the user attempting to access cases. The filter logic used to combine the filters uses the AND and OR operators and parentheses. You can enter multiple sets of filter logic values, each associated with a when rule, so that the filters enforced for a specific user are dynamically determined at run time.

The special comparison operators **All Of** and **One Of** can be used to compare two property values when each is a comma-separated list of one or more values. The comparison values that are referenced in policy condition filters must be existing Requestor properties or requestor-scoped data pages.

The following restrictions apply to column source properties:

- They must be top-level, optimized properties that are available as database columns and can be referenced in generated SQL. For best performance, consider indexing optimized properties that are referenced in policy conditions.
- They must be included among the custom search properties that are stored in the search index in a returnable form if they do not have a text data type or if the **All Of** or **One Of** comparison operator is used. **Caution:** When access policies are inherited by multiple classes, column source properties might need to be optimized and stored in a returnable form in the search index in each class where the policies are enforced. Also, when the list of custom search properties for a class changes, the search index must be rebuilt for the class on the **Search** landing page.

Do not enter case attributes or policy class property values in an Access When rule that is used for conditional logic because doing so causes invalid results or run-time failures.

Target values are restricted to a clipboard page reference or to a nonparameterized data page reference. Primary page properties are not allowed. Target values must be of the same data type as the column source.

- [Attribute-based access control](#)
- [Using security attributes markings](#)
- [Creating an access control policy condition](#)

Using security attributes markings

Attributes are unique security markings, which are assigned to objects and operators. Each attribute has a value associated with it, which means that a user must possess an attribute value to access an object.

Attribute types suggest how to compare the attribute values between the user and the object that the user is requesting access to, and to determine whether to grant access to the object.

These main data types represent an attribute value in the Pega Platform :

- **Single string value** – A simple string equality comparison is made to determine if the subject has access to the object.
- **A list of string values** – A multivalued data type represented by a comma-separated noun string with no spaces. The subject must have either all of the object's attribute values (All Of) or one of the attribute values (One Of).
- **Numeric value** – This attribute type is internally represented by an integer. A simple numeric comparison is made to determine if the subject has access to the object.

You can see the attributes that are assigned to a case on the *pyWorkPage* in the clipboard.

You can create attributes on data pages or work objects by adding special properties to them. These properties can then be referenced by the ABAC engine to enforce security policies.

- [Attribute-based access control](#)
- [Creating an access control policy](#)
- [Creating an access control policy condition](#)

Storing of operator security attributes

For the system of record, security attributes can be stored and maintained internally or externally to Pega Platform. You can access the operator security attributes that are used in policy conditions in several ways.

- If attributes are stored internally, then it is likely that they are stored in the operator record or a new data class, which makes them already accessible to policy conditions.
- If the attributes are stored externally, then these attributes need to be retrieved during login and stored somewhere in Pega Platform where policy conditions can access them.

Custom authentication allows the policy conditions to access the attributes. Depending on their requirements, users can choose to use the custom authentication activity or the **Mapping** tab on the Authentication Service form, or they can use both options. For example, mapping can be used to copy data stored in an LDAP directory directly in properties in the *Data-Admin-Operator-ID* class. If mapping is more complex, then an activity can be used to retrieve the attributes from the external location and copy them to a location within the application created in Pega Platform (for example, a requestor-scoped data page).

- [Attribute-based access control](#)
- [Using security attributes markings](#)

Using One Of and All Of conditions

The **One Of** condition and the **All Of** condition specify how to compare the multivalued attributes between the user and the object that the user requests, in order to determine whether to grant access. You can create attributes on cases to determine who is authorized to access the case.

- **One Of** condition – Requires one of the values in the object's property to match the subject's property.

For example, if a case has an attribute value "Red,Green", then the user with the attribute value "Red" or "Red,Yellow" is granted access. If the user has the attribute value "Purple,Brown", then access is denied. The order of the values is irrelevant.

- **All Of** condition – Requires all values in the object's property to match the subject's property. The subject can have more, but not fewer values.

For example, if a case has an attribute value "Red,Green", then the user with the attribute value "Red,Green" or "Red,Green,Yellow" is granted access. If the user has the attribute value "Red,Blue", the access is denied.

- [Attribute-based access control](#)
- [Creating an access control policy condition](#)

Managing access control policy condition performance

When you define an access control policy condition, use the relationship that gives the best performance for your data profile.

Consider the following performance factors when you define an access control policy condition.

- For policy conditions that use the **All of** and **One of** relationships, ensure that the **Column source** properties are optimized and included in returnable form in the custom search properties that are stored in the search index.
- Test the performance of your conditions by using realistic data. Pega Platform evaluates your filter by using database queries or Elasticsearch, depending on the report definition. Different relationships perform differently, depending on the filter and data profiles.
- For best performance when comparing a scalar value to a comma-delimited list of values, use the **Is equal** relationship.

However, if the number of comparison values in the **Is equal** relationship(s) in your policies is very large, queries against the database or the Elasticsearch index might generate error messages, because these data stores restrict the number of terms allowed in queries. The error message explains the cause of the problem.

When using database queries for **Is equal**, Pega Platform generates SQL statements with IN clauses, which your database might limit. You should identify this limit for your database, and use the **One of** relationship if you expect the limit to be exceeded in your application.

When using Elasticsearch for **Is equal**, about 1,000 comparisons are allowed by default. If you exceed this limit, you can either use **One of**, or increase the limit by updating the dynamic system setting *indexing/distributed/search_maxclausecount*.

- [Attribute-based access control](#)
- [Creating an access control policy condition](#)
- [Using One Of and All Of conditions](#)

Managing hierarchical attributes

An attribute with a specified order of values (hierarchy) is the main attribute type that defines the access level, by being assigned to objects and operators. The value of this attribute can be internally represented by an integer. A simple numeric comparison is made to determine if the subject has access to the object.

You can define the access level with hierarchical attributes in two ways:

- The attribute is represented by a string type property on the object and the user, with one of the text values: Senior Manager, Manager, or User. To achieve a hierarchy, you define a set of conditions, for example:

A	Operator.SecurityClearance = "Senior Manager"
B	Operator.SecurityClearance = "Manager"
C	Operator.SecurityClearance = "User"
D	.SecurityClearance = "Senior Manager"
E	.SecurityClearance = "Manager"
F	.SecurityClearance = "User"

The properties then have to be combined with the following logic:

A or (B and (E or F)) or (C and F)

- For convenience the hierarchical attributes can be represented by a numeric data type. The attribute values must be mapped to a top-level numeric property on both the object and the subject, for example:
 - Senior Manager=1
 - Manager=2
 - User=3

To determine the access level a single condition with a numeric comparison can be used, for example:

.SecurityClearance >= Operator.SecurityClearance

- [Attribute-based access control](#)
- [Creating an access control policy condition](#)

Reviewing access control policies

In your application, you can view access control policies that are based on a specific action a user can perform. By viewing these policies, you can determine whether any new policy needs to be added, changed, or deleted in your application.

Before you begin: To view the Policy Management landing page, you must have the *pzCanManageSecurityPolicies* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.

1. In the header of Dev Studio, click **Configure Org & Security Authorization Policy-Based Access Policy Management**.
2. In the **Application** list, select the application for which you want to view the access control policy.
3. In the **Target class** field, press the Down Arrow key, and then select the class to which the policy applies.
4. In the **Action** list, select the type of policy to view.
 - a. If you select **PropertyRead** in the **Action** field, enter the property name to view in the **Property** field.
5. Click **Search policies**.
6. Review the results, and then update the policy as needed.

If any results are found:

 - a. Hover over the **Policy condition** column to display the conditions that are applied in the policy.
 - b. If you selected a **PropertyRead** action, click the value in the **Policy scope** column to display the properties that the policy secures and which restriction method the policy applies to those properties.
 - c. In the **Actions** column, choose one of the following actions for each policy.
 - Open the policy rule form.
 - Create a policy in the target class that you specified in the search.
 - Withdraw the policy.

If no results are found, you can create a new policy by clicking **Create policy**.
7. If you make any changes to the policies (for example, the policy name or action), click **Search policies** again to refresh the result.

- [Attribute-based access control](#)
- [Verifying access control policies](#)

Verifying access control policies

You can verify access control policies by testing them to see whether they grant or deny access to a specific case for a specific user. By verifying an access control policy, you can see whether a user has required access to a case, and decide whether any changes need to be made to a policy.

Before you begin:

- To view the Policy Verification landing page, you must have the *pzCanManageSecurityPolicies* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.
1. In Dev Studio, click **Configure Org & Security Authorization Policy-Based Access Policy Verification**.
 2. In the **Target class** field, press the Down Arrow key, and then select the class to which the policy applies.
 3. In the **Action** list, select an action to verify in the policy.
 4. If the target class is a *Work-* class, then in the **Case ID** field, enter a case ID to verify in the policy.
 5. If the target class is a *Data-* class, then in the **Class keys** section, enter an ID of a data type (data instance ID) to verify the policy, for example, an employer's name.
 6. In the **Operator Id** field, press the Down Arrow key, and then select a user to evaluate against the policy.
 7. If an operator has more than one access group, in the **Access Group** list, select the access group that the policy should verify.
 8. Click **Verify policies**.
 9. Review the results, and then update the policy as needed.

Note: If there are no results for the user, click **Create policy** to add a policy to the target class.

 - a. In the Actions column, click **View policy condition results** to display the condition logic that is used in the policy.
 - b. In the **Status** column, review whether the operator has passed or failed each condition.
 - c. To update the policy, in the **Actions** column, click **Open policy**.

- [Attribute-based access control](#)
- [Creating an access control policy condition](#)
- [Masking property visibility for users](#)

Understanding Discovery features for access control policies

Access control policies support discovery features that allow users to view limited, customizable information about class instances that fail read policies but satisfy discover policies. Users cannot open or interact with the discoverable instances, but they can see that the instances exist and they can evaluate whether they might need access to certain instances.

To make discovery features available to users, the *pysDiscoveryEnabledForOperator* Access When rule must evaluate to true. These features apply to instances of *Work-* and *Data-* classes when data is retrieved from the Pega database, but apply only to instances of *Work-* classes when data is retrieved from the search index.

The following Discovery gadgets, which are section rules, support discovery features:

- *pxDiscoverableItems* – This section displays discoverable instances for report definitions. By default, this section is included in the Report Viewer when it displays results for list reports. This section can be displayed in the results for summarized reports if you set the dynamic system setting *DiscoverableItemsIncludedForSummaryReport* to true. **Note:** In the Report Viewer, this Discovery gadget shows only records that have the condition *NotReadAndDiscover*.
- *pxDiscoverableSearchItems* – This section displays discoverable class instances for search. By default, this section is included when search results are displayed.

The Discover gadget is shown in the Report Viewer or in search when all of the following are true:

- Read policies are in force for at least one of the classes involved (either defined on the class or inherited by it), and there are policy conditions for those policies (the policy conditions do not have empty filter logic strings).
- Discovery policies are in force for at least one of the classes that also have Read policies, and there are policy conditions for those policies.
- The number of instances that fail the Read policies but satisfy the Discovery policies is greater than zero.

Each of the Discovery gadgets displays a link with the number of class instances for the report or search that fail read policies but meet discover policies for the user. Users can view a list of the discoverable instances by clicking the link. The information that is shown for these instances is determined by the *pyDefaultDiscoverableReport* report definition rule for a class.

Developers can customize these sections to change their labels and behavior. You can also include these sections in other parts of an application's user interface. For example, you can include the *pxDiscoverableItems* section above a list control that is populated by a report definition.

- [Attribute-based access control](#)
- [Customizing of Discovery gadgets](#)
- [Enabling discovery features for access control policies](#)
- [Adding a Discovery gadget to a customized user interface](#)

Customizing of Discovery gadgets

You can customize some of the information in the Discovery gadgets so that the information that you specify is displayed for users.

You can customize the Discovery gadgets in the following ways:

- Specify a custom source for the data that is shown when a user clicks the link to view discoverable instances. You can configure the custom source as a report definition, a virtual report definition, or a property reference.
- Customize the *pyDefaultDiscoverableReport* report definition rule template so that any class can show different information about discoverable instances in that class.
- Override the field values (*pyDiscoverableItems* and *pyDiscoverableItemsNoCount*) for the text that is displayed in the Discovery gadget in the application ruleset.
- Change the harness headers that display the discoverable instances in reports or in search by overwriting the field values for *pyDiscoverableItemsHeader* and *pyDiscoverableItemsHeaderForSearch*.
- If the number of discovered instances is zero, specify whether you want the gadget to show a link for discovered instances.
- [Understanding Discovery features for access control policies](#)

Enabling discovery features for access control policies

Access control policies support discovery features that allow users to view limited, customizable information about class instances that fail Read policies but satisfy Discover policies. To enable these features for users, the *pyIsDiscoveryEnabledForOperator* Access When rule must evaluate to true. You can enable discovery features for all users of an application, or for particular operators.

Follow these steps to enable discovery for all operators.

1. In the navigation panel, click **Records Security Access When**.
2. In the **When Name** column header, click the **Filter** icon and search for *pyIsDiscoveryEnabledForOperator*.
3. Open the *pyIsDiscoveryEnabledForOperator* Access When rule that is displayed by clicking it.
4. On the **Conditions** tab, double-click the condition.
5. In the **Condition** dialog box, change the comparator to the equal sign (=), and click **Submit**.
6. Save your changes.

- [Creating an access control policy](#)
- [Adding a Discovery gadget to a customized user interface](#)
- [Understanding Discovery features for access control policies](#)

Adding a Discovery gadget to a customized user interface

A Discovery gadget enables users to view limited information about selected cases that they do not have Read authority to access. You can include the section rule for a Discovery gadget in any section of your user interface, such as in the layout of a customized search gadget, or above a list control that is populated by a report definition.

Note: For an end user to see a list of discoverable instances, you must have access control policies defined, and the *pyIsDiscoveryEnabledForOperator* Access When rule must evaluate to true. For more information, see [Creating an Access Control Policy](#) and [Enabling discovery features for access control policies](#).

1. In Dev Studio, open the **Application Explorer**.
2. Navigate to and open the user interface that you want to add the Discovery gadget to.
3. Click **Structural Embedded section**, and drag the section into the section where you want to add the Discovery gadget.
4. If you are adding the Discovery gadget to a list control, complete the following steps.
 - a. In the **Section** field, enter *pxDiscoverableItems*.
The *pxDiscoverableItems* section is defined in the *@baseclass* class so you can include this section in any section.

The *pxDiscoverableItems* section shows only records that have the condition *NotReadAndDiscover*.
 - b. Click the **View properties** icon to configure the input parameters.
 - c. On the **Parameters** tab, select the source of the discoverable instances.
 - **Report Definition** – Specify the Applies To class and the name of the report definition rule. When a report definition is the source, the columns in the report are replaced with the columns that you specify in the *pyDefaultDiscoverableReport* report definition rule. When the report runs, the corresponding results are displayed. The report definition name overwrites the default column source in the *pyDefaultDiscoverableReport* report definition rule.
 - **Virtual RD Page** – Enter the name of the virtual report definition page. When a virtual report definition page is the source, the columns in the report are replaced with the columns from the *pyDefaultDiscoverableReport* report definition rule to display discoverable instances.
 - **Property Reference** – Specify the property reference. When a property reference is the source, the columns in the report are replaced with the columns from *pyDefaultDiscoverableReport* to display discoverable instances.
 - d. Select the **Show discoverable items count** check box.
 - e. Click **Submit**.
5. If you are adding the Discovery gadget to a search gadget, complete the following steps.
 - a. In the **Section** field, enter *pxDiscoverableSearchItems*.
The *pxDiscoverableSearchItems* section is defined in the *@baseclass* class, so you can include this section in any section.
 - b. Click the **View properties** icon to configure the input parameters.
 - c. On the **Parameters** tab, select the source of the discoverable instances.
 - **Report Definition** – Specify the following information:
 - In the **Applies To** field, enter *Work*.
 - In the **RD name** field, enter *pyWorkSearch*.
 - **Virtual RD Page** – Enter the name of the virtual report definition page.
 - **Property Reference** – Specify the property reference that you want to use to store the virtual report definition page name.
 - d. For the **Search String**, **Case Type List**, **Include Attachments**, **Type Of Work**, and **Updated On** fields, refer to the clipboard page that is used as the source for the values to enter.
 - e. Click **Submit** and save your changes.
 - f. **Optional:** If **All Of** or **One Of** attributes are used in the policy conditions, you must add all the properties that are used as column sources in the Custom Search Properties.
 1. In Dev Studio, click **Configure System Settings Search**.
 2. For the **All work** index type, click **Re-index**.
 3. Select the **Only Classes listed below** radio button, and specify the work classes to reindex.
 4. Click **OK**.
 5. Click **Submit**.
 6. In the navigation panel, click **Records SysAdmin Custom Search Properties**.
 7. Click **Create**.
 8. Enter a short description and a class name, and click **Create and open**.
 9. Click the **Gear** icon for the property.
 10. In the **Selected Properties** section of the **Property Configurations** dialog box, select properties to add.
 11. Click **Submit**.
 12. Expand the property list, and select the properties to include in search.
 13. Click **Save**.
6. Save your changes.

- [Understanding Discovery features for access control policies](#)
- [Customizing of Discovery gadgets](#)

Enabling attribute-based access control

Attribute-based access control (ABAC) provides another level of control beyond role-based access control, and is enabled by default. If you have disabled ABAC in the past, you can enable ABAC by updating a dynamic system setting.

1. In Dev Studio, search for the dynamic system setting **EnableAttributeBasedSecurity** and open it.
2. In the **Value** field, enter *true* to enable attribute-based access control.
3. Click **Save**.

- [Attribute-based access control](#)

Client-based access control

If your application stores data that might be used to identify a person and you are subject to GDPR or similar regulations, use client-based access control (CBAC) to track and process requests to view, change, or remove the data.

Client-based access control helps you satisfy the data privacy requirements of client protection regulations, such as the European Union (EU) General Data Protection Regulation (GDPR), and the California Consumer Privacy Act (CCPA). In Pega Platform, personal data might be stored in the database or related data sets, and is identified by class name and property name. Personal data is associated with an actual person, not with an abstract entity such as a business.

For more information, see:

- [General Data Protection Regulation](#).
- [California Consumer Privacy Act \(CCPA\)](#).

Data privacy APIs

REST APIs process requests to get, rectify (update), erase (delete), or limit the usage of personal data. The access request processing can be synchronous or asynchronous, but the processing of requests to rectify and erase is asynchronous. Cases handle Access, erase, and rectify requests. When data requests are processed, the decrypted client data is returned to the client using HTTPS in Base64 encoded format. For requests to rectify or erase, the data is modified or deleted as requested.

The REST APIs that define personal data requests are in the Data Privacy category of the *api* service package, which is known as the *Pega API*.

Note:

- Requests to update and delete personal data are one-time requests that do not prevent the data from being changed or added again in the future.
- Client data that is temporarily stored on a CBAC case does not persist after the case has been resolved.
- [Defining client-based access control rules](#)
- [Attribute-based access control](#)
- [Integrating with Pega APIs and services](#)

Defining client-based access control rules

Client-based access control (CBAC) rules define where personal data is stored and how it can be accessed. These CBAC rules are used by the application server that receives and processes the requests.

CBAC rules are only one part of the overall processing of client-based access requests. For information, see [General Data Protection Regulation](#).

Follow these general steps to define CBAC rules for client-based data requests:

1. **List the applications** – Identify the applications that store personal data. By listing the applications, you can determine the rulesets that contain the rules needed for personal data requests. If all your applications are built on the same parent application, you can use the parent application for this purpose. Otherwise, define CBAC rules separately for each application.
2. **List the data elements** – Identify the data elements that contain protected information that could be used to identify an actual person. For example, personal data might include genetic data, health data, Internet cookies, fingerprints, names, addresses, ages, national identification numbers, and personally identifiable data gathered over the Internet. In Pega Platform, identify the class names and property names where this data is stored.
3. **List the identifiers** – Establish how your application identifies the person who is described by the personal data. Your application identifies the person with one or more unique properties such as, for example, a national identification number or, if your application equates an email address with a person, an email address. You must optimize and index these client identifiers on all the classes that contain them.
4. **Create the CBAC rules** – Create the CBAC rules that describe the personal data and identifiers:
 - The applies to class of the CBAC instance is the class where the personal data is stored or where an identifier is referenced. The applies to class can be an abstract class if the data is stored on different concrete classes within the same abstract class. The instances are of *Data-*, *Index-* or *Work-*.
 - The ruleset of the CBAC instance belongs to the application that controls the personal data. You can create CBAC instances in a ruleset that is shared by multiple applications, or in separate rulesets by application.

For detailed steps on creating the CBAC rules, see the steps listed below.

1. [Creating a client-based access control rule](#)

Create client-based access control (CBAC) rules to identify the personal data and personal identifiers in your Pega Platform application. CBAC rules define how an incoming request finds the personal data in your data store. CBAC rules also define the type of access the client has for each data instance (view, modify, or delete).

2. [Configuring a client-based access control rule](#)

Define the personal data properties and personal identifiers for a client-based access control rule (CBAC) so that requests for personal data can be tracked and processed. A CBAC rule defines access, update, and delete permissions for individual data elements.

- [Client-based access control](#)
- [Configuring client-based access control for a non-Pega data source](#)
- [Specifying a client-based access control rule on an abstract class](#)

Creating a client-based access control rule

Create client-based access control (CBAC) rules to identify the personal data and personal identifiers in your Pega Platform application. CBAC rules define how an incoming request finds the personal data in your data store. CBAC rules also define the type of access the client has for each data instance (view, modify, or delete).

Before you begin: To create a client-based access control rule, you must have the *pzCanManageSecurityPolicies* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.

1. In the Dev Studio header, click **Create Security Client Based Access**.
2. Specify the applies to class where the personal data is stored or referenced. This is a class for *Data-*, *Index-*, or *Work-* instances.
3. From the **Add to ruleset** list, select the ruleset to contain the CBAC rule.
4. Click **Create and open**.
5. Configure your access control rule as described in [Configuring a client-based access control rule](#).

- [Defining client-based access control rules](#)

Configuring a client-based access control rule

Define the personal data properties and personal identifiers for a client-based access control rule (CBAC) so that requests for personal data can be tracked and processed. A CBAC rule defines access, update, and delete permissions for individual data elements.

Before you begin: To configure a client-based access control rule, you must have the *pzCanManageSecurityPolicies* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.

- You can create a CBAC rule for each class where personal data is stored, within a ruleset that is accessible to your applications that gather personal data. In the simplest case where your data and identifiers are all in the same class, you can create one CBAC rule for the entire application. In more complex classes, where the personal data is stored on multiple classes, you create a CBAC rule for each class.
- If data is defined in a common abstract class, you can create one CBAC rule for the abstract class.
- You can create CBAC rules at different levels in the class hierarchy. They are added together at run time.
- Creation and update of CBAC rules are logged as client-based access change security events.

1. Create a client-based access control rule, or open an existing rule from the navigation panel by clicking **Records Security Client Based Access**.
2. On the **Data elements** tab, list the personal data properties:
 - a. In the **Property** field, press the Down Arrow key and select a *persistent* property from the applies to class of the rule or one of its ancestor classes, or from a page list

- or page group within that class.
 - b. In the **External label** field, enter a label that is used to resolve personal data requests for this property. This label uniquely identifies the data for the purposes of CBAC. For example, if a person's home phone number is stored in class A as *pyHomePhone* and the same value is stored in class B as *pyHomePhone*, you define a CBAC for class A with an external label equal to *Home Phone*, and another CBAC for class B with the same external label (*Home Phone*).
 - c. **Optional:** In the **External description** column, click the **Pencil** icon, enter a description, and click **Submit**. For example, you might enter "Home phone number." **Note:** When two properties have the same external label, only one of the properties is returned in the CBAC response because both properties have the same data value. As a best practice, enter the same external description for both properties.
 - d. If personal data requests are allowed to change this data, select the **Rectify** check box. You cannot select **Rectify** if the applies to class inherits from *Index-*.
 - e. If personal data requests are allowed to delete this data, select the **Erase** check box. You cannot select **Erase** if the applies to class inherits from *Index-*.
- For example:**
- To allow a client to change the primary email property but not to delete it, select the **Rectify** check box and clear the **Erase** check box.
 - To allow a client to change and delete the secondary email property, select both the **Rectify** and **Erase** check boxes.
 - To prevent a client from changing or deleting the account number, clear both check boxes.
- Note:** View access is granted to every data element that you list.
3. **Optional:** To add more properties to the CBAC rule, click the **Add a row** icon and repeat step 2.
 4. **Optional:** To remove a property from the CBAC rule, click the **Delete this row** icon.
 5. If more than one class contains personal identifiers, on the **Pages & Classes** tab, identify the classes that contain the identifiers.
 6. On the **Identifier mapping** tab, list the personal identifiers. A client making a personal data request will supply one of these identifiers. These identifiers are also used to join multiple classes when needed to find the personal data.
 - a. In the **Identifier** field, press the Down Arrow key and select an identifier from the applies to class of the rule or from one of the classes that you have listed on the **Pages & Classes** tab. Each identifier must be *optimized* and indexed. Identifiers must also be listed as data elements.
 - b. In the **External label** field, enter a label that is used to resolve personal data requests for this property. A client making a personal data request will supply the external label and the identifier value, for example, *Home Phone* and 1234567.
 - c. To define multiclass identifier relationships, in the **Association** field, press the Down Arrow key and select the class and property that contain a value equal to the value in the **Identifier** field. **For example:** A person's address is personal data that is stored in class A, and class A instances are unique by home phone, which is the property *.HomePhone1*. Incoming requests supply a national identifier that is stored on class B, and home phone is stored on class B as *.HomePhone2*. Class B is unique by *.HomePhone2*, and is also unique by national identifier.
 - Create a CBAC rule with an applies to class that is equal to A.
 - On the **Data elements** tab, enter the *.Address* and *.HomePhone1* properties, with the external labels Address and Home Phone.
 - On the **Pages & classes** tab, define PageB for class B.
 - On the **Identifier mapping** tab, enter the *.HomePhone1* identifier with the association PageB.HomePhone2, and an external label Home Phone.
 - Create a second CBAC rule with an applies to class that is equal to B.
 - On the **Data elements** tab, enter the *.NationalID* and *.HomePhone2* properties, with the external labels National ID and Home Phone.
 - On the **Identifier mapping** tab, enter the *.NationalID* identifier with the external label National ID.
 7. **Optional:** To add more identifiers to the CBAC rule, click the **Add a row** icon and repeat step 6.
 8. **Optional:** To remove an identifier from the CBAC rule, click the **Delete this row** icon.
 9. Click **Save**.
- [Planning for property optimization](#)
 - [Page List property](#)
 - [Page Group property](#)
 - [About Declare Index rules](#)
 - [Creating a client-based access control rule](#)

Configuring client-based access control for a non-Pega data source

By default, client-based access control applies to personal data that is stored in the Pega Platform database. By doing additional configuration, you can also apply client-based access control to non-Pega databases and other data sources.

For client-based access control of personal data that is not stored in the Pega Platform database, you can write activities that access, rectify, and erase the personal data on your external data source.

1. Create three activities: one to access, one to rectify, and one to erase data on your external data source. If possible, these activities should return the results in JSON format.
 - For a database, the activity can call one of the RDB methods to operate on a Connect SQL rule.
 - For a data set, the activity can call *DataSet-Execute*.
 - For a data flow, the activity can call *DataFlow-Execute*.

For example, to return the step page in JSON format is similar to the following. After calling this, you remove the step page.

```
tools.sendFile(myStepPage.getJSON(false).getBytes(),"CustomerData.json",false,null,true);
```

2. Create a client-based access control rule, or open an existing rule from the navigation panel by clicking **Records Security Client Based Access** .
3. Configure your access control rule as described in [Configuring a client-based access control rule](#).
4. In the **Activity name** fields, enter the names of the activities that you created in step 1 for **Access**, **Rectify**, and **Erase**.
5. Click **Save**.

- [Creating a client-based access control rule](#)
- [Configuring a client-based access control rule](#)
- [Viewing test coverage reports](#)
- [About Connect SQL rules](#)
- [About Data Set rules](#)
- [Creating a data flow](#)

Specifying a client-based access control rule on an abstract class

You can define client-based access control (CBAC) rules on concrete classes and on abstract classes. At run time, Pega Platform combines multiple CBAC rules, starting from the concrete class. If the CBAC rule is on an abstract class and not on any of its concrete classes, you must do extra configuration to ensure that the abstract class is included.

1. Create a data transform and append the abstract class name to *.pxResults*. **For example:** You have defined a CBAC rule on the abstract class *MyApp-Work-AbsClass* but not on any of its concrete classes. Create a data transform named *MyCBACTransform* with the values shown below.

Data transform step	Action	Target	Relation	Source
1	Append and Map to	<i>.pxResults</i>	a new page	
1.1	Set	<i>.pyClassName</i>	equal to	<i>MyApp-Work-AbsClass</i>

2. Customize the data page *D_pyCBACPolicyClassList* by adding the data transform that you created in step 1 as a data source. **For example:** For the data transform example in step 1 above, update the data page with the following values.
 1. In the **Source** list, click **Data Transform**.
 2. In the **Data transform name** field, enter *MyCBACTransform*.

- [Client-based access control](#)
- [Data transforms](#)
- [Understanding data page definition](#)

Operators

An operator defines a unique identifier, password, preferences, and personal information for a user. Create operators so that people and processes can access your application.

For example, in an application for reviewing loan requests, you can create operators for customer service representatives (CSRs), reporting managers, and an administrator to

manage the application from the technical side.

By creating operators you provide access for users and processes to your application so that processing work can happen, and so that business processes can reach successful resolution. You can create operators not only for users, but also for automated, unattended processes. Unattended operators are robotic automation virtual machines (VMs). The system generates unattended operators for each registered VM in a robotic process automation (RPA) solution. Creating unattended operators increases the automation of your business processes, lowers costs, and speeds up work resolution.

Creating operators gives you the following benefits:

Increased security

Every operator can have a unique password with which to log in to your application. As a result, you protect your data and operations from inappropriate or random users.

Improved routing of work

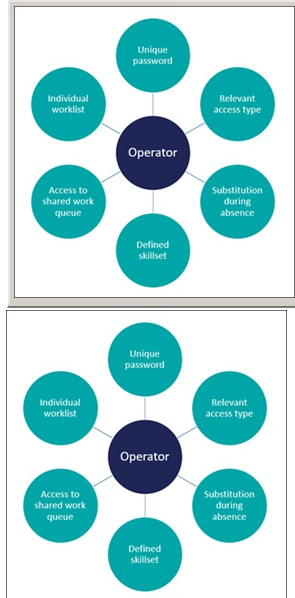
Every operator has an individual queue of assignments to complete that Pega Platform refers to as a worklist. Additionally, you can give an operator access to a work queue that groups assignments that are available for members of a specific team. For every operator, you can specify the level of relevant skills, for example, you can indicate that an operator is a highly advanced Java developer, or speaks French fluently. Grouping assignments and defining skillsets improves the routing of work and ensures that an application assigns tasks to the most appropriate users. For every operator, you can also define periods of unavailability and a substitute worker or work queue to ensure continuous case processing.

Relevant access type

You can define which data and operations an operator can access and perform in your application. For example, you can ensure that only a manager can approve or reject new job candidates, or that a CSR can access all information that a case requires to reach resolution. Consequently, the users of your application can perform only relevant actions and process their work faster.

The following figure summarizes the options that are available for operators in Pega Platform applications:

The characteristics of an operator



Clusters and operator IDs

After you save a new, or update an existing operator ID instance, the change might not be reflected on another node in a cluster until the Pega-RULES agent on that node performs the next system pulse — typically after no more than 60 seconds. Unlike instances of most other *Data-* classes, the system saves operator ID instances to the rule cache. As a result, until the next time that the rule cache is synchronized, one node might access a stale copy from its rules cache.

Bulk operator load

To save time, you can create operator ID instances by importing a comma-separated values (CSV) file, such as a Microsoft Excel spreadsheet.

Operator ID property

After a requestor logs in, the operator ID identifier is available on the *pxRequestor* page as the *pxUserIdentifier* property.

Operator IDs and external identity providers

If you implement authentication by using an external identity provider (IdP), the login process accesses IdP for authentication and ignores the password in this operator ID instance. However, the system still requires an operator ID data instance for each user.

Operator passwords

The system saves passwords for operator IDs as hashed values in the PegaRULES database, by first salting the clear text password value and then hashing it using the default bcrypt algorithm. The system uses two property types when changing the password: Password type for the **New Password** field, and Text type for the **Confirm Password** field. The *Data-Admin-Operator-ID.pyPwdCurrent* property stores the hashed password after the password passes validation.

• [Enabling, disabling and deleting operators](#)

You can control which users can access your application or are authorized to work on assignments in your application. By providing the minimum level of access that each user requires, you can improve the security of your application.

- [Bcrypt hashing algorithm for Password property types](#)
- [Understanding Requestor Type data instances](#)
- [Fields for operator contact information and application access](#)
- [Defining security information for an operator](#)

Enabling, disabling and deleting operators

You can control which users can access your application or are authorized to work on assignments in your application. By providing the minimum level of access that each user requires, you can improve the security of your application.

For example, you can disable accounts for inactive users or classify a user as unavailable for a specific period of time.

The following tasks can help you restrict a user's access to your application:

- [Viewing locked and disabled operators](#)

You can view lists of locked and disabled operators, and unlock or enable them.

- [Disabling an operator](#)

An inactive operator should not be able to log in to the Pega Platform. Each operator ID has a defined number of days of inactivity before being automatically disabled. However, you can manually disable an operator at any time, if necessary.

- [Enabling operators](#)

You can manually enable any disabled or locked operators.

- [Unlocking an operator](#)

An operator can get locked after a certain number of failed login attempts. The security policies specify the default number of failed login attempts before a password is locked. You can unlock the operator manually.

- [Deleting operators](#)

To ensure that only designated workers can access your application, you can disable access for users that are no longer active, for example, when they no longer work in your organization.

- [Operators](#)
- [Fields for operator contact information and application access](#)
- [Defining security information for an operator](#)

Viewing locked and disabled operators

You can view lists of locked and disabled operators, and unlock or enable them.

Before you begin: You must have the *pzViewAuthPoliciesLP* privilege to view the Operator Access tab.

1. In the header of Dev Studio, click **Configure Org & Security Authentication Operator Access**.
2. To view locked operators and to unlock them, view the list in the **Locked operators** section.
3. To view disabled operators, and to enable and disable other operators, view the list in the **Disabled operators** section.

- [Enabling operators](#)
- [Disabling an operator](#)
- [Unlocking an operator](#)

Disabling an operator

An inactive operator should not be able to log in to the Pega Platform. Each operator ID has a defined number of days of inactivity before being automatically disabled. However, you can manually disable an operator at any time, if necessary.

Before you begin: You must have the *pzViewAuthPoliciesLP* privilege to view the Operator Access tab.

1. In the header of Dev Studio, click **Configure Org & Security Authentication Operator Access**.
2. In the **Disable operators** section, click **Disable additional operators**.
3. Click **Add Operator**.
4. Enter the operator ID.
5. Click **Submit**.

- [Viewing locked and disabled operators](#)
- [Enabling operators](#)

Enabling operators

You can manually enable any disabled or locked operators.

Before you begin: You must have the *pzViewAuthPoliciesLP* privilege to view the Operator Access tab. The administrator must enable new default operator IDs.

A new password is not generated for operators that are created by the administrator. These operators must change their password when they log in for the first time.

When you enable an operator ID that is shipped with Pega Platform, after you click **Submit** on the **Enable Operator** dialog, the dialog shows the enabled operator ID and the new password that is automatically generated. Copy the password from this dialog box and send it to the enabled operator.

1. In the header of Dev Studio, click **Configure Org & Security Authentication Operator Access**.
2. In the **Disabled operators** section, select the check box next to the operator ID to enable.
3. Click **Enable selected**.
The **Enable Operator** dialog is displayed.
4. Click **Submit** to confirm that you want to enable the selected operator ID.
5. Click **OK** to close the dialog box.
Note: If you have the **john.johnson@uplusfs.com** operator in the system and you want to start or continue using the operator, you should change its access group to **CLMFSCIB_RM**.

- [Viewing locked and disabled operators](#)
- [Disabling an operator](#)

Unlocking an operator

An operator can get locked after a certain number of failed login attempts. The security policies specify the default number of failed login attempts before a password is locked. You can unlock the operator manually.

Before you begin: You must have the *pzViewAuthPoliciesLP* privilege to view the Operator Access tab.

1. In the header of Dev Studio, click **Configure Org & Security Authentication Operator Access**.
2. In the **Locked operators** section, click **Unlock & Reset** next to the chosen operator.
3. Send the newly generated password to the operator.

- [Security policies](#)
- [Viewing locked and disabled operators](#)
- [Using the login policies settings](#)

Deleting operators

To ensure that only designated workers can access your application, you can disable access for users that are no longer active, for example, when they no longer work in your organization.

You cannot delete an operator when a data instance, such as an organization unit or work queue, still references the operator. To save time, instead of deleting all of the references individually, change the operator password and mark the user as unavailable.

1. Ensure that the user that you want to delete is not logged in.

2. Transfer or complete all assignments in the work queue for the user.
For more information, see [Transferring an assignment](#).
3. Update the password to a value that is unknown to the user, so that the person can no longer log in.
For more information, see [Defining security information for an operator](#).
4. Ensure that the user has no rules checked out. If any rules are checked out, sign on and delete or check in the checked-out rules.
You cannot delete an operator ID if the operator has rules checked out. Ensure that the operator deletes or checks in all rules in their personal ruleset. For more information, see [Checking in a rule](#).
5. Mark the user as unavailable to receive work, and then define the user's departure date and a substitute operator.
For more information, see [Defining operator availability](#).

- [Operators](#)

Managing the operators of your application

Operators represent users in your application. You can adjust an operator's permission by assigning them to an operator group. Operator permissions are important and the more they can access, the more the application data can be at risk. The leading practice is to give operators less permissions and upgrade their access as their role evolves.

Creating an operator ID

Provide users with access to your application by creating operator IDs. When you create an operator ID, you can specify the background and general information about the user, define application access, and configure password credentials for the user.

For example, in a banking application, create operator IDs for customer service representatives (CSRs) who review loan requests, and the manager to whom the CSRs report.
Before you begin: To create an operator ID, you must have the *pxCanManageUsers* privilege, which is part of the *PegaRULES:SecurityAdministrator* role.

1. In the header of Dev Studio, click **Create Organization Operator ID**.
2. In the **Short description** field, enter the full name of the new operator.
The value that you enter populates the **Full name** field in the operator contact information on the **Work** tab. The maximum length of the description is 64 characters.
3. In the **Operator ID** field, enter a unique identifier.
When you enter the identifier, consider the following guidelines:
 - In addition to letters and digits, the identifier can include the following characters: period (.), single quote ('), tilde (~), underscore (_), exclamation point (!), ampersand (&), octothorpe (#) and no more than one at (@) character. Avoid using the forward slash (/) or backslash (\) characters in the identifier.
 - An operator ID can consist of a maximum of 128 characters.
 - Special processing applies to any user identifier that begins with the word *External*. Use such identifiers only when defining external operators that use Directed Web Access (DWA) for one-time processing of assignments.
 - Avoid using *system* as an operator ID name. This term is reserved and refers to agents.
4. Click **Create and open**. **Result:** The operator rule form is displayed on the page and you can continue to add information about the user. For example, you can configure access settings or provide organizational information.

What to do next:

- Define an access group and contact information for the operator. For more information, see [Defining operator contact information and application access](#).
- Define organizational information for the operator. For more information, see [Defining work routing settings for an operator](#).
- Define a password for the operator. For more information, see [Defining security information for an operator](#).

- [Adding an operator by using the organizational chart](#)
- [Operators](#)

Adding an operator by using the organizational chart

You can create operator ID data instances for a selected organization by using the **Organizational Chart** tab on the Organization landing page.

1. Display the organizational chart by doing one of the following steps.
 - Click **Configure Org & Security Organization Organizational Chart**.
 - Open the Organization, Division, or Organization Unit rule form, and go to the **Charts** tab.
2. Right-click to open the pop-up panel, and click **Add Operator**. **Result:** A pop-up window appears. The organization, department, or unit values are automatically filled depending upon which node you choose.
3. Enter any missing organization information. Enter an operator ID, password, and full name. Select the operator's access group. All values are required.
4. Click **OK** to create the operator ID instance.
5. Open the operator ID instance to add or modify the information. Some values such as work group and calendar are copied from a data transform.

- [Creating an operator ID](#)
- [Operators](#)

Defining operator contact information and application access

Define contact information and application access for an operator so that all operators can communicate and access the required applications. The access groups that you specify affect which applications an operator can access and what their role is.

For example, you can configure access settings so that a user can access a Loan requests application both as a manager and as a customer service representative (CSR).
Before you begin:

- Create an operator as described in [Creating an operator ID](#) or [Adding an operator by using the organizational chart](#).
- To view an application's access settings, you must have the *pxCanManageUsers* privilege, which is part of the *PegaRULES:SecurityAdministrator* role.

1. In the header of Dev Studio, click **Configure Org & Security Organization Operators**.
2. On the **Operators** tab, select the operator that you want to edit.
Tip: To find your operator faster, in the **Search Text** field, enter the operator name, and then click **Search**.
3. **Optional:** To more quickly distinguish between operators, on the **Profile** tab, in the **Contact Information** section, provide an operator image:
 - a. Click **Choose File**.
 - b. In the files window, select a graphic file that you want to use, and then click **Open**.
 - c. Click **Upload Image**. **Result:** The image is visible after you save the operator record.
4. Provide basic information about the operator:
 - a. **Optional:** To inform users how to address the operator, in the **Title** field, enter a relevant personal title. **For example:** Enter Mr, Mrs, or Dr.
 - b. **Optional:** To provide personal information about the operator, complete the **First Name** and **Last name** fields.
 - c. In the **Full name** field, enter a value that you want to use to represent the operator across your system.
By default, the system autopopulates the **Full name** field with the value that you provide in the **Short description** field when you create an operator ID. The operator's full name is visible, for example, in the list of operators.
 - d. **Optional:** To inform users about the job title of the operator, complete the **Position / job title** field. **For example:** Enter Software engineer.
 - e. **Optional:** To provide the contact information of the operator, complete the **Phone** and **Email** fields.
The system uses the email address to send email correspondence to the operator.
5. In the **Application Access** section, list the access groups that the operator can access at run time:
 - a. In the **Access Group** field, enter an access group that you want to associate with the operator.
Tip: To see the roles in an access group, click the **Expand row** icon.
 - b. **Optional:** To provide additional roles and access to additional applications, click **Add item**, and then in the **Access Group** field, enter another access group. **For example:** You can provide multiple roles for a user within one application, such as a manager role and a CSR role.
 - c. Indicate the default access group by selecting a relevant radio button. **Result:** When a user logs in, the system directs the user to the application and role that the default access group defines. For example, if you define *LoanRequests:Managers* as a default access group, the user logs into the Loan requests application with a manager role.
6. In the **Localization** section, list the default locale and time zone of the operator.

