

Pega 8.1 Multitenancy Administration Guide



©2018 Pegasystems Inc., Cambridge, MA. All rights reserved.

Trademarks

For Pegasystems Inc. trademarks and registered trademarks, all rights reserved. All other trademarks or service marks are property of their respective holders.

For information about the third-party software that is delivered with the product, refer to the third-party license file on your installation media that is specific to your release.

Notices

This publication describes and/or represents products and services of Pegasystems Inc. It may contain trade secrets and proprietary information that are protected by various federal, state, and international laws, and distributed under licenses restricting their use, copying, modification, distribution, or transmittal in any form without prior written authorization of Pegasystems Inc.

This publication is current as of the date of publication only. Changes to the publication may be made from time to time at the discretion of Pegasystems Inc. This publication remains the property of Pegasystems Inc. and must be returned to it upon request. This publication does not imply any commitment to offer or deliver the products or services described herein.

This publication may include references to Pegasystems Inc. product features that have not been licensed by you or your company. If you have questions about whether a particular capability is included in your installation, please consult your Pegasystems Inc. services consultant.

Although Pegasystems Inc. strives for accuracy in its publications, any publication may contain inaccuracies or typographical errors, as well as technical inaccuracies. Pegasystems Inc. shall not be liable for technical or editorial errors or omissions contained herein. Pegasystems Inc. may make improvements and/or changes to the publication at any time without notice.

Any references in this publication to non-Pegasystems websites are provided for convenience only and do not serve as an endorsement of these websites. The materials at these websites are not part of the material for Pegasystems products, and use of those websites is at your own risk.

Information concerning non-Pegasystems products was obtained from the suppliers of those products, their publications, or other publicly available sources. Address questions about non-Pegasystems products to the suppliers of those products.

This publication may contain examples used in daily business operations that include the names of people, companies, products, and other third-party publications. Such examples are fictitious and any similarity to the names or other data used by an actual business enterprise or individual is coincidental.

This document is the property of:

Pegasystems Inc.
One Rogers Street
Cambridge, MA 02142-1209, USA
Phone: 617-374-9600 Fax: 617-374-9620

www.pega.com

Document: Multitenancy 8.1 Administration Guide

Feedback

If you have comments for how we can improve our materials, send an email to DocRequest@Pega.com.

Contents

Multitenancy.....	4
Benefits of multitenancy.....	4
Multitenancy planning.....	5
Multitenant architecture.....	6
Multitenant core concepts.....	6
Tenant representation in multitenancy.....	7
Multitenancy Installation.....	8
Multitenant application development.....	9
Application layering in multitenancy.....	9
Tenant life cycle.....	10
Multitenant SDLC model.....	10
Tenant development restrictions.....	10
Multitenant security.....	14
Roles, access groups, and privileges.....	14
File system temporary directory.....	15
Multitenant system deployment planning.....	16
Tenant life cycle administration.....	17
Tenant management.....	17
Viewing tenant information.....	17
Adding tenants and administrators.....	17
Managing tenants.....	18
Deleting tenants.....	18
Multitenancy SOAP services.....	18
CreateTenant service.....	19
DeleteTenants service.....	20
GetDeleteTenantsStatus service.....	20
GetTenantDetails service.....	20
GetTenantStatus service.....	20

Multitenancy

A multitenant system is a single system that is logically partitioned into multiple heterogeneous business processing environments, which each operate as if they were on a dedicated, stand-alone system. In multitenancy, a tenant is a secure region of a multitenant system. A tenant shares a single database, rulesets, and code that support the operations of the other tenants in the system. However, tenant specific content is stored in a tenant-safe database component visible only to the user of that tenant. A tenant is stored as an instance of the `Data-Admin-Tenant` class.

Pega Platform provides multitenancy in the following ways:

- Multi-instancing where each tenant has its own system. In this model the applications are managed per system. Operations and management are typically costlier.
- Multitenant Edition achieves multitenancy through architectural means. The database schema is enhanced so that tenant-specific information is isolated to the data access layer. Logically, a multitenant system is partitioned into a shared region (code and rules available to all tenants) and tenant region (with tenant-specific applications or overrides). The system is aware of the tenant and ensures that tenants are isolated from each other. Tenants are restricted from using certain features that might inadvertently or maliciously expose another tenant's data or content.
- Applications can build in multitenancy as part of the application design. In this model a tenant ID is added to the class model of the application. It was the primary way to deliver multitenant application prior to Multitenant Edition.

In all cases, a multitenant provider deploys and operates a multitenant system. The multitenant provider is responsible for all aspects of:

- System planning, operation, and maintenance
- Security
- Tenant life cycle
- Provisioning of shared content
- Tenant application development and customization

Benefits of multitenancy

Multitenant Edition provides a variety of financial and operational benefits that increase over time as more tenants are added and share resources. Specific benefits include:

- Better use of system resources (memory, CPU, disk)
- Smaller hardware footprint
- Ideal for cloud or remote operations
- Improved economy of process (upgrades, hotfixes) that benefits all tenants at once

Multitenant Edition is ideal for Software as a Service (SaaS) where hosted applications have high reuse potential across multiple customers and strict data separation is required. It is a deployment strategy that achieves infrastructure and process efficiencies and yet allows for tenant-specific customizations.

Tenants share the entire computing infrastructure, including the hardware, database, file system, application server, Pega Platform, and shared applications and provider. Because of this setup, tenants are subject to security measures that limit the use of traditional development tools and features in Pega Platform.

The multitenant provider can implement customizations on behalf of tenants or can delegate the customization process to the tenant. Multitenant providers are encouraged to implement all tenant customizations on behalf of the tenant.

Ultimately, the question of whether multitenancy is right for an organization is a business decision. While multitenancy imposes additional governance considerations, the infrastructure investment is equivalent to a stand-alone (non-multitenant) system. And, with multitenancy, ROI is accelerated as new tenants are added.

Multitenancy planning

When planning for multitenancy, consider the following items:

- **Multitenant SDLC:** A multitenant provider must establish the infrastructure and guidelines to support a multitenant software development life cycle. This effort can involve deploying and managing an instance of multitenancy in multiple environments, and working with tenants to deploy, test, and monitor applications.
- **Multitenant application architecture:** A multitenant provider must establish