What to do next: Before you can save the operator record, you need to define the organizational unit of the operator in your organization. For more information, see [Defining work routing settings for an operator](#).

- [Fields for operator contact information and application access](#)
- [Learning about access groups](#)
- [Operators](#)

Fields for operator contact information and application access

Complete the **Profile** tab to define the contact information, application access, and localization information for an operator. The access groups that you specify affect which rulesets, ruleset versions, and portals the user can access.

Contact Information

Field	Description
Operator image	Optional. Associate a photograph, graphic, or other image with this operator. If no image is specified, a default image from the binary file rule <i>images.dialogprofileimage.gif</i> appears. For best results: 1. Prepare a small square image about 25 to 56 pixels on each side, in a JPG, GIF, or PNG format. 2. Click Browse... or Choose File to navigate to the image file. 3. Click Upload Image... to upload the image file. 4. Save the rule. Note: • When you save the Operator ID form, an image content data instance (<i>Data-Content-Image class</i>) is created to hold the uploaded image file. The image is not part of the operator ID instance. • If you delete an operator ID instance, any associated image content data instance is also deleted. If you copy an operator ID instance using Save As, the newly created operator ID instance is not linked to the image. • Property <i>pyImageFileName</i> links the operator ID to an image content data instance. • The Delete Image option only disassociates the <i>Data-Content-Image</i> instance from the operator. The image is not deleted.
Title	Optional. Enter a title, such as Mr. or Ms.
First name	Optional. Enter the first name for this user.
Last name	Optional. Enter the last name for this user.
Full name	Enter the full name of this user. This text is present in the standard portal displays including Dev Studio, Case Manager, and Case Worker portals. Enter up to 50 characters. A few standard portal displays use the Short Description field — not the Operator Full Name value — to identify operators. To have the system present the full names on such displays, copy the Operator Full Name value to the Short Description field.
Position/job title	Optional. Enter the job title or position of this user.
Phone	Optional. Enter the business phone number of the user. (If your system includes the optional PegaCALL component, a different property <i>pyExtension</i> holds the phone number; that property does not appear on the Operator ID form.)
Email	Optional. Enter an email address for messages sent from this user or sent by an application to this user. Note: • The <i>pyEmailAddress</i> property has a maximum length of 40 characters. You can overwrite this property with a higher maximum length if needed. • The system does not check the format or validity of the address.

Application access

Specify which applications this operator will access by populating a list of access groups.**Note:** To view the **Application Access** section, you must have the *pxCanManageUsers* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.

Field	Description
Access Groups	For each row in the grid, select the name of an access group (a <i>Data-Admin-Operator-AccessGroup</i> instance) for this user. Designate a default access group by clicking the radio button next to the desired access group name. Use the expand icon to see a list of roles associated with the access group. Note: • If this operator will use a composite portal to enter work items for multiple applications, create and add a distinct access group for each application • Application scoping control <i>preferences</i> are stored on the access group level. If you delete (and later re-add) an access group from the list, the Application Explorer will default to displaying results in all application layers.
Application	Displays the name and version of the application this operator has access to. Click to open the rule.
Roles	Expand the Access Group item to display the access roles that are available to this operator. Click to open the access role rule.

Localization

Field	Description
Default locale	Optional. You can associate an initial locale with this operator that affects the processing and presentation of dates, times, and numbers. For example, enter <i>de_DE</i> to use the German display mode for dates (DD.MM.YY) rather than the American mode (MM/DD/YY). If the Internet Options locale setting for the user's workstation should apply for the application, leave this field blank. If the field is not blank and the system contains locale-specific rulesets corresponding to rulesets that this user can access, the user can work with an entire localized application.
Time zone	Optional. Select a time zone to be used for presenting dates and times to this user in: • Work object history • Reports on work items and assignments For example, select <i>America/New_York</i> for the US Eastern time zone; the display adjusts correctly for daylight savings time. Select <i>Europe/London</i> for London time. As a best practice, if all or nearly all operators in an organization are in a single time zone and follow a common business day calendar, leave this field blank for those operators who share the organizational calendar and time zone. Complete the Calendar and Time Zone fields only for the operators who are exceptions. These values are grouped by zone, not alphabetically, in the drop-down list, starting with the zone that is 12 hours away from Greenwich, England. Time zone codes follow the Olson TZ database, used in the ICU4J 3.4.4 library and most UNIX systems. Details on the Olson TZ database are available from http://en.wikipedia.org/wiki/Tz_database. Do not type a three-letter code in this field, as such codes are ambiguous in certain cases and may produce incorrect presentation of daylight savings time in other

Field	Description
	cases. For details, see the Pega Community article <i>Issue: Avoid three-letter time zone codes because they do not support Daylight Savings Time</i> .
	Settings in this field only affect the presentation of Date/Time values. Within the database, all Date/Time values are recorded in universal time, informally known as GMT. As a result, a work item resolved in one country on Tuesday may be presented as resolved on Monday when the history is reviewed in another country. As a best practice, if your organization has all its staff and servers in a single time zone, leave the Calendar and Time Zone fields blank. If most staff and all servers are in a common time zone, leave the Calendar field blank but complete a Time Zone value for the exceptional operators. If the organization has operators and servers across many time zones, create a Calendar for each time zone and specify both the Calendar and Time Zone fields in every operator ID instance.

- [Defining operator contact information and application access](#)
- [Operators](#)

Defining work routing settings for an operator

For continuous and efficient workload management and routing, define skills, work groups, and work queues for operators. As a result, an operator can receive work based on skill set and team membership. For reporting purposes and advanced routing use cases, configure the reporting structure.

For example, when you define the reporting manager of an operator, the manager might receive notifications about the operator's progress on a case.

Before you begin: Create an operator so that a user can access your application. For more information, see [Creating an operator ID](#) or [Adding an operator by using the organizational chart](#). Every operator has a unique worklist that accumulates tasks that the specific operator can retrieve. Additionally, an operator can access a work queue that collects tasks for a team to which the operator belongs. Any team member can pick up tasks from a work queue. Another name for a team is a work group.

For an automated selection of tasks to process, operators can use the Get Next Work logic. Get Next Work automatically prompts users with an assignment that currently has the highest urgency in work queues and work groups that the operator can access.

- In the header of Dev Studio, click **Configure Org & Security Organization Operators**.
- On the **Operators** tab, select an operator that you want to edit.
Tip: To find your operator faster, in the **Search Text** field, enter the operator name, and then click **Search**.
- On the **Work** tab, in the **Organizational unit** section, define the affiliation of the operator within the organization by clicking **Update**, and then provide a relevant organization, division, and unit. **For example:** For an organization, enter the name of your company, for example, UPlusTelco, and then provide a division and unit, such as BankingOperations for a division and Mortgages for a unit.
- In the **Team** field, enter a work group to which the user belongs.
A work group defines routing of tasks and a reporting manager of the user.
For example: If you create Team A to review mortgage requests and Team B to investigate issues with mortgage operations, the two teams can use separate areas of your application so that the work is logically separated.
- Optional:** To assign a user to multiple work groups, click **Add a work group**, in the text box enter a work group, and then select the radio button next to the work group that you want to set as the default.
- Optional:** To enhance the reporting options of your application, in the **Reports to** field, enter the reporting manager for the user.
- Optional:** To enable routing of tasks based on a skillset, in the **Skill** field, enter the ability of the user, and then in the **Rating** field, enter the level of the ability.
You can apply a rating from 1 to 10, where 10 implies the highest level of the skill.
For example: To indicate that a user is a fluent speaker of French, in the **Skill** field, enter French, and then in the **Rating** field, enter 10.
- Optional:** To allow the user to process assignments from a work queue that corresponds with the user's work group, in the **Work queue** section, click **Add item**, and then provide a work queue and an urgency threshold.
During Get Next Work processing, your application ignores assignments with urgency lower than the urgency threshold. For more information about Get Next Work, see [Customizing the Get Next Work logic](#).
- Optional:** To assign the user to multiple work queues, click **Add item**, and then repeat step 8.
- Optional:** To trigger the Get Next Work algorithm to retrieve assignments from the work queues of the user first, select the **Get from work queues first** check box. Otherwise, Get Next Work picks up the top assignment on the user worklist, and accesses work queues only if this user's worklist is empty.
For example: If a user is a member of the Team A to review mortgage requests, the Next Get Work algorithm suggests assignments that the application routes to the Team A work queue first and then moves to the assignments from the user's individual worklist.
- Optional:** To trigger the Get Next Work algorithm to retrieve assignments from all work queues that correspond with the team that you provided in step 4, select the **Use all work queue assignments in user's team** check box.
- Optional:** To trigger the Next Assignment item to consolidate assignments from all of the work queues of the user, sort by assignment urgency, and return the most urgent assignment in any work queue, select the **Merge work queues** check box.
If you select the **Use all work queue assignments in user's team** check box in step 11, the system selects the **Merge work queues** check box by default and you cannot clear this check box.
- Click **Save**.

- [Fields for operator teams, work queues, and schedules](#)
- [Operators](#)

Fields for operator teams, work queues, and schedules

Use the **Work** tab to identify the skills, teams, work queues, and work schedule for an operator. The work queues are searched for assignments when users click **Next assignment** on the Case Manager portal.

Routing

In this section, you identify properties of the operator that relate to the assignment of work, including: organization, skill, work queue, and whether the operator is able to receive work assignments.

Field	Description
Organizational Unit	Click Update to select an organization, division, and unit from the pop-up window.
Teams	Select the name of a team (a <i>Data-Admin-Team</i> instance). Teams are the basis of many standard management reports. The system does not constrain team membership based on organization structure. Members of a team may belong to different organization units, divisions, or even different organizations. It is mandatory to have only one team selected, and this has an impact on what Work Queues are visible in Case Manager Portal for the user. Click the plus sign to add the additional teams to which this user belongs.
Reports to	Optional. Select the operator ID of another user to whom this operator reports. The specified value is used by the display shown when you click Reporting structure . The field has no additional uses.
Reporting structure	Click to display the reporting structure for this operator, based on the values specified in the Reports To field for this operator and for other operators. A new window opens displaying the reporting structure. The system assembles and displays a hierarchy (tree) view. In the displayed hierarchy, the names of operators who are not available are marked with a distinctive light blue background.
Skill	Optional. Identify a skill name — the name of a skill rule (<i>Rule-Admin-Skill</i> rule type) associated with this user. Routing activities and security tests can access skill information recorded here to determine which assignments this operator can perform. Order in this array is not significant.
Rating	Optional. Select a user proficiency rating for this skill between 1 and 10, where 10 indicates highest proficiency. (Some skill rules may allow only a

Field	Description
Work queue	narrower range of ratings, such as 1 to 3.) Optional. Enter a list of work queues that may contain assignments for this user. When this user clicks the <i>Get Next Work</i> menu item to obtain an assignment, the system searches the work queues listed here in order (unless the Merge work queues box is selected.) The Get from work queues first check box controls whether the system searches the work queues or the worklist first, and other aspects of the search algorithm. This array is visible only if the Use all work queue assignments in user's team check box is not selected. Your changes to this array may take effect only after the LookupList cache is rebuilt and the later operator logs in. This list does not restrict the ability of users to open and complete assignments placed in other work queues. Normally, users can fetch assignments directly from any work queue defined for their own organization unit, bypassing the <i>Get Next Work</i> facility. For general information on this facility, see <i>Maximizing user productivity with GetNextWork</i>.
Operator is available to receive work	Select to allow applications to route additional assignments to the worklist of the user who is identified by this operator ID data instance. Clear to prevent routing shapes in flow executions from adding assignments to this user's worklist. This check box affects only routing activities. This check box does not affect the ability of this user to log in, enter work items, or complete assignments already on the worklist. It does not prevent another operator or manager from transferring assignments to the worklist. If selected and at runtime a router activity (with parameter <i>CheckAvailability</i> set to true) attempts to send an assignment to the worklist of this operator ID, the substitute operator if any is used instead. If no substitute is found, the router activity adds the assignment to the worklist of the manager of this operator's team.
Get from work queues first	Select to cause the <i>Get Next Work</i> menu item retrieve assignments first from work queues listed below. - When selected, system retrieves an assignment from the user's worklist only when all of the work queues listed in the Work queues array are empty. - If not selected, Next Assignment, when clicked, retrieves the top assignment on the user worklist, and accesses work queues only if this user's worklist is empty. Work queues can be set up so that only users who possess one of a list of access roles can remove assignments from them. Processing for the Next Assignment facility skips over any work queue that this user cannot fetch assignments from. This processing ignores Error: assignments (<i>problem assignments</i>); they are never selected as most urgent. Your application can alter or override the standard activities that support this button, providing a sophisticated means of prioritizing which assignments users work on. See <i>Maximizing user productivity with GetNextWork</i>.
Use all work queue assignments in user's team	Select to have Get Next Work processing assemble and sort assignments in any work queue for which the Teams field (on the Work queue tab of the Work queue form) is the same as the Teams field of this operator (as set on the Operator tab). This field is visible only when the Merge work queues check box is selected. When enabled, selection and sorting occurs directly through efficient SQL operations on tables in the PegaRULES database. This is typically faster than activity-based searching and sorting
Merge work queues	Select to cause Next Assignment item to consolidate assignments from all the work queues in the Work queues array below, then sorted by assignment urgency, returning the most urgent in any work queue. When enabled, selection and sorting occurs directly through efficient SQL operations on tables in the PegaRULES database. This is typically faster than activity-based searching and sorting.

Scheduling

This area records future scheduled absences. Standard routing activities can examine this information before routing an assignment to a work queue, based on a Boolean parameter *CheckAvailability*. If the parameter is true and this operator is not available when the router activity runs, the assignment is routed to a substitute.

Complete these fields to control how assignments are routed for this operator when this operator is marked absent or unavailable. Routing rules can redirect the assignments to a substitute operator or to a work queue during those periods.

When an assignment is linked to a service-level agreement, the dynamic system setting *Pega-ProCom.pyUseGoalTimeForAvailabilityCalculation* affects this availability calculation.

Field	Description
Business Calendar	Optional. Identify the first key part (Calendar Name) of a calendar instance that identifies the working days and holidays of this operator. At runtime, the system uses the current date in format YYYYMMDD as the second key part to find a calendar data instance. At runtime, if no calendar with a second key part exactly matching today's date is found, the calendar that matches the first key part exactly and has as a second key part the latest date not greater than today's date is used. Select a valid calendar carefully. To simplify and reduce maintenance over multiple years, this value is not validated when you save an Operator ID form. As a best practice, if all or nearly all operators in an organization are in a single time zone and follow a common business day calendar, leave this field blank for those operators who share the organizational calendar and time zone. Complete the Calendar and Time Zone fields only for the operators who are exceptions. Few standard rules depend on the value in this field; it is provided for application use. The calendar does not affect the ability of the operator to sign on, enter work, or perform assignments. Your application can use this field to access a calendar data instance for the operator when calling date functions. For example, if the value of this field is HONGKONG and the current date is March 31, 2006, the system first searches for a calendar with key HONGKONG.20060331. If no calendar data instance exactly matches this key, the immediately preceding calendar instance if any that matches HONGKONG (such as HONGKONG.20060101) is used. During log-in, if this field is not blank, both the calendar and the time zone are set on the requestor page as the calendar and default time zone (property <i>pyRequestorTimeZone</i>). If blank, the system uses the calendar from the Organization data instance and does not set a time zone; the time zone is then typically defaulted to the time zone of the server node.
Unavailable From	Optional. Identify one or more future date intervals during which this operator is not available. In testing the current date time against the date ranges here, the system uses the calendar (and the time zone of the calendar) associated with this operator. Click the + icon to add a row. Click the calendar icon to select a date from the pop-up calendar. Most operators cannot update this information directly. A team manager can update this information, for operators in the group. (The standard privilege named <i>Data-Admin-Operator-ID.UpdateWorkgroupAvailability</i> conveys this capability.) The system purges past time intervals from this list each time the operator logs in.
To	Identify the end date of the future interval.

Field	Select:
Substitute operator type	- Operator — To attempt to route new assignments to another operator when this operator is not available to accept new assignments. - Work queue — To place new assignments for this operator into a work queue when this operator is not available to accept new assignments.
Decision tree to find substitute	Optional. Identify the name (second key part) of a decision tree that the system can use to identify a substitute who can receive assignments sent to this operator. Select a decision tree that returns an operator ID or one that returns a work queue name, matching your selection in the Assignee Type field. To find this decision tree, the system uses <i>Data-Admin-Operator-ID</i> as the initial value of the Applies To class.
Default to assignee	Optional. Identify the operator ID or work queue name of a substitute for new assignments routed to the user identified by this operator ID instance when that user is unavailable. The system uses the substitute operator or work queue if the First, use this Decision Tree field is blank, or the decision tree does not result in a valid operator ID or work queue value.

- [Operators](#)

Defining security information for an operator

Ensure that your organization complies with security policies by defining security information for the operators in your system. You can manage operator authentication, passwords, and license types, to allow rule check out, and enable or disable the operator. As a result, operators can access your application safely, and then perform only the actions that are relevant to their roles.

1. In the header of Dev Studio, click **Configure Org & Security Organization Operators**.
2. On the **Operators** tab, select an operator that you want to edit.
Tip: To find your operator faster, in the **Search Text** field, enter the operator name, and then click **Search**.
3. On the **Security** tab, in the **Access Settings** section, secure access to your application with a password:
 - a. Click **Update password**.
 - b. In the **Change Operator ID Password** dialog box, in the **New password** field, enter the new password.
 - c. In the **Confirm new password** field, reenter the password.
 - d. Click **Submit**.

Note: If the operator is provided with Pega Platform, enter a password that is consistent with your security policies, and then send the new password to the enabled operator.

The system converts the password to a hash value by using the salted bcrypt algorithm. The Storage Stream (BLOB) column of the pr_operators table contains the hashed value. By using the **View XML** action, you can discover only the hashed form of any operator password.

For more information about security policies, see [Security policies](#).

4. If the operator is an unattended operator, select the **This is an unattended operator (robot)** check box.
Unattended operators are robotic automation virtual machines (VMs). The system generates unattended operators for each registered VM in a robotic process automation (RPA) solution.
5. **Optional:** To allow this operator to update rules in rulesets that use rule checkout, select the **Allow rule check out** check box.
6. **Optional:** To authenticate this operator only through external authentication facilities, select the **Use external authentication** check box.
7. **Optional:** To prompt the operator to change their password the next time the operator logs in, select the **Force password change on next login** check box.
8. **Optional:** To disable the operator, select the **Disable Operator** check box.
9. **Optional:** In the **Starting activity to execute** field, specify the first activity that the system runs after authentication for this user is complete.
The default is Data-Portal.ShowDesktop.
10. In the **License type** list, indicate the operator type:
 - To indicate that the operator is a person who does business operations by using an application or customer-created interface, select **Named**.
 - To indicate that the operator is an abstract user to run agents, services, and other background processes, or an external user that interacts with the application through the Directed Web Access feature, select **Invocation**.

For unattended operators, the system selects **Invocation** by default.

11. Click **Save**.

- [Operators](#)
- [License compliance](#)
- [Checking out a rule](#)

Auditing

With Pega Platform, you can track many types of security events, such as failed logins, password changes, and changes to rules and data. By tracking all of these events, you can understand how your system functions and detect any potential problems.

System auditing

Pega Platform provides comprehensive security information and event management (SIEM) features with which you can:

- Monitor all security-related activity in the system.
- Create reports that analyze patterns of system usage.
- Identify patterns of suspicious behavior.
- Determine the scope of the damage if any vulnerabilities are exploited.

Data auditing

The Pega Platform *History*- class supports auditing by capturing all data changes in rules and cases. The *History*- class automatically captures the following updates:

- For rules and cases - changes to the operator ID
- For standard properties - any changes to field-level tracking

For more information, see:

- [Enabling field-level auditing](#).
- [Tracking and auditing changes to data](#).
- [Monitoring security alerts and events](#).

Audit user and developer actions

In addition to tracking data changes in rules and cases, you can audit user and developer actions that might affect the security of your application. This information might potentially indicate suspicious behavior by a developer or user.

All security events include the following information:

- Date and time
- Application name
- Node
- IP address
- Tenant ID

- Operator ID
- Event class (authentication or authorization)
- Event type

Event types that can be audited

In Security Event Configuration, there are 3 types of events you can audit: Authentication events, Data access events, and Security administration events. Specific information about these events is available below.

To access the Security Event Configuration, in the header of Dev Studio, click **Configure Org & Security Tools Security Security Event Configuration**.

Authorization events

Authorization events assist developers by tracking:

- Successful and failed login attempts
- Password changes
- Session terminations
- Logouts
- Changes to operator records

The table below describes the **Authorization events** on the **Security Event Configuration** tab. **Note:** Any **authorization event** that is selected by default. You cannot stop these changes from being tracked.

Authorization event	Default setting
Successful and failed login attempts	Not selected
Password changes	Not selected
Session terminations	Selected
Logouts	Selected
Changes to operator records	Selected

Data access events

Data access events assist developers by tracking:

- Successful attempts to open cases
- Attempts to open cases if the attempt fails because of security policies
- SQL queries to the database
- Changes to report filters
- Full-text searches

The table below describes the **Data access events** on the **Security Event Configuration** tab. **Note:** Any **data access event** that is selected by default. You cannot stop these changes from being tracked.

Data access event	Default setting
Every open of a work- class object on the clipboard that succeeds	Not selected
Every SQL query that executed	Not selected
Changes to report definition filters	Not selected
Search queries	Not selected
Every open of a work- class object on the clipboard that fails due to security policies	Selected
Every report definition that executed	Selected
Every malformed request received from client	Selected

Security administration events

Security administration events assist developers by tracking:

- Changes to security authentication policies
- Changes to attribute-based access control (ABAC) policies and policy conditions
- Changes to role-based access control (RBAC), including changes to *Rule-Access-Role-Obj* (RARO) rules
- Changes to dynamic system settings
- Changes to content security policies (CSP)
- Changes to access groups
- Changes to work queues
- Invocations of Access Manager

The table below describes the **Security administration events** on the **Security Event Configuration** tab. **Note:** Any **security administration events** that is selected by default. You cannot stop these changes from being tracked.

Security administration event	Default setting
Every invocation of access manager	Not selected
Every BIX form changes and executions	Not selected
Every change to ABAC security policies	Selected
Every change to CBAC security policies	Selected
Every change to dynamic system settings	Selected
Every change to content security policy (CSP)	Selected
Every change to security authentication policies	Selected
Every change to security event configuration	Selected
Every change to RBAC security policies (including RADO and RARO)	Selected
Every change to access group settings	Selected
Every change to workbasket role settings	Selected

Every request to Disable/Enable operator	Selected
Every request to add/update/removal of servlet	Selected

OAuth 2.0 events

OAuth 2.0 events assists developers by tracking:

- Token requests
- Token revocations
- Invalid tokens
- API requests
- Client rule form changes
- Dynamic client registration

The table below describes the **OAuth 2.0 events** on the **Security Event Configuration** tab. **Note:** Any **OAuth 2.0 events** that is selected by default. You cannot stop these changes from being tracked.

OAuth 2.0 events	Default setting
Invalid token requests	Selected
API requests with invalid client credentials	Selected
Token revocation from Rest API	Selected
Regeneration of client secret from rule form	Selected
Token revocation from rule form	Selected
Delete client instance from rule form	Selected
Dynamic client registration	Selected
Resource API invocation using invalid access token	Selected

Custom events

You can toggle custom events **ON** and **OFF**.

You can define your own custom security events that you want to log.

For more information, see [Tracking and auditing actions by developers and users](#).

- [Tracking and auditing changes to data](#)

Pega Platform maintains a historical record of changes to certain data classes and rule types. You can use this history to diagnose system issues and to demonstrate compliance to internal and external auditors.

- [Tracking and auditing actions by developers and users](#)

The security event configuration feature is part of security information and event management (SIEM), which combines security information management (SIM) and security event management (SEM). Use the Security Event Configuration landing page to configure the logging of security events so that you can diagnose system issues and demonstrate compliance to auditors.

- [Monitoring security alerts and events](#)

Pega Platform generates security alerts and events for situations such as attempts to hijack a user session. You can review the security alerts and events by viewing their respective logs.

- [Logging each use of harness and flow action rules](#)

Your application can create an audit record each time an operator requests either a harness form or a flow action.

- [Authentication](#)
- [Authorization](#)
- [Security operations](#)
- [Security](#)

Tracking and auditing changes to data

Pega Platform maintains a historical record of changes to certain data classes and rule types. You can use this history to diagnose system issues and to demonstrate compliance to internal and external auditors.

For example, you can record the application rule name that is referenced in an access group each time a user updates the Access Group form and adds, changes, or deletes an application rule name.

The security audit capability supports change auditing for sensitive fields on selected rule or data objects.

The details of each change are displayed in the History Details section and are summarized in several standard reports.

When enabled, each save operation on a rule instance or data instance (whether through a form or through an activity) triggers a standard activity. The activity compares the current values of the tracked properties with their previous values, and writes a history detail instance for each value that was added, deleted, or updated.

The history detail identifies the following items:

- The property name
- The values added (if any are added)
- Changed values (the from value and the to value) (if any change)
- The deleted values (if any are deleted)

For aggregate properties, the history detail identifies the following items:

- Two entries when a value is changed: one entry about deleting the prior value and another about adding the new value.
- Only one level of nested PageList mode properties, because only one level is supported. Nested PageGroup mode properties are not supported.

The system saves rule changes as an instance of the *History-Rule* class. Changes to data instances are saved in an instance of a subclass of the *History-Data* class. For example, the system records changes to access groups (*Data-Admin-Operator-AccessGroup*) in instances of the *History-Data-Admin-Operator-AccessGroup* class.

- [Auditing field-level changes to security rule and data instances](#)

By simply creating a model and a trigger rule, you can generate history memos that automatically audit field-level changes to both rules and data instances. For example, you can create a model and trigger that will track changes to user RuleSet lists in an access group. Pega Platform™ also provides preconfigured reports to collectively view the history data.

- [Extending the security auditing feature to record changes to additional properties](#)

Pega Platform auditing can be extended to other properties such as a Page List and a Single Value property.

- [Enabling security auditing for a data class or rule type](#)
- [Customizing the memo text for an audited rule](#)
- [Auditing changes to aggregate properties](#)
- [Tracking and auditing actions by developers and users](#)

Enabling security auditing for a data class or rule type

You can record changes for single values and aggregate properties when you enable security auditing.

To enable security auditing for a data class or a rule type, complete the following steps.

1. In Dev Studio, create a data transform named *pyTrackSecurityChanges* with an applies to class equal to the class you want to audit.
Note: Examples with this name are provided as standard data transforms for tracking changes to access groups, operator ID instances, Access of Role to Object rules, flows, case type rules, and application rules. You can save copies of these standard data transforms into your ruleset and update the list of properties to be tracked.
2. On the **Definition** tab, in the **Action** field, select **Set**.
3. Specify the property in the **Target** field.
 - For value list or value group properties, enter a numeric index to identify one value.
 - For page mode properties, enter a reference to a single-value property within the page structure.
4. In the **Source** field, enter a word or phrase that will be displayed on the **History Details** section.
5. Click **Save**.
6. Create a Declare Trigger rule named *TrackSecurityChanges*.
The **Apply to (class)** field and **Add to ruleset** field are filled out automatically, but can be changed if needed.
Optional: Leave the Value column blank for a row to apply custom text from a field value rule.
7. On the **Trigger** tab, in the **Trigger when an instance is** field, click **Save**.
9. In the **Trigger activity** section, in the **Name** field, select *TrackSecurityChanges* for the activity name.
10. In the **Execute** field, click **Immediately**.
11. Click **Save**.
12. **Optional:** Test the feature by changing a rule or data instance and review the history.

- [Tracking and auditing changes to data](#)
- [Customizing the memo text for an audited rule](#)
- [Auditing changes to aggregate properties](#)
- [Tracking and auditing actions by developers and users](#)

Customizing the memo text for an audited rule

You can customize the memo text to provide a more detailed description of the change that was made to a rule or data object. Use field value rules to change the memo text that is displayed on the history details reports for a property.

1. In Dev Studio, in the navigation panel, click **Records Data Model Data Transform**.
2. Search for the data transform named **pyTrackSecurityChanges** and open it.
3. On the **Definition** tab, leave the **Source** field blank for the row that sets the property.
4. Create three field value rules. In the **Label** field, enter the following values.
 - TrackSecurityChange_Add
 - TrackSecurityChange_Change
 - TrackSecurityChange_Remove
5. In the **Field Name** field, for each field value, enter the name of the field where the memo text will be displayed.
6. In the **Apply to (class)** field, for each field value, set the class that the data transform applies to.
7. Click **Create and open** for each field value.
8. On the **Localized Label** tab, in the **To** field, enter the text to display in the memo field.
 - a. In the **TrackSecurityChange_Add** field value, enter {1} to refer to the new value of the tracked property.
 - b. In the **TrackSecurityChange_Change** field value, enter {1} to refer to the old value and {2} to refer to the new value.
 - c. In the **TrackSecurityChange_Remove** field value, enter {1} to refer to the old value of the tracked property.

- [Tracking and auditing changes to data](#)
- [Enabling security auditing for a data class or rule type](#)
- [Auditing changes to aggregate properties](#)

Auditing changes to aggregate properties

The history detail for aggregate properties does not indicate changed values as it does for single-value (scalar) properties. Instead, the history details shows two entries when a value is changed: one entry about deleting the prior value and another about adding the new value.

When you create a field value rule for a message for a Page List or Page Group property, do the following steps.

- For the field value rule's Applies To class, use the class of the embedded objects.
- For the field value rule's Field Name (the second key part of the field value rule), match what is entered in the data transform for that property.

- [Tracking and auditing changes to data](#)
- [Enabling security auditing for a data class or rule type](#)
- [Parts of aggregate property references](#)

Auditing field-level changes to security rule and data instances

By simply creating a model and a trigger rule, you can generate history memos that automatically audit field-level changes to both rules and data instances. For example, you can create a model and trigger that will track changes to user RuleSet lists in an access group. Pega Platform™ also provides preconfigured reports to collectively view the history data.

This process includes the following steps:

1. [Opening the appropriate model to review the default properties](#)
2. [Creating a declarative trigger](#)
3. Optional: [Verifying your new auditing changes](#)

1. [Opening the appropriate model to review the default properties](#)

The Pega-RULES ruleset contains four models (Rule-Obj-Model) named *pyTrackSecurityChanges* that allow you to quickly set up audit functionality for security-related records.

2. [Creating a declarative trigger](#)

In this step, you create a declarative trigger (Rule-Declare-Trigger) for each class you want to track.

3. [Verifying your new auditing changes](#)

Verify your configuration by changing a field (defined in the model) in the appropriate rule instance, and then opening the history memo.

Opening the appropriate model to review the default properties

The *Pega-RULES* ruleset contains four models (*Rule-Obj-Model*) named *pyTrackSecurityChanges* that allow you to quickly set up audit functionality for security-related records.

The four security-related records are:

- **Operators** (*Data-Admin-Operator-ID*)
- **Access Groups** (*Data-Admin-Operator-AccessGroup*)
- **Access of Roles to Objects** (*Rule-Access-Role-Obj*)
- **Applications** (*Rule-Application*)

Each model contains a default list of properties that are tracked. For example, to see the default properties that are tracked in an access group, select *Data-Admin-Operator-AccessGroup*.pyTrackSecurityChanges.

The default properties are listed in the left column; their alias values (the names that appear in the memos and reports) on the right. Note that all complex property types such as **Value List** and **Page List** must have an index subscript value, as all the items in the property list are tracked. The value of 1 is used by default.

You can use the models in the *Pega-RULES* ruleset if you want to track the standard properties. If you want to add or remove fields from the model lists, copy the rule to your application RuleSet.

What to do next: Complete the [Creating a declarative trigger](#) process.

Creating a declarative trigger

In this step, you create a declarative trigger (*Rule-Declare-Trigger*) for each class you want to track.

Before you begin: Complete the [Opening the appropriate model to review the default properties](#) procedure. Each class you want to track must have a corresponding model as defined in the [Opening the appropriate model to review the default properties](#) process. This rule executes the activity *TrackSecurityChanges* whenever you save an instance in the class.

1. Select **Application New Decision Declare Trigger** to open the **New Rule Instance** form.
2. In the **Applies To** field, enter the security-related class you want to audit. **For example:** Select *Data-Admin-Operator-AccessGroup* from the Smart Prompt.
3. In the **Purpose** field, enter any meaningful name, select your application RuleSet, and click **Create** to open the rule form.
4. In the **Declare Trigger** rule form:
 - a. Select **Saved** in the **Trigger When An Instance Is** field.
 - b. Enter *TrackSecurityChanges* in the field **Trigger Activity —.Name**.
5. Save the rule.

What to do next: Complete the [Verifying your new auditing changes](#) process.

Verifying your new auditing changes

Verify your configuration by changing a field (defined in the model) in the appropriate rule instance, and then opening the history memo.

Before you begin: Complete the [Creating a declarative trigger](#) procedure.

1. Using the *Rule-Obj-Model* in the [Creating a declarative trigger](#) procedure, open a *Data-Admin-Operator-AccessGroup* instance in your application.
2. Add a **Work Pool**.
3. Click the **History** tab and click the **History Details** button at the bottom of the form. **Result:** The memo appears.

Result:

The memo provides a timestamp of the event, the action performed on the property, and the ID of the operator who made the change. The standard memos take the following forms:

- Added {ValueFromModel}[NewValue]
- Changed {ValueFromModel} from {OldValue} to {NewValue}
- Removed {ValueFromModel}[NewValue]

The memos track modifications in the following manner:

- Single value properties, such as *.pyDefaultAppName*, are tracked as **Changed**. For example, if the Application Version was changed from 8.2 to 8.1, the memo would show

Time	Memo	Handled By
2/26/2020 3:41 PM	Changed application version from '8.2' to '8.1'	Administrator@AlphaCorp.com

- Complex property types such as value lists and pages lists are tracked as **Added** or **Removed**. For example, if Work Pools originally contained:
 - PegaSample
 - History-PegaSampleAnd was modified to:
 - AlphaCorp-OrderEntrySystem
 - History-PegaSample

Extending the security auditing feature to record changes to additional properties

Pega Platform auditing can be extended to other properties such as a *Page List* and a *Single Value* property.

Pega Platform provides security auditing features that automatically generate history memos that track field-level updates to rules and data instances. (See [Auditing field-level changes to security rule and data instances](#).) This functionality can be applied to any rule in the system, and is not limited to security-related rules.

The audit function is preconfigured to track the fields of most interest in typical applications. However, you can add new fields, such as a new aggregate property (a *Page List*) and a *Single Value* property.

- [Tracking changes to a Page List property](#)

Pega Platform auditing can be extended to other properties such as a *Page List* property.

- [Tracking changes to a Single Value property](#)

Pega Platform auditing can be extended to other properties such as a *Single Value* property.

- [Tracking changes to a Page List property](#)
- [Tracking changes to a Single Value property](#)

Tracking changes to a Page List property

Pega Platform auditing can be extended to other properties such as a *Page List* property.

Before you begin: Create a **Declare Trigger** rule named *TrackSecurityChanges* for the class you want to audit; for example *Data-Admin-Operator-AccessGroup*.

1. Open the rule form that contains the field you want to include in the history memo.

You can quickly determine the property and type underlying the **Portal Layout** field by clicking the **Rule Data** button on the toolbar to open the form's XML.
2. Override the *pyTrackSecurityChanges* model for *Data-Admin-Operator-AccessGroup* and add a new row.
3. In the **Property** field, enter *.pyUserPortals(1).pyPortalLayout*. Leave the **Value** field blank as the field value from the property list will be displayed in the memo.
4. Save the **Model** form.
5. Because aggregate properties record **Add** and **Remove** actions, you must create two field value rules to associate the type of update with the field (which is noted in the memo): *TrackSecurityChange_Add* and *TrackSecurityChange_Remove*.
 - a. In the **Developer** portal, click the **New rule** icon to create a new **Field Value** rule and enter the following values:
 - The class *Embed-PortalLayout* in the **Applies To** field

- `pyPortalLayout` in the **Field Name** field
 - `TrackSecurityChange_Add` in the **Field Value** field (do not modify this value)
 - Any meaningful **Short Description**
 - The **RuleSet** for your application
- b. Click **Create**.
 - c. In the **Localized Label** field, enter the text you want to see in the history memo followed by the property name in brackets (`{}`).
 - d. Save the rule.
 - e. Copy the rule (**Save As**) and use a **Field Value** of `TrackSecurityChange_Remove`.
 - f. Modify the **Short Description** appropriately.
 - g. In the **Localized Label** field, change `Added to Removed`.
 - h. Click **Save**.

What to do next:

Test the modifications by opening an instance of *Data-Admin-Operator-AccessGroup* and changing one or more values in the **Secondary Portal Layouts** section. Then click the **History Details** button (on the **History** tab) to review the history.

Tracking changes to a Single Value property

Pega Platform auditing can be extended to other properties such as a Single Value property.

To turn on security auditing for a Single Value property, simply add the property name to the model. If you are uncertain of the property name underlying a form field, click the **Rule Data** button on the toolbar to open the form's XML.

In the following example, the property `pyAuthenticationTimeout` is added to the model. This tracks changes to the **Authentication Usage (seconds)** field in the **Settings** tab.

Because the field **Authentication Timeout** is initially blank (has no default or required value), the first value entered is noted as *Added*. Subsequent changes to the value are noted as *Changed*. If the value is eliminated and the field is left blank, the update is noted as *Removed*.

Tracking and auditing actions by developers and users

The security event configuration feature is part of security information and event management (SIEM), which combines security information management (SIM) and security event management (SEM). Use the **Security Event Configuration** landing page to configure the logging of security events so that you can diagnose system issues and demonstrate compliance to auditors.

Security events include actions performed by any requestor who accesses the application, data accesses or changes, changes to security policies or security-related rules or landing pages, and so on. With the logged security events you can then monitor inappropriate access to your data or system.

All security events include the following information:

- Date and time
- Application name
- Node
- IP address
- Operator ID
- Event class (authentication or authorization)
- Event type

Authorization events can also include the following information:

- The entered search string.
- Report name, class, and filter condition.
- The class, ID, and class- of the case state:
 - Open.
 - Unopened.
 - Blocked by an access policy.
- [Selecting a security event to monitor](#)
- [Adding a custom security event](#)
- [Tracking and auditing changes to data](#)
- [Enabling security auditing for a data class or rule type](#)

Selecting a security event to monitor

To monitor and analyze security events, use the Security Event Configuration feature. You can select individual events to be automatically captured in logs for every user session.

Before you begin: To view the Security Event Configuration landing page, your access group must include the `pzViewSecEventConfigLP` privilege. You cannot deselect any event that is selected by default.

1. In the header of Dev Studio, click **Configure Org & Security Tools Security Security Event Configuration**.
2. Select the events that you want Pega Platform to capture.
3. In the **Custom event** section, click **ON** to log custom events, or click **OFF** to not log custom events.
4. Click **Submit**.

- [Tracking and auditing actions by developers and users](#)
- [Adding a custom security event](#)

Adding a custom security event

The Security Event Configuration landing page might not have all the events you want to monitor. However, you can define custom security events that are specific for your application in order to monitor them in the logs.

1. In the navigation panel, click **Records Technical Activity**.
2. Open the activity that processes the action or event you want to monitor.
3. Click **Check out**.
4. On the **Steps** tab do one of the following actions:
 - If a Java step does not exist, click **Add a step** to insert a Java step where you want to log the event.
 - If a Java step exists, click the **Arrow** icon to the left of the **Method** field to expand the **Method Parameters** section for the Java method.
5. In the **Method Parameters** section, enter the following call to the API: `tools.getSecEventLogger().logCustomEvent(PublicAPI tools, String eventType, String outcome, String message, Map<String, String> customFlds)` with the following parameter values.
 - `eventType` – Name of the event type to keep track of custom events
 - `outcome` – The outcome of the event
 - `message` – Any message that a user needs to log as part of the event
 - `customFlds` – A map of key-value pairs that log extra information for the event

Note: Add the call to the API at the end of the section if a Java step exists.
6. Click **Save as**.

- [Tracking and auditing actions by developers and users](#)
- [Selecting a security event to monitor](#)
- [Viewing test coverage reports](#)

Security events Log

Use the items displayed in the Security events log to view login attempts and other security events to improve application security and ensure application integrity.

Before you begin: To display the security audit log, you must have the *pzViewAuthPoliciesLP* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.

The **Security Events Log** report includes a default filter to display all audit events, which is set to *pxCreateDateTime* is Less or Equal to Current Month. To adjust the date range, click the link next to **Filters** in the report header.

To select security events to monitor, use the **Security Event Configuration** landing page.

To select whether to report successful or unsuccessful login attempts, set the **Audit log level** setting on the **Security Policies** landing page.

To display the Security Audit Log, complete the following steps.

1. In the header of Dev Studio, click **Configure System Settings Security Policies**.
2. Click **Display Audit Log**. **Result:** The Security Event log is displayed. For each logged event, the log captures the following information:

Captured in the log	Description
Create Date/Time	The time of the attempt.
Remote IP Address	The IP address of the system from which the attempt was made.
Remote Host	The name of the computer used.
User Name	The operator ID used in the login attempt.
Message	The status message returned during authentication of the login.
Browser (User-Agent)	HTTP Request Header "User-Agent".
Referrer	HTTP Request Header "User-Agent".
Via	Records the proxies through which the request was sent.

- [Security policies](#)
- [Monitoring security alerts and events](#)
- [Selecting a security event to monitor](#)

Tracking login failures by using a report

Login failures are recorded on the Pega log and are tracked in the database. To view the login failures, you can use a log viewer or you can write a report.

Pega Platform records login failures for all requestors, and can also record login successes if you configure that event on the **Security Event Configuration** landing page. Login successes and failures are tracked as instances of the *Log-SecurityAudit* class.

You can also view login failures on the Security Audit Log if you have configured the **Audit log level** on the Security Policies landing page.

1. Create a report with an applies to class equal to *Log-SecurityAudit*.
2. In the **Edit columns** section, list the columns to appear on the report.
3. **Optional:** In the **Edit filters** section, define filters for the report. **For example:** In the **Column source** field, enter *.pxCreateDateTime*. In the **Relationship** list, click **Greater than or equal to**. Click **Select values** and then click a time period.
4. Save and run the report.

- [Creating advanced reports](#)
- [Selecting a security event to monitor](#)
- [Viewing log files in an external log viewer](#)
- [Security events Log](#)
- [Security policies](#)

Monitoring security alerts and events

Pega Platform generates security alerts and events for situations such as attempts to hijack a user session. You can review the security alerts and events by viewing their respective logs.

Before you begin:

In Pega Platform, there are two different types of logs, one for alerts and one for events, and the distinction between these logs is important.

Alerts identify urgent items that are happening within your application. A Security Administrator must deal with these items as soon as possible. A Security Administrator is responsible for periodically examining and addressing any alerts.

Events general items that are happening within your application. For example, a user changing their password can be a good thing, especially if you have a policy in which they must change their passwords every six months.

To view either the *PegaRULES-ALERTSECURITY* log or the *PegaRULES-SECURITYEVENT* log:

1. In the header of Dev Studio, click **Configure System Operations Logs**.
2. Choose the log you would like to view:

For	Select by
Security alerts	Click the <i>PegaRULES-ALERTSECURITY</i> log for the list of Security Alerts.
Security events	Click the <i>PegaRULES-SECURITYEVENT</i> log for the list of Security Events.

What to do next:

Security alert codes start with *SECU*. For a list and explanation of security alert codes, see [List of performance and security alerts in Pega Platform](#).

- [Alert format](#)

Logging each use of harness and flow action rules

Your application can create an audit record each time an operator requests either a harness form or a flow action.

Logging occurs upon initial display of the form; it does not require that the user update any fields. Your application adds a data instance to the *pr_log_dataaccessaudit* table (corresponding to the *Log-DataAccessAudit* class). The instance identifies the following:

- Operator ID
- Harness rule or flow action rule instance
- Customer work party role (if any)
- Work object ID

This information enables you to support compliance auditing, reporting, and analysis. From a security standpoint, this feature lets you identify instances of unauthorized operator access to specific work objects. You can create list view rules and summary rules to report on the table instances, or export the data for analysis. This article describes how you enable the data access capability.

Enable this feature only if it is needed. Auditing increases the database workload of your application and can produce large volumes of log data.

To set the audit rule access, complete the following procedure:

1. Open the **Display Options** tab on the harness rule form, or the **Security** tab on the flow action form.
2. In the **Auditing** section, select the **Audit Use?** checkbox. **Result:** The **When** field and **Audit Activity** field (containing the default value *Audit*) appear.
3. Select a *when condition* rule in the **When** field.
The rule is evaluated each time a user displays a work object form based on this harness or flow action rule. This *when condition* operates in conjunction with the activity in the **Audit Activity** field. If the when rule evaluates to *true* or there is no when rule, the activity is executed.
For example: You can configure a rule that evaluates the **DeptName** property on the clipboard for the value *Finance*. If true, the audit activity is invoked.
4. By default, the standard activity *Work-Audit* is entered in the **Audit Activity** field. This activity is configured to run when a user requests a work object form based on the harness or flow action rule (and the when condition, if not blank, evaluates to true) and to add an instance to the *Log-DataAccessAudit* class. You can modify this activity or create a similar one in your work class to meet the needs of your application. For example, you can choose to record a role other than the default *Customer* or add other facts about the work object. Your activity should use the *Obj-Save* method with *WriteNow* enabled to ensure that logging occurs even when the work object is displayed but not updated.
5. If you build an activity that uses input parameters, click **Params** to provide values for each parameter. **Result:** The system automatically completes the following four parameter values; you do not need to complete them:
 - **pyRuleKey** — Handle (pzInsKey) of this harness rule.
 - **pyRuleClass** — Rule-HTML-Harness
 - **pyRuleAppliesToClass** — The *Applies To* key part of this harness rule
 - **pyRulePurpose** — The *Action Name* key part of this harness rule.

Mitigate common (OWASP Top 10) security vulnerabilities

Pega Platform offers policies on the Security Policies landing page, as well as additional security restrictions that control cross-site request forgery (CSRF), content security policies (CSP), cross-origin resource sharing (CORS), and other types of vulnerabilities. Use these features to ensure that your system is as secure as possible.

According to their official site, the OWASP Top 10 is a standard awareness document for developers and web application security specialists. It represents a broad consensus about the most critical security risks to web applications.

Pegasystems uses the 2017 [OWASP Top 10 Web Application Security Risks](#) as a means of focusing on the most effective steps towards producing more secure code and applications.

OWASP Top 10 Web application security risks

The 2017 OWASP Top 10 Web application security risks include the following:

1. **Injection:** Pega Platform prevents execution of unintended commands or access to data without proper authorization. For more information, see:
 - [Configuring the Java injection check](#)
 - [Implementing security guidelines for custom HTML](#)
2. **Broken authentication:** Pega Platform can prevent authentication and session management from being implemented incorrectly. For more information, see:
 - [Authentication](#)
 - [Authentication time-out](#)
 - [Content security policies](#)
3. **Sensitive data exposure:** Pega Platform aids in proper configuration to protect sensitive data. For more information, see:
 - [Compliance with regulatory standards](#)
 - [Encryption](#)
 - [Security policies settings](#)
 - [Using the login policies settings](#)
 - [Encrypting the values of sensitive properties](#)
4. **XML external entities (XXE):** Older or poorly configured XML processors evaluate external entity references within XML documents. Pega Platform follows leading practices in all of our code, in which XML parsing prevents XXE. As part of the security development life cycle (SDLC), Pega Platform has code scanners that check new or modified code and merges it into the repository. Through this process, bad code is blocked from the repository and must be addressed before the merge can be complete.
5. **Broken access control:** Pega Platform restricts what authenticated users are allowed to do and the policies surrounding user access as properly enforced. For more information, see [Using Access Control Checks](#).
6. **Security misconfiguration:** As the most common security issue, Pega Platform applications must be securely configured and updated and patched in a timely fashion. For more information, see:
 - [Content security policies](#)
 - [Using HTTP response headers](#)
 - [Defining cross-origin resource sharing policies](#)
7. **Cross-site scripting XSS:** Pega Platform includes features to prevent cross-site scripting, a client-side code injection attack, in which an attacker can run malicious scripts on a legitimate website or web application. For more information, see [Understanding cross-site scripting](#).
8. **Insecure deserialization:** Pega Platform allows for proper deserialization, which prevents remote code execution. For more information, see [Configuring the deserialization filter](#).
9. **Using components with known vulnerabilities:** Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts. There are several layers related to this issue. The easiest way to combat components with known vulnerabilities is to verify that all of the pieces of your applications are the most up to date, secure versions available. This includes installation of all upgrades and patches. This should also be done with all external services, Pega Platform, and computer operating systems. Pega Platform uses third-party components that also need to be secured. Scanners run in the background that analyze libraries used in the product and see whether those libraries are at risk, which is then reported.
10. **Insufficient logging and monitoring:** Pega Platform has sufficient logging and monitoring coupled with effective incident response. For more information, see:
 - [Auditing](#)
 - [Tracking and auditing changes to data](#)
 - [Tracking and auditing actions by developers and users](#)
 - [Monitoring security alerts and events](#)

Cross-site request forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they are currently authenticated. CSRF was part of the 2013 OWASP Top 10. Pega Platform continues to provide protections against CSRF. For more information, see [Understanding cross-site request forgery](#).

- [Configuring the Java injection check](#)

At design time and at run time, Pega Platform checks activities, functions, and stream rules for particular Java injection vulnerabilities.

- [Implementing security guidelines for custom HTML](#)

The Rule Security Analyzer can find specific JavaScript or SQL coding patterns that might indicate a security vulnerability. The most effective way to search for vulnerabilities is to run the Rule Security Analyzer several times, each time matching against a different regular expression rule. If the Rule Security Analyzer finds problems, you can fix them to make your system more secure.

- [Compliance with regulatory standards](#)

Regulatory compliance ensures that organizations are aware of and comply with relevant laws, policies, and regulations. Regulatory compliance is when a business follows international and local laws and regulations that are relevant to its operations.

- [Using Access Control Checks](#)

Use access control checks to identify broken custom code that must be fixed. During development, it is easy to introduce risks into your application by implementing custom code. By using access control checks, you help proactively fix your code by identifying potential issues.

- [Using HTTP response headers](#)

To improve the security of your application against client-based attacks, you can use the HTTP response headers that are supported by your browser.

- [Defining cross-origin resource sharing policies](#)

Cross-origin resource sharing (CORS) policies define a method that enables a browser and server to interact and determine whether it is safe to allow a cross-origin request. For example, a client using a Pega Marketing application running in a browser, may see advertisements from third-parties, and if they click one of these advertisements, the CORS policy will record that the advertisement was viewed or clicked on.

- [Understanding cross-site scripting](#)

Cross-site scripting is a client-side code injection attack, in which an attacker can run malicious scripts on a legitimate website or web application.

- [Configuring the deserialization filter](#)

In Pega Platform, a global filter checks a list of blocked classes that are not allowed to be deserialized. You can add classes to the global deserialization filter to increase the security of your application by preventing unauthorized access.

- [Understanding cross-site request forgery](#)

Configuring the Java injection check

At design time and at run time, Pega Platform checks activities, functions, and stream rules for particular Java injection vulnerabilities.

Pega Platform reports errors at design time and run time, and does not run any activity, function, or stream rule that was created in Pega Platform 8.3 or later and includes any of the following commands:

- Runtime.getRuntime()
- new ProcessBuilder()
- JavaCompiler
- org.dita.dost.invoker

For rules that were created before version 8.3, the system behavior depends upon the value of the dynamic system setting *security/enableJavaInjectionMitigation*.

- If a vulnerability is found and the dynamic system setting is not defined or is false, the rule runs and security alert SECU0018 appears on the security alert log.
- If a vulnerability is found and the dynamic system setting is true, an error is reported and the rule does not run.

Important: As a security best practice, set the dynamic system setting *security/enableJavaInjectionMitigation* in the owning ruleset *Pega-Engine* to true to prevent running vulnerable rules.

Optional: To check for Java injection vulnerabilities in addition to the default checks listed above, set the JVM system property named *JavaInjection* equal to a Regex pattern to flag as vulnerabilities. For Example: `-DJavaInjection="new Foo()"`

- [Mitigate common \(OWASP Top 10\) security vulnerabilities](#)
- [Creating a dynamic system setting](#)

Implementing security guidelines for custom HTML

The Rule Security Analyzer can find specific JavaScript or SQL coding patterns that might indicate a security vulnerability. The most effective way to search for vulnerabilities is to run the Rule Security Analyzer several times, each time matching against a different regular expression rule. If the Rule Security Analyzer finds problems, you can fix them to make your system more secure.

1. In the header of Dev Studio, click **Configure Org & Security Tools Security Rule Security Analyzer**.
2. Complete the **Search Criteria** section.
 - a. **Rulesets** - To scan all rulesets, make sure the **All Rulesets** check box is selected. To scan specific rulesets, clear the check box and select one or more rulesets.
 - b. **Optional: Ruleset version** - To analyze all versions, leave this field blank. To limit the analysis, enter the version information in one of the following ways.
 - Major version only (05)
 - Major and minor version (05-05)
 - Major version, minor version, and patch (05-05-05)
 - c. **Allow highest version only** - To scan only the highest version of each rule, make sure the check box is selected. To scan all versions, clear the check box.
 - d. **Optional: Updated Since** - To scan rules regardless of update date and time, leave this field blank. To scan only rules updated after a certain date and time, click the **Calendar** button and enter the date and time.
 - e. **Rule Types** - To scan all ruletypes within the chosen ruleset or rulesets, make sure the **All Ruletypes** check box is selected. To scan specific rule types within the chosen ruleset or rulesets, clear the check box and select one or more rule types.
 - f. **Allow unauthenticated activities visited in the list** - If you keep this check box selected, the tool analyzes activities that have **Allow direct invocation from the client or service** selected and **Require authentication to run** unselected on the **Security** tab of the Activity rule form.
 - g. **Expression List** - Click **Add expression** and select the regular expression to use for analyzing rules.
3. Click **Run Analyzer**.

The summarized search statistics are displayed in the **Search Statistics** section. For each rule type, the number of vulnerable rules and the number of analyzed rules are shown.
4. **Optional:** To see the detailed results in an Excel spreadsheet, click **Export as Excel**.

An Excel file is downloaded to your browser. For each vulnerability that is found, the Excel file lists ruleset name and version, rule name, and other information.

What to do next: If you generated a spreadsheet as described above, use it as a worksheet for analyzing and correcting potential vulnerabilities. The spreadsheet lists "Unknown risk" in the vulnerability assessment column for all rows. You are responsible for evaluating the risk for each finding and updating the spreadsheet. If you determine that the at-risk value cannot be altered by an end-user, update the row's risk level to "False positive" and provide an explanation. Otherwise, set the risk level to "High risk" or "Low risk," depending on how easy it is to exploit the finding and the impact of a successful exploit. Depending on your analysis, fix the at-risk vulnerabilities as described in [Analyzing security vulnerability search results](#).

- [Analyzing security vulnerability search results](#)
- [Regular Expression rules](#)

Analyzing security vulnerability search results

To make Pega Platform applications more secure, you can run the Rule Security Analyzer. This tool searches through non-autogenerated rules to find specific JavaScript or SQL coding patterns that match regular expression rules, some of which might indicate security vulnerabilities, as detailed below.

To use the analyzer, you must have the *Rule-SecurityVA* privilege in your access group's role. Standard developer roles such as *SysAdm4* include this privilege.

Note:

- The Rule Security Analyzer tool examines only custom code, not autogenerated rules.
- Blocked rules are ignored. These rules are identified by the property `.pyRuleAvailable = "Blocked"`.
- The tool scans rules in your own applications, not rules in standard Pega Platform rulesets.

The Rule Security Analyzer tool finds potential security vulnerabilities in code by searching for matches to regular expressions that are defined in Rule Analyzer Regular Expression rules. The system provides the following standard regular expressions. You can supplement these standard regular expressions with regular expressions that you create.

Expression Name	Description
	Finds possible cross-site scripting vulnerabilities from unencoded output to the browser. Calls to <code>tools.getActiveValue()</code> should be wrapped in <code>StringUtils.crossScriptingFilter</code> or converted to the appropriate non-string value. For example, the following expressions are allowed:

Expression Name	Description
<i>pyCrossSiteScriptingActiveValue</i>	- String x = StringUtils.crossScriptingFilter(tools.getActiveValue()) - int y = Integer.parseInt(tools.getActiveValue()) For more information, see Understanding cross-site scripting.
<i>pyCrossSiteScriptingFromParam</i>	Finds possible cross-site scripting vulnerabilities from unencoded output to the browser. Calls to <code>tools.getParamValue()</code> should be wrapped in <code>StringUtils.crossScriptingFilter</code> or converted to the appropriate non-string value. For example, the following expressions are allowed: - String x = StringUtils.crossScriptingFilter(tools.getParamValue("x")) - int y = Integer.parseInt(tools.getParamValue("y")) For more information, see Understanding cross-site scripting.
<i>pyDatabase</i>	This is an old version of the <i>pySQLInjection</i> rule, which is described below. Finds direct calls to the database, for example: <code>executeRDB()</code> , <code>createRDBListSpec()</code> , <code>RDBExecute()</code> .
<i>pyLogStatements</i>	Rules that call <code>oLog</code> or <code>System</code> are flagged, and should be reviewed manually to ensure that no critical or confidential data is logged.
<i>pySQLInjection</i>	Finds patterns and APIs that might inject unfiltered user input to an SQL query. Direct SQL queries that could be maliciously modified at run time should not be allowed, for example: <code>executeRDB()</code> , <code>createRDBListSpec()</code> , <code>RDBExecute()</code> , <code>executeDBSchemaUpdates()</code> . Validate all input from the user and use parameterized queries with bind variables where possible.
<i>pySystemCall</i>	Finds system calls, for example: <code>getRuntime()</code> , <code>ProcessBuilder</code> . These calls are permitted but should be manually reviewed.
<i>pyUnsafeURL</i>	To prevent data from being misinterpreted as having a special meaning, finds URLs and query parameters that are not encoded.
<i>pyXMLExternalEntity</i>	Finds instances of XML parsing with <code>DocumentBuilderFactory</code> Or <code>XMLInputFactory</code> (among others) that are not protected against XML External Entity (XXE) attack vectors. For a complete list of the validations performed by this rule, see the GitHub web site for OWASP Cheat Sheet Series (XML_External_Entity_Prevention_Cheat_Sheet.md). For example, the following are considered safe when using <code>DocumentBuilder.newInstance()</code>: - setExpandedEntityReferences(false) - setFeature("http://apache.org/xml/features/disallow-doctype-decl",true) - setFeature("http://xml.org/sax/features/external-general-entities",false)

The following regular expressions are also provided and can be used with the analyzer, although they are not specifically related to security.

Expression Name	Description
<i>pyCustomJS</i>	Finds rules that attempt to obtain node information and other information that might interfere with an upgrade. For example, rules cannot contain expressions such as <code>CHILDNODES</code> , <code>PARENTNODE</code> , <code>FIRSTCHILD</code> , <code>LASTCHILD</code> , <code>NEXTSIBLING</code> , <code>PREVIOUSIBLING</code> , and so forth.
<i>pyGetThreadInRules</i>	Finds references to <code>getThread()</code> .
<i>pyNbsplnDSDefaultCaption</i>	Finds <code>DynamicSelect</code> controls that contain a default caption that includes <code>&nbsp; </code> .
<i>pyNbsplnSections</i>	Finds controls that contain the value <code>&nbsp; </code> .
<i>pyUnsafeURL</i>	Finds dynamic generation of URLs that do not use the encoding API.

- [Implementing security guidelines for custom HTML](#)
- [Security guidelines for custom HTML](#)
- [Regular Expression rules](#)

Security guidelines for custom HTML

As a best practice, do not include custom HTML in your application because it is not autogenerated. If you need to use custom HTML, ensure that you take precautions to protect your application from hackers.

The following guidelines can help you minimize the security vulnerabilities in the custom HTML that you create:

- Use the `SafeURL` function to generate Pega Platform request URLs in custom JavaScript.

For more information, see [URL JavaServer Page tag](#).
- Minimize or remove hidden properties (type=hidden), because these properties are common targets for malicious users.
- Remove comments or other sensitive information so that users cannot see them. Use `<%...` instead of `<!--...` to tag comments so that users do not receive comment text.
- For custom Pega JavaServer Page tags, either omit the mode attribute or set `mode=normal`.
- Minimize custom buttons because their URL construction can bypass filtering for potentially hazardous characters.
- Minimize the use of `mode=literal`, because the text that it references does not filter potentially hazardous characters.
- Minimize custom Java steps in activities, especially those that call Parse XML rules, to prevent possible XML injection.
- Replace dynamic SQL statements with prepared statements that have parameterized queries to prevent possible SQL injection.
- When using parameters in a JavaServer Page tag or Pega Platform reference tag, use the `StringUtils.crossScriptingFilter` method.
- If you are using the inline Microsoft Word editing feature, be aware that it downloads Microsoft Word documents to the user desktop without notification.
- When deploying a Pega Web Mashup application, consider configuring your deployment as a web node, and use the activity allow list feature to enhance security.

For more information, see [Creating cases from a web mashup](#).

- Use strong authentication for Web Mashup users. Do not hardcode credentials in your web mashup gadget.
- Define Web Mashup security by specifying trusted origins that can use and communicate with the Web Mashup gadget.

For more information, see [Securing your application for mashup communication](#).
- Consider securing your application by using the Access Group setting Rule security mode that locks down access to specific features in your application.

For more information, see [Adding custom tools to Access Manager](#).

- Consider implementing the when condition *pyAllowDoUIAction* to further lock down actions that the *DoUIAction* activity provides.

For more information, see [When form - Completing the Conditions tab](#).

Regular Expression rules

Regular Expression rules support operation of the Rule Security Analyzer by defining patterns of text that when found in the code being examined, cause the analyzer to report potential security issues.

Open a regular expression rule from the navigation panel by clicking **Records Security Rule Analyzer Regular Expression** and selecting a rule from the instance list.

On the Regular Expression tab, enter the regular expression (regex). Note that the system does not check that the regular expression is properly formed, and does not warn you if it contains a problem in its syntax.

Category

Regular Expression rules are part of the Security category. A regular expression rule is an instance of the *Rule-SecurityVA-Regex* rule type.

- [Analyzing security vulnerability search results](#)

Creating a regular expression rule

Records can be created in various ways. You can add a new record to your application or copy an existing one. You can specialize existing rules by creating a copy in a specific ruleset, against a different class or (in some cases) with a set of circumstance definitions. You can copy data instances but they do not support specialization because they are not versioned.

Create a regular expression rule by clicking **Create Security Rule Analyzer Regular Expression** . Enter an identifier that starts with a letter and uses only letters, digits and underscore characters.

- [Regular Expression rules](#)

Testing regular expressions

Test for potential security issues that custom code can introduce into your application. Use the Regular Expression tester to test whether a text pattern matches the regular expression.

1. In the navigation panel of Dev Studio, click **App**.
2. Use the Application Explorer to open the *Code-Pega-Parse.RegExpTester* standard activity.
3. Click **Actions Run** . The system displays a test input form.
4. On the test input form, in the **Page** list, click **Empty test page**.
5. Leave the parameter values blank.
6. Click **Run**.
7. Complete the following fields on the detail form:

Field	Enter
Regular Expression	Enter a regular expression to use, using syntax conforming to the java implementation of regular expressions (Java.util.regex.Pattern).
Source	Enter source text to use to search for matches to the regular expression.
UNIX Lines	Select this check box to recognize only newline characters (\n, also called line feed) as end-of-line delimiters. Clear to recognize additional characters or character pairs as end-of-line delimiters: carriage return (\r), carriage return followed by newline, next-line (\u0085), line-separator (\u2028) or paragraph-separator (\u2029).
Multilines	Select this check box to cause the ^ and \$ expressions to match immediately after or immediately before a line terminator or the end of the input sequence respectively. Clear to have these expressions only match at the beginning and the end of the entire input sequence.
Canonical Equivalence	Select this check box to cause two characters to match if their full canonical decompositions match. For example, the expression a\u030A matches the string "á" when you select this flag. By default, matching does not take canonical equivalence into account.
Comments	Select this check box to ignore all white space (tabs and spaces) within the Source text, and to ignore all material starting with a # comment character through the end of the line.
Dot Matches All	Select this check box to indicate that a period character matches any character, including a line terminator. Clear to indicate that a period character matches any character except a line terminator.
ASCII Case Insensitive	Select this check box to match uppercase with lowercase ASCII characters.
Unicode Case Insensitive	Select this check box to match uppercase with lowercase ASCII characters.

8. Click **Test Expression**.
- Result:** Pega Platform generates a table that lists the results of each match of regular expressions found in the source text.
- [Regular Expression rules](#)

Compliance with regulatory standards

Regulatory compliance ensures that organizations are aware of and comply with relevant laws, policies, and regulations. Regulatory compliance is when a business follows international and local laws and regulations that are relevant to its operations.

The specific requirements can vary, depending on the industry and type of business. Regulatory compliance also pertains to specific industries. No matter the industry or company size, all businesses must adhere to certain laws and regulations as part of operations.

The specific requirements can also vary depending on the country in which the business operates. For example, if your company is based in the United States but serves customers in Europe, you must follow the local standards (such as the General Data Protection Regulation) that ensure the rights of your European customers.

Regulatory compliance should not be confused with compliance with company policies and procedures, and compliance with internal requirements set forth by the business. While all three types of compliance are important to ensure integrity, safety, and ethical behavior in businesses, it helps to understand the difference.

Pega clients need to comply with local regulations when developing and managing their applications. For example:

- A healthcare provider in the United States must keep the Health Insurance Portability and Accountability Act (HIPAA) standards in mind when developing an application, because personal identifiable information must be encrypted to be kept secure and in compliance.
- A United States-based financial services company that does business in Europe would have to abide by the regulations set forth in the General Data Protection Regulation (GDPR) for their European customers. GDPR regulations need to be considered during development, but also when the application is live, because GDPR gives European customers the right to have their data deleted.

California Consumer Privacy Act - United States regulation

The California Consumer Privacy Act (CCPA) is a state-wide data privacy law that regulates how businesses all over the world are allowed to handle the personal information (PI) of California residents.

Note: For more information, see the official [California Consumer Privacy Act](#) website.

Health Insurance Portability and Accountability Act - United States regulation

HIPAA was created primarily to modernize the flow of healthcare information in the United States, stipulate how personally identifiable information is maintained by the healthcare and insurance industries, and address limitations on healthcare insurance coverage. Dominantly, HIPAA provides security provisions and data privacy to keep patients' medical information safe.

Note: For more information, see the official [Health Insurance Portability and Accountability Act](#) website.

Federal Risk and Authorization Management Program - United States regulation

The Federal Risk and Authorization Management Program (FedRAMP) is a United States government-wide program that provides a standardized approach to security assessment, authorization, and continuous monitoring for cloud products and services.

FedRAMP simplifies security for the digital age by providing a standardized approach to security for the cloud.

Note: For more information, see the official [FedRAMP](#) website.

- [General Data Protection Regulation](#)

Implementing client-based access control (CBAC) helps you satisfy the data privacy requirements of the European Union (EU) General Data Protection Regulation (GDPR) and similar regulations. Personal data is associated with an actual person, not with an abstract entity such as a business.

General Data Protection Regulation

Implementing client-based access control (CBAC) helps you satisfy the data privacy requirements of the European Union (EU) General Data Protection Regulation (GDPR) and similar regulations. Personal data is associated with an actual person, not with an abstract entity such as a business.

If your application stores data that could be used to identify a person and you are subject to GDPR or similar regulations, use client-based access control to track and process requests to view, change, remove, and restrict the use of personal data, and to show auditors that you have done so.

Note: For more information about GDPR, see the [EU General Data Protection Regulation](#) (GDPR) website.

Terminology

This article and related articles use the following terminology:

Pega application

An application that is built on Pega Platform and contains personally identifiable data

Client

A person who is a customer of yours and whose personal data you manage. (In GDPR, this person is known as the “data subject.”)

Repository

A Pega or non-Pega application that stores personally identifiable data, within which client requests must be enforced. This type of repository is different from the repositories that are used in the CI/CD pipeline.

Types of requests

Rectify and erase requests are one-time operations. They do not prevent data from being changed or added again in the future. Pega Platform can be configured to automatically support the following types of personal data requests:

Personal data request types	Configuration behavior
Request to access	Find all the personal data for a client and return the values to the client.
Request to rectify	Correct personal data for a client for properties that you support changing.
Request to erase	Delete personal data for a client for properties that you support deleting.
Request to restrict usage	Prohibit particular functions from accessing data for a specific client. For example, a client might consent to use your banking system but not to receive marketing communications.

- [Personal data restrictions for GDPR](#)

The General Data Protection Regulation (GDPR) imposes strict requirements for protecting personal data. When you develop Pega applications, you can restrict personal information from certain functions.

Personal data restrictions for GDPR

The General Data Protection Regulation (GDPR) imposes strict requirements for protecting personal data. When you develop Pega applications, you can restrict personal information from certain functions. You can restrict the access to personal information from internal and external users by using access groups.

Function restrictions

Pega Platform uses instances of the function restrictions class (*ClientData-FunctionRestrictions*) to store the restrictions on the database. Function restrictions are unique by client ID and application name. If no restrictions are requested by a customer, there are no instances.

When a customer requests restrictions, a function restrictions instance is created. The instance specifies each function in your application, and within each function, each channel where functions can be used. For each function and channel, the instance defines whether that function can be executed. A customer can also request to update or delete function restrictions.

You can view the detailed specification for this REST API in the Pega API specification under the Data Privacy category.

For example: For example, John Doe, a customer included in your MyCRM application, does not want to receive marketing phone calls. The function restrictions for John Doe are shown in the following example:

ClientID: JohnDoe **Application:** MyCRM

Function: Billing

- **Channel:** Email **Is Restricted?** No
- **Channel:** Phone **Is Restricted?** No

Function: Marketing

- **Channel:** Email **Is Restricted?** No
- **Channel:** Phone **Is Restricted?** Yes

Function restrictions data transform

You customize the *pyClientRestrictedFunctions* data transform to define the functions and channels that are used by your application and their default restriction values. For sample code, refer to the *pyClientRestrictedFunctions* data transform that is in the *Pega-RulesEngine* ruleset.

When rule

The function *pzIsFunctionRestricted* identifies when a function is restricted for an app, client, function, and optional channel. You create a when rule that calls this function, and you invoke the when rule at the appropriate place in your application so that the function restrictions take effect. For example, the when rule can be used when calculating the next best action or when rendering a section of the user interface so that John Doe does not receive marketing phone calls.

Using Access Control Checks

Use access control checks to identify broken custom code that must be fixed. During development, it is easy to introduce risks into your application by implementing custom code. By using access control checks, you help proactively fix your code by identifying potential issues.

If you do not fix broken access controls, when you enable security protections and the application goes into hardening, the broken features may stop working or may not work properly.

1. In the header of Dev Studio, click **Configuration Org and Security Tools Security Access Control Check**.
2. **Optional:** On the **Search Criteria** tab, select from the following check boxes:
 1. Select an **Application** check box to define the application on which you will run the access control check, as shown in the following figure:

Defining the application to run an access control check

Search Criteria

2 applications

Rt

☐ CDX89App

Ru

☒ Cosmos Rules

☐ Pega Developer

☐ PegaSSOT

☒ Pega Platform

Apply

Search Criteria

2 applications

Rt

☐ CDX89App

Ru

☒ Cosmos Rules

☐ Pega Developer

☐ PegaSSOT

☒ Pega Platform

Apply

- Note:** The current application is selected by default. If you have multiple applications configured, you can run tests on the applications at the same time.**Note:** Running tests for multiple applications at the same time takes longer to process the request. The more applications that you check, the longer the Access Control Check will take.
2. Select the **Rulesets** check box to define the rulesets on which you want to run the Access Control Check. By default, all rulesets are selected. However, to only run checks on certain rulesets, clear the **Ruleset** check box to manually indicate which rulesets you want to run the check on.**Note:** When you make small changes to a ruleset that you want to verify, it is more effective to run the Access Control Check on just the rulesets that you change, to ensure that they do not cause any unintended issues.
 3. **Optional:** To select whether to display and run the Access Control Check on rules that call custom code in custom calls or on a Standard API, select one of the following radio buttons:

- Custom calls in custom code

Runs the rules in your applications that call custom code from custom calls.
- Standard API calls in custom code

Runs the rules in your applications that call custom code from Standard API.

4. Click **List Rules for mitigation** button.
The results display as shown in the following figure:

List rules for mitigation results

Rule Type	Rule Label	Ruleset	Preview
Rule-HTML-Property	DDOACTORSALTOCOMPLETECONFIRM	Pega-AppDefinition	☒
Rule-HTML-Property	DYNAMICSELECTFORAPPLICATION	Pega-AppDefinition	☒
Rule-HTML-Property	PXREFRESHACTORS	Pega-AppDefinition	☒
Rule-HTML-Property	PXREFRESHREQUIREMENTS	Pega-AppDefinition	☒
Rule-HTML-Property	PXTOGGLEBUTTON	Pega-AppDefinition	☒
Rule-HTML-Property	PZMPACKAGEPOLLING	Pega-AppDefinition	☒
Rule-HTML-Property	PZMOBILEBUILDPOLLING	Pega-AppDefinition	☒
Rule-HTML-Property	PZMOBILEBUILDPROGRESS	Pega-AppDefinition	☒
Rule-HTML-Property	PZREFRESHDOCUMENTFORPACKAGE	Pega-AppDefinition	☒
Rule-HTML-Property	REFRESHLEMENTFORSECTION	Pega-AppDefinition	☒
Rule-HTML-Property	SIZINGWIZARDSAVEASLABEL	Pega-AppDefinition	☒
Rule-HTML-Fragment	ACTIONSORENCAPTUREFIELDS	Pega-AppDefinition	☒
Rule-File-Text	WEBWPZAGLESANITIZEDBACKQS	Pega-AppDefinition	☒
Rule-File-Text	WEBWPZDICO_PROCESSQS	Pega-AppDefinition	☒

Group Fields Density Refresh			
Rule Type	Rule Label	Ruleset	Preview
Rule-HTML-Property	DCACTORSAUTOCOMPLETESCONFIRM	Pega-AppDefinition	Preview
Rule-HTML-Property	DYNAMICSELECTFORAPPLICATION	Pega-AppDefinition	Preview
Rule-HTML-Property	PXREFRESHACTORS	Pega-AppDefinition	Preview
Rule-HTML-Property	PXREFRESHREQUIREMENTS	Pega-AppDefinition	Preview
Rule-HTML-Property	PXTOGGLEBUTTON	Pega-AppDefinition	Preview
Rule-HTML-Property	PZAPPPACKAGEPOLLING	Pega-AppDefinition	Preview
Rule-HTML-Property	PZMOBILEBUILDPOLLING	Pega-AppDefinition	Preview
Rule-HTML-Property	PZMOBILEPUBLISHPROGRESS	Pega-AppDefinition	Preview
Rule-HTML-Property	PZREFRESHDOCWIZFORPACKAGE	Pega-AppDefinition	Preview
Rule-HTML-Property	REFRESHELEMENTFORUCSECTION	Pega-AppDefinition	Preview
Rule-HTML-Property	SIZINGWIZARDSAVEASLABEL	Pega-AppDefinition	Preview
Rule-HTML-Fragment	ACTIONSSCREENCAPTUREFIELDS	Pega-AppDefinition	Preview
Rule-File-Text	WEBWIPZAGILESMARTFEEDBACKJS	Pega-AppDefinition	Preview
Rule-File-Text	WEBWIPZDCO_PROCESSJS	Pega-AppDefinition	Preview

5. **Optional:** To view the matched strings for each line result, click the Preview icon for that line. The following figure shows an example of the line preview:

Line preview results

Group Fields			
Rule Type	Rule Label	Ruleset	Preview
Rule-HTML-Property	DCACTORSAUTOCOMPLETESCONFIRM	Pega-AppDefinition	Preview
Rule-HTML-Property	DYNAMICSELECTFORAPPLICATION	Pega-AppDefinition	Preview
Rule-HTML-Property	PXREFRESHACTORS	Pega-AppDefinition	Preview
Rule-HTML-Property	PXREFRESHREQUIREMENTS	Pega-AppDefinition	Preview
Rule-HTML-Property	PXTOGGLEBUTTON	Pega-AppDefinition	Preview
Rule-HTML-Property	PZAPPPACKAGEPOLLING	Pega-AppDefinition	Preview
Rule-HTML-Property	PZMOBILEBUILDPOLLING	Pega-AppDefinition	Preview
Rule-HTML-Property	PZMOBILEPUBLISHPROGRESS	Pega-AppDefinition	Preview
Rule-HTML-Property	PZREFRESHDOCWIZFORPACKAGE	Pega-AppDefinition	Preview
Rule-HTML-Property	REFRESHELEMENTFORUCSECTION	Pega-AppDefinition	Preview
Rule-HTML-Property	SIZINGWIZARDSAVEASLABEL	Pega-AppDefinition	Preview
Rule-HTML-Property	ACTIONSSCREENCAPTUREFIELDS	Pega-AppDefinition	Preview
Rule-File-Text	WEBWIPZAGILESMARTFEEDBACKJS	Pega-AppDefinition	Preview
Rule-File-Text	WEBWIPZDCO_PROCESSJS	Pega-AppDefinition	Preview

6. **Optional:** To group rules based on Rule Type or Ruleset, click **Group**.
7. Open each rule to refactor and address its issues.

Result: The system will display a table of Access Control alerts that you need to mitigate.

Content security policies

The Content Security Policy (CSP) is a set of directives that inform the user's browser of locations from which an application is allowed to load resources. These locations are provided in the form of URL schemes, including the use of an asterisk (*) to represent all URLs. Each directive governs a specific resource type that affects what is displayed in a browser. Collectively, the directives are sent to the client in the Content-Security-Policy HTTP header. Each browser type and version obey as much of the policy as they can. If a browser does not understand a directive, it is ignored; otherwise it is explicitly followed.

Content Security Policy

The **Policy Definition** tab contains a section for each source directive. Set the policy directives for each category displayed by clicking the category title to display or hide its fields.

You can create a content security policy or open an existing instance from the navigation pane by clicking **Records Security Content Security Policy** and selecting an instance.

For most of the source directives, you can choose from the following restrictions.

- **None** – Do not allow any location or URL.
- **Allow-All** – Allow all locations for this resource.
- **Self** – Match the location of the containing document but not subdomains or other domains.
- **Data** – Match inline resources defined using data.

Note: Exceptions are described in the relevant section below.

For most source directives, you can also list individual allowed websites. For more information, see [Allowing a website as an exception to a content security policy directive](#).

You can configure the following source directives on the **Policy Definition** tab.

Default-Src

The Default-Src directive serves as a fallback for the other CSP fetch directives. For each of the following directives, if the directive is not explicitly stated, the user or agent applies the value for the default-src directive: child-src, connect-src, font-src, img-src, media-src, object-src, script-src, style-src.

Base-URI

The Base-URI directive governs the document base URL. Most websites use a relative link system, which informs a web browser where a resource is relative to its location. A browser uses the document's base Uniform Resource Identifier (URI) and the relative link to create a full path to that resource. An attacker who controls the base URI can force the user's browsers to pull potentially malicious content from the attacker's site.

Connect-Src

The Connect-Src directive restricts the URLs that the protected resource can load. This restriction includes, but is not limited to, EventSource, WebSocket, and XMLHttpRequest (the driving source behind AJAX) connections. Limiting the XMLHttpRequest connection sources can protect users from attacks where an attacker forces a user's browser to make connections without alerting the user. Use this directive if your application's users make cross-origin requests as part of a cross-origin resource sharing (CORS) system.

Font-Src

The Font-Src directive controls the locations from which fonts can be loaded. An attacker can exploit several vulnerabilities that target the browser's font generation. Such an attack could compromise a user's browser.

Form-Actions

The Form-Actions directive governs the URLs that can be used as an action of the HTML <form> element. An attacker who gains access to this directive could compromise a user's data and submit potentially confidential information to the attacker's website.

Frame-Ancestors

The Frame-Ancestors directive restricts access from websites that can embed your application by using a <frame>, <iframe>, <object>, <embed>, or <applet> element. An attacker can embed your application in a malicious site, and log each keystroke and mouse click made by users who visit the site to use your application.

Child Frame-Src

The Child Frame-Src directive manages the content sources that your application can include in <frame> and <iframe> elements. An attacker can control the frame source and make it pull malicious data, including cross-site scripting attacks, into your application user's browser. Avoid using **Allow-All**.

Image-Src

The Img-Src directive controls the sources from which your application can load images. Attackers can use the HTML tag to extract confidential information through a cross-site scripting attack and make an image request to their own malicious site to request a non-existent image and append the page content. The attacker can then view the malicious site's logs to read your page's content.

Media-Src

The Media-Src directive manages sources from which your application can download media such as videos and audio files. An attacker can compromise a page to load a malicious object that can compromise a user's computer.

Object-Src

The Object-Src directive manages sources from which your application can download objects by using the <object>, <embed>, and <applet> elements. Such objects include Flash files, Java applets, scripts in other languages, and generic text documents. Flash files and Java applets can run any kind of code; if an attacker can compromise a page to load a malicious object, the user's computer could be compromised.

Plugin-Types

The Plugin-Types directive contains a list of allowed resource types that can be retrieved and used to instantiate plug-ins. This directive in conjunction with other directives, particularly the Object-Src directive, can ensure that all the content that is loaded by the plug-in is the correct content. An attacker is able to upload malicious content, such as a Java applet, to a source that is defined in Object-Src.

Sandbox

The Sandbox directive specifies an HTML sandbox policy that the user applies to the protected resource. These policy settings are closely aligned with the World Wide Web Consortium (W3C) specification. If these settings are too permissive, an attacker can load a malicious site through a frame.

Script-Src

The Script-Src directive restricts the scripts that the protected resource can run, protecting users against script injection attacks (for example, XSS). Pegasystems recommends adding specific websites to the Allowed Websites list. Avoid using **Allow-All**. Pega Platform applications must have *unsafe-eval* and *unsafe-inline* checked. For more information on specifying *unsafe-eval* and *unsafe-inline*, see [CSP: script-src](#).

Style-Src

The Style-Src directive governs the sources of styles (stylesheets) that can be used. Attackers can use the <style> tag to describe CSS style sheet content or external sources of style sheets. A style sheet loaded from a malicious site might make your application unusable by overriding content with images, odd colors, decreasing opacity, or by entirely removing your content.

Pegasystems recommends adding specific websites to the Allowed Websites list. Avoid using **Allow-All**.

Pega Platform applications must have *unsafe-eval* and *unsafe-inline* checked. For more information on specifying *unsafe-eval* and *unsafe-inline*, see [CSP: script-src](#).

- [Securing your application with a content security policy](#)
- [Application URL alias](#)
- [Allowing a website as an exception to a content security policy directive](#)

Securing your application with a content security policy

You can use content security policies to indicate from where your application can load resources, which makes your application more secure. To view or update the content security policies in your application, or to view the content security policies that are available in Pega Platform, do one of the following actions.

- To specify a content security policy for your application, complete the following steps.
 - a. In the Dev Studio header, click **Application name Definition**.
 - b. On the Application form, click the **Security** tab.
 - c. In the **Policy name** field, press the Down Arrow key, and then select the name of a content security policy.
 - d. Specify whether to enforce the policy, or to merely report usage of the policy, by clicking one of the following.
 - **Reject and report** – Enforces the policy.
 - **Report only** – Reports, but do not enforce the policy.
 - e. Click **Save**.
- To view the content security policy for your application, in the header of Dev Studio, click the name of the application, and then click **Definition**. Then click the **Security** tab.
- To list all the content security policies that are available to you, in the navigation panel, click **Records Security Content Security Policy**.

- [Creating a content security policy](#)
- [Content security policies](#)

Creating a content security policy

Create a content security rule so that browsers load resources only from authorized web sites.

1. In the Dev Studio header, click **Create Security Content Security Policy**.
2. Enter a short description.
3. Click **Create and open**.
4. Configure the content security policy as described in [Content security policy directives](#).

- [Content security policies](#)

Allowing a website as an exception to a content security policy directive

For various directives in a content security policy, you can explicitly grant access to selected websites. These websites have access to the application as an exception to the configured policy.

Before you begin: You must complete the following task before you can add a website to a content security policy directive: [Content security policy directives](#)

1. Create a content security policy, or open an existing instance from the navigation panel by clicking **Records Security Content Security Policy** and selecting an instance.
2. On the **Policy Definition** tab, expand the section for the directive for which you want to list allowed websites.
3. Under **Allowed websites**, click the **Add a row** icon.
4. In the **Allowed websites** field, enter the URL of the website for which to grant access.
5. **Optional:** In the **Notes** field, enter a short description about why the site should have access.
6. Click **Save**.

- [Configuring a setting](#)
- [Content security policies](#)

Using HTTP response headers

To improve the security of your application against client-based attacks, you can use the HTTP response headers that are supported by your browser.

Verify that you test every custom header that you create for your application. In some situations, custom headers can cause problems with how the application operates.

Pega Platform supports the ability to add custom headers. However, it is important to note that Pega Platform only sends these headers on dynamic content requests and not on static content requests.

You might consider adding the following security headers to your application:

- **X-XSS-Protection** – Prevents cross-site scripting. Prevents attackers from injecting client-side scripts into the website that is viewed from the user side.
- **Strict-Transport-Security** – Allows a website to tell browsers that they should communicate only by using HTTPS, not HTTP.

For browsers other than Internet Explorer, do not attempt to set a custom X-Frame-Options response header. Instead, the correct security setting approach is to use a Content Security Policy. If you use both X-Frame-Options and content security policy, be sure to test in order to verify that the options function as intended.

For more information, see [Content security policies](#).

- [Creating a custom HTTP response header](#)

You can create a custom application header to improve the security of your application to protect it from client-based attacks. However, use caution when using custom application headers because they might interfere with how the application operates. Be sure to test the application after implementing custom application headers.

- [Testing a custom HTTP response header](#)

To determine whether a custom application header has been correctly applied, you need to test it.

- [Understanding dynamic system settings](#)

To enable greater security in your application, configure the following dynamic system settings to enable greater security in your application before moving your application from development to a production environment.

Creating a custom HTTP response header

You can create a custom application header to improve the security of your application to protect it from client-based attacks. However, use caution when using custom application headers because they might interfere with how the application operates. Be sure to test the application after implementing custom application headers.

1. In the navigation panel, click **Records SysAdmin Dynamic System Settings**.
2. In the **Setting Purpose** field, click the **Filter** icon.
3. In the **Search Text** field, enter `http/responseHeaders` and click **Apply**.
4. Click the instance that contains the name.
5. On the **Settings** tab, in the **Value** field, enter the header parameters in the format: `("header name":"header value")`, or for multiple headers, `("header1 name":"header1 value","header2 name":"header2 value")`.

Following are some examples:

```
("X-Content-Type-Options":"nosniff")
("X-XSS-Protection":"1; mode=block")
("Strict-Transport-Security":"max-age=31536000; includeSubDomains")
("X-Content-Type-Options":"nosniff", "X-XSS-Protection":"1; mode=block")
```

You can add a Content-Security-Policy in a format such as `("Content-Security-Policy":"default-src 'self'")`, but best practice is to define content security policies as described in [Securing your application with a content security policy](#).

Note: For browsers other than Internet Explorer, do not attempt to set a custom X-Frame-Options response header. The correct security setting to use instead is Content Security Policy. For more information, see [Content security policies](#). If you use both X-Frame-Options and content security policy, be sure to test to verify that the options function as intended.

6. **Optional:** To see an example configuration, click the **History** tab.

- [Using HTTP response headers](#)
- [Testing a custom HTTP response header](#)

Testing a custom HTTP response header

To determine whether a custom application header has been correctly applied, you need to test it.

1. Open a case in your application.
2. In the navigation pane, click **Records SysAdmin Dynamic System Settings**.
3. In the **Key Contains** field, enter `response`.
4. Click the instance that contains the `http/responseHeaders` name.
5. Press the **F12** function key.
6. In the **Developers Tool**, click the **Network** tab.
7. Click a request.
8. Click the **Headers** tab. In the **Response Headers** section, you can see the customized header.

- [Creating a custom HTTP response header](#)
- [Using HTTP response headers](#)

Understanding dynamic system settings

To enable greater security in your application, configure the following dynamic system settings to enable greater security in your application before moving your application from development to a production environment.

Dynamic system setting information

As a leading practice, before moving your application from development to a production environment, configure the following dynamic system settings to enable greater security in your application.

Note: When you define security settings using dynamic system settings, the values are stored in the Pega database and are applied to all nodes and are therefore not node specific.

Purpose	Default	Secure setting	Security Implications
prconfig/alerts/database/operationTimeThreshold/suppressInserts/default	true	true	Recommended for all deployments. Prevents SQL statements from being written to the alert log in clear text. By default, all entries in the alert log show all data associated with the alert, including customer ID numbers, passwords, and other sensitive data. Setting this entry to true prevents sensitive data from being written to the alert log. Prevents SQL injection attacks and prevents exposing sensitive information about how data is written to the database.
prconfig/alerts/parameterpage/allowedKeywords/default	Blank	Blank	Eliminates PII data from the alert log, making it potentially more difficult to resolve the issue reported by the alert. The following keywords are supported: <i>pyActivity, pyStream, action, harnessName, StreamClass, StreamName, ViewClass, ViewPurpose, ViewOwner, objClass, insName, Format, openHandle, ActivityClassToExecute, ActivityNameToExecute, TaskStatus, FlowClass, FlowType, flowType, CustomActivityName, CustomActivityClassName, actionName, productName, productVersion, portal, pyAction, pyClassName, primaryPageClass, ViewInsKey, InsKey, pyReportName, pyReportClass.</i>
prconfig/alerts/parameterpage/remoteFilterType/default	Allowed	Allowed	Eliminates all clear-text information in the alert log, making it potentially more difficult to resolve the issue reported by the alert.
prconfig/crypto/onewayhashalgorithm/default	bcrypt	bcrypt	Hashing algorithm for operator password storage. As a best practice, set this setting before creating the operator that is used during testing. The bcrypt default is salted.
prconfig/Database/dumpStats/default	false	false	Recommended for all development and testing deployments. This is a high-volume-output tool only for use in development and testing environments. Do not use it in production. Prevents exposing sensitive information that could otherwise aid a hacker in predicting system behavior.
prconfig/HTTP/UseNoCacheHeaders/default	true	true	Recommended for all deployments. Prevents dynamic content and sensitive information from being cached on the client, regardless of expiration time. Also disables tracer functionality and forces fresh loading of the dynamic content from the server for each request. Prevents session hijacking, injection attacks, and cross-site scripting.
prconfig/initialization/DisableAutoComplete/default	false	true	Recommended for all deployments. This setting prevents client-side storage of user name and password combinations. Use this setting in conjunction with clearing any existing stored sensitive information in the browser.
prconfig/initialization/DisplayExceptionTraceback/default	false	false	Recommended for all deployments. This setting prevents display of stack-trace when an error occurs, and removes the Show Exception Details button, which could expose sensitive information in a production environment.
prconfig/initialization/ProfileApplication/default	false	false	Recommended for all deployments. This setting turns off the Application Profiler, which writes sensitive information to log files.
prconfig/initialization/PromoteEmbeddedPortals/default	false	true	Recommended for all deployments. This setting prevents a Pega Platform HTML frame from being embedded in an invisible additional frame that

Purpose	Default	Secure setting	Security implications
prconfig/initialization/OnErrorInvalidThreadName/default	true	true	Rejects requests that contain invalid characters in the threadname of the URL that potentially can be malicious, for example, symbol characters.
prconfig/Timeout/Browser/default	900	900 (or fewer)	Specifies the time-out value (in seconds) for the user session. If the user does not perform any system action specified by the time-out value, the user session is terminated.
prconfig/Cookie/HTTPOnly/default	true	true	Prevents client-side JavaScript access to the PegaRULES cookie, for example, session identifier.
prconfig/security/showSQLInListPage/default	true	false	Suppresses visibility of generated SQL on the clipboard page.
prconfig/security/UnexpectedInputPropertyAlert/default	true	true	Ignores unexpected properties in a request.
prconfig/security/CSP/PolicyEnabled/default	true	true	Enables Content Security Policy (CSP) support.
EnableAttributeBasedSecurity	true	true	Enables enforcement of access control policies and access control policy conditions (ABAC).
DiscoverableItemsIncludedForSummaryReport	false	false	Enables the discoverability feature associated with read-type access control policies.
KeyStoreCacheSize	150	50 (or more)	Restricts keystore cache size to control how much memory is used. For more information, see Changing the default keystore caching settings .
security/enableJavaInjectionMitigation	false	true	Enables java mitigation detection for all ruleset versions.
security/validateReloadParameters	false	true	Controls the HTTP response when the <i>pyBlockUnregisteredRequests</i> when rule evaluates to true. For more information, see Verify requests at the application layer .
prconfig/initialization/SubmitObfuscatedURL/default	optional	required	Recommended for all deployments. This setting also requires the <i>urlencryption</i> entry to be enabled. These two entries work as a pair, and causes Pega Platform to reject clear-text URLs.
prconfig/initialization/Urldebug/default	none	none	Recommended for all deployments. This setting prevents obfuscated URLs from being written to the log file. This prevents exposing potentially sensitive information.
prconfig/initialization/Urlencryption/default	false	true	Recommended for all deployments. This setting works as a pair with <i>SubmitObfuscatedURL</i> . The setting enables or disables the encryption of the URLs. Note: URL encryption only works in the same session.

- [Creating a dynamic system setting](#)

Add a dynamic system settings rule to change default system behavior.

- [Bcrypt hashing algorithm for Password property types](#)

To provide extra protection against brute-force attacks, Pega Platform uses salted bcrypt as the default hashing algorithm for Password property types. Bcrypt uses a modified key setup algorithm that requires a long time to process. Key strengthening makes a password more secure against brute-force attacks, meaning potential attackers must spend a substantial amount of time testing every possible key.

Creating a dynamic system setting

Add a dynamic system settings rule to change default system behavior.

For example, by default, encryption and authentication are not enabled for communication among search nodes. To enable encryption, create the `indexing/distributed/enable_inter_node_encryption` dynamic system setting and set it to `true`.

1. In the header of Dev Studio, click **Create SysAdmin Dynamic System Settings**.
2. Enter a short description.

3. Enter the **Owning Ruleset**. **For example:** Pega-RULES

If no ruleset is associated with the data instance, when the data instance is saved, the system assigns a ruleset from the current application, which assists with application packaging.

4. In the **Setting Purpose** field, enter the name of the setting. **For example:** `indexing/distributed/enable_inter_node_encryption`, where the key structure has the following components:
 - **File type** – Specifies the kind of setting for the instance.
 - `prconfig` – Used for `prconfig.xml` settings.
 - `properties` – Used for `prbootstrap.properties` settings.
 - `blank` (no value entered) – Used for unspecified *Data-Admin-System-Settings*.
 - **Setting name** – Specifies the name of the setting and can include long strings that are divided by slashes or periods.
 - **Node classification** – Specifies the node classification for the instance. For `prconfig.xml` settings, use default. For example, the `prconfig.xml` setting `asyncservices/enable` as a dynamic system setting is `prconfig/asyncservices/enable/default`.

- 5. Click **Create and open**.
- 6. In the **Value** field, enter the value for the setting. **For example:**
 - true for a Boolean value
 - 100 timeout value
- 7. Click **Save**.

- [Configuring dynamic system settings](#)
- [Creating a rule specialized by circumstance](#)

Bcrypt hashing algorithm for Password property types

To provide extra protection against brute-force attacks, Pega Platform uses *salted bcrypt* as the default hashing algorithm for Password property types. Bcrypt uses a modified key setup algorithm that requires a long time to process. Key strengthening makes a password more secure against brute-force attacks, meaning potential attackers must spend a substantial amount of time testing every possible key.

Bcrypt is an adaptive hashing algorithm based on the Blowfish symmetric block cipher cryptographic algorithm.

Changing the encryption algorithm

For on-premises deployments, ensure that you are using the salted bcrypt algorithm by removing all cryptographic-related configuration settings from the `prconfig.xml` file:

- `crypto/v5oneway`
- `crypto/v5onewahsha1`
- `crypto/v5portable`
- `crypto/onewayhashalgorithm`
- `crypto/updatehash`

Convert preexisting password hashes to use the new algorithm by editing or creating the following Dynamic System Settings in Dev Studio:

Dynamic System Setting	Owning ruleset	Setting purpose	Value
one way hash algorithm	<i>Pega-Engine</i>	<code>prconfig/crypto/onewayhashalgorithm</code>	bcrypt
update hash	<i>Pega-Engine</i>	<code>prconfig/crypto/updatehash</code>	true

Defining cross-origin resource sharing policies

Cross-origin resource sharing (CORS) policies define a method that enables a browser and server to interact and determine whether it is safe to allow a cross-origin request. For example, a client using a Pega Marketing application running in a browser, may see advertisements from third-parties, and if they click one of these advertisements, the CORS policy will record that the advertisement was viewed or clicked on.

The purpose of a CORS policy is to enable cross-domain requests, and is only applicable for cross-domain browser requests. In Pega Platform, CORS policies can only be associated with REST services. If the request is sent via server-side logic, such as in Java code or with a non JavaScript client such as postman, CORS is not applicable.

Using CORS policies results in reduced costs and implementation times while providing increased security as other systems or websites interact with your application.

To configure a CORS policy, you complete two main tasks:

- Define the CORS policy for a REST service by specifying the allowed origins, allowed headers, exposed headers, allowed methods, credential usage, and preflight expiration time.
 - Map the CORS policy to an endpoint (URL or path) for the REST service that you want to protect.
- [Creating a cross-origin resource sharing policy](#)

By creating a cross-origin resource sharing (CORS) policy and subsequently mapping it to an application endpoint (path or URL) for an API or service, you control whether and how other systems or websites (origins) are permitted to access that resource.

- [Mapping an endpoint to a cross-origin resource sharing policy](#)

The purpose of a CORS policy is to enable cross-domain requests. In Pega Platform, CORS policies can only be associated with REST services. When setting up cross-origin resource sharing (CORS) policies, you must map to a REST endpoint to specify which CORS policies apply to it. By doing so, you define which domains are allowed to access these resources within your Pega application.

Creating a cross-origin resource sharing policy

By creating a cross-origin resource sharing (CORS) policy and subsequently mapping it to an application endpoint (path or URL) for an API or service, you control whether and how other systems or websites (origins) are permitted to access that resource.

Before you begin: To create or modify CORS policies you must have the *pzCanManageSecurityPolicies* security privilege, which the *PegaRULES:SecurityAdministrator* role includes.

1. In the header of Dev Studio, click **Create Security Cross Origin Resource Sharing**.
2. On the Cross Origin Resource Sharing form, in the **Short description** field, enter a description of the CORS policy.
3. In the **Policy name** field, enter a name for the CORS policy. You might name the policy for an endpoint, or for the API or REST service that you want to protect.
4. Click **Create and open**.
5. On the Cross Origin Resource Sharing form, on the **Policy Definition** tab, select the **Allow credentials** check box to indicate that requests to the endpoint can include credentials.
6. In the **Allowed origins** field, enter a comma-separated list of domains (origins) that are allowed to make a request against the API or REST service.
At run time, the system evaluates all origins that you specify for this setting until it finds a match for the origin header of the request. Wildcard characters are also supported.
For example:
 - **www.abc.com** – Allows requests from the host that you specify.
 - ***.abc.com** – Allows requests from any site hosted in the abc.com domain.
 - ***** – Allows requests from any website. Use this value only if you want to give public access to the API or REST service.
 - **Null (blank)** – Does not allow any access requests.
7. In the **Maximum age** field, enter a number to specify how long, in seconds, the results of a preflight request can be cached.
This is the time period between two consecutive preflight requests, within which you do not want the web browser to send a new preflight request. A longer period reduces the frequency of browser preflight OPTIONS method call requests.
For example: A web browser sends a preflight request to the GET /cases service 300 seconds after accessing the GET /assignments service.
 - If you set the maximum age to 400, the browser does not send another preflight request for the GET /cases service.
 - If you set the maximum age to 200, the browser sends a preflight request for the GET /cases service.
8. In the **Allowed methods** section, select one or more check boxes to specify which request methods are allowed: **GET**, **POST**, **PUT**, **PATCH**, or **DELETE**.
9. In the **Allowed headers** section, enter a comma-separated list of the request header values that the origin domain is allowed to use for a CORS request.
The authorization and content-type headers are required for Pega Platform applications.

The default value is: authorization, content-type

10. In the **Exposed headers** section, enter a comma-separated list of response headers that clients of this API or service can access.
11. Click **Save**.

What to do next: After you create a CORS policy, you must map the CORS policy to an endpoint to determine where the policy is applied.

- [Mapping an endpoint to a cross-origin resource sharing policy](#)

Mapping an endpoint to a cross-origin resource sharing policy

The purpose of a CORS policy is to enable cross-domain requests. In Pega Platform, CORS policies can only be associated with REST services. When setting up cross-origin resource sharing (CORS) policies, you must map to a REST endpoint to specify which CORS policies apply to it. By doing so, you define which domains are allowed to access these resources within your Pega application.

Before you begin: To map endpoints to CORS policies you must have the *pzCanManageSecurityPolicies* security privilege, which the *PegaRULES:SecurityAdministrator* role includes.

1. In the header of Dev Studio, click **Configure Integration Services Endpoint-CORS Policy Mapping**.
2. On the Endpoint-CORS policy mapping form, click **Add endpoint**.
3. In the **Endpoint-CORS policy mapping** dialog box, in the **Endpoint** field, specify a valid endpoint (path or URL).
4. Click **Add policy**.
5. In the **CORS Policies** field, specify one or more CORS policies that map to this endpoint.

The sequence in which you list the CORS policies is significant. At run time, the system checks the CORS policy rules in the order that they are listed on the CORS-Endpoint Security form, until a match is found. Matching is based on the request method and the origin header value, which the system compares to the allowed request methods and allowed origins.

- A request from the origin that is specified in the CORS policy is recognized as secure for this REST endpoint. Requests that satisfy the policy are sent responses with the appropriate headers, as defined in the CORS policy.
 - A request from an origin that is not specified in the CORS policy is not considered secure, and the browser returns an error message stating that the cross-origin request was denied.
6. Click **Submit**.

- [Creating a cross-origin resource sharing policy](#)

Understanding cross-site scripting

Cross-site scripting is a client-side code injection attack, in which an attacker can run malicious scripts on a legitimate website or web application.

Cross-site scripting

Cross-site scripting

Cross-site scripting (XSS) is a web security vulnerability that allows an attacker to

- compromise the interactions that users have with a vulnerable application
- circumvent the same origin policy, which is designed to segregate different websites from each other.

Cross-site scripting vulnerabilities normally allow an attacker to masquerade as a victim user, to carry out any actions that the user is able to perform, and to access any of the user's data. If the victim user has privileged access within the application, then the attacker might be able to gain full control over all of the application's functionality and data.

Cross-Site Scripting attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page.

Leading practices to avoid cross-site scripting vulnerabilities

Pega Platform™ applications depend on JavaScript functions that are run in a web browser. As a result, application users potentially can be exposed to JavaScript-based cross-site scripting security issues. Cross-site scripting is a client-side code injection attack, in which an attacker can run malicious scripts on a legitimate website or web application. If a malicious or infected site (or user) injects unauthorized JavaScript code into a browser form, execution of that JavaScript, on the same or another browser session, can cause serious security problems, including:

- Hijacking the user session
- Defacing websites
- Introducing worms, bots, or viruses
- Sending spam email

Many JavaScript functions are built into standard text file rules, HTML fragment rules, and HTML Property rules. Additional JavaScript files can be created as part of an application.

Pega Platform includes features and facilities that prevent many security vulnerabilities, and Pegasystems routinely tests for and analyzes potential security issues. However, the many software layers in use (network, email, application server, database, firewalls, and so on) make security an ongoing challenge. The practices described in this article address only one dimension of security, but one that has been heavily exploited when ignored.

Suggested approach

Pega recommends using the following best practices to protect against the most common ways that cross-site scripting (XSS) attacks gain access to a client browser:

- Maintain guardrail compliance (see below).
- [Filtering all inputs](#).
- [Filtering HTML and XML outputs](#).
- [PublicAPI methods for XSS filtering](#).

These practices are always important whether or not your application:

- Is used by unauthenticated users
- Is exposed to a large community or embedded as a web mashup in other applications
- Runs in a less-secure network
- Runs in a less-secure physical environment
- Includes stream rules that are not autogenerated

The best practices and the built-in features of Pega Platform are intended to help you build more secure applications by avoiding specific, well-known vulnerabilities.

Maintain guardrail compliance

In a guardrail-compliant application, application rulesets contain only autogenerated flow actions, autogenerated sections, and autogenerated harness rules. The application might use standard rules that are not autogenerated.

Your application eliminates many potential security vulnerabilities, and is easier to maintain and document, by avoiding custom HTML, custom JavaScript, and custom Java.

For more information, see [Security Checklist additional tasks](#).

- [Filtering all inputs](#)
- [Filtering HTML and XML outputs](#)
- [PublicAPI methods for XSS filtering](#)

Filtering all inputs

Prevent invalid data from entering a work object or a work object attachment. Filter and validate input data as thoroughly as possible, including input that is submitted from browser forms, input from service requests (such as email), and input from connector responses. Do not rely on client-side editing, which is in the browser, because a hacker can easily bypass it. Perform validation in your server-side application logic.

Use the following features to validate individual values:

Individual values	
Property types	- For Single Value, Value List, and Value Group properties, select the property type carefully. - Use Integer, Double, Decimal, DateTime, Date, TimeOfDay, and TrueFalse when appropriate. - Rather than Text property type, use the Password or Identifier property type. - Special characters that are common in JavaScript code (such as quotation marks), are never displayed in a numeric, date or time value, but might legitimately be displayed in Text, Password, or Identifier value.
Allow lists	Several features let you constrain a property value to one of a fixed list or pattern of values, including the property table edits (local list, field values, class key values) on the General tab in Pega Platform 5.5 and later versions, or the Table Edit tab in earlier releases.
Restrictions	Complete the Max Length field for Text, Password or Identifier fields. For example, it is difficult to fit a malicious JavaScript program into a small number of characters.
Not declarative	If applicable, select the Cannot be a Declarative Target check box. This option is helpful, because a declarative expression could assemble JavaScript source code.
Special properties	Select the Cannot be included as an input field check box if the property is always computed from other values.
Validation	Identify an edit input rule and an edit validate rule when possible. Do not accept angle brackets, quotation marks, ampersands, or other special characters in fields unless necessary for a sound business reason. For example, the standard validation rule LetterorDigit limits values to only letters and digits. On the PRPC 5.5 property form, the Max Length and Validate fields are displayed on the Advanced tab.
	Use map value rules, validation rules, and constraints rules to validate inputs.
	To test arriving email attachments or other file attachments for software viruses and malicious JavaScripts, override the extension point activity Data-WorkAttach-File. CallVirusCheck with an activity that calls your third-party virus software. Your activity can call a Java class, or in a Microsoft Windows server, a Dynamic Linked Library routine
	Ensure that files that are uploaded from application user workstations and text files processed by a file listener are checked against viruses.
	Input filtering is worthwhile even if an application has 100% coverage of output filtering for the HTML that the application sends to a browser. Applications often send output to other systems, rather than to a browser, which could contain a malicious JavaScript function, such as an HTML-rendered email message. Infecting another system, which might belong to another department, a customer, or to a supplier is a security failure that is no less serious than infecting an application user.
	Avoid using the URL JavaScript Page (JSP) tag in manually created stream rules. If you must use a URL tag, call the SafeURL JavaScript functions to obfuscate the URL query string.

Filtering HTML and XML outputs

Not all invalid data can be detected and 100% excluded during input processing. Some input fields legitimately and necessarily contain quotation mark characters, angle bracket characters, and other sensitive characters. For example, a user-entered text area on a form might contain a snippet of JavaScript or Java code that was entered to describe a software issue.

If invalid data, such as malicious JavaScript source code, somehow reaches a Pega Platform clipboard, applications that are built following the guardrails reduce the risk that the JavaScript code is ever sent to a browser in a form that can be executed.

Output from auto generated harness, section, and flow action rules is automatically output-filtered. Application developers do not need a special approach for such rules. (This is applicable to Pega Platform 5.3 SP1 or later versions.)

In any non-auto generated (manually created) stream rules (correspondence, paragraph rules, XML, HTML, flow actions, harness, or sections), use only Pega Platform JSP tags to ensure that output filtering occurs. Note these specific cases:

- Even when you manually create only a portion of a full HTML or XML document, make the HTML code that your rules produce well-formed in terms of matching begin-end tags, matched quotation marks, use of only legitimate HTML tags, and correct nesting of tags. Various browser versions render malformed HTML in unpredictable ways, and some browsers become vulnerable to bugs and quirks after rendering malformed HTML code.
- For the <pega:reference> or <p:r> tag, omit the mode attribute to provide complete XSS filtering. This is the default when your tag omits the mode attribute. Avoid mode=literal, which disables XSS filtering. Use mode=javascript when truly necessary, but with extreme care.
- For the <pega:lookup> tag in PRPC 5.5 applications, avoid mode=literal. In versions prior to PRPC 5.5, the mode attribute is not available, and all uses of the <pega:lookup> tag provide XSS filtering.
- Filter any potential-risk text value for XSS vulnerability using a Java scriptlet that calls one of two PublicAPI methods.

PublicAPI methods for XSS filtering

Output filtering involves replacing sensitive characters in the source HTML code such as angle brackets (< and >), single and double quotation marks, ampersands, and others by the equivalent HTML character entities, such as < for < or ' for ' an apostrophe. If the source text is legitimate, output filtering does not prevent correct HTML rendering by the browser.

Use these two Java methods to provide output filtering:

- StringUtils.crossScriptingFilter(aValue) – Filters the contents of aValue and returns a filtered (encoded) version. Available in PRPC 5.3 SP1 and later versions.
- StringUtils.reversibleCrossScriptingFilter(aValue) – Filters the contents of aValue and reverse-filters the value in the server if the same value is resubmitted in subsequent requests. This method ensures that the friendly user's intent is preserved.

Use reversibleCrossScripting() when the internal clipboard value is not to be modified, but the value sent to the browser is to be filtered. For example, if an application user needs to search a customer master file for a customer named Molly O'Brian, the SQL statement (on the server) needs to process using the apostrophe (single quotation mark) character ('), but the browser display uses the HTML entity ' or an equivalent. In this case, a separate escape mechanism for SQL also applies, converting the name O'Brian to O\Brian.

The following examples are portions of manually created stream rules that are modified to filter a value using these two PublicAPI functions in an inline Java scriptlet.

Example 1

Incorrect

```
<%tools.appendString(tools.getActiveValue()); %>
```

Correct

```
<%tools.appendString(StringUtils.crossScriptingFilter(tools.getActiveValue()));%>

or

<%tools.appendString(StringUtils.reversibleCrossScriptingFilter(tools.getActiveValue()));%>
```

Example 2

Incorrect

```
<%tools.appendString(tools.getSaveValue("savename")); %>
```

Correct

```
<pega:reference name=$save(savename) />

or

<%tools.appendString(StringUtils.crossScriptingFilter(tools.getSaveValue("savename"))); %>

or

<%tools.appendString(StringUtils.reversibleCrossScriptingFilter(tools.getSaveValue("savename"))); %>
```

Example 3

Incorrect

```
<%tools.appendString(tools.getParamValue("paramname")); %>
```

Correct

```
<pega:reference name=param.paramname/>

or

<%tools.appendString(StringUtils.crossScriptingFilter(tools.getParamValue("paramname"))); %>

or

<% tools.appendString(StringUtils.reversibleCrossScriptingFilter(tools.getParamValue("paramname"))); %>
```

Configuring the deserialization filter

In Pega Platform, a global filter checks a list of blocked classes that are not allowed to be deserialized. You can add classes to the global deserialization filter to increase the security of your application by preventing unauthorized access.

- 1. In the header of Dev Studio, click **Configure System Settings Deserialization Blacklist** .
- 2. To add a class to the list of blocked classes, click **Add gadget class to blacklist**, and enter a class name. Pega Platform does not deserialize classes that match this class name or pattern. Repeat this step to add multiple class names or patterns. You can use wildcards to specify a pattern for the class names to block.

Wildcard	Function
<package name>.*	Match any class in the package and all subpackages.
<package name>.*	Match any class in the package.
<partial name>.*	Match anything that starts with <partial name>.

- 3. Click **Save**.

- [Java deserialization](#)

Java deserialization

Deserialization is the process of rebuilding a data stream into a Java object. The Open Web Application Security Project (OWASP) has identified insecure deserialization as one of the top ten security vulnerabilities for web applications. Pega Platform protects against this vulnerability by using features in the Java JDK.

In Pega Platform, a global filter checks a list of blocked classes that are not allowed to be deserialized. This global filter also adds known internal classes to the allow list. You can add classes to the global deserialization filter. If the filter flags a data stream as invalid, a security event is written to the security event log and the stream is not deserialized.

By default, the filter blocks the following classes:

- com.sun.org.apache.xalan.internal.xsltc.trax.TemplatesImpl
- org.apache.commons.collections.functors.InvokerTransformer
- org.apache.commons.collections.functors.InstantiateTransformer
- org.apache.commons.collections4.functors.InvokerTransformer
- org.apache.commons.collections4.functors.InstantiateTransformer
- org.apache.xalan.xsltc.trax.TemplatesImpl
- org.codehaus.groovy.runtime.ConvertedClosure
- org.codehaus.groovy.runtime.MethodClosure
- org.springframework.beans.factory.ObjectFactory

- [Configuring the deserialization filter](#)

Understanding cross-site request forgery

Cross-site request forgery (CSRF) is an attack that forces a user to execute unwanted actions on a web application in which the user is currently authenticated. CSRF specifically targets state-changing requests, not data theft, because the attacker cannot see the response to the forged request. With a little help of social engineering (sending a link in email or chat), an attacker can trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force users to perform state-changing requests including transferring funds or changing their email address. If the victim is an administrative account, CSRF can compromise the entire web application.

Mitigating CSRF attacks

Pega Platform uses session tokens to mitigate the risk of CSRF attacks. Each user session is assigned one or more unique tokens. These tokens are made available to the browser for inclusion in the URL of all requests. Each request is examined for a valid token and is rejected, if either no token or an invalid one is provided. Exceptions can be configured based on the values of the HTTP Referrer header contained in the request.

Cross-site request forgery settings

To obtain the values to include in the cross-site request forgery (CSRF) mitigation lists, enable CSRF mitigation, and then perform a detailed test of the application. You can capture the requests that fail because of the CSRF protection. From the requests, identify the activities, streams, and referrer headers, and add them to these settings.

Use the Cross-Site Request Forgery landing page to modify these settings, the information settings is defined in the table below.

Purpose	Default	Secure	Security Implications
Security/CSRF/mitigation	false	true	Enables the CSRF mitigation feature.
Security/CSRF/secured/Activities	blank	Specify a comma-delimited list of activities to secure.	A request for an activity in this list must include a valid CSRF token.
Security/CSRF/securedStreams	blank	Specify a comma-delimited list of streams to secure.	A request for a stream in this list must include a valid CSRF token.
Security/CSRF/validReferrers	blank	Specify a comma-delimited list of valid referrers for incoming requests.	If CSRF token and activity/stream validations fail, the referrer header is validated against this list. The request fails if the referrer header is not on the list.
Security/CSRF/secureall	false	true	Indicates that all activities and streams must include a valid CSRF token, with no exceptions.

- [Enabling and configuring Cross-Site Request Forgery settings](#)

Configure cross-site request forgery settings (CSRF) to prevent users from unintentionally making changes because of a CSRF attack. You can set validation for activities and streams, add host names to an allow list, and specify host names that you want checked for a CSRF token.

Enabling and configuring Cross-Site Request Forgery settings

Configure cross-site request forgery settings (CSRF) to prevent users from unintentionally making changes because of a CSRF attack. You can set validation for activities and streams, add host names to an allow list, and specify host names that you want checked for a CSRF token.

1. In the header of Dev Studio, click **Configure System Settings Cross-Site Request Forgery**.
2. **Optional:** To prevent the browser from submitting the PegaRULES cookie in a request from a non-originating site, complete the following steps:
 - a. Select the **Enable samesite cookie attribute** checkbox.
 - b. In the **Samesite Options** list, choose one of the following options:
 - **None** – If you select this option, Pega Platform offers no protection. The browser attaches the cookies in all cross-site browsing contexts.
 - **Lax** – If you select this option, Pega Platform provides a reasonable balance between security and usability for websites that want to maintain logged-in sessions after users arrive from an external link. The browser does not send cookies in requests from non-originating sites.
 - **Strict** – If you select this option, Pega Platform prevents the browser from sending cookies to the target site in all cross-site browsing contexts, including when following a regular link.For more information about samesite cookies, refer to the documentation from [owasp.org](#).
3. To enable CSRF settings, select **Enable CSRF token check**.
Selecting this check box will enable the CSRF token validation.
4. If you have enabled CSRF token check, select one of the following **Secure** fields:
 - All activities & streams
CSRF validation checks all activities and streams for CSRF tokens in your system. If you select this option, you can specify certain streams and activities to be excluded from CSRF token validation by entering them in the **Allowed Activities** field and the **Allowed Streams** field. Separate multiple activities and streams with commas.
 - Specific activities & streams
CSRF validation checks the activities and streams that you specify in the **Secured Activities** and **Secured Streams** fields for CSRF tokens. Separate multiple activities and streams with commas.
5. **Optional:** To add names to a safe "allow list" of host names to ignore during CSRF token validation, perform the following actions.
 - a. In the **Referrer Settings** section, select **Enable referrer check**.
 - b. Select the **Allow domains only if matches exactly with Referrer** check box to allow requests which exactly match the URL given in the **Allowed referrers** field.
 - If **Enable referrer check** is checked and [http://pega.com](#) is provided in **Allowed referrers**, then [http://pega.com.xyz](#) is valid, but [http://xyz.pegacom](#) is not valid.
 - If **Allow domains only if matches exactly with Referrer** is checked in addition to the conditions above, only the exact match [http://pega.com](#) is valid, but not [http://pega.com.xyz](#) or [http://xyz.pegacom](#).
 - c. In the **Allowed referrers** field, enter host names that you want to be checked for a CSRF token. Separate multiple host names with commas.
6. Click **Submit**.
7. If you changed the value of **Enable CSRF token check**, you must restart your system for the new value to take effect.

- [Enabling cross-site request forgery support](#)

Enabling cross-site request forgery support

You can enable cross-site request forgery (CSRF) settings for the system from the Cross-Site Request Forgery landing page. These settings help prevent attacks that might cause a user to perform unintended actions in the system, for example, unintentionally changing a password.

You can do several tasks on the Cross-Site Request Forgery landing page. For example, you can enable a CSRF token check and manage Referrer settings. CSRF tokens are unique tokens that are generated by a random number generator and assigned to Pega URLs. For information about performing these tasks, see [Enabling and configuring Cross-Site Request Forgery settings](#).

- [Enabling and configuring Cross-Site Request Forgery settings](#)

Secure your application user interface

As a security administrator, you permit or restrict groups of users to access various actions in an application, such as having access to a case type, flow action, or button.

When you create an application, some access groups are created automatically. You can create additional access groups and assign users to access groups based on the type of work they do. For example, in most applications, managers have permission to do tasks that ordinary employees cannot.

As a security administrator, you configure an access group for managers and an access group for regular employees. When a new employee is hired, human resources staff assigns the employee to the proper access group.

Use case

The examples below assume that you have a human resources application named HRApp in which various access groups, such as managers and human resources staff, can do different actions. The examples also assume that you have access to the Dev Studio portal for HRApp and have the *PegaRULES:SecurityAdministrator* role. Some of the examples assume that you have created specific case types and access groups, which are described in each example.

Securing an application user interface involves these sorts of tasks:

- [Controlling access to an entire case type](#)

Authorized users process salary reviews by using the SalaryReview case type in the HRApp application. You need to ensure that only human resources staff and managers can access the SalaryReview case type.

- [Restrict who can use a flow action](#)

It is a best practice to limit flow actions to the users who really require them.

- [Controlling access to flow actions](#)

Authorized users approve salary changes by using the SalaryReview case type in the HRApp application. You need to ensure that only managers have access to the flow action for salary approval.

- [Controlling access to sections, buttons, and other UI controls](#)

Authorized users approve salary changes by using the SalaryReview case type in the HRApp application. You need to ensure that only managers have access to the button that is used to approve salary changes.

- [Controlling access to reports](#)

Ensure that only senior executives can run reports in the HRApp application, because these reports include confidential information related to hiring and compensation.

- [Validating user input and preventing invalid values](#)

Ensure that only numbers can be entered for employee Social Security numbers.

- [Controlling access to individual cases](#)

Ensure that only the employee, the employee's manager, and the human resources staff can view an employee's timesheet.

- [Encrypting the values of sensitive properties](#)

In the HRApp application, ensure that the Social Security number and salary properties are encrypted in all Pega Platform data stores (the database and Elasticsearch index files, in memory, and on the clipboard). Ensure that they are decrypted only when they are displayed in the user interface.

- [Masking the values of sensitive properties](#)

You need to ensure that sensitive data such as social security number (SSN) are visible only to human resources staff and to the employee.

- [Securing your application for mashup communication](#)

Define the external URLs that are allowed to access Pega Platform so that the host page can communicate with the mashup gadget page, if you use the mashup feature to embed Pega Platform content in an external application.

- [Securing Cosmos React-UI applications](#)

If your application uses a Cosmos React-UI, then it authenticates operators using one of the newer (PRAuth) types of Pega Platform Authentication schemes. Requests are typically submitted using a URL that includes an application alias, for example: https://<host>:<port>/prweb/app/Alias. For an unauthenticated user, this type of request presents a page showing a list of authentication services available for login to the application. If the user chooses Basic authentication, then the password, logout, and CAPTCHA policies are

Controlling access to an entire case type

Authorized users process salary reviews by using the SalaryReview case type in the HRApp application. You need to ensure that only human resources staff and managers can access the SalaryReview case type.

Before you begin: Assume that these steps have already been done:

1. Access groups are defined for human resources staff (**HRApp:HRStaff**) and managers (**HRApp:Managers**).
2. A role is defined for salary review processing (**HRApp:SalaryReview**).
3. The **HRApp:SalaryReview** role is assigned to the **HRApp:HRStaff** and **HRApp:Managers** access groups.

Do these steps to ensure that only operators with the **HRApp:SalaryReview** role have permission to open the **SalaryReview** case type.

1. In Dev Studio, go to the **Work & Process** tab of the **Access Manager** landing page by clicking **Configure > Org & Security > Access Manager > Work & Process**.
2. In the **Access group** field, enter **HRApp:Managers**. The tab lists permissions that the **HRApp:Managers** access group has for various roles, case types, and actions.
3. Expand the **SalaryReview** case type and the **Open** action, and ensure that a green check mark is shown for the **HRApp:SalaryReview** role. A green check mark indicates that permission is granted. You can change the value by clicking the icon.
4. Repeat the steps above for the **HRApp:HRStaff** access group.
5. Repeat the steps above for the other access groups, but ensure that a red X is shown for the **Open** action of the **SalaryReview** case type and that the **HRApp:SalaryReview** role is not listed, showing that no access is permitted. You can change the value by clicking the icon.

Restrict who can use a flow action

It is a best practice to limit flow actions to the users who really require them.

To restrict a flow action:

- Update the **Security** tab of the **Flow Action** form to identify one or more privileges that allow run-time use of the action.
- Update one or more *Rule-Access-Role-Obj* rules, for specific access roles, to convey privileges to access roles.

When processing an assignment at, users can only view and select from the flow actions that match their privileges.

Pega Platform provides a privilege for each standard flow action. You can follow this model and create a privilege for each new flow action that you create.

To make design easier, the *Work-.AllFlowActions* privilege allows access to all standard flow actions. This privilege is supplied for expedient debugging and power users and should not be used for normal work.

Controlling access to flow actions

Authorized users approve salary changes by using the SalaryReview case type in the HRApp application. You need to ensure that only managers have access to the flow action for salary approval.

Before you begin: Assume that these steps have already been done:

1. Access groups are defined for human resources staff (**HRApp:HRStaff**) and managers (**HRApp:Managers**).
2. A role is defined for salary approval processing (**HRApp:SalaryApproval**).
3. The **HRApp:SalaryApproval** role is assigned to the **HRApp:Managers** access group.
4. A flow action named **SalaryChangeApproval** is used to approve salary changes in the **HRApp** application.

Do these steps to ensure that only operators with the **HRApp:SalaryApproval** role have permission to select the **SalaryChangeApproval** flow action.

1. In Dev Studio, go to the **Work & Process** tab of the **Access Manager** landing page by clicking **Configure > Org & Security > Access Manager > Work & Process**.
2. In the **Access group** field, enter **HRApp:Managers**. The tab lists permissions that the **HRApp:Managers** access group has for various roles, case types, and actions.
3. For the **SalaryReview** case type and the **SalaryChangeApproval** flow action, ensure that a green check mark is shown for the **HRApp:SalaryApproval** role. A green check mark indicates that permission is granted. You can change the value by clicking the icon.
4. Repeat step 2 for **HRApp:HRStaff** and other access groups that might have access to the **SalaryReview** case type but cannot approve salary changes. Ensure that a red

X is shown for the **SalaryChangeApproval** flow action of the **SalaryReview** case type and that the **HRApp:SalaryApproval** role is not listed, showing that no access is permitted. You can change the value by clicking the icon
For more information, see [Editing authorizations for case type flows and flow actions in a single access group](#).

Controlling access to sections, buttons, and other UI controls

Authorized users approve salary changes by using the **SalaryReview** case type in the **HRApp** application. You need to ensure that only managers have access to the button that is used to approve salary changes.

Before you begin:

Assume that these steps have already been done:

1. Access groups are defined for human resources staff (**HRApp:HRStaff**) and managers (**HRApp:Managers**).
2. A role is defined for salary approval processing (**HRApp:SalaryApproval**).
3. The **HRApp:SalaryApproval** role is assigned to the **HRApp:Managers** access group.
4. The user interface includes a button named **pyApproveSalary**, which a user clicks to approve a salary change.

Do these steps to ensure that only operators with the **HRApp:SalaryApproval** role have permission to click the **pyApproveSalary** button.

1. Create the **CanApproveSalary** privilege by doing the following steps:
 - a. In Dev Studio, click **Create Security Privilege**.
 - b. In the **Label** field, enter **CanApproveSalary**.
 - c. In the **Apply to** field, enter **SalaryReview**, and then click **Create and open**.
 - d. Click **Save**.
2. Associate the **CanApproveSalary** privilege with the **pyApproveSalary** button by doing the following steps:
 - a. In Dev Studio, open the section that contains the button **pyApproveSalary**, and select the **pyApproveSalary** button.
 - b. Click the **View Properties** icon.
 - c. In the **Privilege** field, enter **CanApproveSalary**.
 - d. Click **Submit**.
 - e. Click **Save**.
3. In Dev Studio, go to the **Privileges** tab of the Access Manager landing page by clicking **Configure Org & Security Access Manager Privileges**.
4. In the **Role** field, enter **HRApp:SalaryApproval**, and in the **Case type** field, enter **SalaryReview**. The tab lists privileges that apply to this role and case type.
5. Click the **Add a privilege** icon and in the **Privilege** field, enter **CanApproveSalary**.
6. In the dialog box for this privilege, click **Full Access**, and then click **OK**.
7. For other roles for the **SalaryReview** case type, repeat the above steps, but ensure that the **CanApproveSalary** privilege is not listed or that the privilege is shown with a red X icon. You can change the value by clicking the icon.

Controlling access to reports

Ensure that only senior executives can run reports in the **HRApp** application, because these reports include confidential information related to hiring and compensation.

Before you begin:

Assume that these steps have already been done:

1. An access group is defined for senior executives (**HRApp:SeniorExec**).
2. A role is defined for running reports in this application (**HRApp:RunAllReports**).
3. The **HRApp:RunAllReports** role has been assigned to the **HRApp:SeniorExec** access group.

Do these steps to ensure that only operators with the **HRApp:SeniorExec** access group have permission to run all reports in the **HRApp** application.

1. In Dev Studio, go to the **Tools** tab of the Access Manager landing page by clicking **Configure Org & Security Access Manager Tools**.
2. In the Access group field, enter **HRApp:SeniorExec**. The table lists common tools that can be conditionally available in all applications and the roles that have access in the **HRApp:SeniorExec** access group.
3. In the table, expand **User & manager Reports** and find the value for **Provide criteria on reports** that is listed for the **HRApp:RunAllReports** role. Ensure that a green check mark is shown, which means that access is granted. You can change the value by clicking the icon.
4. For access groups that should not be able to run reports, repeat the above steps, but ensure that the **HRApp:RunAllReports** role is not listed and that any other roles show a red X icon, which means that access is not granted. You can change the value by clicking the icon.

For more information, see [Editing tools authorization for a single access group](#).

Validating user input and preventing invalid values

Ensure that only numbers can be entered for employee Social Security numbers.

Before you begin: Assume that in the **Employee** class, a property named **SSN** defines the employee's Social Security number. Do these steps to ensure that only numbers can be entered for the **SSN** property.

1. In the navigation pane of Dev Studio, click **App**, and in the **class** field, enter **Employee**.
2. In the navigation area for **Employee**, expand **Data Model Property** and click the **SSN** property.
3. On the **Advanced** tab, enter the following values:
 - a. To ensure that 9 characters must be entered for **SSN** values, enter 9 in the **Max length** and **Expected length** fields.
 - b. To ensure that only numbers can be entered for **SSN** values, in the **Use validate** field, press the Down arrow key and click **IsPositiveInteger**. The **IsPositiveInteger** edit validate rule, which is part of Pega Platform, ensures that only positive integers can be entered. You can create additional edit validate rules for custom edit checks.

For more information, see [Property form - Completing the Advanced tab](#).

Controlling access to individual cases

Ensure that only the employee, the employee's manager, and the human resources staff can view an employee's timesheet.

Before you begin:

Assume that these steps have already been done:

1. A case type named **Timesheet** is created.
2. In the **Timesheet** class, the **EmployeeID** property identifies the employee and the **EmployeeManagerID** property identifies the employee's manager.

Do these steps to ensure that only the employee, the employee's manager, and human resources staff can view the employee's timesheet.

1. In Dev Studio, create an access control policy for an **Apply to class** equal to **Timesheet** and **Action** equal to **Read**.

For more information, see [Creating an access control policy condition](#).

2. Next to the **Permit access if** field, click the **Open** icon to create a new Access control policy condition instance.
3. Create an access control policy condition named **CanViewTimesheet** to define who can view the timesheet. Enter the following values:

For more information, see [Creating an access control policy condition](#).

- a. Policy condition A = **Requestor.AccessGroup** = **HRApp:HRStaff** (the user works in human resources)
- b. Policy condition B = **Requestor.OperatorID** = **EmployeeID** (the user is looking at the user's own timesheet)
- c. Policy condition C = **Requestor.OperatorID** = **EmployeeManagerID** (the user is the manager of the employee on the timesheet)

- d. Conditional logic = A OR B OR C
4. On the Access control policy instance, in the **Permit access if** field, enter `CanViewTimesheet`. Only users who satisfy the condition in step 3d can view the timesheet. Access control policies apply not only to the application user interface, but to most Pega Platform features. For example, *PropertyRead* policies also apply to reports, searches, and even to custom SQL that you write.

Encrypting the values of sensitive properties

In the HRApp application, ensure that the Social Security number and salary properties are encrypted in all Pega Platform data stores (the database and Elasticsearch index files, in memory, and on the clipboard). Ensure that they are decrypted only when they are displayed in the user interface.

Before you begin: Complete the following steps:

1. An encryption key is defined in a key management system (KMS) outside of Pega Platform.
2. A keystore instance is defined in Pega Platform that refers to the encryption key.
3. The **Keystore** field in the Application data encryption section of the Data Encryption landing page refers to the keystore in step 2, and the Activate button has been clicked to activate the keystore.
4. In the **Employee** class, a property named **SSN** defines the employee's Social Security number and a property named **Salary** defines the employee's salary.

Do these steps to ensure that the **SSN** and **Salary** properties are encrypted in all data stores, in memory, and on the clipboard. You can combine property encryption with property masking.

1. In Dev Studio, create an access control policy for an Apply to class equal to **Employee** and Action equal to `PropertyEncrypt`.
2. Click **Add property** and in the **Property** field, enter `SSN`.
3. Click **Add property** and in the **Property** field, enter `Salary`.
For more information, see:

[Configuring the platform cipher.](#)

[Creating an access control policy.](#)

Masking the values of sensitive properties

You need to ensure that sensitive data such as social security number (SSN) are visible only to human resources staff and to the employee.

Before you begin: Assume that in the Employee class, a property named SSN defines the employee's social security number.

1. In Dev Studio, create an access control policy for an Apply to class equal to *Employee* and Action equal to `PropertyRead`. For more information, see [Creating an access control policy](#).
2. Next to the **Permit access if** field, click the **Open** icon to create a new Access control policy condition instance.
3. Create an access control policy condition named *CanViewSSN* to define who can view the *SSN* value. Enter the following values. For more information, see [Creating an access control policy condition](#).
 - a. Policy condition A = `Requestor.AccessGroup = HRApp:HRStaff` (the user works in human resources)
 - b. Policy condition B = `Requestor.OperatorID = EmployeeID` (the user is looking at the user's own employee record)
 - c. Conditional logic = A OR B
4. On the Access control policy instance, in the Permit access if field, enter `CanViewSSN`.
5. Click **Add property** and in the **Property** field, enter `SSN`.
6. In the **Restriction Method** list, select whether to fully mask all values or to mask only the values in a certain position. For example, you might want to permit viewing the last 4 digits of the SSN. The value is masked for everyone except the users who satisfy the condition in step 3c.
You can combine property encryption with property masking.

Securing your application for mashup communication

Define the external URLs that are allowed to access Pega Platform so that the host page can communicate with the mashup gadget page, if you use the mashup feature to embed Pega Platform content in an external application.

1. In the header of Dev Studio, click **Application name Definition**.
2. On the Application form, click the **Security** tab.
3. In the **Mashup security** section, in the list of trusted origins (URLs) that can access your application, click **Add a row** and enter a URL that can be trusted.
When entering URLs, you need to enter a host-page domain, including `https` or `http`.

For example:

`https://www.google.com`

or

`https://example:4444`

- [Pega web mashups for embedding Pega Platform UI in external web pages](#)
- [Creating cases from a web mashup](#)
- [Harness and Section forms - Configuring an External web component section](#)

Securing Cosmos React-UI applications

If your application uses a Cosmos React-UI, then it authenticates operators using one of the newer (PRAuth) types of Pega Platform Authentication schemes. Requests are typically submitted using a URL that includes an application alias, for example: `https://<host>:<port>/prweb/app/Alias`. For an unauthenticated user, this type of request presents a page showing a list of authentication services available for login to the application. If the user chooses **Basic authentication**, then the password, logout, and CAPTCHA policies are applied by default.

Cosmos React-UI security

When using Cosmos React-UI, use the following conventions, or you will encounter errors and issues with authentication:

- HTTPS is required if the application is marked to use Cosmos React-UI.
- Authorized access tokens (AAT) should be marked as *HttpOnly* and secured using a dynamic system setting. For more information, see [Creating a dynamic system setting](#).
- You can customize token lifetime configurations on the OAuth 2.0 Client Registration rule form.
 - The auto-generated OAuth 2.0 client for an application is identified with the name `PegaAPP_<ApplicationName>`. Where `<ApplicationName>` is the name that is used to automatically generate OAuth 2.0 Client Instance when an application is saved.
 - The default expiry time is 15 minutes for access token and 24 hours for refresh token.
 - For more information, see:
 - [Understanding authorized access tokens.](#)
 - [Creating and configuring an OAuth 2.0 client registration.](#)
 - [Enhanced refresh token strategy.](#)
- URL patterns & authentication
 - Cosmos React-UI supports only app-specific URLs and *PRAuth*-based authentication-schemes. For example, `https://<host>:<port>/prweb/app/Alias` is valid.
 - Cosmos React-UI does not support non app-specific URLs and for custom authentication and any authentication-schemes other than *PRAuth*. For example, URLs such as the following will cause an error: `https://hostname:port/prweb/PRWebLDAP1/app/Alias`.
 - If a Cosmos React-UI application is exported to higher environments, instances of OAuth 2.0 clients that are specific to the application must be included in the package.
- API Security
 - All Cosmos React-UI service-packages must be configured to use OAuth 2.0 authentication. For more information, see *OAuth 2.0 client registrations*.
- Authorization
 - An operator must have `PegaRULES:PegaAPI` role to perform Digital Experience (DX) and data API calls.
 - If the application uses DX API, an OAuth 2.0 Client is generated automatically when the application is saved.

- Session management
 - It is a leading practice to set the **Access group timeout** as a longer time period than the **Refresh token timeout**. For example, the **Refresh token timeout** is set to 15 minutes and the **Access group timeout** is set to 1 hour.
 - Even if the **Access group timeout** is configured to be less than access token expiry, then re-authentication is required if the access token expires.
- Custom authentication is supposed in Cosmos React-UI applications. For more information, see [Configuring custom or Kerberos login authentication](#).
- [OAuth 2.0 Management Services](#)
- [Understanding dynamic client registration](#)
- [Setting up an OAuth 2.0 client registration](#)
- [Enhanced refresh token strategy](#)

Verify requests at the application layer

Pega Platform protects access to information in your application by using role-based settings and access control policies. Pega Platform provides additional request verification when you use autogenerated controls.

As a security best practice, and to conform to platform guardrails, use autogenerated controls. You can manually configure custom (non-autogenerated) controls for increased security.

[Verifying requests when using custom controls](#) describes how to manually configure non-autogenerated controls for increased security.

[Using Access Control Checks](#) describes how to identify broken custom code that must be fixed. If you do not fix broken access controls, when you enable security protections and the application goes into hardening, the broken features may stop working or may not work properly.

You can block unauthorized requests by using three *when* rules, which are defined on *@baseclass*. The following list describes the *when* rules that are used to enable the application protection feature.

Note: When *pzSecureFeatures* is false, no access checking is performed and the other two *when* rules are ignored.

pzSecureFeatures

When rule name: *pzSecureFeatures*

- **Description:** Turns application level checking on or off
 - **Default value:** True when
 - the portal is not Dev Studio, App Studio, Admin Studio, or Prediction Studio.
 - AND
 - the client is not mobile or hybrid.
- In other words, the access control check is not done when the client is using Dev Studio, App Studio, Admin Studio, or Prediction Studio and the client is mobile or hybrid. The check is done everywhere else.
- **Behavior when true:** Application level checking is on. When an access violation is found, a security alert is logged that says "Unregistered request encountered"; default behavior.
 - **Behavior when false:** Application level checking is off.

pyShowSecureFeatureWarnings

When rule name: *pyShowSecureFeatureWarnings*

- **Description:** Controls display of a warning to the end user
- **Default value:** False
- **Behavior when true:** When an access violation is found and *pyBlockUnregisteredRequests* is false, a Pega warning is displayed to the user saying "URL tampering vulnerability detected."
- **Behavior when false:** The access control warning is not displayed to the user; default behavior.

pyBlockUnregisteredRequests

When rule name: *pyBlockUnregisteredRequests*

- **Description:** Controls the HTTP response
- **Default value:** *pxProcess.pzProductionLevel* ≥ 4 **Note:** The default value is False when the production level is < 4. When production level is changed to 4, the value changes to True.
- **Behavior when *pxProcess.pzProductionLevel* ≥ 4:** When an access violation is found, the server responds with HTTP status 403, and the user sees a browser error saying the request is forbidden.
- **Behavior when *pxProcess.pzProductionLevel* < 4:** The request is processed normally; default behavior.

Note:

Pega Platform version 8.5 introduced this feature. When *pyBlockUnregisteredRequests* is enabled in development, the application development team can detect and respond to any problems with unauthorized or unregistered requests well ahead of application staging and production deadlines. In Pega Platform versions 8.5.6 and higher and 8.6.3 and higher, the *when* rule blocks unauthorized requests by default at production level values of 4 (4 is Staging, 5 is Production) or higher. In versions prior to 8.5.6 and 8.6.3, the *when* rule blocks unauthorized requests by default at production level values of 4 (4 is Staging, 5 is Production) or higher. See [Verifying requests when using custom controls](#) for registration steps for the user request based on the various scenarios. As a security best practice, ensure that you properly enable *pyBlockUnregisteredRequests* to return a value of True to block unregistered requests.

If the *pyBlockUnregisteredRequests* *when* rule evaluates to true, the following dynamic system setting becomes applicable:

security/validateReloadParameters

Dynamic system setting name: *security/validateReloadParameters*

- **Description:** Controls the HTTP response. This dynamic system setting is installed as part of the Pega Platform version 8.6.6 update and is available in the Pega-Engine ruleset.
- **Default value:** false
- **Behavior when the dynamic system setting is true:** If an access violation is found, Pega Platform rejects the HTTP request and security alert SECU0019 appears on the security alert log.
- **Behavior when the dynamic system setting is false:** If an access violation is found, Pega Platform accepts the HTTP request and security alert SECU0019 appears on the security alert log.
- [Verifying requests when using custom controls](#)

Pega Platform protects access to information in your application by using role-based settings and access control policies. Pega Platform provides additional request verification when you use autogenerated controls.

Verifying requests when using custom controls

Pega Platform protects access to information in your application by using role-based settings and access control policies. Pega Platform provides additional request verification when you use autogenerated controls.

To identify the rules that you need to configure, see [Identifying the custom controls to configure](#).

When you see the message "Unregistered request encountered" in the security alert log, you need to fix the behavior of the non-autogenerated control that invokes the specified activity. There are three ways to register the rule or encrypt the URL to prevent this alert from being logged, depending on the rule type.

1. [Configuring the custom control by using the rule form](#) — Use the rule form when a control such as a button calls the **Run Script** action, which in turn calls a custom function.
2. [Configuring the custom control by encrypting the URL in JavaScript](#) — Use JavaScript when you cannot use the rule form and your control calls custom JavaScript.
3. [Configuring the custom control by registering the action page in Java](#) — Use Java when your control calls a custom rule (other than a function) and you do not have access to JavaScript.

- [Identifying the custom controls to configure](#)

Pega Platform protects access to information in your application by using role-based settings and access control policies. Pega Platform provides additional request verification when you use autogenerated controls. When your application supports access checking and includes non-autogenerated controls, you need to identify and fix the controls.

- [Configuring a custom control by using the rule form](#)

When you see the message "Unregistered request encountered" in the security alert log, you need to fix the behavior of the non-autogenerated control that invokes the specified rule.

- [Configuring the custom control by encrypting the URL in JavaScript](#)

When you see the message "Unregistered request encountered" in the security alert log, you need to fix the behavior of the non-autogenerated control that invokes the specified rule. There are various ways to register the rule or encrypt the URL to prevent this alert from being logged. Use JavaScript when you cannot use the rule form and your control calls custom JavaScript.

- [Configuring the custom control by registering the action page in Java](#)

When you see the message "Unregistered request encountered" in the security alert log, you need to fix the behavior of the non-autogenerated control that invokes the specified rule. There are various ways to register the rule or encrypt the URL to prevent this alert from being logged. Use Java when your control calls a custom rule (other than a function) and you do not have access to JavaScript.

Identifying the custom controls to configure

Pega Platform protects access to information in your application by using role-based settings and access control policies. Pega Platform provides additional request verification when you use autogenerated controls. When your application supports access checking and includes non-autogenerated controls, you need to identify and fix the controls.

Perform one or more of the following steps to identify the rules to be configured:

1. Scan the security alert log for the message "Unregistered request encountered," and then configure the rule that is identified in the message.
2. Review the [attached list](#) of Pega Platform rules that have been configured in the Pega rulesets. If your application overrides any of these rules, configure the custom rules.
3. Use the attached branch ruleset to find sections and navigation rules that invoke the **Run Script** action. To use the branch ruleset, complete the following steps.
 1. In Dev Studio, import the [attached archive](#) file. The archive defines rules in the `FindRunScript` branch.
 2. Add the `FindRunScript` branch to your application definition, and save the application.
 3. Run the activity `pzPopulateRulesWithActions`, and check the box to exclude Pega internal rulesets. This activity outputs a spreadsheet named `RuleList_RunScript.xlsx`, which contains a list of rules that you need to examine and configure.

For information on how to configure the rules, see [Verifying requests when using custom controls](#).

Configuring a custom control by using the rule form

When you see the message "Unregistered request encountered" in the security alert log, you need to fix the behavior of the non-autogenerated control that invokes the specified rule.

There are various ways to prevent this alert from being logged:

- [Register the action.](#)
- [Submit the action request in encrypted form.](#)

Use the rule form when a control such as a button calls the **Run Script** action, which in turn calls a custom function. This rule form configures registration of the custom rules that are invoked from the custom function.

For example, a button calls the **Run Script** action, which calls the `setStockPrice` function. This function runs the data transform `setStockPriceDT`. In the rule form, you register `setStockPriceDT`.

1. [Registering the action and preventing the security alert](#)

When you see the message "Unregistered request encountered" in the security alert log, you need to fix the behavior of the non-autogenerated control that invokes the specified rule. One way to prevent this alert from being logged is to register the action.

2. [Submitting the action request content in encrypted form](#)

An alternate to registering the data transform is to encrypt the submitted action request content. The developer can write code in a non-auto section or html fragment and include it in the harness

Registering the action and preventing the security alert

When you see the message "Unregistered request encountered" in the security alert log, you need to fix the behavior of the non-autogenerated control that invokes the specified rule. One way to prevent this alert from being logged is to register the action.

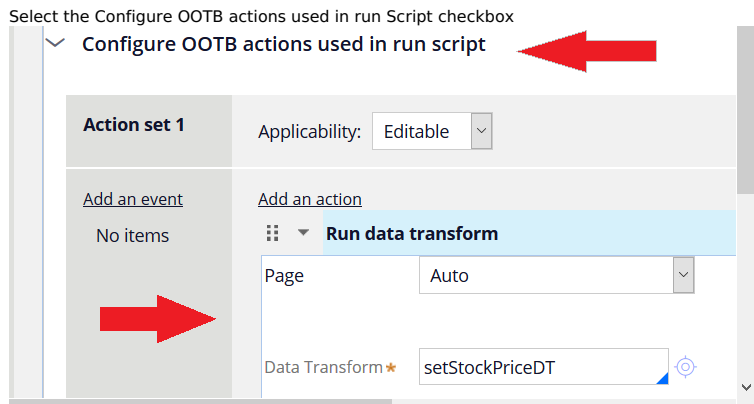
This procedure uses `setStockPrice` as a sample action/function name.

1. Define the action using the following steps:
 - a. On the **Action** tab for the button, click **Add an action**.
 - b. Click **All actions**, and then click **Run script**.
 - c. In the **Function Name** field, enter `setStockPrice`.

The Function Name Field with `setstockprice` added

Run Script	
Function Name*	setStockPrice
Parameters	
Name	Value
stockId	
price	

2. Register the data transform and prevent the security alert:
 - a. Select the **Configure OOTB actions used in run script** check box.
 - b. Click **Add an action**.
 - c. Select the **Run data transform** action.
 - d. In the **Data Transform** field, enter `setStockPriceDT`.



e. Click **Submit**.

Note: Your function might call numerous rules or rules other than a data transform. Steps 2c and 2d in this procedure will be different when you configure your non-autogenerated control with your custom function.

Submitting the action request content in encrypted form

An alternate to registering the data transform is to encrypt the submitted action request content. The developer can write code in a non-auto section or html fragment and include it in the harness.

To ensure the action request content is submitted in encrypted form, complete the following procedure:

1. Use the rule utility function `pega_rules_utilities.pzEncryptURLActionString` to encrypt the action. This function requires parameters in the following format: `pega_rules_utilities.pzEncryptURLActionString(tools, "Thread", pyActivity=activityName¶m1=value1¶m2=value2)` **For example:** `pega_rules_utilities.pzEncryptURLActionString(tools, "Thread", pyActivity=pzRunActionWrapper&pzDataTransform=setStockPriceDT)`
2. Enter a code snippet. **For example:** Examples of a specific and a generic code snippet are shown below:
 - generic code snippet: `pega_rules_utilities.pzEncryptURLActionString(tools, "Thread", "pyActivity=activityName¶m1=value1¶m2=value2")`
 - specific code snippet: `pega_rules_utilities.pzEncryptURLActionString(tools, "Thread", "pyActivity=pzRunActionWrapper&pzDataTransform=setStockPriceDT")`
3. Click **Save**.

For example:

Example from `@baseclass.SpellCheckerMoreDialog`:

```
var oSafeUrl= SafeURL_createFromURL("<%tools.appendstring(pega_rules_utilities.pzEncryptURLActionString(tools, "Thread", "pyActivity=Data-SpellChecker.Spellcheck"));%>");
var strSpellCheckXML = httpRequestAsynch(oSafeUrl.toURL(), strPost, 50, 100);
```

Configuring the custom control by encrypting the URL in JavaScript

When you see the message "Unregistered request encountered" in the security alert log, you need to fix the behavior of the non-autogenerated control that invokes the specified rule. There are various ways to register the rule or encrypt the URL to prevent this alert from being logged. Use JavaScript when you cannot use the rule form and your control calls custom JavaScript.

Call `pzEncryptURLActionString` to encrypt the URL, as shown in the following example:

Example: Encrypting the URL in JavaScript

```
<%
String myStyle = (tools.getParamValue("GadgetStyle").equals("")) ? "width:160" : tools.getParamValue("GadgetStyle");

String stringToEncrypt1 = "pyActivity=Data-Admin-WorkBasket.LookupList&pyListName=List&pyObjClass=Data-Admin-WorkBasket" +
    "&p1=pyWorkGroup&v1=<pega:reference name='pxRequestor.pxWorkGroup' mode='normal' />";

String encryptedAction1 = pega_rules_utilities.pzEncryptURLActionString(tools, "Thread", stringToEncrypt1);

String stringToEncrypt2 = "pyActivity=Data-Admin-Operator-ID.getListOfAvailable&pyListName=OperatorNameList&pyObjClass=Data-Admin-Operator-ID" +
    "&sortByPropertyName=pyUserName&p1=pyWorkGroup&v1=<pega:reference name='$save(wg)' mode='normal' />";

String encryptedAction2 = pega_rules_utilities.pzEncryptURLActionString(tools, "Thread", stringToEncrypt2);

%>
```

Configuring the custom control by registering the action page in Java

When you see the message "Unregistered request encountered" in the security alert log, you need to fix the behavior of the non-autogenerated control that invokes the specified rule. There are various ways to register the rule or encrypt the URL to prevent this alert from being logged. Use Java when your control calls a custom rule (other than a function) and you do not have access to JavaScript.

Register the action page by calling `getUIAction().register()`, as shown in the following example.

Example: Registration of custom flow action

```
ClipboardPage actionPage = pega.getUIEngine().getUIActionPage("createWork");

actionPage.getPage("pyActionAPI").putString("pyClassName", "O8CB4Y-GGApp-Work-TestCase");
actionPage.getPage("pyActionAPI").putString("pyFlowName", "pyStartCase");

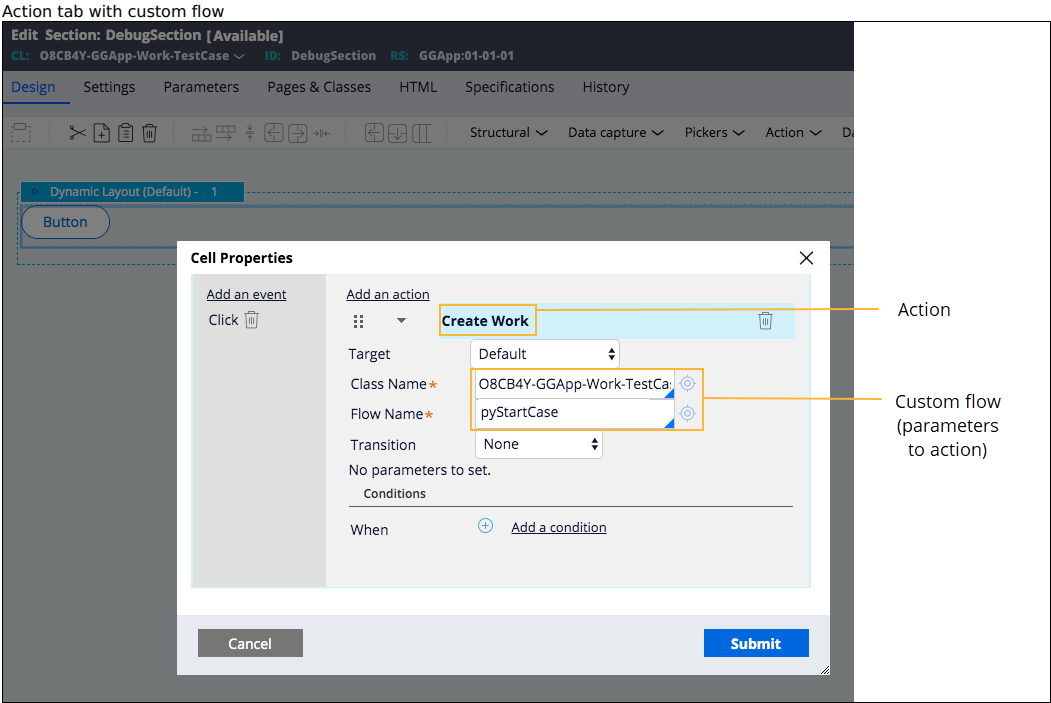
pega.getUIEngine().getUIAction(actionPage).register();
```

Registration of action page

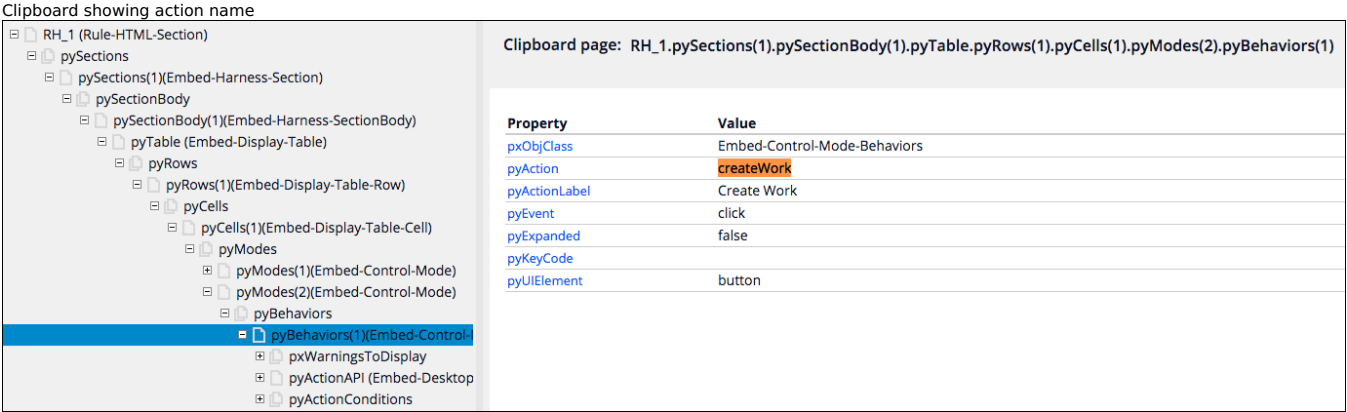
Parameter names and values

The following images show how to find the action name, parameter names, and parameter values when you are viewing the **Action** tab for the control in Dev Studio.

In the following image, the **Action** tab for the button shows that the **Create Work** action calls the *pyStartCase* flow on the *O8CB4Y-GGApp-Work-TestCase* class. The screen shows these values in a user-friendly format. To register the action in Java, you need the technical names of the action and its parameters.



While you view the **Action** tab, open the clipboard. From the current section on the clipboard, expand the entries until you find the action that is labelled *Create Work*, as shown in the following image. The action label that is displayed on the **Action** tab (*pyActionLabel*) is *Create Work* and the technical name for this action (*pyAction*) is *createWork*. When you register the action page in Java, use *createWork* as the action name.



To find the parameter names and values, open Live UI and find the **Class Name** and **Flow Name** fields. The following image shows the technical names for the class name parameter (*pyClassName*) and the flow name parameter (*pyFlowName*). When you register the action page in Java, use *pyClassName* and *pyFlowName* as the parameter names.

Class parameter name and value

Encryption

Encryption is a way to protect sensitive data within your application, without affecting the functionality of Pega Platform. Examples of typically encrypted data include Social Security numbers, credit card numbers, account numbers, and addresses.

Note: This guidance is specifically for UI centric applications where all the data is dealt with wholly within the Pega Platform run-time. For example, it would not apply to Pega Marketing or Customer Decision Hub deployments that must perform many asynchronous background processing tasks on the data, including asking relational databases to run SQL queries that make comparisons against data that is stored encrypted by Pega Platform.

You can choose one of the following methods to encrypt case data:

- Class-level (BLOB) encryption
- Property-level encryption.

The best practice depends on the capabilities of your application and the amount of data that needs to be encrypted.

Class-level encryption

Class-level (BLOB) encryption occurs when Pega Platform retrieves and opens an instance. Decryption occurs when Pega Platform retrieves and opens an instance.

The difference between class-level (BLOB) and property-level encryption is performance and functionality. BLOB encryption and decryption are more straightforward and efficient than the individual encryption and decryption of a large number of properties that store sensitive case data. BLOB encryption does not, however, encrypt properties that have been exposed as columns for reporting, properties that are stored on the clipboard when you open a case, or properties that are stored in secondary data stores such as the *Elasticsearch* index.

Encryption of the *pzPVStream* (BLOB) column for one class does not affect the *pzPVStream* column of other classes, even those that occupy the same table in the Pega Platform database.

For more information, see [Storage stream encryption of selected classes](#).

Property-level encryption

Note: Do not configure properties for encryption if they are used for data segmentation purposes in Pega Marketing or Customer Decision Hub. Segmentation cannot be accomplished on encrypted values.

By using access control policies, you can encrypt any property type by listing it in a *PropertyEncrypt* access control policy. You can define such policies on *Work-*, *Data-*, and *Index-* classes. The property is encrypted in the database, clipboard, logs, and search indexes. If there is no *PropertyRead* policy that obfuscates the property, then the decrypted property value is visible to the user in a UI control. Properties specified in a *PropertyEncrypt* policy are encrypted unconditionally. Define *PropertyRead* policies to obfuscate or mask these values depending on who is viewing them. To get the *cleartext* value outside of UI controls (for example, in the background processing of cases), call the appropriate decryption method. You can define *PropertyEncrypt* access control policies for properties that are optimized for reporting only if the property type is equal to Text. To define a report filter for an encrypted property that is not Text, convert the values to text and store them in a Text property that is encrypted and optimized.

For more information, see [Creating an access control policy](#).

Data Encryption

Data encryption in Pega Platform provides the following benefits:

- Gives sensitive data in your applications an additional layer of security while preserving critical Pega Platform functionality
- Makes it easier to comply with privacy policies, regulatory requirements, and contractual obligations for handling private data

Encryption uses a cipher algorithm to turn readable text (plaintext) into an unreadable secret format (ciphertext). The ciphertext can be decrypted only through the use of the correct encryption key. Every time that Pega Platform commits changes to the Pega Platform database and *Elasticsearch* indexes, data encryption occurs. Encryption is unrelated to the encryption that your database software or encryption software provides.

Decide what data needs to be encrypted and which encryption method to use before your application goes into production. If you decide to encrypt certain properties or change the encryption method after your application is in production, you must write activities to encrypt or reencrypt the existing data.

To avoid loss of encrypted data, do not delete data from the *pr_data_admin_sec** tables.

Custom Cipher

Using a custom cipher also makes it difficult to support best practices such as key rotation, which is a complex task that requires maintaining information about every key ever used to encrypt data. Mistakes can easily lead to loss of data.

Defining a custom cipher requires that you define the Java classes that need to be installed on your Pega cluster.

This process is complex and requires very careful testing and assistance from Global Customer Support staff. Custom ciphers are not supported for use by Pega Cloud customers. Using a custom cipher also makes it difficult to support best practices such as key rotation, because the encryption key is compiled into a .jar file. If the key changes after production data is encrypted, you might not be able to decrypt the data without making significant changes to the custom cipher.

Use a custom cipher only when your organization's security standards require the use of a cipher that is different from the Pega Platform cipher.

For more information, see [Creating a custom cipher in Pega Platform](#).

Encryption key management

When you use BYOK, your master key is never directly accessed or stored by Pega Platform, using a two-tier encryption strategy.

Pega Platform creates an initial customer data key (CDK), and other CDKs based on the key rotation policy that you specify on the **Data Encryption** landing page. The CDKs are stored in encrypted format in the Pega database, and never appear in clear text on the clipboard or anywhere in memory that is accessible by you or your application.

On node start up, your KMS is called to decrypt the CDKs stored in the Pega database, using your master key. The CDKs are then reencrypted using a key generated by Pega Platform, which is not accessible to you or within your application. The reencrypted CDKs are only decrypted within Pega Platform as needed during operations to encrypt or decrypt your data.

Protection of other sensitive data

Pega creates and stores cookies on client browsers to assist in authenticating client requests. The cookie does not contain any operator-specific or case-specific data, only a randomly generated session ID. By default, this cookie is also encrypted using a Pega generated key. You can replace this key by configuring BYOK for the master key that is used to encrypt cookies on the **Data Encryption** landing page, in the section titled **System data encryption**.

For more information, see [Configuring the platform cipher](#).

Passwords that are manually entered during development or application use (for example, login passwords and ruleset passwords) are not encrypted, but are stored as hashes, using the bcrypt hashing algorithm, in the Pega database.

Most other types of sensitive configuration data will be encrypted by using the BYOK feature if you configure it, otherwise these will be encrypted using the internal Pega Platform cipher and Pega Platform-generated master key, for example:

- Authentication passwords for services, connectors, agents, and email.
- Client secrets in keystores, authentication services, and authentication profiles.

However, user IDs and passwords for database connections or command-line utilities are not encrypted by using BYOK. Instead, they are encrypted by using an internal Pega Platform cipher and Pega Platform-generated master key that is not accessible by you or your application.

Encryption of Passwords and Other Secrets

Pega Platform stores and automatically encrypts a variety of passwords and other sensitive configuration data, such as:

Operator passwords

Operator passwords uses the *Password* property type. Automatically hashed, no developer actions are required.

Ruleset version passwords

Ruleset version passwords uses the *Password* property type. Automatically hashed, no developer actions are required.

Database connection passwords

When you run any of the command-line utilities, you must provide database credentials in the *prbootstrap.properties* file and in the *prconfig.xml* file. By default, Pega Platform stores database connection passwords as clear text in these files. To secure these passwords, you can do the following:

- Generate encrypted passwords for the *prbootstrap.properties* file by using the **PassGen** tool.
- Generate encrypted passwords for the *prconfig.xml* file in a *pegarules.keyring* file with the **Keyring** tool.

For more information, see [How To Encrypt Database Passwords Using a JCE Keyring file](#).

Rule resolution passwords

Some command-line utilities require that you specify a Pega Platform user name and password for rule resolution. You can store these Pega Platform credentials in the same key ring file that stores the database connection password for the *prconfig.xml* file.

Passwords for external system authentication

Passwords for integration services and connectors, email and database connectivity, and agents require external authentication. Pega stores encrypted passwords using key information controlled by Pega and are decrypted as needed for authenticating with these external systems.

Sensitive information in configuration data

Pega encrypts certain configuration information, such as authentication profile data, keystore instances, by using key information also controlled by Pega Platform.

Note: By default, encryption uses key information that is controlled by Pega Platform. If your application data encryption is configured by using BYOK, then encryption of this data is also accomplished by using the BYOK configuration.

- [Encrypting application data](#)

Help secure your data by selecting the type of encryption to use in your application to encrypt and decrypt passwords, properties, and BLOBs. When you encrypt sensitive data at the application level, it inherently protects data stored in your application, and making it so only authorized parties can read it.

Encrypting application data

Help secure your data by selecting the type of encryption to use in your application to encrypt and decrypt passwords, properties, and BLOBs. When you encrypt sensitive data at the application level, it inherently protects data stored in your application, and making it so only authorized parties can read it.

Caution: After you enable application data encryption, you cannot disable it and restore the default settings. However, you can change the initial encryption type to another encryption type as necessary.

You can access the **Data Encryption** tab from the navigation pane of Dev Studio by clicking **Configure System Settings Data Encryption**.

Note: The **Data Encryption** tab is visible to operators who have the *pxCanManageDataEncryption* privilege in their access roles. This privilege is part of the *PegaRULES:SecurityAdministrator* role.

The following encryption options are available:

Application data encryption (exposed by default)

You can change the encryption type for your application at any time by switching between the platform cipher and a custom cipher.

- **Platform cipher** – The platform cipher uses the AES256-CBC with PKCS7 Padding cryptographic algorithm to encrypt and decrypt sensitive case data in your application. You need to use your own Customer Master Key (CMK), managed by your external key management service (KMS). The keys stored in AWS KMS support time-based and on-demand data key rotations. You do not need to create any custom cipher code for this encryption option.

Caution: When changing the KMS keystore, you must activate the new keystore before you delete or disable the active Customer Master Key. **Caution:** If you delete the KMS master key or the KMS master key expires, Pega Platform cannot decrypt the previously encrypted data, which can result in data loss.

- **Forced data key rotation** – You can rotate the internal encryption key at any time. This is typically done if the Customer Data Key in the platform cipher has been compromised. The key can be rotated regardless of the configured key rotation period in the keystore data instance.
- **Custom cipher** – The Custom cipher is used when the platform cipher does not suit your company's needs. To use this encryption type in your application, you need to create your own custom encryption cipher. For more information, see [Creating a custom cipher](#).

You can switch between the platform cipher and a custom cipher to change the encryption type for your application at any time.

Caution: When you switch between cipher types, do not delete the original custom cipher or encryption keys. Deleting the previous custom cipher or encryption keys will make Pega Platform unable to decrypt previously encrypted data.

After you configure and activate the cipher, you specify the classes and properties to encrypt. For more information, see [Encrypting the storage stream \(BLOB\)](#), and [Creating an access control policy](#) for the PropertyEncryption action.

Application master key usage history

You can see read-only data based on the application data encryption settings, which shows activation and rotation history.

System data encryption (exposed by default)

Allows you to tell Pega Platform what to use or who is providing the master key for system data encryption. The only data that is encrypted is the session cookie. There are two master key providers:

- **Pega Platform** – Pega will generate the key for you.
- **Keystore** – You can provide your own key by specifying a key and selecting a keystore instance.

Master key usage history

You can see read-only data based on the system data encryption settings, which shows the activation and regeneration history.

Caution: Do not delete data from the *pr_data_admin_sec** tables. Doing so might result in loss of encrypted data.

For more information about encryption, see [Encryption](#).

- [Configuring the platform cipher](#)

- [Configuring a custom cipher](#)
- [Configuring an Amazon Key Management Service \(KMS\) key rotation](#)
- [Encrypting the storage stream \(BLOB\)](#)
- [Enabling and disabling encryption for communication among search nodes](#)
- [Implementing and using the TextEncrypted property type](#)
- [Creating an access control policy](#)
- [Enabling password encryption for BIX command-line extractions in on-premises systems](#)

Configuring the platform cipher

Use the platform cipher to encrypt and decrypt sensitive data in your application without the need to create and code your own custom cipher. By using the platform cipher, you do not need to share knowledge about sensitive data in your application with Pega staff, because their assistance is not required to install a cipher.

Before you begin: To configure ciphers, you must have the *pxCanManageDataEncryption* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.

1. Create a keystore to reference the Customer Master Key (CMK) that is stored in the Amazon Web Services Key Management Service (AWS KMS) or another key management service. For more information, see [Creating a keystore](#).
2. In the header of Dev Studio, click **Configure System Settings Data Encryption**.
3. To encrypt application data, in the **Application data encryption** section, do the following steps:
 - a. Select **Platform cipher** to encrypt data with the default cryptographic algorithm that is used by Pega Platform.
 - b. In the **Keystore** field, press the Down Arrow key, and then select the name of your keystore.

Caution:

 - To change the Customer Master Key, you need to create a new Keystore instance, and then reference it on the Data Encryption landing page.
 - When changing the keystore, you must activate the new keystore before you delete or disable the currently active Customer Master Key. Otherwise, the data encrypted by the old keystore cannot be reencrypted by the new keystore.
 - c. Click **Activate** to start using your keystore for encryption purposes.
4. In the **System data encryption** section, encrypt system data by selecting the source of the master key:

Choices	Actions
Pega Platform	Select Pega Platform , and then click Regenerate .
Keystore	a. Select Keystore. b. In the Select keystore field, press the down arrow key, select a keystore that is sourced from a data page, and then click Activate.
5. On the Warning dialog box that informs you about the changed encryption keystore, click **Apply** to confirm the keystore activation.
6. In the navigation panel, click **Records SysAdmin Agents**.
7. In the **RuleSet Name** column, click *Pega-IntegrationEngine*.
8. In the **Schedule Agents** section, select the **Enable** check box for the *pyValidateKMSMetadata* agent. This agent checks whether the Customer Master Key that is stored in AWS is available. If the key is disabled or selected for deletion, the agent logs a security alert in the SECURITYALERT log file.
9. **Optional:** To receive an email notification about an inactive Customer Master Key, modify the *pyHandleKMSValidationError* activity.
 - a. In the first step, in the Method field, press the Down Arrow key and select Property-Set.
 - b. Click the **Collapse method parameters** icon and enter the following method parameters.
 - **Param.To** - Enter the email recipient.
 - **Param.From** - Enter the sender email.
 - **Param.Subject** - Enter the title of the email.
 - **Param.Message** - Enter the message that the email provides.
 - **Param.Password** - Enter the password to the sender email.
 - **Param.smtpHost** - Enter the sender email server address.
 - **Param.HTMLmessage** - Enter "true" to send the message as HTML. Enter "false" to send the message in plain text.
 - c. Click **Add a step**.
 - d. In the **Method** field, press the Down Arrow key and enter Call SendEmailNotification.
 - e. Select the **Pass current parameter page** check box.
 - f. Click **Save**.
10. **Optional:** If you are using an AWS KMS, you can create an Amazon CloudWatch Alarm that informs you if the Customer Master Key that is being used is pending for deletion. For more information, see the Amazon article *Creating an Amazon CloudWatch Alarm to Detect Usage of a Customer Master Key that is Pending Deletion*.

- [Viewing keystore usage history in the platform cipher](#)
- [Forcing data key rotation in the platform cipher](#)
- [Encrypting application data](#)
- [Encrypting application data by using a custom key management service](#)
- [Encrypting system data by using a custom key management service](#)

Viewing keystore usage history in the platform cipher

View the keystore usage history to see which keystores were used previously or by whom the keystores were activated in the platform cipher. Viewing the keystore usage history can help you determine which keystore can be used in the future.

Before you begin: To view keystore history, you must have the *pxCanManageDataEncryption* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.

1. In the header of Dev Studio, click **Configure System Settings Data Encryption**.
2. Click **Platform cipher**.
3. In the **Keystore usage history** section, in the **Application data** table, click **View all**.
4. Review the information about the keystores that are used in the platform cipher.

- [Configuring the platform cipher](#)
- [Encrypting application data](#)

Forcing data key rotation in the platform cipher

If the Customer Data Key (CDK) that you are using in the platform cipher has been compromised, you can immediately rotate the internal encryption key. You can rotate the key regardless of the key rotation period that you configure in the keystore data instance.

Before you begin: To force data key rotation, you must have the *pxCanManageDataEncryption* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.

1. In the header of Dev Studio, click **Configure System Settings Data Encryption**.
2. In the **Application data encryption** section, perform the following steps:
 - a. Click **Platform cipher**.
 - b. Click **Force data key rotation** to immediately rotate the internal encryption key.

- [Creating a keystore for application data encryption](#)
- [Encrypting application data](#)

Creating a custom cipher

Pega Platform provides encryption of sensitive data while the data is at rest. This encryption can be performed using a platform cipher or by using a custom cipher that you define.

Before you begin:

Before you implement a custom cipher, ensure that you have completed the following requirements:

- Pega Platform is installed.
- The Pega Platform distribution image is extracted to a local hard drive where you have write permission, and prweb.jar has been expanded.

- You have update permission for the Pega rules database.
- The JAVA_HOME environment variable is set.

To implement a custom cipher, you create the cipher class, create a .jar file, and load the .jar into the database. Follow these steps to implement a custom cipher.

1. On the hard drive where you extracted the distribution image, remove the build name suffix from the jar file names in the prweb\WEB-INF\lib folder. **Result:** You should have files similar to the following:
 - o jsr94-1.0.jar
 - o pradapter.jar
 - o prbootstrap-api.jar
 - o prbootstrap.jar
 - o prdbcp.jar
2. Update the prconfig.xml and prbootstrap.properties files in your expanded image directory to include database login credentials, if they are not already present.
3. Perform the following steps to determine which cryptographic capabilities are supported in your environment and to select the algorithm for the custom cipher.
 - a. From the scripts directory, run runPega.bat OR runPega.sh with the following parameters:

--driver <path to the database driver file>

--prweb <path to the archives/prweb folder>

--propfile <path to the bootstrap properties file>

com.pegarules.exec.internal.util.crypto.JCECapabilities

For example:

```
./runPega.sh
--driver /Users/yourname/Develop/apache-tomcat-8.0.33/lib/postgresql-9.4.1208.jar
--prweb /Users/yourname/Develop/PegaPlatformDistro/archives/prweb
--propfile /Users/yourname/yourname-prbootstrapX.properties
com.pegarules.exec.internal.util.crypto.JCECapabilities
```

- b. The output from step 3a is a list of supported capabilities. From that list, choose an algorithm, key length, and provider. The algorithm you select must have both a Cipher and KeyGenerator of the same name listed in the output. Depending on your algorithm, you might also specify a mode and padding for your transformation. Refer to the javadoc for javax.crypto.Cipher for more information on transformations. Consult your security administrator to ensure that your algorithm, key length, and provider satisfy your company's security policies. If the combinations available do not yield a key length of sufficient strength for your needs, you may need to update your Java Cryptography Extension (JCE) jurisdiction policy file to include unlimited key strength. For further details on the jurisdiction policy file, see the JCE documentation. If you update the jurisdiction policy file, run the JCECapabilities utility again to view the new combinations of algorithm, key length, and provider.
4. Use the PRCipherGenerator tool to generate Java source for your cipher class.
 - a. From the scripts directory, run runPega.bat OR runPega.sh with the following parameters:

--driver <path to the database driver file>

--prweb <path to the archives/prweb folder>

--propfile <path to the bootstrap properties file>

com.pegarules.exec.internal.util.crypto.JCECapabilities

<transformation>

<key length>

<provider> (optional)

For example:

```
./runPega.sh
--driver /Users/yourname/Develop/apache-tomcat-8.0.33/lib/postgresql-9.4.1208.jar
--prweb /Users/yourname/Develop/PegaPlatformDistro/archives/prweb
--propfile /Users/yourname/yourname-prbootstrapX.properties
com.pegarules.exec.internal.util.crypto.PRCipherGenerator
blowfish/cfb/pkcs5padding
128
```

Some examples of transformation, key length, and provider are shown below.

Transformation	Key length	Provider
blowfish/cfb/pkcs5padding	128	
blowfish	128	
SKIPJACK/ECB/PKCS7Padding	128	BC
DESede/CBC/PKCS5Padding	168	

Result: Java code is displayed.

- b. Copy and save the generated code in a .java file for your custom cipher class and replace the placeholders with the package name and class name for your custom cipher.
 - c. You can create a test version of your cipher that contains logging statements that you can use to verify the implementation. To create a test version of your cipher, add the following two methods.
- Note:** You must remove the logging code prior to using the cipher in production, as described in step 11e.

```
public byte[] encrypt(byte[] alnput) {
    System.out.println("encrypting");
    return super.encrypt(alnput);
}
```

```
public byte[] decrypt(byte[] alnput) {
    System.out.println("decrypting");
    return super.decrypt(alnput);
}
```

5. Determine which codeset and version will hold your custom cipher. It is recommended that you use the customer codeset. However, you can choose to use a Pega Platform engine codeset. **Note:** If you use a Pega Platform engine codeset and you later upgrade your engine version, you will have to migrate your cipher to the new version.
6. Compile the cipher class and create a jar file from the command line by using compileAndLoad.bat OR compileAndLoad.sh with the following parameters:

--basedir <path to your .java source folder>

--driver <path to the database driver file>

--jarfile <name of the jar file to output, without path>

--prweb <path to the archives/prweb folder>

--propfile <path to the bootstrap properties file>

--codeset customer OR pega-enginecode

--codesetversion 06-01-01 for a customer codeset; otherwise, <relevant engine codeset version>

--prprivate

<Package and class name in quotes for your cipher source code>

For example:

/compileAndLoad.sh --basedir /Users/yourname/Develop/Projects/SiteCipherProject/src --driver /Users/yourname/Develop/apache-tomcat-8.0.33/lib/postgresql-9.4.1208.jar --jarfile MyCustomCipher --prweb /Users/yourname/Develop/PegaPlatform

Note: This command is very sensitive to spaces. Follow the example shown above.

7. Verify that the generated .jar file has been imported to the Pega Platform engine classes database table by entering the following SQL query against the rules schema:

```
select pzjar, pzpackage, pzclass, pzlastmodified, pzcodeset, pzcodesetversion, pzpatchdate from <rules-schema>.pr_engineclasses
where pzjar like '<jar file name>'.jar'
```

8. Log on to Pega Platform and add the Dynamic System Setting with the following values:

Owning ruleset = Pega-Engine

Setting purpose = prconfig/crypto/sitecipherclass/default

Setting value = The qualified class name of your custom cipher.

For example:

```
Pega-Engine
prconfig/crypto/sitecipherclass/default
com.yourname.CustomCipherBC
```

For more information, see [Creating a Dynamic System Setting](#).

9. Restart Pega Platform for your custom cipher to take effect.
10. Log on to Pega Platform and perform the following steps to activate your cipher:
- Dev Studio System Settings Data Encryption**.
 - Click **Custom cipher**.
 - Click **Activate**.
11. If you added logging statements to your cipher in step 4c, verify that encryption is working for your instance of Pega Platform. If you did not add logging statements, skip this step.
- Create a case type.
 - Open the Class definition for the case type and select **Encrypt BLOB**.
 - From one of the **Case** portals, create an instance of the case type. Close and redisplay the instance.
 - Review the console log.
- You should see messages that say *encrypting* and *decrypting*. These messages are generated from your custom cipher class.
- Prior to going into production, remove any logging statements from your cipher code and repeat steps 5 through 8 so that your log does not grow excessively.

What to do next: You need to configure your custom cipher. For more information, see [Configuring a custom cipher](#).

Configuring a custom cipher

Although Pega Platform has an integrated platform cipher, you can implement a custom cipher to address policies that are specific to your company. You can use your own custom cipher in Pega Platform as a data encryption type.

Before you begin:

- To configure ciphers, you must have the *pxCanManageDataEncryption* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.
- Complete the [Creating a custom cipher in Pega Platform](#) procedure.

- In the header of Dev Studio, click **Configure System Settings Data Encryption**.
- In the **Application data encryption** section, encrypt application data:
 - Encrypt data with your custom cipher by selecting **Custom cipher**.
 - In the **Site specific cipher class** field, enter the name of the class of your cipher.
You defined the cipher class in the [Creating a custom cipher in Pega Platform](#) procedure.
 - Start using this custom cipher for encryption purposes by clicking **Activate**.
- In the **System data encryption** section, encrypt system data by selecting the source of the master key:

Choices	Actions
Pega Platform	Select Pega Platform , and then click Regenerate .
Keystore	- Select Keystore. - In the Select keystore field, press the down arrow key, select a keystore that is sourced from a data page, and then click Activate.

- In the **Warning** dialog box that informs you about the changed encryption cipher, confirm the custom cipher activation by clicking **Apply**.

- [Encrypting application data](#)

Encrypting the storage stream (BLOB)

In Pega Platform, you can encrypt the storage stream (BLOB) by using a platform or custom cipher.

- Update the class forms. Perform the following actions:
 - Open each class form that is to hold instances that have encrypted Storage Stream values and select the **Encrypt BLOB** check box on the **General** tab.
 - Click **Save** on the class form.
 - Repeat steps a and b for all affected classes.

Result: Encryption is enabled. Your application can create the class instances normally.

- [Storage stream encryption of selected classes](#)
- [Organizing rules into classes](#)
- [Encrypting application data](#)
- [Implementing and using the TextEncrypted property type](#)
- [Creating an access control policy](#)

Storage stream encryption of selected classes

You have the option to enable encryption of the Storage Stream or BLOB (*pxPVStream*) column for rows in the Pega Platform database that correspond to specific classes. Use this option when you have a particular data class that you want to protect throughout the application.

Encryption of the Storage Stream column for one class does not affect the Storage Stream of other classes, even those that occupy the same table in the Pega Platform database. Encryption occurs when Pega Platform saves an instance of the class. Decryption occurs when Pega Platform retrieves and opens an instance.

Storage Stream encryption is unrelated to any database encryption that is provided by database software, or third-party encryption software.

CPU processing for encryption and decryption can affect overall system performance, depending on data volumes and algorithms. To assess the impact, check the record time and count properties on the Full Details display of the Performance tool. Elapsed time and CPU times are components of other statistics that measure database times. CPU statistics are available only for Windows platforms. All times are in seconds. The following statistics are available:

Property	Meaning
<i>pxEncryptCPU</i>	The required CPU time to encrypt Storage Stream contents.
<i>pxDecryptCPU</i>	The required CPU time to decrypt Storage Stream contents.
<i>pxEncryptElapsed</i>	The required elapsed time to encrypt Storage Stream contents.
<i>pxDecryptElapsed</i>	The required elapsed time to decrypt Storage Stream contents.
<i>pxEncryptCount</i>	Number of Storage Streams encrypted.
<i>pxDecryptCount</i>	Number of Storage Streams decrypted.

- [Organizing rules into classes](#)
- [Using the full details display of Performance Analyzer](#)

Encrypting individual properties

Pega Platform provides encryption of sensitive data while the data is at rest. You select the data that you want to encrypt, and Pega Platform encrypts the selected components. You can choose to encrypt individual properties or entire BLOB columns, based on performance considerations and data sensitivity. This article describes how to encrypt specific properties.

For information on encrypting the entire BLOB, see [Encrypting the storage stream \(BLOB\)](#).

Prior to Pega Platform 7.4, the TextEncrypted property type was used to encrypt properties. As a best practice, use a PropertyEncrypt access control policy as described below.

Defining rules in that use your cipher by using PropertyEncrypt access control policies

To use the *PropertyEncrypt* access control policy in Pega Platform, which is available beginning with Pega 7.4, do the following:

1. If you are using Pega 8.1 or earlier, enable attribute-based access control for an instance of Pega Platform. This is enabled by default starting in Pega 8.2. For more information, see [Enabling attribute-based access control](#).
2. Create a *PropertyEncrypt* access control policy and list the properties that you want to encrypt. For more information, see [Creating an access control policy](#).

The property is encrypted in the database, clipboard, logs, and search indexes. If there is no *PropertyRead* policy that obfuscates the property, the decrypted property value is visible to the user in a UI control. The property is automatically encrypted when a value is assigned to the property and saved to the database.

In report definitions, the property is displayed in report results and can also be referenced on the left side of filter conditions that use the *Is equal* and *Is not equal* operators. You cannot use this property elsewhere in report definitions (for example, to sort, rank, or group results, in SQL functions, and so forth).

Key management system for application data encryption

Create a keystore that references keys other key management services, such as Microsoft Azure Key Vault, HashiCorp Vault, and Google Cloud KMS, and Amazon KMS through the use of a data page. By supporting additional key management services, Pega Platform offers you increased flexibility when defining keys that are used for encryption of application and internal system data.

Build in encryption at every layer as best you can. Your encryption strategy is only as good as your ability to protect encryption keys. Encryption keys are the secret that has to be protected.

- [Creating a keystore for application data encryption](#)

Create a keystore instance for your keystore file, which contains the keys and certificates that are used, for example, to support Web Services Security and outbound email security.

- [Encrypting system data by using a custom key management service](#)

Encrypt system data using an encryption key that is sourced from a Custom Key management service (KMS) that is accessed from a data page. For system data encryption, you can only use the Custom Key management service.

- [Keystores](#)

A keystore is a file that contains keys and certificates that you use for encryption, authentication, and serving content over HTTPS. In Pega Platform, you create a keystore data instance that points to a keystore file.

- [Changing the default keystore caching settings](#)

You can change the values of the KeyStoreCacheExpireTime and KeyStoreCacheSize settings to control how often the keystore cache is refreshed and to restrict cache size. The lower the values, the less memory is used, but processing power is reduced.

- [Creating a keystore instance for an external key management service](#)

You can encrypt application and system data in Pega Platform™ by using either the platform cipher or a cipher that is stored within an external key management service (KMS). Use an external KMS to control the ownership, creation, and rotation of your master key.

- [Importing an X.509 certificate](#)

You can import X.509 certificates that are defined in keystore instances of type JKS or PKCS12. They become active without your having to restart the server.

Creating a keystore for application data encryption

Create a keystore instance for your keystore file, which contains the keys and certificates that are used, for example, to support Web Services Security and outbound email security.

Before you begin: Obtain a keystore file that is signed by a certificate authority or is self-signed, and make it available as a file or as a URL. Obtaining the keystore file is done outside of Pega Platform. If the file has a password, you also need the password. For more information, see your security administrator.

1. In the header of Dev Studio, click **Create Security Keystore**.
2. In the **Short description** field, enter a name for the keystore.
3. In the **Keystore** field, enter an ID for the keystore.
4. Click **Create and open**.
5. In the **Keystore location** field, press the Down arrow key and select the key management system or keystore source:

KEY MANAGEMENT SYSTEM (KMS) FOR APPLICATION DATA ENCRYPTION

- **Amazon Key Management Service (KMS)** — Reference an encryption key that is stored in Amazon Web Services Key Management Service (AWS KMS).
- **Microsoft Azure Key Vault** — Reference an encryption key that is stored in Microsoft Azure Key Vault.
- **HashiCorp Vault** — Reference an encryption key that is stored in HashiCorp Vault.
- **Google Cloud KMS** — Reference an encryption key that is stored in Google Cloud KMS.
- **Custom — Source master key from other KMS using a data page** — Reference an encryption key that is stored in an external custom source and is retrieved by using a data page. For details on configuring a custom KMS for application data encryption, see [Encrypting application data by using a custom key management service](#).

KEY MANAGEMENT SYSTEM (KMS) FOR SYSTEM DATA ENCRYPTION

- **Custom — Source master key from other KMS using a data page** – Reference an encryption key that is stored in an external custom source and is retrieved by using a data page. For details on configuring a custom KMS for system data encryption, see [Encrypting system data by using a custom key management service](#).

KEYSTORE

- **Upload file** — Upload the keystore file, such as a Java KeyStore (JKS) file.
 - **Reference to file** — Reference the keystore file from a file location.
 - **Reference to URL** — Reference the keystore file that contains public keys from a URL address.
 - **Reference to data page** — Reference the keystore that is stored in a data page.
6. Configure the keystore based on the keystore location that you selected.
 - [Configuring an Amazon Key Management Service \(KMS\) key rotation](#)
 - [Configuring a Microsoft Azure Key Vault keystore](#)
 - [Configuring a HashiCorp Vault keystore](#)
 - [Configuring a Google Cloud KMS keystore](#)
 - [Configuring a keystore by uploading a file](#)
 - [Configuring a keystore by using a file reference](#)
 - [Configuring a keystore by using a URL reference](#)
 - [Configuring a keystore by using a data page reference](#)

- [Configuring the platform cipher](#)
- [Keystores](#)
- [Changing the default keystore caching settings](#)

Configuring an Amazon Web Services Key Management Service keystore

To use the Amazon Web Services Key Management Service (AWS KMS) in Pega Platform, you create the master key in AWS KMS, and then you create a keystore instance in Pega Platform that refers to the KMS.

This procedure shows an example of creating a customer master key and an access key within AWS KMS. For detailed instructions, refer to the documentation from AWS. The AWS documentation includes the *AWS Developer Guide for KMS* and the *Managing Access Keys for IAM Users* guide.

1. In AWS, under Identity and Access Management (IAM), create an alias for the customer master key.
2. Add the **Tag key** and **Tag value** to the master key.
3. Assign administrators for the master key.
4. Assign users for the master key.
5. Click **Create key** to generate the access key.
6. Copy the **ARN** for the key. You use the ARN to create the Pega keystore instance.
7. **Before** clicking **Close**, copy the **Access key ID** and **Secret access key**. You use these to create the Pega keystore instance.
8. Create a keystore instance in Pega Platform, using the [Creating a keystore for application data encryption](#) procedure and enter the following:
 - a. In the **Access key ID** field, enter the value from step 7.
 - b. In the **Secret access key** field, enter the value from step 7.
 - c. In the **Customer master key ID** field, enter the **ARN** value from step 6.

Configuring an Amazon Key Management Service (KMS) key rotation

To configure a keystore, you can reference the encryption key that is stored in the Amazon Web Services Key Management Service (AWS KMS).

Before you begin: You must [create a keystore](#) data instance in Pega Platform with **Keystore location** equal to *Amazon Key Management Service (KMS)* before you can configure the keystore.

1. If you have not yet defined your keys in Amazon, log in to your Amazon Web Services account, and under Identity and Access Management (IAM), create a Customer Master Key (CMK) and access key.
 - For information about how to create a Customer Master Key, see the AWS Developer Guide that describes the AWS Key Management Service and [Configuring an Amazon Web Services Key Management Service keystore](#).
 - The access key provides the access key ID and secret access key that you need to enter in the keystore form. For more information, see the Amazon guide *Managing Access Keys for IAM Users*.
 - When you create the encryption key, select the same geographic region for your key that your application is deployed in. Selecting the same geographic region gives your application the best network performance.
2. Open a keystore from the navigation panel by clicking **Records Security Keystore** and selecting a KMS keystore from the instance list.
3. In the **Access key ID** field, enter the access key ID that you created in AWS KMS.
4. In the **Secret access key** field, enter the secret access key that you created in AWS KMS.
5. In the **Customer master key ID** field, enter the Amazon Resource Name (ARN) of the customer master key created in AWS KMS.
6. In the **Customer data key rotation in days** field, enter the number of days after which the customer data key (CDK) rotates.

Note: The recommended (default) value is 90 days. You can set the rotation to any time between 30 and 365 days.
7. Click **Test connectivity** to verify that all fields are filled out correctly and that Pega Platform is able to connect to AWS KMS.
8. Click **Save**.

- [Keystores](#)
- [Creating a keystore for application data encryption](#)

Configuring a Microsoft Azure Key Vault keystore

Configure a keystore by referencing an encryption key that is stored in a Microsoft Azure Key Vault.

Before you begin: You must [create a keystore](#) data instance in Pega Platform with **Keystore location** equal to *Microsoft Azure Key Vault* before you can configure the keystore.

1. If you have not yet defined your cryptographic key in Azure, log in to your Microsoft Azure Key Vault account and create a key with an RSA algorithm. For details, see your Azure Key Vault documentation and [Creating a keystore for application data encryption](#).
2. Open a keystore from the navigation panel by clicking **Records Security Keystore** and selecting an Azure Key Vault keystore from the instance list.
3. In the **Client ID** field, enter the client ID of the application that you created in Azure.
4. In the **Client key** field, enter the client secret for the application that you created in Azure.
5. In the **Customer master key ID** field, enter the key identifier of the master key that you created in Azure Key Vault.
6. In the **JSON Web Algorithm (JWA)** list, select the algorithm for the JSON web token.
 - **RS1_5**
 - **RSA-OAEP**
 - **RSA-OAEP-256**
7. In the **Customer data key rotation in days** field, enter the number of days after which the customer data key (CDK) rotates.

Note: The recommended (default) value is 90 days. You can set the rotation to any time between 30 and 365 days.
8. Click **Test connectivity** to verify that all fields are filled out correctly and that Pega Platform can connect to Key Vault and find your key.
9. Click **Save**.

- [Keystores](#)
- [Creating a keystore for application data encryption](#)

Configuring a HashiCorp Vault keystore

Configure a keystore by referencing an encryption key that is stored in a HashiCorp Vault.

Before you begin: You must [create a keystore](#) data instance in Pega Platform with **Keystore location** equal to *HashiCorp Vault* before you can configure the keystore.

1. If you have not yet defined your cryptographic key in HashiCorp Vault, log in to your HashiCorp Vault account and create an encryption key. The key should be accessible with the AppRole authentication method. For details, see your HashiCorp Vault documentation and [Creating a keystore for application data encryption](#).

2. Open a keystore from the navigation panel by clicking **Records Security Keystore** and selecting a HashiCorp Vault keystore from the instance list.
3. In the **AppRole Role ID** field, enter the *Role ID* for accessing Vault with the AppRole authentication method.
4. In the **AppRole Secret ID** field, enter the *Secret ID* for accessing Vault with the AppRole authentication method.
5. In the **Authentication service endpoint** field, enter the endpoint (URL) for accessing Vault with the AppRole authentication method.
6. In the **Encryption service endpoint** field, enter the endpoint (URL) for encryption that uses your Vault encryption key.
7. In the **Decryption service endpoint** field, enter the endpoint (URL) for decryption that uses your Vault encryption key.
8. In the **Customer data key rotation in days** field, enter the number of days after which the customer data key (CDK) rotates.
Note: The recommended (default) value is 90 days. You can set the rotation to any time between 30 and 365 days.
9. Click **Test connectivity** to verify that all fields are filled out correctly and that Pega Platform can connect to Key Vault and find your encryption key.
10. Click **Save**.

- [Keystores](#)
- [Creating a keystore for application data encryption](#)

Configuring a Google Cloud KMS keystore

Configure a keystore by referencing an encryption key that is stored in Google Cloud key management service (KMS).

Before you begin: You must [create a keystore](#) data instance in Pega Platform with **Keystore location** equal to *Google Cloud KMS* before you can configure the keystore.

1. If you have not yet defined your cryptographic key in Google Cloud KMS, create a Google project, a service account, and a keyring. For details, see your Google Cloud KMS documentation and [Creating a keystore for application data encryption](#).
 - a. Create a service account with a role equal to *Cloud KMS CryptoKey Encrypter/Decrypter*, and download the account credentials as a .json file.
 - b. Create a keyring and a symmetric key, and copy the key ID in Google resource name format.
2. Open a keystore from the navigation panel by clicking **Records Security Keystore** and selecting a Google Cloud KMS keystore from the instance list.
3. Click **Upload file**, and select the service account credentials file that you downloaded in step 1a.
4. In the **Customer master key ID** field, enter the key in Google resource name format that you copied in step 1b.
5. In the **Customer data key rotation in days** field, enter the number of days after which the customer data key (CDK) rotates.
Note: The recommended (default) value is 90 days. You can set the rotation to any time between 30 and 365 days.
6. Click **Test connectivity** to verify that all fields are filled out correctly and that Pega Platform can connect to Google Cloud KMS and find your key.
7. Click **Save**.

- [Keystores](#)
- [Creating a keystore for application data encryption](#)

Encrypting application data by using a custom key management service

You can encrypt application data by using an encryption key that is sourced from a custom key management service that is accessed from a data page. You source a key in this way when you use a key management service that is not one of the supported keystore platforms.

The master key in the custom KMS must be a 128-bit AES key.

1. Create an activity that accesses the custom KMS, configures a *CustomMasterKey* object, and loads the master key into *KeyStoreUtils*.
 - a. In the header of Dev Studio, click **Create Technical Activity**.
 - b. In the **Apply to (class)** field, enter *Data-Admin-Security-Keystore*, and then click **Create and open**.
 - c. In an activity step, enter **Method** equal to *Java*, and in the **Java Source** field, enter a code snippet similar to the example shown in step 2 of the sample activity *pzSampleGetCustomMasterKey*.
 - d. Click **Save**.
2. Create a data page that is loaded by the activity that you created in step 1.
 - a. In the header of Dev Studio, click **Create Data Model Data Page**.
 - b. In the **Apply to (class)** field, enter *Data-Admin-Security-Keystore*, and then click **Create and open**.
 - c. In the **Object type** field, enter *Data-Admin-Security-Keystore*.
 - d. In the **Mode** list, select **Read-Only**.
 - e. In the **Scope** list, select **Thread**.
 - f. In the **Source** list, select **Activity**.
 - g. In the **Activity name** field, enter the name of the activity that you created in step 1.
 - h. On the **Parameters** tab, select the **Pass current parameter page** check box.
 - i. On the **Load Management** tab, in the **Refresh strategy** section, select the **Reload once per interaction** check box.
 - j. Click **Save**.
3. Create a keystore that is loaded from the data page that you created in step 2.
 - a. In the header of Dev Studio, click **Create Security Keystore**.
 - b. In the **Keystore location** field, press the Down arrow key, and under **KEY MANAGEMENT SYSTEM (KMS) FOR APPLICATION DATA ENCRYPTION**, select **Custom - Source master key from other KMS using a data page**.
 - c. In the **Source data page** field, enter the name of the data page that you created in step 2.
 - d. Click **Save**.
4. Identify and activate the key for application data encryption.
 - a. In the header of Dev Studio, click **Configure System Settings Data Encryption**.
 - b. In the **Application data encryption** section, in the **Keystore** field, enter the name of the keystore that you created in step 3.
 - c. Click **Activate**.

- [Configuring the platform cipher](#)
- [Keystores](#)
- [Understanding data page definition](#)
- [Viewing test coverage reports](#)

Encrypting system data by using a custom key management service

Encrypt system data using an encryption key that is sourced from a Custom Key management service (KMS) that is accessed from a data page. For system data encryption, you can only use the Custom Key management service.

The master key in the custom KMS must be a 128-bit AES key.

1. Create an activity that accesses the custom KMS, configures a *CustomMasterKey* object, and loads the master key into *KeyStoreUtils*.
 - a. In the header of Dev Studio, click **Create Technical Activity**.
 - b. In the **Apply to (class)** field, enter *Data-Admin-Security-Keystore*, and then click **Create and open**.
 - c. In an activity step, in the **Method** field, enter *Java*, and in the **Java Source** field, enter a code snippet similar to the example in step 1 of the sample activity *pzSampleGetCustomMasterKey*.
Instead of the first Java command shown in the sample, your activity can use a *Connect-REST* step that accesses the master key from the custom KMS.
 - d. Click **Save**.
2. Create a data page that is loaded by the activity that you created in step 1.
 - a. In the header of Dev Studio, click **Create Data Model Data Page**.
 - b. In the **Apply to (class)** field, enter *Data-Admin-Security-Keystore*, and then click **Create and open**.
 - c. In the **Object type** field, enter *Data-Admin-Security-Keystore*.
 - d. In the **Mode** list, select **Read-Only**.
 - e. In the **Scope** list, select **Node**.
 - f. In the **Source** list, select **Activity**.
 - g. In the **Activity name** field, enter the name of the activity that you created in step 1.
 - h. On the **Parameters** tab, select the **Pass current parameter page** check box.
 - i. Click **Save**.
3. Create a keystore that is loaded from the data page that you created in step 2.
 - a. In the header of Dev Studio, click **Create Security Keystore**.
 - b. In the **Keystore location** field, press the Down arrow key, and under **KEY MANAGEMENT SYSTEM (KMS) FOR SYSTEM DATA ENCRYPTION**, select **Custom - Source master key from other KMS using a data page**.

- c. In the **Source data page** field, enter the name of the data page that you created in step 2.
 - d. Click **Save**.
4. Identify and activate the key for system data encryption.
 - a. In the header of Dev Studio, click **Configure System Settings Data Encryption**.
 - b. In the **System data encryption** section, in the **Keystore** field, enter the name of the keystore that you created in step 3.
 - c. Click **Activate**.

- [Configuring a keystore for a master key from a custom source](#)

Configure a keystore for a master encryption key that is stored in an external source, such as a key management service. Use keystores to encrypt, authenticate, and serve content over HTTPS. Master keys can encrypt data that is stored temporarily, for example, cached requestor IDs, or data that is persisted, such as data in a database.

- [Creating a data page for a master key from a custom source](#)

Encrypt system data using an encryption key that is sourced from a Custom Key management service (KMS) that is accessed from a data page. To configure a keystore for a master key by using a data page reference, create the data page, and then use the data page to retrieve a master key from an external source.

- [Creating a data page activity for a master key from a custom source](#)

Encrypt system data using an encryption key that is sourced from a Custom Key management service (KMS) that is accessed from a data page. To configure a keystore for a master key by using a data page reference, create the data page, and then use the data page to retrieve a master key from an external source.

- [Configuring the platform cipher](#)
- [Keystores](#)
- [Understanding data page definition](#)
- [Viewing test coverage reports](#)

Configuring a keystore for a master key from a custom source

Configure a keystore for a master encryption key that is stored in an external source, such as a key management service. Use keystores to encrypt, authenticate, and serve content over HTTPS. Master keys can encrypt data that is stored temporarily, for example, cached requestor IDs, or data that is persisted, such as data in a database.

Before you begin: You must create a keystore instance for your keystore file. For more information, see [Creating a keystore for application data encryption](#).

1. Open a keystore from the navigation pane by clicking **Records Security Keystore** and selecting a keystore for a master key from a custom source from the instance list.
2. In the **Keystore location** field, select **Source master key from other KMS using a data page**.
3. For the keystore to function properly, [Create a data page for a master key from a custom source](#).
4. In the **Source data page** field, enter or select the data page that you created in step 3.
5. Click **Save**.

- [Keystores](#)

Creating a data page for a master key from a custom source

Encrypt system data using an encryption key that is sourced from a Custom Key management service (KMS) that is accessed from a data page. To configure a keystore for a master key by using a data page reference, create the data page, and then use the data page to retrieve a master key from an external source.

Before you begin: You must create a keystore instance for your keystore file. For more information, see [Creating a keystore for application data encryption](#).

1. On the **Main** tab of the keystore, next to the **Source data page** field, click the **Add** icon.
2. Enter the **Label**, **Identifier**, and **Apply to** class for the data page, and then click **Create and open**.
3. In the **Structure** list, select **Page**.
4. In the **Object type** field, enter *Data-Admin-Security-Keystore*.
5. In the **Mode** list, select **Read-Only**.
6. In the **Scope** list, select **Node**.
7. In the **Data sources** section, in the **Source** list, select **Activity**.
8. If a Java code step that creates and populates the keystore data page does not exist, create an activity to do so. For more information, see [Creating a data page activity for a master key from a custom source](#).
9. In the **Activity name** field, enter the name of the activity that creates and populates the keystore data page.
10. Click **Parameters**.
11. Select the **Pass current parameter page** check box, and then click **Submit**.
12. Click **Save**.

- [Understanding data page definition](#)
- [Configuring a keystore for a master key from a custom source](#)

Creating a data page activity for a master key from a custom source

Encrypt system data using an encryption key that is sourced from a Custom Key management service (KMS) that is accessed from a data page. To configure a keystore for a master key by using a data page reference, create the data page, and then use the data page to retrieve a master key from an external source.

1. On the **Definition** tab of the data page that you are creating the activity for, next to the **Activity name** field, click the **Add** icon.
2. Enter a label and identifier, and accept the default properties in the **Apply to** class and *Data-Admin-Security-Keystore*.
3. Click **Create and open**.
4. In the **Method** field, enter *Java*.
5. Click the **Expand** icon, and then in the **Method Parameters** section, in the **Java Source** field, enter a code snippet appropriate to your use case:

Caution: Do not include both code snippets.

- To retrieve a master key directly from an external key management service, enter a code snippet similar to this one:

```
// Get master key from remote location
byte[] masterKey = null; // TODO: assign 16 byte master key
KeyStoreUtils keyUtil = pega.getKeyStoreUtils();
CustomMasterKey customMasterKey = keyUtil.getMasterKeyObject();
customMasterKey.setMasterKey(masterKey);
keyUtil.loadMasterKeyForSystemDataEncryption(customMasterKey);
```

- To retrieve REST APIs that encrypt and decrypt data keys from an external key management service, enter a code snippet similar to this one:

```
KeyStoreUtils keyUtil = pega.getKeyStoreUtils();
CustomMasterKey customerMasterKey = keyUtil.getMasterKeyObject();

//Enable remote encryption mode
customerMasterKey.setRemoteEncryptionMode(true);

//If CustomMasterKey object contains Encrypted CDK, then isEncryptedCustomDataKey() method returns true
if(customerMasterKey.isEncryptedCustomDataKey()){
    String ecdk =new String(customerMasterKey.getECDK());
    //oLog.infoForced("decrypting ECDK: ");

    byte[] cdk = null; // TODO: decrypt the above ecdk at remote KMS and assign it to the cdk variable
    customerMasterKey.setCDK(cdk);
    customerMasterKey.setECDK(null);
    customerMasterKey.setCustomDataKeyType(false);
}else{
```

```
//CustomMasterKey object contains plaintext CDK
byte[] cdk = customerMasterKey.getCDK();
if(cdk!= null){
    String str = new String(cdk);
    //oLog.infoForced("encrypting CDK.");

    byte[] ecdk = null;//TODO: encrypt the above cdk at remote KMS and assign it to the ecdk variable
    customerMasterKey.setECDK(ecdk);
    customerMasterKey.setCDK(null);
    customerMasterKey.setCustomDataKeyType(true);
}
}
keyUtil.loadMasterKeyForSystemDataEncryption(customerMasterKey);
```

6. Click **Save**.

- [Configuring a keystore for a master key from a custom source](#)
- [Creating a data page for a master key from a custom source](#)

Keystores

A keystore is a file that contains keys and certificates that you use for encryption, authentication, and serving content over HTTPS. In Pega Platform, you create a keystore data instance that points to a keystore file.

You can reference keystore data instances in the **Keystore** and **Truststore** fields for record types such as Web Services (WS) Security Profile data instances and REST connectors.

You source a keystore from a data page, a URL, a keystore management service, or an external file. The keystore source determines which of the standard formats are available: JKS, JWK, PKCS12, KEYTAB, or KEY.

Pega Platform includes several keystore examples, such as *BoxComTrust*.

- [Configuring a keystore by uploading a file](#)

To configure a keystore, you can upload a keystore file, such as a Java KeyStore (JKS) file.

- [Configuring a keystore by using a file reference](#)

To configure a keystore, you can reference a keystore file from a file location.

- [Configuring a keystore by using a URL reference](#)

To configure a keystore, you can reference a keystore file that contains public keys from a URL address.

- [Configuring a keystore by using a data page reference](#)

To configure a keystore, you can reference a keystore that is stored in a data page.

- [Creating a keystore for application data encryption](#)
- [Changing the default keystore caching settings](#)
- [Importing an X.509 certificate](#)
- [Encrypting application data by using a custom key management service](#)
- [Encrypting system data by using a custom key management service](#)

Configuring a keystore by uploading a file

To configure a keystore, you can upload a keystore file, such as a Java KeyStore (JKS) file.

Before you begin: You must [create a keystore](#) data instance in Pega Platform with **Keystore location** equal to *Upload file* before you can configure the keystore.

1. Open a keystore from the navigation panel by clicking **Records Security Keystore** and selecting a file upload keystore from the instance list.
2. Click **Upload file**.
3. Click **Choose File**, browse to the keystore file, and select it.
4. Click **Upload file**.
5. In the **Keystore type** field, enter the keystore file type: JKS, JWK, PKCS12, KEYTAB, or KEY.
6. In the **Keystore password** field, enter the password to the keystore file.
7. Click **Save**.

- [Keystores](#)
- [Creating a keystore for application data encryption](#)
- [Changing the default keystore caching settings](#)

Configuring a keystore by using a file reference

To configure a keystore, you can reference a keystore file from a file location.

Before you begin: You must [create a keystore](#) data instance in Pega Platform with **Keystore location** equal to *Reference to file* before you can configure the keystore.

1. Open a keystore from the navigation panel by clicking **Records Security Keystore** and selecting a file reference keystore from the instance list.
2. In the **Keystore file path** field, enter the path to the keystore file, for example: C:\keystores\JWTKeys.jks.
3. In the **Keystore type** list, select the keystore file type.
 - **Java KeyStore (JKS)**
 - **Public Key Cryptography Standards (PKCS12)**
 - **JSON Web Key (JWK)**
4. Select the password source for the keystore.
 - **Fixed password** – After you select **Fixed password**, in the **Keystore password** field, enter the password to the keystore file.
 - **Clipboard property** – After you select **Clipboard property**, in the **Clipboard password property** field, enter the name of the clipboard property that you want to get the password from, for example, *operatorID.pyPassword*.
 - **Function** – After you select **Function**, in the **Function** field, enter the rule utility function to be invoked to get the password. The function must comply with the following syntax: @(RuleSetName:RuleUtilityLibraryName).RuleUtilityFunctionName, for example, @(myApp:PasswordUtil).getPassword().
5. Select the refresh strategy for your keystore.
 - **Reload once per interaction** – Disable caching of the keystore file. Choose this if the keystore might change unexpectedly.
 - **Cache keystore** (default) – Cache the keystore, with the caching time set to 3600 seconds. You can override the caching time in the **Reload after** field.
6. Click **Save**.

- [Keystores](#)
- [Creating a keystore for application data encryption](#)
- [Function rule](#)

Configuring a keystore by using a URL reference

To configure a keystore, you can reference a keystore file that contains public keys from a URL address.

Before you begin: You must complete the following tasks before you can configure a keystore:

- [Create a keystore](#) data instance in Pega Platform with **Keystore location** equal to *Reference to URL*.

- Ensure that the URL is accessible. If the required HTTPS certificate is not in the cacerts keystore or an equivalent application server-level truststore, you must import the certificate into the cacerts keystore or equivalent application server-level truststore, and restart the server.
1. Open a keystore from the navigation panel by clicking **Records Security Keystore** and selecting a URL reference keystore from the instance list.
 2. In the **Keystore url** field, enter the URL address of the keystore file, for example: <https://www.pegasystems.com/prweb/PRRestService/keys/v1/jwt/JWTGenerationProfile>.
 3. In the **Keystore type** list, click the keystore type.
 - **JSON Web Key (JWK)**
 - **Microsoft authentication metadata**
 4. Select the refresh strategy for your keystore.
 - **Reload once per interaction** – Disable caching of the keystore file. Choose this if the keystore might change unexpectedly.
 - **Cache keystore** (default) – Cache the keystore, with the caching time set to 3600 seconds. You can override the caching time in the **Reload after** field.
 5. Click **Save**.
- [Keystores](#)
 - [Creating a keystore for application data encryption](#)

Configuring a keystore by using a data page reference

To configure a keystore, you can reference a keystore that is stored in a data page.

Before you begin: You must [create a keystore](#) data instance in Pega Platform with **Keystore location** equal to *Custom – Source master key from other KMS using a data page or Reference to data page* before you can configure the keystore.

1. Open a keystore from the navigation panel by clicking **Records Security Keystore** and selecting a data page keystore from the instance list.
2. Create a data page that contains keystore data, if one does not exist. For more information, see [Creating a data page for a keystore](#).
3. In the **Source data page** field, enter the name of the data page that contains keystore data, for example, *D_KeystoreJKS*.
4. In the **Keystore type** list, select the keystore file type.
 - **Java KeyStore (JKS)**
 - **Public Key Cryptography Standards (PKCS12)**
 - **JSON Web Key (JWK)**
5. Select the password source for the keystore.
 - **Fixed password** – After you select **Fixed password**, in the **Keystore password** field, enter the password to the keystore file.
 - **Clipboard property** – After you select **Clipboard property**, in the **Clipboard password property** field, enter the name of the clipboard property that you want to get the password from, for example, *operatorID.pyPassword*.
 - **Function** – After you select **Function**, in the **Function** field, enter the rule utility function to be invoked to get the password. The function must comply with the following syntax: *@(RuleSetName:RuleUtilityLibraryName).RuleUtilityFunctionName*, for example, *@(myApp:PasswordUtil).getPassword()*.
6. Click **Save**.

- [Creating a data page for a keystore](#)

To configure a keystore with a data page reference, you must create the data page. You can create the data page by clicking the Add icon on the keystore **Main** tab.

- [Creating a data page activity for a keystore](#)

To configure a keystore with a data page reference, you must create the activity that loads the keystore into the data page. You can create the activity by clicking the Add icon on the data page **Definition** tab.

- [Understanding data page definition](#)
- [Keystores](#)
- [Creating a keystore for application data encryption](#)
- [Changing the default keystore caching settings](#)
- [Function rule](#)

Creating a data page for a keystore

To configure a keystore with a data page reference, you must create the data page. You can create the data page by clicking the **Add** icon on the keystore **Main** tab.

1. On the keystore **Main** tab, next to the **Source data page** field, click the **Add** icon.
2. Enter the Label, Identifier, and Apply to class for the data page, and then click **Create and open**.
3. In the **Structure** list, select **Page**.
4. In the **Object type** field, enter *Data-Admin-Security-Keystore*.
5. In the **Mode** list, select **Read-Only**.
6. In the **Scope** list, select **Node**.
7. In the **Data sources** section, in the **Source** list, select **Activity**.
8. Create an activity with a Java code step that creates and populates the keystore data page, if such an activity does not exist. For more information, see [Creating a data page activity for a keystore](#).
9. Do the following steps to associate the activity with the data page.
 - a. In the **Activity name** field, enter the name of the activity that creates and populates the keystore data page.
 - b. Under the activity name, click **Parameters**.
 - c. Select **Pass current parameter page**.
 - d. Click **Submit**.
10. **Optional:** In the **Load Management** tab, **Refresh strategy** section, specify how long you want the keystore to remain in the cache.
11. Click **Save**.

- [Understanding data page definition](#)
- [Creating a data page activity for a keystore](#)
- [Configuring a keystore by using a data page reference](#)

Creating a data page activity for a keystore

To configure a keystore with a data page reference, you must create the activity that loads the keystore into the data page. You can create the activity by clicking the **Add** icon on the data page **Definition** tab.

1. On the data page **Definition** tab, next to the **Activity name** field, click the **Add** icon.
2. Enter the label and identifier, accept the default for Apply to class (*Data-Admin-Security-Keystore*), and then click **Create and open**.
3. In the **Method** field, enter *Java*.
4. Click the **Expand** icon, and then enter the **Java Source**.
 - To load a JKS or PKCS12 type keystore into the cache, in the Method Parameters section, in the **Java Source** field, enter a code snippet similar to the following example:

```
try {
    java.io.FileInputStream file = new java.io.FileInputStream("
        JKS_or_PKCS12_keystore_file_with_complete_path");
    KeyStoreUtils keystoreUtils = pega.getKeyStoreUtils();
    keystoreUtils.loadKeystore(file);
} catch (Exception ex) {
    oLog.error("Failed to load keystore into cache "+ex);
}
```

- To load a JWK type keystore into the cache, in the Method Parameters section, in the **Java Source** field, enter a code snippet similar to the following example:

```
try {
    java.net.URL url = new java.net.URL("

```

```

        JWK_keystore_URL_location");
        KeyStoreUtils keystoreUtils = pega.getKeyStoreUtils();
        keystoreUtils.loadKeystore(url.openConnection().getInputStream());
    } catch (Exception ex) {
        oLog.error("Failed to load keystore into cache "+ex);
    }
}

```

5. Click **Save**.

- [Creating an activity](#)
- [Configuring a keystore by using a data page reference](#)
- [Creating a data page for a keystore](#)

Changing the default keystore caching settings

You can change the values of the *KeyStoreCacheExpireTime* and *KeyStoreCacheSize* settings to control how often the keystore cache is refreshed and to restrict cache size. The lower the values, the less memory is used, but processing power is reduced.

When you create a keystore data instance, you select the reference from which the keystore entries are sourced.

- If you choose **Reference to file** or **Reference to URL** and select **Cache keystore**, Pega Platform caches the keystore entries for the time that you specify.
- If you choose **Upload file**, Pega Platform caches the keystore entries for the time set in the *KeyStoreCacheExpireTime* dynamic system setting.
- If you select **Reference to data page** and do not populate the Load Management tab, Refresh strategy area, Pega Platform caches the keystore entries for the time set in the *KeyStoreCacheExpireTime* dynamic system setting.

To change the values of the *KeyStoreCacheExpireTime* and *KeyStoreCacheSize* settings, do the following steps.

1. In the Dev Studio search field, enter the name of the dynamic system setting that you want to modify and press Enter.
2. In the list of results, click the dynamic system setting that you want to modify.
 - *KeyStoreCacheExpireTime* – The caching time of the keystore entries (in seconds). The default value is 3600.
 - *KeyStoreCacheSize* – The number of keystore entries that can be held in cache. The default value is 150. Set this value to no less than 50.
3. Change the value in the **Value** field.
4. Click **Save**.
5. Repeat steps 1 through 4 for each dynamic system setting that you want to modify.

- [Keystores](#)
- [Creating a keystore for application data encryption](#)

Creating a keystore instance for an external key management service

You can encrypt application and system data in Pega Platform™ by using either the platform cipher or a cipher that is stored within an external key management service (KMS). Use an external KMS to control the ownership, creation, and rotation of your master key.

To configure Pega Platform to use an external KMS, complete the following tasks.

1. In the external key management service, create the master key.
2. In Pega Platform, create an instance of the keystore rule to access the external KMS.
For step-by-step examples for various key management services for tasks 1 and 2, see:
 - [Configuring an Amazon Key Management Service \(KMS\) key rotation](#)
 - [Configuring a Microsoft Azure Key Vault keystore](#)
 - [Configuring a HashiCorp Vault keystore](#)
 - [Configuring a Google Cloud KMS keystore](#)
3. In Pega Platform, configure and activate the platform cipher to reference the keystore instance.
For more information, see [Configuring the platform cipher](#).
4. In Pega Platform, identify the classes and properties to encrypt.

For more information, see:

- [Creating an access control policy](#)
- [Encrypting the storage stream \(BLOB\)](#)

Importing an X.509 certificate

You can import X.509 certificates that are defined in keystore instances of type JKS or PKCS12. They become active without your having to restart the server.

A typical application server is deployed with a truststore file, which contains the trusted certificates. To add a certificate, you use the `keytool` utility to update the `cacerts.jks` file or the equivalent server-level truststore file, and then you restart the server.

Pega Platform includes a platform truststore to which you can import X.509 certificates. When a certificate needs to be used for transport layer security, Pega Platform looks for the certificate in the platform truststore, and then in the application server-level (JVM) truststore. You can add certificates to the platform truststore without having to restart the server, which is useful when TLS certificates are changed for reasons such as key expiration.

The platform truststore is used for rules that reference a truststore, which include certificates used for transport layer security in connectors and integration services, keystores, authentication services, OAuth 2.0 providers, and web service policies.

To view and update the platform truststore, run the activities described below using Dev Studio.

1. To view the contents of the platform truststore, run the *pyInstanceList* report of the *Data-Admin-Security-Certificate* class.
2. To add, update, and delete certificates in the platform truststore, run the appropriate activities in the *Data-Admin-Security-Certificate* class. For more information, see [Managing X.509 certificates](#).

- [Managing X.509 certificates](#)

Beginning with Pega Platform 8.4, you can import your X.509 certificates directly into the Pega Platform truststore.

- [Enabling encrypted communication between nodes](#)

For on-premises deployments, enable encryption for communication among nodes to prevent unauthorized access to the data that is transported across node clusters.

- [Keystores](#)
- [Connect EJB form - Completing the Service tab](#)
- [Creating a Connect REST rule](#)
- [Connect EJB form - Completing the Service tab](#)

Managing X.509 certificates

Beginning with Pega Platform 8.4, you can import your X.509 certificates directly into the Pega Platform truststore.

Before you begin: The operator must have the Security Administrator role to use the `<?pxCanManageCertificates>` privilege.

Before you can begin managing x.509 certificates in the Pega Platform truststore, you must perform the following tasks:

- Obtain an x.509 certificate.
- Make the certificate available as a file or URL.
- Create a keystore file instance to import the certificate. For more information about importing the certificate by creating a keystore, see [Creating a keystore](#).

- Open the certificate management activities. For more information about running an activity rule, see [Introduction to activities](#).
- Run the certificate management activities as needed.

When your application makes a secure outbound connection using HTTPS, the external host presents a certificate for secure connection authentication. Your application checks this certificate against the certificates in the Pega Platform truststore. If the certificate is not present, the external host is not authenticated, and an exception is thrown. The Pega Platform truststore holds both the public keys and certificates of your trusted external systems.

At runtime, Pega Platform looks for certificates to load in the following order: first from the Pega Platform truststore, then from the application server truststore, and finally from the JVM truststore. After loading a certificate, Pega Platform syncs updates to the certificate in real time and presents them to applications for use with secure outbound connections.

Pega Platform features that require x.509 certificates include the functions described below.

- Authentication services that import identity metadata exposed over an HTTPS URL
- Connectors that access external REST API over HTTPS

If you require use of certificates for your applications outbound connections that do not use HTTPS, speak to your regional Pega support team.

1. In the left navigation pane of Dev Studio, click **Records Technical Activity**.
2. In the **Applies To** column, click the search icon (▼).
3. In the **Search Text** field, enter Data-Admin-Security-Certificate.

The Record page displays the certificate management activities.

- The activity **Add certificates to Platform Truststore from Pega Keystore** (*pxAddCertificatesToPlatformTruststore*) adds certificates from a Pega keystore rule into the Pega Platform truststore. The activity contains the following parameters:
 - **keystoreName**: String. The Java KeyStore (JKS) or Public-Key Cryptography Standards (PKCS212) instance from which to import the certificate.
 - **overwriteDuplicates**: Boolean. When enabled, this activity overwrites the existing Platform truststore certificate with a new certificate of the same alias. When disabled, this activity excludes duplicate certificates from the import.
 - **checkExpiryDate**: Boolean. Select this check box for this activity to exclude adding expired certificates to the Platform truststore.
 - The activity **Change Certificate Status** (*pxChangeCertificateStatus*) changes the status of a certificate to Active or Inactive. The activity contains the following parameters:
 - **certificateAliasName**: String. The alias name given to the certificate of which you want to change the status.
 - **certificateStatus**: String. Enter Active or Inactive to apply the respective status to the certificate.
 - The activity **Delete Certificate** (*pxDeleteCertificate*) removes the specified certificate from the Platform truststore. The activity contains the following parameter:
 - **certificateAliasName**: String. The alias name of the certificate to remove from the Platform truststore.
4. Select an activity to add certificates, change certificate status, or remove certificates from the Platform truststore.
 5. Run the respective certificate activity by clicking **Actions Run**.
 6. Complete each field that is defined in the parameters for the activity that you ran.

- [Importing an X.509 certificate](#)
- [Configuring login authentication with basic credentials](#)

Enabling encrypted communication between nodes

For on-premises deployments, enable encryption for communication among nodes to prevent unauthorized access to the data that is transported across node clusters.

Enabling encryption between nodes secures the data that is transferred across nodes so that it cannot be accessed by unauthorized hosts. Node encryption is useful when compliance rules require all intracluster communication to be encrypted, such as when working under HIPAA regulations.

Enabling encryption involves the following high-level steps:

1. [Creating the keystore.jks and truststore.jks files](#)

Enabling encryption between nodes secures the data that is transferred across nodes so that it cannot be accessed by unauthorized hosts. Create a keystore.jks for the private key and the associated certificate or certificate chain.

2. [Uploading the keystore and truststore files](#)

Enabling encryption between nodes secures the data that is transferred across nodes so that it cannot be accessed by unauthorized hosts. Upload a keystore.jks for the private key and the associated certificate or certificate chain.

3. [Enabling encryption](#)

Enabling encryption between nodes secures the data that is transferred across nodes so that it cannot be accessed by unauthorized hosts. To enable encryption between nodes, use dynamic system settings.

4. [Restarting nodes](#)

Enabling encryption between nodes secures the data that is transferred across nodes so that it cannot be accessed by unauthorized hosts. To complete node encryption between nodes, restart nodes.

Creating the keystore.jks and truststore.jks files

Enabling encryption between nodes secures the data that is transferred across nodes so that it cannot be accessed by unauthorized hosts. Create a keystore.jks for the private key and the associated certificate or certificate chain.

Before you begin: If you already have a self-signed certificate (SSL), go to step 2.

1. Create a self-signed certificate by entering the following command:

```
keytool -genkey -alias <alias> -keyalg RSA -keysize <enter size> -keystore <password>
-keystore cluster-keystore.jks -storepass <password>
```

2. Export the self-signed certificate so that it can be added to the truststore as the trusted certificate by entering the following command:

```
keytool -export -alias <alias> -file <certificate> -keystore cluster-keystore.jks
-storepass <password>
```

3. Create the cluster-truststore.jks file by entering the following command:

```
keytool -import -alias <alias> -file client.cer -keystore cluster-truststore.jks
-storepass <password>
```

What to do next: [Uploading the keystore and truststore files](#).

Uploading the keystore and truststore files

Enabling encryption between nodes secures the data that is transferred across nodes so that it cannot be accessed by unauthorized hosts. Upload a keystore.jks for the private key and the associated certificate or certificate chain.

By default, the file names are **cluster-keystore.jks** and **cluster-truststore.jks**. Change the file names by modifying the `cluster/encryption/keystoreName` and `cluster/encryption/truststoreName` prconfign.xml file settings, respectively. Upload the keystore and truststore files into the Pega Platform as instances of *Data-Admin-Security-Keystore*.

1. In Dev Studio, click + **Create > Security > Keystore**.
2. Enter a short description.
3. In the **Keystore** field, enter the keystore name.

4. Click **Create and open**.
5. Enter the keystore file type, either JKS or PKCS12.
6. Enter the keystore password.
7. Click **Upload file** and upload the actual keystore file, not the certificate.
8. Click **Save**.
9. Repeat this procedure to upload the truststore file.

What to do next: [Enabling encryption](#).

Enabling encryption

Enabling encryption between nodes secures the data that is transferred across nodes so that it cannot be accessed by unauthorized hosts. To enable encryption between nodes, use dynamic system settings.

Enable encryption by using the `cluster/encryption/enabled` dynamic system setting.

1. In Dev Studio click **Create > SysAdmin > Dynamic System Settings**.
2. Enter a short description.
3. Enter Pega-Engine in the **Owning Ruleset** field.
4. Enter `cluster/encryption/enabled` in the **Setting Purpose** field.
5. Click **Create and open**.
6. On the **Settings** tab, enter `true` in the **Value** field.
7. Click **Save**.

You can also enable encryption in the `prconfig.xml` file by setting `cluster/encryption/enabled` to `true`. The setting in the `prconfig.xml` file takes precedence over the Dynamic System Setting unless there is no entry in the `prconfig.xml` file.

```
<env name="cluster/encryption/enabled" value="true" />
```

What to do next: [Restarting nodes](#).

Restarting nodes

Enabling encryption between nodes secures the data that is transferred across nodes so that it cannot be accessed by unauthorized hosts. To complete node encryption between nodes, restart nodes.

After you complete all the preceding tasks, restart all nodes in the cluster.

1. Restart all nodes in the cluster.
If either the `keystore.jks` or the `truststore.jks` certificate is not available in Pega Platform, then Pega Platform fails to start. Follow the instructions in the error message to resolve the problem.

Token profile data instance

Create a JSON Web Token (JWT) profile data instance to confirm a user's identity between two different processes. You can configure Pega Platform to act as both a producer or a consumer of JWT.

JWT is an open standard that defines a compact and self-contained method to securely exchange information between different parties as a JSON object. For example, the token can contain information about a user that another party can use to validate the identity of the user.

The Pega Platform server can act as a trusted third-party that generates or validates a JWT after you complete fill out the token profile form and provide the required information.

Token profiles

Use a JWT to securely exchange information between two different parties. For example, a JWT can carry information about a user that another party can use to authenticate the identity of the user.

Token profiles have two purposes:

- generation
- processing

Generation token profiles

A generation token profile specifies how Pega Platform generates a given JWT.

The generation JWT profile data instance consists of the following pieces:

- one or more headers
- claims and a security configuration specifying no security
- a signed token
- an encrypted token
- or a combination that uses both signing and encryption

You can use the profile to specify the following information:

- token lifetime
- timeout option
- if the profile should include a processing JWT profile

You can use JWTs to exchange information securely between Pega Platform and another party. The following table lists common uses for JWTs in Pega Platform:

JWT uses in Pega Platform	JWT
Authentication	Holds user information that can be used by another party to authenticate the identity of the user presenting the token.

By default Pega Platform automatically adds the following claims to the JWT header when it generates the token:

JTW header claims	Description
alg	The configured JWS algorithm, if you added one in the Security section, on the Generation tab.
typ	The type is always JWT.
kid	The Key ID is a unique ID that the JWT run time process generates for each token generated.
crit	Headers that are marked as critical on the Generation tab.

When you create a generation token profile, Pega Platform supports the following fields:

Security Configuration	Support Fields
Signature	Signature type, Signature algorithm, Keystore, Alias, and Password.
Encryption	Encryption type, Key encryption algorithm, Password, Truststore, Alias, and Content encryption type.
Signature & Encryption	For JSON Web Signature (JWS): Signature type, Signature algorithm, Keystore, Alias, and Password Encryption type: Encryption type, Key encryption algorithm, Password, Truststore, Alias, and Content encryption type
None	N/A

To create a Generation JSON Web Token, see [Creating a generation JSON Web Token profile](#).

Processing token profiles

A processing token profile specifies how Pega Platform validates the signature and decrypts each JSON Web Token that it receives.

The processing JWT profile data instance consists the following:

- one or more claims validations
- claims mappings
- security configuration that specifies one of the following:
 - signing
 - decryption
 - combination that uses both signing and decryption
 - no security

You can use JWTs to exchange information securely between Pega Platform and another party. The following table lists common uses for JWTs in Pega Platform:

JWT uses in Pega Platform	JWT
Authentication	Holds user information that can be used by another party to authenticate the identity of the user presenting the token

When you create a processing token profile, Pega Platform supports the following fields:

- **Signature configuration**
- **Signature type**
- **Truststore**

To create a Processing JSON Web Token, see [Creating a processing JSON Web Token profile](#).

Access token profile data instance

To access the token profile data instance in the navigation panel Dev Studio, click **Records Security Token Profile**.

- [Creating a token profile](#)

Use a JSON Web Token (JWT) to exchange information securely between two different parties. For example, a JWT can carry information about a user that can be used by another party to authenticate the identity of the user.

- [Creating a generation JSON Web Token profile](#)
- [Creating a processing JSON Web Token profile](#)
- [Previewing the JSON Web Token profile](#)

Creating a token profile

Use a JSON Web Token (JWT) to exchange information securely between two different parties. For example, a JWT can carry information about a user that can be used by another party to authenticate the identity of the user.

There are two purposes for token profiles: Generation and Processing. Each is a separate token profile.

- To create a Generation JSON Web Token, see [Creating a generation JSON Web Token profile](#).
- To create a Processing JSON Web Token, see [Creating a processing JSON Web Token profile](#)

- [Creating a generation JSON Web Token profile](#)

Create a generation JSON Web Token (JWT) profile to specify how Pega Platform generates a given JWT.

- [Creating a processing JSON Web Token profile](#)

Create a processing JSON Web Token (JWT) profile to specify how Pega Platform validates and decrypts each JSON Web Token it receives.

- [Token profile data instance](#)

Creating a generation JSON Web Token profile

Create a generation JSON Web Token (JWT) profile to specify how Pega Platform generates a given JWT.

Before you begin: Configure cryptographic components in instances of an existing keystore. Pega Platform can then generate the JWT from this profile using these components to calculate the token's signature as well as encrypt the token. For details, see [Creating a keystore for application data encryption](#).

To generate a JWT, Pega Platform processes this generation JWT profile with the `pxGenerateJWT` activity. The generation JWT profile data instance consists of one or more headers, claims and a security configuration specifying no security, a signed token, an encrypted token, or a combination that uses both signing and encryption. The profile can specify a token lifetime, and timeout option, and if the profile should include a processing JWT profile.

You can use JWTs to exchange information securely between Pega Platform and another party and to encrypt the data to be transmitted. Common uses for JWTs in Pega Platform include:

JWT uses in Pega Platform	The JWT
Authentication	holds user information that can be used by another party to authenticate the identity of the user presenting the token.
Session management	ensures the secure exchange of information during user authenticated sessions.

By default Pega Platform automatically adds the following parameters to the JWT header when it generates the token:

- `alg` – The configured JWS algorithm, if you added one, in the **Security** section, on the **Generation** tab.

- typ – The type is always JWT.
- kid – The Key ID is a unique ID generated by the JWT run time process for each token generated.
- crit – Headers that are marked as critical on the **Generation** tab.

After you create a generation JWT profile, you can view the details of the token that the profile will generate by clicking **Preview**.

1. In the navigation panel, click **Records Security Token Profile**.
2. Click **Create**.
3. In the **Token Type** field, select **JSON Web Token**.
4. In the **Purpose** field, select **Generation** to create a token profile for generating a new JWT.
5. Enter a token profile name and short description.
6. Click **Create and open**.
7. In the **Headers** section, you can supplement the default JWT header parameters with additional parameters by clicking the **Add** icon and completing the **Name**, **Description**, **Map from** fields.
 - If you selected **Constant** in the **Map from** field, then enter a text string in the **Map from key** field.
 - If you selected **Clipboard** in the **Map from** field, then enter a property name on the clipboard in the **Map from key** field.
8. In the **Claims** section, add a set of **Registered claims** to the payload of the token.
 - a. Click the **Add** icon.
 - b. In the **Name** field, press the Down arrow key, and select one of the available claims:
 - Issuer (iss) – Specifies the principal that issued the JWT.
 - Audience (aud) – Specifies the intended recipients of the JWT.
 - Subject (sub) – Specifies the principal that is the subject of the JWT. **Note:** Each claim must have a different name.
 - c. In the **Map from** field, select whether to map the claim value to text or to a clipboard property:
 - If you selected **Constant** in the **Map from** field, then enter a text string in the **Map from key** field.
 - If you selected **Clipboard** in the **Map from** field, then enter a property name on the clipboard in the **Map from key** field.
9. In the **Claims** section, add a set of **Custom claims** to the payload of the token to share information between parties that agree to use these claims.
 - a. Click the **Add** icon.
 - b. Complete the **Name** and **Description** fields.
 - c. In the **Map from** field, select whether to map the claim value to text or to a clipboard property:
 - If you selected **Constant** in the **Map from** field, then enter a text string in the **Map from key** field.
 - If you selected **Clipboard** in the **Map from** field, then enter a property name on the clipboard in the **Map from key** field.
10. In the **Security configuration** field, specify whether to generate a signed JWT for a signature, or JSON Web Encryption (JWE) token for content encryption.
 - **Signature** – Configure the JWT signature for validation purposes by a recipient by completing step 11.
 - **Encryption** – Configure the JWT encryption in order to obscure its content by completing step 12.
 - **Signature & encryption** – Use this token for both signature and data encryption configurations by completing steps 11 and 12.
 - **None** – Disable security by skipping steps 11 and 12.
11. For JWT signature configuration, in the **JSON Web Signature (JWS)** section, complete the following steps.
 - a. In the **Signature type** list, select either **Asymmetric** or **Symmetric**.
 - b. In the **Signature algorithm** list, press the Down arrow key, and select an algorithm for signing the token.
 - c. In the **Keystore** field, press the Down arrow key, and select an existing keystore name.
 - d. In the **Alias** field, press the Down arrow key, and select the alias name of the private key in this keystore used to sign the JWT generated from this JWT profile.
 - e. In the **Password** field, enter the keystore password.

Note: After you save a token rule for decrypting tokens that use asymmetric encryption, the **JSON web key set URI** field exposes the public key endpoint for retrieving the public key.
 - f. **For Asymmetric signatures only:** To suppress the key ID header, select the **Suppress generation of Key ID (kid) header** check box.
12. For JWE content encryption, in the **JSON Web Encryption (JWE)** section, select either **Asymmetric** or **Symmetric**

Choices	Actions
For asymmetric encryption	a. In the Key Encryption algorithm field, press the Down arrow key, and select an algorithm for encrypting the secret key. b. In the Truststore field, press the Down arrow key, and select the truststore name. c. In the Alias field, enter the alias name of the private key in this keystore used to encrypt the JWT generated from this JWT profile. d. In the Content encryption algorithm field, press the Down arrow key, and select an algorithm for encrypting the content. e. To suppress the key ID header, select the Suppress generation of Key ID (kid) header check box.
For symmetric encryption	a. In the Key Encryption algorithm field, press the Down arrow key, and select an algorithm for encrypting the secret key. b. In the Keystore field, press the Down arrow key, and select an existing keystore name. c. In the Alias field, press the Down arrow key, and select the alias name of the private key in this keystore used to encrypt the JWT generated from this JWT profile. d. In the Password field, enter the keystore password. e. In the Content encryption algorithm field, press the Down arrow key, and select an algorithm for encrypting the content.

13. In the **Token lifetime** section, complete the following steps.
 - a. In the **Valid till (in seconds)** field, specify the number of seconds that the token remains valid after the token is issued. The default value is 900 seconds.
 - b. In the **Processed not before (in seconds)** field, enter the time before which the receiver of the JWT cannot use the token. The default coded in value for this field is 0 seconds.
14. Click **Save**.
15. If you want to add a processing profile to this generation JWT profile, in the **Advanced** section, select **Process generated token**.
16. Click the **Processing** tab.

Based on the token generation profile configuration, some sections and fields autofill.

To complete the remaining fields, go to [Processing a JSON Web Token](#), and begin at step 7.

- [Configuring activities to generate a JSON Web Token](#)
- [Previewing the JSON Web Token profile](#)
- [Creating a processing JSON Web Token profile](#)
- [Token profile data instance](#)

Configuring activities to generate a JSON Web Token

To generate a JSON Web Token in Pega Platform, you need to call the *pxGenerateJWT* activity. The activity can then generate the token.

1. Create a custom activity that you will use to generate the JSON web token.
2. In your activity, add a step to call the activity *pxGenerateJWT*, and pass these parameters.
 - **profileName** - Enter the token profile name that you created.
 - **jwtPageName** - Enter the JWT clipboard page name.
 - **inputPageName** - Enter the page name that contains input data used in a custom claim.
 - **payloadPageName** - Enter the page name from which you want to map properties.
 - **payloadCustomKeyName** - Enter the node name where the data is generated. The default name is *pyData*.
 - **excludedPropertiesJWTList** - Enter a comma-separated list of properties to exclude from the node.
 - **outputPegaName** - Enter the generated JWT.
3. Click **Save**.
4. Run your custom activity to generate the token.

- [Token profile data instance](#)
- [Creating a token profile](#)
- [Calling another activity](#)

Previewing the JSON Web Token profile

Preview the JSON Web Token that is generated from the token profile instance to view all the claims and headers in the token. You can also view the values that are mapped from the clipboard and resolved at run time. The preview function is useful for debugging. For example, if a token is unexpectedly rejected, you can view the entire content of the token to determine the cause.

If the token profile is configured for generating a signed JWT, then the signature generated using the configured JSON Web Algorithm can be viewed. Also, in the Encoded section, you can see the encoded compact JWT.

1. In the navigation panel, click **Records Security Token Profile**.
2. Click the token profile that you want to preview.
3. On the **Generation** tab, click **Preview**.

- [Token profile data instance](#)
- [Creating a token profile](#)

Creating a processing JSON Web Token profile

Create a processing JSON Web Token (JWT) profile to specify how Pega Platform validates and decrypts each JSON Web Token it receives.

Before you begin: Configure cryptographic components in instances of an existing keystore. Pega Platform can then use the keystore components to validate the token's signature as well as decrypt the token when Pega Platform receives it from an external source. For details, see [Creating a keystore for application data encryption](#).

The processing JWT profile data instance consists of one or more claims validations, claims mappings, and a security configuration specifying no security, signing, decryption, or a combination that uses both signing and decryption. The profile can also specify a token lifetime and timeout option.

You can use JWTs to exchange information securely between Pega Platform and another party and to decrypt the data to be transmitted. Common uses for JWTs in Pega Platform include:

JWT uses in Pega Platform	The JWT
Authentication	holds user information that can be used by another party to authenticate the identity of the user presenting the token
Session management	ensures the secure exchange of information during user authenticated sessions.

1. In the navigation panel, click **Records Security Token Profile**.
2. Click **Create**.
3. In the **Token Type** field, select JSON Web Token.
4. In the **Purpose** field, select **Processing**, to create a token profile for processing JWTs.
5. Enter a token name and short description.
6. Click **Create and open**.
7. On the **Processing** tab, in the **Claims validation** section, click the **Add** icon to configure validations that are applied to the expected claims.
 - a. In the **Name** field, press the Down arrow key and select one of the available claims:
 - Issuer (iss) – Specifies the principal that issued the JWT.
 - Audience (aud) – Specifies the intended recipients of the JWT.
 - Subject (sub) – Specifies the principal that is the subject of the JWT.
 - b. In the **Map from** field, select **Constant** or **Clipboard** to indicated if the claim is mapped from a text string you enter or a property name on the clipboard.
 - c. In the **Compare with** field, enter the constant or clipboard property with which the claim should be compared.
8. In the **Claims mapping** section, click the **Add** icon to configure mapping for custom claims to clipboard properties.
 - a. In the **Claim name** field, enter the custom claim name.
 - b. In the **Property** field, press the Down arrow key, and select the property name to which this custom claim is mapped.
9. In the **Security Security configuration** field, specify whether to process JSON Web Encryption (JWE) token for content decryption, or a signed JWT for a signature.
 - **Signature verification** – Specify how to use signature verification of the token by completing step 10.
 - **Decryption** – Specify how to decrypt the token by completing step 11.
 - **Decryption and signature verification** – Specify details for processing a token that has been signed and encrypted by completing step 10 and step 11.
 - **None** – Disable security by skipping steps 10 and 11.
10. For JWT signature verification, in the **JSON Web Signature (JWS)** section, do the following steps. If you previously entered values on the **Generation** tab, some of these fields might be already populated.

Choices	Actions
For asymmetric signature verification	In the Truststore field, press the Down arrow key, and select the Truststore that is used for the JWT signature validation.
For symmetric signature verification	1. In the Keystore field, press the Down arrow key, and select an existing keystore name. 2. In the Alias field, press the Down arrow key, and select the alias name of the private key in this keystore used to decrypt the JWT. 3. In the Password field, enter the keystore password

11. For JWE content decryption, in the **JSON Web Decryption (JWE)** section, do the following steps. If you previously entered values on the **Generation** tab, some of these fields might be already populated.
 - a. In the **Encryption type** list, select either **Asymmetric** or **Symmetric**.
 - b. In the **Key encryption algorithm** list, select an algorithm for decrypting the metadata.
 - c. In the **Keystore** field, press the Down arrow key, and select the keystore name.
 - d. In the **Alias** field, press the Down arrow key and select the alias name of the private key in this keystore used to decrypt the JWT.
 - e. In the **Password** field, enter the keystore password.

Note: After you save a token rule for decrypting tokens that use asymmetric encryption, the **JSON web key set URI** field exposes the public key endpoint for retrieving the public key.
12. In the **Token lifetime** section, in the **Allowed time to account for clock skew (in seconds)** field, enter the time difference between two different servers that are out of sync.
13. In the **Advanced mapping** section, in the **Header as JSON string** field, enter the clipboard property that the decoded header is assigned to and that can be used for any user-defined JWT processing logic.
14. In the **Payload as JSON string** field, enter the clipboard property the decoded payload is assigned to and can be used for any user-defined JWT processing logic.
15. Click **Save**.

- [Configuring an activity to process a JSON Web Token](#)
- [Token profile data instance](#)
- [Keystores](#)

Configuring an activity to process a JSON Web Token

To process a JSON Web Token, you need to call the `pxProcessJWT` activity. The activity can then process the token.

1. Create a custom activity that you will use to process the JSON web token.
2. In your activity, add a step to call the activity `pxProcessJWT`, and pass these parameters.
 - **profileName** - Enter the token profile name that you created.
 - **jwtPageName** - Enter the JWT clipboard page name.
 - **inputPageName** - Enter the page name that contains input data used in a custom claim.
 - **payloadPageName** - Enter the page name from which you want to map properties.
 - **payloadCustomKeyName** - Enter the node name where the data is generated. The default name is `pyData`.
 - **excludedPropertiesJWTList** - Enter a comma-separated list of properties to exclude from the node.
 - **outputPegaName** - Enter the generated JWT.
3. Click **Save**.
4. Run your custom activity to process the token.

- [Token profile data instance](#)
- [Creating a processing JSON Web Token profile](#)
- [Calling another activity](#)

Securing an activity

You can better protect your application by limiting how an activity can be run and who may run it by configuring activity-specific access control.

In most cases, the standard [Authentication](#) in Pega Platform is sufficient to secure your application.

To alter the Activity-specific access control settings, use the **Restrict access** section of the **Security** tab, as shown in the following figure.

The Security tab of an activity

Restrict access

☐ Allow direct invocation from the client or a service

☒ Require authentication to run

	Privilege Class	Privilege Name
1	@baseclass	OpenDeveloperForm

+

Three security fields are available in the **Restrict access** section:

- **Allow invocation from browser**
- **Require authentication to run**
- **Privileges** (Privilege Class and Privilege Name)

Allow invocation from browser

Select the **Allow invocation from browser** check box when your activity meets one or more of the following criteria:

- You need to be able to call this activity directly from JavaScript code in a **non-autogenerated** Section or **non-autogenerated** Control or *Text File* rule.
- You need to be able to call this activity from a *URL Mappings* rule.
- You need to use the deprecated List Views or Summary Views and you need to call an Activity in the **Content** field. To access these deprecated views:
 - In the panel of Dev Studio, click **Record Deprecated: Summary view Format tab**. Then, right-click **Smart info settings**, and then click the **Content** field.

Allow invocation from browser is not selected by default. If you select **Allow invocation from browser** check box, you must add a privilege to the activity.

For more information, see [Setting a privilege to secure an activity](#).

One use case where you should not select the **Allow invocation from browser** check box is when you are configuring an external system to call this activity by URL. Instead, create a Service REST rule.

- For more information, see [Creating a REST integration](#).

Note: Beginning with Pega Platform 7.1.5, there is no longer a requirement to select the **Allow invocation from browser** check box for activities called by Service rules.

Require authentication to run

Require authentication to run is selected by default. In most cases, do not change this selection.

Clear the **Require authentication to run** check box to allow guest users to run this activity if they meet other security and access criteria. Guest users — unauthenticated requestors — typically have access to rules in the RuleSets in the *PRPC:Unauthenticated* access group. In many instances, clients overwrite this with their own unauthenticated authentication group. Therefore, your unauthenticated access group might have another name, similar to *<application name>:Unauthenticated*. **Note:** A session remains unauthenticated until the user signs in by using credentials, unless you are using an anonymous authentication service. For more information, see:

- [Configuring login authentication for an anonymous operator](#).
- [Activating your anonymous authentication service](#).

Note: If you clear the **Require authentication to run** check box and select **Allow invocation from browser**, any person with access to your server can execute this activity. The default [Authorization](#) model stops unauthenticated users from having read and write access, so it is important to consider if your activities do something other than read and write.

Privileges: Privilege Name and Privilege Class

Privilege inheritance simplifies the process of defining privileges and access settings that are relevant in multiple classes.

Use privileges to differentiate the capabilities of different groups of users within your application. The privilege form defines only the existence (name and *Applies To* class) of a new privilege. It contains no other information and conveys no capabilities by itself.

Privileges has two fields that you can enter information into, the **Privilege Name** field and **Privilege Class** field.

A privilege is an application-specific access control element associated with a class and access role. A privilege rule only names a privilege; it does not convey the privilege to anyone. A privilege has two parts: the privilege class and the privilege name.

To have access to a privilege, a requestor session must "hold," in the access role list, at least one of the access roles that grant access to the privilege. The association between access roles and privileges is defined by instances of the Rule-Access-Role-Obj rule type.

Using privilege rules in an application is optional, but they can offer finer control over access than using access roles alone.

The **Privilege Class** field contains the auto-generated rule classes that are associated with the privilege.

The **Privilege Name** field contains the auto-generated list of privilege names.

Standard privileges

The following tables list the most common end-user and administrative privileges.

Standard end-user privileges are shown in the following table.

Privilege	Privilege class	Role names	Comments
AllFlows	@baseclass	<i>PegaRULES:SysAdm4Pega</i> <i>RULES:WorkMgr4</i> <i>PegaRULES:User4</i>	Use this privilege during case processing. You would not normally use this privilege because most users already have it. If, however, you created an Access Group and Role only used for reporting purposes, users with that access group should not do any flow processing. You can add <i>AllFlows</i> to your activities to ensure that reporting users cannot run those activities.
UpdateLimitedForm	@baseclass	<i>PegaRULES:WorkMgr4</i>	Use this privilege for rules visible for non-Dev Studio users. Rule delegation uses this privilege.
pxViewLimitedForm	@baseclass	<i>PegaRULES:ViewerCollaboratorVariousDecisionManager Roles</i>	Use this for read-only rule forms, not in Dev Studio.
Perform	Work-	<i>PegaRULES:SysAdm4Pega</i> <i>RULES:WorkMgr4</i>	Restricts the ability to take action on an assignment. This privilege allows a user to perform an assignment on another individual's list (including workbaskets). Without this privilege, you can take action only on cases assigned to you.
WorkUpdate	Work-	<i>PegaRULES:SysAdm4Pega</i> <i>RULES:WorkMgr4</i>	None
WorkReopen	Work-	<i>PegaRULES:SysAdm4Pega</i> <i>RULES:WorkMgr4</i>	None
PerformBulk	Work-	<i>PegaRULES:SysAdm4Pega</i> <i>RULES:WorkMgr4</i>	None
AccessAuditTrail	Work-	<i>PegaRULES:SysAdm4Pega</i> <i>RULES:WorkMgr4</i>	None

Standard administrative privileges are shown in the following table.

Privilege	Privilege class	Role names	Comments
OpenDeveloperForm	@baseclass	<i>PegaRULES:SysAdm4</i>	Restricts anything related to rule form usage in Dev Studio. This is the most generic Dev Studio privilege. It is often used with <i>UpdateLimitedForm</i> and <i>pxViewLimitedForm</i> privileges. Rule delegation can also use this privilege.
clipboardViewer	@baseclass	<i>PegaRULES:SysAdm4</i>	Allows the user to see details in memory data called clipboard pages.
ActionEditAttachment	Work-	<i>PegaRULES:SysAdm4</i>	Runs <i>EditAttachment</i> or <i>ActionEditAttachment</i> flow actions.
ReviewPolicyOverrides	Work-	<i>PegaRULES:SysAdm4</i>	Performs assignments on a suspended work object.
PerformanceTools	Code-Pega-	<i>PegaRULES:SysAdm4</i>	Allows administrators to do specific expensive performance-related analysis. By default, this privilege is not available in a production environment.

Using privileges to control access

Most access control is granted through authorization, which controls many common use cases around creating, updating, and deleting cases and data. However, when using privileges, consider the following two areas:

- When to use a privilege
- [Setting a privilege to secure an activity](#)

When to use a privilege

In some cases, you must add a privilege to an activity to keep your application secure, for example:

- If an activity performs a task that only specific access groups or access roles should perform.
 - For example, a bulk processing activity for administrators should have a privilege so that users cannot run the activity.
- When **Allow invocation from browser** is selected, a user must have a privilege.

1. [Setting a privilege to secure an activity](#)

To secure an activity, determine the correct privilege, then assign that privilege to the role that is authorized to run that activity.

Setting a privilege to secure an activity

To secure an activity, determine the correct privilege, then assign that privilege to the role that is authorized to run that activity.

In this example, an activity is needed that gives a user the ability to reopen a case.

For information about privileges in standard Role Name rules in Pega Infinity, see *Standard privileges* in [Securing an activity](#).

Note: Your access role might have a dependent role. Users inherit all privileges from all dependent roles. To view dependent roles, click **Manage dependent role**.

Dependent role

PegaRULES:WorkMgr4

+ Add dependent role

1. Determine which access roles should and should not have the ability to run the activity.

1. To view the access role, in the panel of Dev Studio, click **Records Security Access Role Name** . Then, select *your Role Name* in the list.**Note:** In this example, your Role Name would be *ApplicationName:User*.
 - In this example, users with the role *PegaRULES:User4* should have the ability to update cases but not to reopen them.
 - Users with the *PegaRULES:WorkMgr4* role should have the ability to both update cases and reopen them.
2. In the **Privileges** column, choose the privilege that describes the action you need to take that is available for this role but not available for other roles.**Note:** If you are building a Component or Application reuse layer, adding new privileges could cause compatibility issues with upgrades, because it is possible to create a role name by using the **Clone from** feature instead of using role dependencies. When you add new privileges you must document this in upgrade instructions because those cloned roles will not inherit the new privilege after an upgrade.
In this example, there is an existing privilege called *WorkReopen* near the bottom of the *PegaRULES:WorkMgr4* role.

The work reopen privilege in the PegaRULES:WorkMGR4 role

Perform (5)
WorkReopen (5)
WorkUpdate (5)
PerformBulk (5)
AllFlows (5)
AllFlowActions (5)
AccessAuditTrail (5)

3. In the same row, in the first column, note the class that is associated with that privilege.
In this example, the class is *Work-*.

The Workreopen privilege shown in the Work- class

Work-	5	5	5	InMyRuleSetList	InMyRuleSetList	5	5	Perform (5) WorkReopen (5) WorkUpdate (5) PerformBulk (5) AllFlows (5) AllFlowActions (5) AccessAuditTrail (5)	None
-------	---	---	---	-----------------	-----------------	---	---	--	------

4. Open your activity rule form, and enter the privilege class and name, for example:

1. Privilege Class: *Work-*
2. Privilege Name: *WorkReopen*

Note: When an activity has multiple privileges listed, the user needs at least one of the privileges to be granted access.

Adding new privileges to roles after an upgrade

Pega Platform version 8.5 has introduced new security privileges to align with leading practices and Pega has provided guidance on how to strengthen your security architecture by adding new privileges to roles after an upgrade.

The new privileges will automatically be integrated into your application, if your roles are dependent upon the roles shipped in Pega Platform (the shipped roles are dependent roles for your roles).

For more information, see [Dependent roles](#).

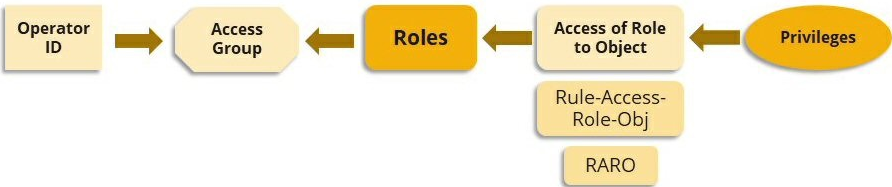
If you do not use dependent roles, you must add the new privileges manually into your application. To assist you with this process, Pega provides the [Privilege Information Utility](#).

Download the [Privilege Information Utility](#) from [Pega Marketplace](#) and install it according to the directions detailed in [Using the Privilege Information Utility](#), which describes how you import the utility, run an activity, and the export the data to Excel. From there, you can create pivot tables in Excel to best determine where to put a new privilege.

Privileges background

Privileges are assigned using access groups and roles.

A flow diagram explaining the relationship between the operator ID, access groups, roles, and privileges



When new privileges are added to Pega Platform applications, these privileges must be associated with roles in order to be implemented in that application.

For more information, see [Security of access roles](#).

Manually adding a new privilege to an application

To manually adding a new privilege to an application, you need to do the following steps:

- Determine where to add the privilege
 - What is the privilege?
 - What users/personas should have access to that privilege?
 - For those users/personas, how will that translate into access groups and roles?
- Use the *Privilege Information Utility*

Determine where to add the privilege

When adding a new privilege into an application, start with the following information:

- What is the privilege?

- What users/personas should have access to that privilege?
- For those users/personas, how will that translate into access groups and roles?

What is the privilege?

Determine the context of the privilege: what functionality it provides. It may protect some types of data from being accessed by ineligible users, or provide users with the ability to run certain features, like tracing.

In addition, determine where this privilege would be used. As stated above, it might affect data, or managerial functions in the application (like the ability to run a report), or system administration functions, such as the ability to view logs.

The new privilege might also be dependent upon an already-existing privilege.

For example, the ability to edit a report might depend on the user’s ability to view the report. If the privilege works with another privilege, then it is important to verify that the roles being granted that privilege also have been granted the underlying privilege.

What users/personas should have access to that privilege?

It is also important to determine what kind of user should be granted the privilege: work user, manager, system administrator, or others. Decide who would need the access – or who should be denied the access – that the privilege represents.

Note: The addition of a privilege for some users to gain access means that other types of users will be denied access.

For those users/personas, how will that translate into access groups and roles?

As shown above, access for users flows from the Operator ID through the access groups to the roles. Privileges are associated with those roles through the *Rule-Access-Role-Obj* records.

Once you have determined which users/personas should have access to this privilege, you must determine which role the privilege should be associated with, in order for the correct personas to receive that access.

Similar to the situational layer cake for rules, when you view your application as a series of layers, you should create new rules in the correct layer, so that functionality is available for all the layers above, and you don’t have to duplicate the rule around the system, which could cause future problems.

Likewise, if there is one role which is associated with all the access groups who need access to the privilege functionality, then it is easier to associate the privilege with that one role, rather than loading it into every role for every access group.

Example: Your access groups have the following roles:

Access Group	Roles
<i>PegaHC:Administrators</i>	AGM:AGCoordinator PegaRULES:Feedback PegaRULES:SecurityAdministrator PegaRULES:SysAdm4
<i>PegaHC:Administrators4010</i>	Pega4010:Administrator PegaRULES:SecurityAdministrator PegaRULES:SysAdm4
<i>PegaHC:AppSetup</i>	PegaRULES:SysAdm4
<i>PegaHC:HCUser</i>	PegaHCCore:User PegaRULES:Feedback PegaRULES:User4
<i>PegaHC:HCUserVIP</i>	PegaHC:View_Employee_PII PegaHCCore:User PegaRULES:Feedback PegaRULES:User4
<i>PegaHC:HealthCheck</i>	PegaHC:View_Employee_PII PegaRULES:Feedback PegaRULES:SysAdm4

In this example, a privilege having to do with the ability to see trace logs, or the ability to stop and start agents, would be a System Administrator privilege.

The following access groups in the example above have the *PegaRULES:SysAdm4* role:

- PegaHC:Administrators
- PegaHC:Administrators4010
- PegaHC:HealthCheck

If the users in those access groups are the one who should have this ability to see trace logs, then associating the *SeeTraceLogs* privilege with the *PegaRULES:SysAdm4* role would be the easiest way to make sure all those users have the appropriate access.

Now let’s suppose that the *HealthCheck* access group should *not* have access to see trace logs, but the two Administrator access groups should. Associating the *SeeTraceLogs* privilege with the SysAdm4 role would not work, because the *HealthCheck* users would also get that privilege. In that case, the *PegaRULES:SecurityAdministrator* role might be a better choice, as only the two administrators have that role.

Determining the best role to associate a privilege with requires that you be able to see all the access groups with their roles. The new Privilege List Utility provides this ability.

Use the Privileges Information Utility

The *Privileges Information Utility* helps you determine how best to integrate these new privileges into your existing security architecture, by displaying your access group and

role structure in one report.

An example privilege information spreadsheet

The screenshot shows an Excel spreadsheet with a PivotTable. The PivotTable is based on the 'AGM:AGCoordinator' data source. The PivotTable fields are set to show 'Access group' and 'Role name' in the Rows area. The PivotTable data is as follows:

Access group	Role name
PegaHC:Administrators	AGM:AGCoordinator
PegaHC:Administrators	AGM:CaseManager
PegaHC:Administrators	PegaHC:View_Employee_PII
PegaHC:Administrators	PegaHC:View_VIP_PII
PegaHC:Administrators	PegaHealthCodes:Administrator
PegaHC:Administrators	PegaRULES:Feedback
PegaHC:Administrators	PegaRULES:HighAvailabilityAdministrator
PegaHC:Administrators	PegaRULES:HighAvailabilityQuiesceInvestigator
PegaHC:Administrators	PegaRULES:PegaAPIDataPrivacyAdmin
PegaHC:Administrators	PegaRULES:SecurityAdministrator
PegaHC:Administrators	PegaRULES:SysAdm4
PegaHC:Administrators	PegaRULES:SysOpsAdministrator
PegaHC:Administrators	PegaRULES:SysOpsObserver
PegaHC:Administrators4010	Pega4010:Administrator
PegaHC:Administrators4010	PegaRULES:SecurityAdministrator
PegaHC:Administrators4010	PegaRULES:SysAdm4
PegaHC:Administrators5010	Pega5010:Administrator
PegaHC:Administrators5010	PegaHC:View_Employee_PII
PegaHC:Administrators5010	PegaHC:View_VIP_PII
PegaHC:Administrators5010	PegaRULES:Feedback
PegaHC:Administrators5010	PegaRULES:SecurityAdministrator
PegaHC:Administrators5010	PegaRULES:SysAdm4
PegaHC:AppSetup	PegaRULES:SysAdm4

• [Using the Privilege Information Utility](#)

In keeping with leading practices, Pega Platform version 8.5 has introduced new security privileges to maintain a strong security architecture.

- [Securing an activity](#)
- [Setting a privilege to secure an activity](#)

Using the Privilege Information Utility

In keeping with leading practices, Pega Platform version 8.5 has introduced new security privileges to maintain a strong security architecture.

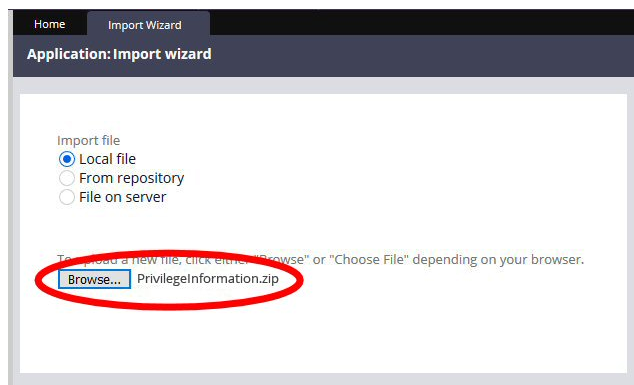
The Privilege Information Utility helps you determine how best to integrate these new privileges into your security architecture, by providing information on the existing structure of access groups, roles, and privileges in your application. Implementing the utility is done in two main steps:

1. Importing the utility.
2. Running the utility.
 1. Creating a pivot table in Excel

1. Importing the Privilege Information Utility

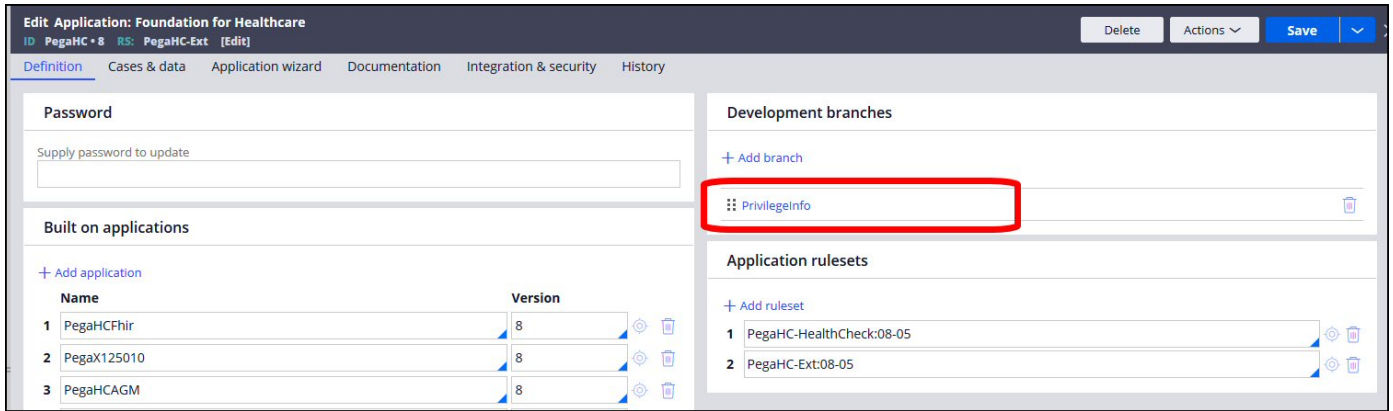
To import the Privilege Information Utility, follow the steps below:

1. Download the Privilege Information Utility from [Pega Marketplace](#).
2. Import this file into your Pega application. In the header of Dev Studio, click **Configure Application Distribution Import**.
3. Choose the `PrivilegeInformation.zip` file.



4. Click **Next** to import the utility. **Note:** This utility contains 9 rule instances in the `Pega-LP-Reports_Branch_PrivilegeInfo` ruleset.

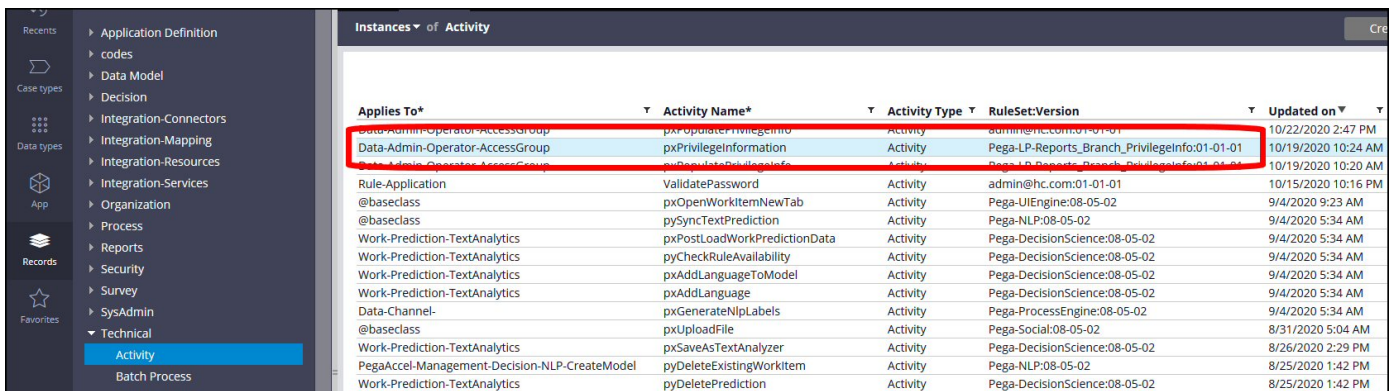
- After the import is complete, add the **PrivilegeInfo** branch into your application on the **Definition** tab, in the **Development branches** section, so the rules will be available.



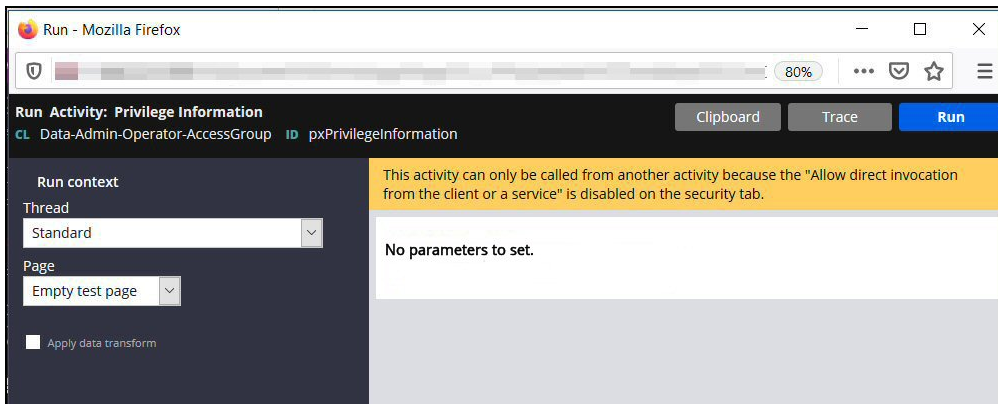
2. Running the Privilege Information Utility

To run the Privilege Information Utility, follow the steps below:

- In the side pane of Dev Studio, click **Records > Technical > Activity**.
- Open the activity **pxPrivilegeInformation** in the **Applies To** class **Data-Admin-Operator-AccessGroup**.



- From the **Actions** menu, click **Run**. **Note:** The **Run Activity** form will display.



- Click **Run** to produce the access group & roles report.
- RESULT:** The report will be generated. **Note:**
 - Depending upon your browser, the Excel spreadsheet will either open (Mozilla Firefox) or download in the corner for you to click on to open (Google Chrome).
 - The file is safe – you may open the file. However, a warning message may display:

“The file format and extension of ‘PrivilegeInformation.xls’ don’t match. The file could be corrupted or unsafe. Unless you trust its source, don’t open it. Do you want to open it anyway?”

Creating a pivot table in Excel

After the report is exported to Excel, a spreadsheet will display, which includes the following data for each privilege:

- Access group
- Role
- Privileges Class
- Name of Privilege

PrivilegeInformation.xlsx - Excel			
File Home Insert Page Layout Formulas Data Review View Acrobat Tell me what you want to do			
Clipboard Font Alignment Number Styles			
A1 Access group			
A	B	C	D
1	Access group	Role name	Privilege class
2	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
3	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
4	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
5	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
6	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
7	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
8	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
9	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
10	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
11	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
12	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
13	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
14	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
15	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
16	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
17	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
18	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass
19	PegaHC:Administrators	PegaRULES:SysAdm4	@baseclass

As there are many privileges in Pega applications, there will be many lines in this spreadsheet. To make this data easier to read, Pega recommends that you create a pivot table.

1. From the **Insert** menu in Excel, choose **PivotTable**.
2. In the **Create PivotTable** window, click **Select a table or range**, and highlight all the data in the worksheet.

The screenshot shows the Excel 'Create PivotTable' dialog box. The 'Insert' tab is selected in the ribbon. The dialog box has the following options:

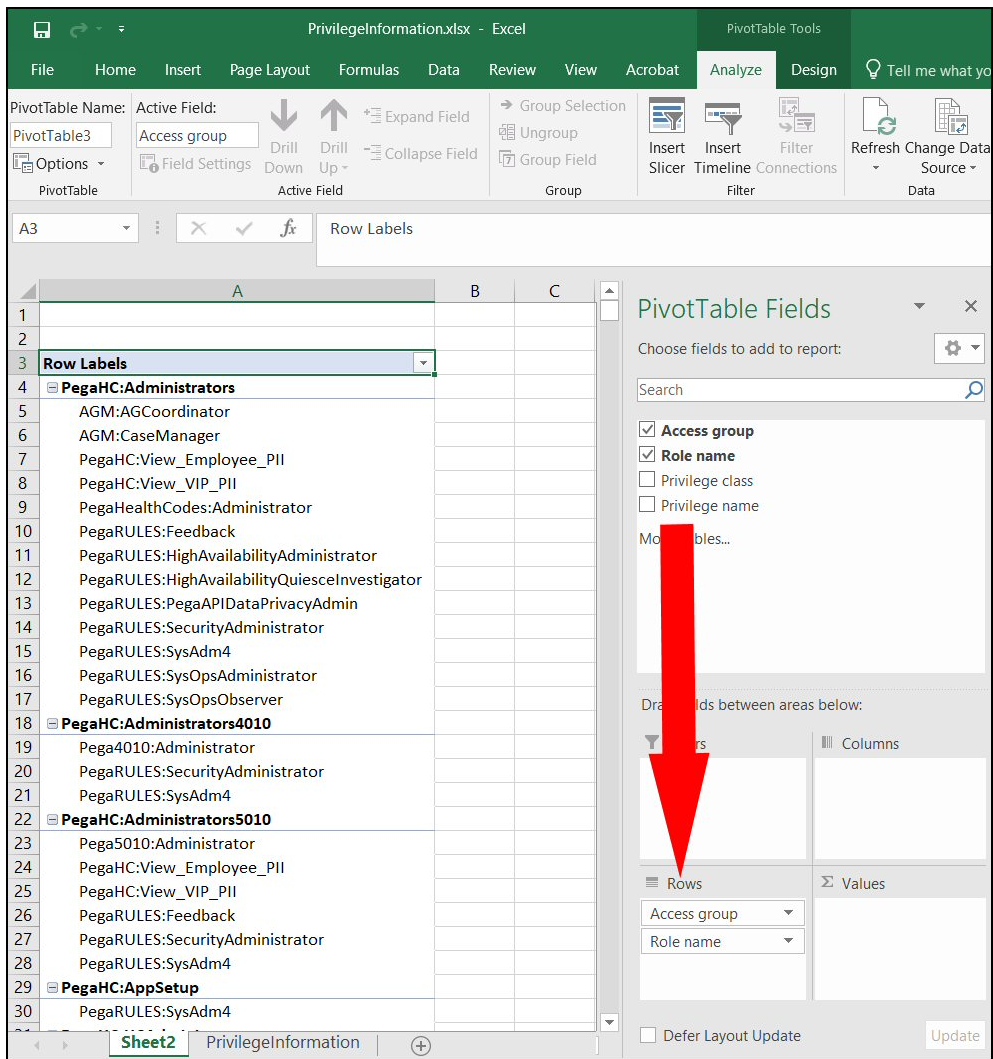
- Choose the data that you want to analyze:**
 - ☒ Select a table or range: Table/Range: Data!\$A\$1:\$D\$1000
 - ☐ Use an external data source: Choose Connection...
 - ☐ Use this workbook's Data Model
- Choose where you want the PivotTable report to be placed:**
 - ☐ New Worksheet
 - ☒ Existing Worksheet: Location: Data!\$G\$6
- Choose whether you want to analyze multiple tables:**
 - ☐ Add this data to the Data Model

The 'OK' button is highlighted.

3. Click **OK**. **Note:** The PivotTable creation structure will display.

At this point, you can rearrange the information as needed.

One suggestion for using this utility is to drag the **Access Group** and the **Role Name** into the **Rows** area, so the table displays the roles used by each **Access Group**.



- [Adding new privileges to roles after an upgrade](#)

Servlet management

Use the Servlet Management landing page in your Pega Cloud applications to securely add, modify, and disable servlet configurations without editing the web.xml file.

You can customize the servlet name, authentication type, and parameters. Additionally, you can add new servlets and disable and delete unused servlets, or create new servlets by cloning existing servlets.

A servlet is a Java program that extends capabilities of the server, responds to any incoming requests, and construct responses. Servlets are used during authentication in commonly used protocols, such as Lightweight Directory Access Protocol (LDAP), Security Assertion markup Language (SAML), and OpenID connect, and other.

Pega Cloud supports system-created servlets (default) and custom (user-defined) servlets, which are created by the security administrator.

The **Availability** column displays whether the servlet has been enabled or disabled, and whether a restart is required for changes to take effect.

When you add a servlet on the Servlet Management landing page, the URL pattern that you configure for the servlet adds servlet mappings for the corresponding servlet. For example, if there is a URL pattern named PRServlet under a servlet WebStandard, then two servlet mappings are created: /PRServlet and /PRServlet/*.

When you make servlet management changes, you need to restart your system for the changes to take effect. To request a restart of the entire environment, select **New request** in [My Support Portal](#). For more information, see:

- [My Support Portal: New Design, Streamlined Features](#)
- [Restarts in Pega Cloud environments](#)

The image below shows the *Servlet Management* landing page.

The Servlet management landing page

System: Servlet Management

Refresh

Help

Servlet Management

Use Servlet Management to add, modify, and disable the servlet configurations. Note: A system restart is required for these changes to take effect: adding a new servlet and modifying or deleting an existing servlet.

Add servlet

Name	Authentication type	URL pattern(s)	Created by	Availability
WebStandard	PRBasic	*,PRServlet	System	ENABLED
sso1	PRCustom	sso1	System	ENABLED
WebStandardLDAP1	PRExternal	PRServletLDAP1	System	ENABLED
WebStandardContainerAuth	J2EEContext	PRServletContainerAuth	System	ENABLED
sso	PRCustom	sso	System	ENABLED
PRChat	PRCustom	PRChat	System	ENABLED
WebStandardExternal	PRExternal	PRServletExternal	System	ENABLED
WebLDAP3	PRCustom	PRWebLDAP3Auth,PRWebLDAP3	System	ENABLED
IAC	PRCustom	IAC	System	ENABLED
sso2	PRCustom	sso2	System	ENABLED
WebStandardLDAP2	PRExternal	PRServletLDAP2	System	ENABLED
WebStandardLDAP3	PRExternal	PRServletLDAP3	System	ENABLED

1 2 Next

To the right of the servlets, there are vertical ellipsis, when clicked, the following options are displayed: **Clone**, **Configure**, and **Copy URL**.

Note: System-generated servlets have different configuration options than user-created servlets. For example, you cannot delete system-generated servlets, but you can delete user-generated servlets.

An image of the options available on a system-generated servlet is shown below:

The options for a system-generated servlet: Clone,Configure, Copy URL, Delete

Availability

ENABLED	
ENABLED	Clone
ENABLED	Configure
ENABLED	Copy URL

An image of the options available on a user-created servlet is shown below:

The options for a user-created servlet: Clone, Configure, Copy URL, Delete

Availability

ENABLED	
ENABLED	Clone
ENABLED	Configure
ENABLED	Copy URL
ENABLED	Delete

Servlet cloning

Clicking **Clone** launches a dialog box similar to adding a new servlet. However, the **Purpose** and **Servlet authentication type** fields have the same information as the servlet you are cloning. Any **Initialization parameters** will also be carried over when a servlet is cloned.

When you want to rename an existing servlet or create a new servlet, with similar properties as another servlet, you can clone that servlet.

When you rename an existing servlet, you can clone the new servlet with the updated name, and then disable the original servlet so that only the new servlet is accessible.

The following figure shows a example of a cloned servlet configuration:

The Cloning a servlet dialog box

The cloning a servlet dialog box

Cloning a servlet

☒ Enable this servlet configuration

Servlet name*

Purpose*

Servlet WebStandard used for Authentication using PRBasic

Servlet authentication type*

PRBasic

Initialization parameters

Name	Value
No items	

+ Add

URL Patterns*

No items

+ Add

Cancel

Add

Servlet configuration

The **Configure** option provides a summary of the servlet information and shows controls for editing and making changes to the configured servlet.

The following figure shows an example of the servlet configuration dialog box:

The Editing a servlet dialog

The editing a servlet dialog box

Editing a servlet

☒ Enable this servlet configuration

Servlet name

WebStandard

Purpose*

Servlet WebStandard used for Authentication using PRBasic

Servlet authentication type*

PRBasic

Initialization parameters

Name	Value
No items	

+ Add

URL Patterns*

PRServlet	
*	

+ Add

Cancel

Update

Servlet copy URL copying

You can use the **Copy URL** option to copy the servlet URL and use that URL for testing or other purposes.

Servlet name

The standard URL to the specific servlet on the server side, and the Copy URL option provides access to the corresponding servlet URL in a browser without manually typing the IP address.

*

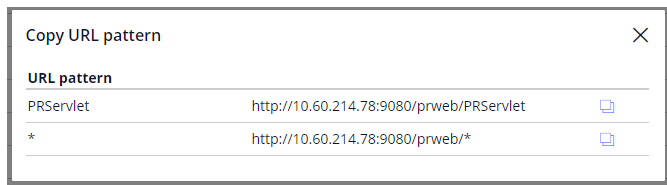
The default servlet URL pattern. If you access a URL after `/prweb` and the servlet is not found, you are redirected to the default servlet where `*` is configured as the URL pattern.

If you do not provide the `*` URL, then system becomes inaccessible. This setting is the minimum servlet configuration that is required for an application. Without this configuration, a 403 error is displayed.

You can copy the URL pattern of the Servlet name, as shown in the following example:

The Copy URL pattern dialog box

The copy URL pattern dialog box



- [Adding a servlet](#)

Add a new servlet, without editing the web.xml file, by cloning and modifying existing servlets where necessary.

- [Cloning a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. Clone a servlet to create a new servlet without having to recreate all the parameters.

- [Configuring a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. Create new servlets by cloning existing servlets and modifying where necessary.

- [Disabling a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. If there is a servlet that is not commonly used within the application and that you want to keep in the system, you can disable the servlet.

- [Deleting a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. You can delete user-defined servlets that are no longer used in the system and that you do not need to keep within the system.

- [Moving the * URL pattern between servlets](#)

The * is a wildcard character that is not used in the URL. * is the default servlet URL pattern that you can replace with another servlet.

Adding a servlet

Add a new servlet, without editing the web.xml file, by cloning and modifying existing servlets where necessary.

Note: When you make changes in Servlet Management, you need to restart your system for the changes to take effect.

1. In the header of Dev Studio, click **Configure System Settings Servlet Management**.
2. On the **Servlet Management** landing page, click **Add servlet**.
3. In the **Servlet name** field, enter a label for the servlet.
4. In the **Purpose** field, enter a description why you want to add the new servlet.
5. In the **Servlet authentication type** field, enter the authentication type used for the servlet.
6. In the **Initialization parameters** section, click the **Add** link, and then enter the name and value of the initialization parameters that you want to add to the servlet.
7. In the **URL patterns** section, click the **Add** link, and then define any additional URL pattern to the servlet.
The first URL pattern is the same as the pattern that appears in the **Servlet name** field.
8. Click **Add**.
9. For the changes to take effect, you must restart your entire environment.

What to do next: To request a restart of the entire environment, select **New request** in [My Support Portal](#). For more information, see:

- [My Support Portal: New Design, Streamlined Features](#)
- [Restarts in Pega Cloud environments](#)

- [Cloning a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. Clone a servlet to create a new servlet without having to recreate all the parameters.

- [Configuring a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. Create new servlets by cloning existing servlets and modifying where necessary.

- [Disabling a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. If there is a servlet that is not commonly used within the application and that you want to keep in the system, you can disable the servlet.

- [Deleting a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. You can delete user-defined servlets that are no longer used in the system and that you do not need to keep within the system.

- [Moving the * URL pattern between servlets](#)

The * is a wildcard character that is not used in the URL. * is the default servlet URL pattern that you can replace with another servlet.

Cloning a servlet

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. Clone a servlet to create a new servlet without having to recreate all the parameters.

1. In the header of Dev Studio, click **Configure System Settings Servlet Management**.
2. Identify the servlet that you want to clone, click the **More** icon on the right-side of the row, and then select **Clone**.
3. In the **Servlet name** field, provide a label for the new servlet.
4. In the **Cloning a servlet** dialog box, edit the servlet as needed.
5. Click **Add**.
6. For the changes to take effect, you must restart your entire environment.

What to do next: To request a restart of the entire environment, select **New request** in [My Support Portal](#). For more information, see:

- [My Support Portal: New Design, Streamlined Features](#)
- [Restarts in Pega Cloud environments](#)

- [Adding a servlet](#)

Add a new servlet, without editing the web.xml file, by cloning and modifying existing servlets where necessary.

- [Configuring a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. Create new servlets by cloning existing servlets and modifying where necessary.

- [Disabling a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. If there is a servlet that is not commonly used within the application and that you want to keep in the system, you can disable the servlet.

- [Deleting a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. You can delete user-defined servlets that are no longer used in the system and that you do not need to keep within the system.

- [Moving the * URL pattern between servlets](#)

The * is a wildcard character that is not used in the URL. * is the default servlet URL pattern that you can replace with another servlet.

Configuring a servlet

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. Create new servlets by cloning existing servlets and modifying where necessary.

1. In the header of Dev Studio, click **Configure System Settings Servlet Management**.
2. Identify the servlet that you want to edit, click the **More** icon on the right side of the row, and then select **Configure**.
3. Make your changes to the servlet:
 - To edit the servlet, select the **Editing a servlet** check box, and then make your edits.
 - To disable the servlet, at the top of the **Editing a servlet** dialog, clear the **Editing a servlet** check box.
4. Click **Update**.
5. For the changes to take effect, you must restart your entire environment.

What to do next: To request a restart of the entire environment, select **New request** in [My Support Portal](#). For more information, see:

- [My Support Portal: New Design, Streamlined Features](#)
- *Restarts in Pega Cloud environments*

- [Adding a servlet](#)

Add a new servlet, without editing the web.xml file, by cloning and modifying existing servlets where necessary.

- [Cloning a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. Clone a servlet to create a new servlet without having to recreate all the parameters.

- [Disabling a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. If there is a servlet that is not commonly used within the application and that you want to keep in the system, you can disable the servlet.

- [Deleting a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. You can delete user-defined servlets that are no longer used in the system and that you do not need to keep within the system.

- [Moving the * URL pattern between servlets](#)

The * is a wildcard character that is not used in the URL. * is the default servlet URL pattern that you can replace with another servlet.

Disabling a servlet

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. If there is a servlet that is not commonly used within the application and that you want to keep in the system, you can disable the servlet.

1. In the header of Dev Studio, click **Configure System Settings Servlet Management**.
2. Identify the servlet that you want to edit, click the **More** icon on the right side of the row, and then select **Configure**.
3. At the top of the **Editing a servlet** dialog box, clear the **Enabling a servlet configuration** check box.
4. Click **Update**.
5. For the changes to take effect, you must restart your entire environment.

What to do next: To request a restart of the entire environment, select **New request** in [My Support Portal](#). For more information, see:

- [My Support Portal: New Design, Streamlined Features](#)
- *Restarts in Pega Cloud environments*

- [Adding a servlet](#)

Add a new servlet, without editing the web.xml file, by cloning and modifying existing servlets where necessary.

- [Cloning a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. Clone a servlet to create a new servlet without having to recreate all the parameters.

- [Configuring a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. Create new servlets by cloning existing servlets and modifying where necessary.

- [Deleting a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. You can delete user-defined servlets that are no longer used in the system and that you do not need to keep within the system.

- [Moving the * URL pattern between servlets](#)

The * is a wildcard character that is not used in the URL. * is the default servlet URL pattern that you can replace with another servlet.

Deleting a servlet

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. You can delete user-defined servlets that are no longer used in the system and that you do not need to keep within the system.

1. In the header of Dev Studio, click **Configure System Settings Servlet Management**.
2. Identify the servlet that you want to delete, click the **More** icon on the right side of the row, and select **Delete**.
3. In the **Confirm delete servlet** dialog box, click **Delete**.
4. For the changes to take effect, you must restart your entire environment.

What to do next: To request a restart of the entire environment, select **New request** in [My Support Portal](#). For more information, see:

- [My Support Portal: New Design, Streamlined Features](#)
- [Restarts in Pega Cloud environments](#)

- [Adding a servlet](#)

Add a new servlet, without editing the web.xml file, by cloning and modifying existing servlets where necessary.

- [Cloning a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. Clone a servlet to create a new servlet without having to recreate all the parameters.

- [Configuring a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. Create new servlets by cloning existing servlets and modifying where necessary.

- [Disabling a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. If there is a servlet that is not commonly used within the application and that you want to keep in the system, you can disable the servlet.

- [Moving the * URL pattern between servlets](#)

The * is a wildcard character that is not used in the URL. * is the default servlet URL pattern that you can replace with another servlet.

Moving the * URL pattern between servlets

The * is a wildcard character that is not used in the URL. * is the default servlet URL pattern that you can replace with another servlet.

Moving the * is optional and is done to change to the default and set up automatic redirection. For example, if one of your teams decides to move the * from default WebStandard to DeltaTeam, all access is automatically redirected from <https://deltateam.pega.com> to <https://deltateam.pega.com/Prweb/deltateam> by default.

If a request is received that does not match any of the defined URL patterns, then the definition for * is used to determine to what servlet the request is dispatched.

If you access something after /prweb and the servlet is not found, you are redirected to the default servlet, where * is configured as the URL pattern.

If the * URL pattern is not provided, then system becomes inaccessible. * is the minimum servlet requirement for a servlet to be accessible. Without *, the servlet produces a 403 error. The Servlet Management landing page provides a validation to ensure that at least one servlet is in the Enabled state with * as the URL pattern.

1. Access the servlet from which you want to move the * URL pattern:
 - a. In the header of Dev Studio, click **Configure System Settings Servlet Management**.
 - b. Identify the servlet from which you want to remove the * URL pattern, click the **More** icon on the right side of the window, and then select **Configure**.
 - c. Under **URL patterns**, click the **delete** icon beside the * URL pattern.
 - d. Click **Update**.
2. Add the * URL pattern to the servlet that you want to update:
 - a. Identify the servlet to which you want to add the * URL pattern, click the **More** icon on the right side of the window, and then select **Configure**.
 - b. Under **URL patterns**, click **Add**.
 - c. Enter the * URL pattern.
 - d. Click **Update**.
 - e. For the changes to take effect, you must restart your entire environment.

What to do next: To request a restart of the entire environment, select **New request** in [My Support Portal](#). For more information, see:

- [My Support Portal: New Design, Streamlined Features](#)
- [Restarts in Pega Cloud environments](#)

- [Adding a servlet](#)

Add a new servlet, without editing the web.xml file, by cloning and modifying existing servlets where necessary.

- [Cloning a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. Clone a servlet to create a new servlet without having to recreate all the parameters.

- [Configuring a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. Create new servlets by cloning existing servlets and modifying where necessary.

- [Disabling a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. If there is a servlet that is not commonly used within the application and that you want to keep in the system, you can disable the servlet.

- [Deleting a servlet](#)

Servlet Management is only available for Pega Cloud, and allows you to add, modify, and disable servlet configurations without editing the web.xml file. You can delete user-defined servlets that are no longer used in the system and that you do not need to keep within the system.

Security in App Studio

To increase system usability and better accommodate multi-functional teams, some parts of the security framework are now available in App Studio.

Only administrators with the *pzAdvancedSecurityUser* privilege can map authentication services to an application. By default, the *PegaRULES:SecurityAdministrator* role includes the *pzAdvancedSecurityUser* privilege.

- [Creating a Google authentication service](#)

Create a Google SSO authentication service so that users can authenticate using Google as the identity provider.

- [Creating a SAML SSO authentication service in App Studio](#)

Create a Security Assertion Markup Language (SAML) single-sign on (SSO) authentication service so that users can authenticate using a SAML identity provider.

- [Creating an OIDC SSO authentication service in App Studio](#)

Create an Open ID Connect (OIDC) single-sign on (SSO) authentication service so that users can authenticate using an OIDC identity provider.

- [Mapping authentication services in App Studio](#)

As an administrator, configure authentication services in an application by using App Studio. This feature streamlines functionality and provides an effective and user-friendly way to configure authentication for users who are not familiar with Dev Studio.

Creating a Google authentication service

Create a Google SSO authentication service so that users can authenticate using Google as the identity provider.

Before you begin: To create an authentication service, you must have the *pzCanCreateAuthService* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.

Before you create a Google authentication service, you must register with the identity provider and obtain a client ID and client secret. You do this outside of Pega Platform.

This is a quick way to define an authentication service with Google as the OIDC provider. The result is the same as creating an OIDC authentication service and manually specifying the values needed for Google.

1. In the navigation pane of App Studio, click **Users Single sign-on (SSO)**.
2. Click **New**, and then click **Google**.
3. Enter a **Name** for the service. **Result:** The value that you enter is used to populate the **Login URL** field. Users access this URL to log in to your application.
4. **Optional:** The **Client ID** and **Client secret** field default to the values that you used to log in. If you need different values, enter the values that were assigned by your identity provider.
5. **Optional:** To automatically create an operator when the operator who is logging in does not already exist in the Pega database, do the following steps.
 - a. Select the **Create operators for new users** check box.
 - b. In the **Access role** list, click the access role for the new user.
6. Click **Submit**.
7. To enable the authentication service, on the **Single sign-on (SSO)** landing page, where the new service is listed, turn on the switch.

- [Creating a SAML SSO authentication service in App Studio](#)

Create a Security Assertion Markup Language (SAML) single-sign on (SSO) authentication service so that users can authenticate using a SAML identity provider.

- [Creating an OIDC SSO authentication service in App Studio](#)

Create an Open ID Connect (OIDC) single-sign on (SSO) authentication service so that users can authenticate using an OIDC identity provider.

- [Mapping authentication services in App Studio](#)

As an administrator, configure authentication services in an application by using App Studio. This feature streamlines functionality and provides an effective and user-friendly way to configure authentication for users who are not familiar with Dev Studio.

Creating a SAML SSO authentication service in App Studio

Create a Security Assertion Markup Language (SAML) single-sign on (SSO) authentication service so that users can authenticate using a SAML identity provider.

Before you begin: To create an authentication service, you must have the *pzCanCreateAuthService* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.

Before you create a SAML authentication service, you must register with the identity provider. You do this outside of Pega Platform.

1. In the navigation pane of App Studio, click **Users Single sign-on (SSO)**.
2. Click **New**, and then click **SAML 2.0**.
3. Enter a **Name** for the service. **Result:** The value that you enter is used to populate the **Login URL** field. Users access this URL to log in to your application.
4. Click **Import metadata** to import metadata from your identity provider.
 - To select a URL where the metadata is stored, select **via URL**, enter a URL, and click **Submit**.
 - To select a file where the metadata is stored, select **via file**, enter a file name, and click **Submit**.
5. Select how to map the user identifier from the SAML assertion to the Pega operator ID.
 - To map from the standard name identifier attribute, select **NameID**.
 - To map from any other SAML attribute, select **Attribute** and enter an attribute name. Enclose the attribute name in curly braces, for example, {uid}.
6. **Optional:** To automatically create an operator when the operator who is logging in does not already exist in the Pega database, do the following steps.
 - a. Select the **Create operators for new users** check box.
 - b. In the **Access role** list, click the access role for the new user.
7. To display the newly created metadata in a new browser window, click the URL that is displayed under **Configure your IdP**. To complete the SSO configuration, you must configure Pega Platform as a service provider within your identity provider. To do this, you will need to provide this URL or some of its content.
8. Click **Submit**.
9. **Optional:** To configure advanced functionality, on the **Single sign-on (SSO)** landing page, where the new service is listed, click the More icon and then click **Open in Dev Studio**. **Result:** The authentication service opens in Dev Studio. For more information, refer to the help in Dev Studio.
10. To enable the authentication service, on the **Single sign-on (SSO)** landing page, where the new service is listed, turn on the switch.

- [Creating a Google authentication service](#)

Create a Google SSO authentication service so that users can authenticate using Google as the identity provider.

- [Creating an OIDC SSO authentication service in App Studio](#)

Create an Open ID Connect (OIDC) single-sign on (SSO) authentication service so that users can authenticate using an OIDC identity provider.

- [Mapping authentication services in App Studio](#)

As an administrator, configure authentication services in an application by using App Studio. This feature streamlines functionality and provides an effective and user-friendly way to configure authentication for users who are not familiar with Dev Studio.

Creating an OIDC SSO authentication service in App Studio

Create an Open ID Connect (OIDC) single-sign on (SSO) authentication service so that users can authenticate using an OIDC identity provider.

Before you begin: To create an authentication service, you must have the *pzCanCreateAuthService* privilege, which is included in the *PegaRULES:SecurityAdministrator* role.

Before you create an OIDC authentication service, you must register with the identity provider and obtain a client ID and a client secret. You do this outside of Pega Platform.

1. In the navigation pane of App Studio, click **Users Single sign-on (SSO)**.
2. Click **New**, and then click **OpenID Connect**.
3. Enter a **Name** for the service. **Result:** The value that you enter is used to populate the **Login URL** field. Users access this URL to log in to your application.
4. Click **Import metadata** to import metadata from your identity provider.
 - To select a URL where the metadata is stored, select **via URL**, enter a URL, and click **Submit**.
 - To select a file where the metadata is stored, select **via file**, enter a file name, and click **Submit**.
5. In the **Client ID** and **Client secret** fields, enter the values that were assigned by your identity provider.
6. In the **Map operator ID from claim** field, enter the attribute name from the claim that is mapped to the Pega Platform operator ID. Enclose the attribute name in curly braces, for example, {name}.
7. **Optional:** To automatically create an operator when the operator who is logging in does not already exist in the Pega database, do the following steps.
 - a. Select the **Create operators for new users** check box.
 - b. In the **Access role** list, click the access role for the new user.
8. Copy the redirect URL that is displayed under **Configure your IdP**. To complete SSO configuration, you must register Pega Platform as a client (relying party) with your identity provider, using the redirect URI that you copy.
9. Click **Submit**.
10. **Optional:** To configure advanced functionality, on the **Single sign-on (SSO)** landing page, where the new service is listed, click the More icon and then click **Open in Dev Studio**. **Result:** The authentication service opens in Dev Studio. For more information, refer to the help in Dev Studio.
11. To enable the authentication service, on the **Single sign-on (SSO)** landing page, where the new service is listed, turn on the switch.

- [Creating a Google authentication service](#)

Create a Google SSO authentication service so that users can authenticate using Google as the identity provider.

- [Creating a SAML SSO authentication service in App Studio](#)

Create a Security Assertion Markup Language (SAML) single-sign on (SSO) authentication service so that users can authenticate using a SAML identity provider.

- [Mapping authentication services in App Studio](#)

As an administrator, configure authentication services in an application by using App Studio. This feature streamlines functionality and provides an effective and user-friendly way to configure authentication for users who are not familiar with Dev Studio.

Mapping authentication services in App Studio

As an administrator, configure authentication services in an application by using App Studio. This feature streamlines functionality and provides an effective and user-friendly way to configure authentication for users who are not familiar with Dev Studio.

This functionality is available in both App Studio and Dev Studio. For more information, see [Mapping authentication services in Dev Studio](#).

Note: Any changes that you make to the authentication services in App Studio will display in Dev Studio, and any changes that you make to the authentication services in Dev Studio will display in App Studio. However, you must first save your changes and then refresh both studios to see the changes. **Note:** In App Studio, you can only create Google, OIDC, and SAML 2 authentication services. If you want to create any other type of authentication service, such as a custom authentication service, you must use Dev Studio.

In App Studio, you can add a new authentication service to an application. Additionally, you can configure, copy the login URL, remove associations and disable the authentication services.

Only administrators with the *pzAdvancedSecurityUser* privilege can map authentication services to an application. By default, the *PegaRULES:SecurityAdministrator* role includes the *pzAdvancedSecurityUser* privilege.

Administrators with this access role can select the default authentication service or interactive screen on which user can then select an authentication service.

- If the administrator selects the default configuration, then the user is automatically redirected to the default authentication service during login.
- If the administrator selects the interactive configuration, then the system will display the authentication selector screen with mapped authentication services.

You can map an existing authentication service in App Studio by following the steps below.

1. In the navigation pane of App Studio, click **Users Authentication**.
2. Click **+ Add authentication service**.
3. Select the existing authentication service that you want to map.
4. **Optional:** To update the authentication service, click the **Configure** icon to the right of the new authentication service.
5. Click **Save**.

- [Creating a Google authentication service](#)

Create a Google SSO authentication service so that users can authenticate using Google as the identity provider.

- [Creating a SAML SSO authentication service in App Studio](#)

Create a Security Assertion Markup Language (SAML) single-sign on (SSO) authentication service so that users can authenticate using a SAML identity provider.

- [Creating an OIDC SSO authentication service in App Studio](#)

Create an Open ID Connect (OIDC) single-sign on (SSO) authentication service so that users can authenticate using an OIDC identity provider.

You can create a new authentication service in App Studio by following the steps below.

1. In the navigation pane of App Studio, click **Users Authentication**.
2. Click **+ Add authentication service**.
3. In the dialog box, click **Create new**.
4. Choose the type of authentication service that you want to create.
In App Studio, you can only create Google, OIDC, and SAML 2 authentication services. To create other types of authentication service, such as a custom authentication service, you must use Dev Studio.
5. In the **Name** field, enter a name for your authentication service.
6. In the **Identity Provider metadata** dialog box, click the **Import metadata** button, and import the IdP metadata via a URL or by uploading a file.
7. In the **Identity Provider metadata** dialog box, click **Submit**.
8. In the **Create new SSO** dialog box, click **Submit**.
9. Click **Save**.

- [Creating a Google authentication service](#)

Create a Google SSO authentication service so that users can authenticate using Google as the identity provider.

- [Creating a SAML SSO authentication service in App Studio](#)

Create a Security Assertion Markup Language (SAML) single-sign on (SSO) authentication service so that users can authenticate using a SAML identity provider.

- [Creating an OIDC SSO authentication service in App Studio](#)

Create an Open ID Connect (OIDC) single-sign on (SSO) authentication service so that users can authenticate using an OIDC identity provider.

- [Creating a Google authentication service](#)

Create a Google SSO authentication service so that users can authenticate using Google as the identity provider.

- [Creating a SAML SSO authentication service in App Studio](#)

Create a Security Assertion Markup Language (SAML) single-sign on (SSO) authentication service so that users can authenticate using a SAML identity provider.

- [Creating an OIDC SSO authentication service in App Studio](#)

Create an Open ID Connect (OIDC) single-sign on (SSO) authentication service so that users can authenticate using an OIDC identity provider.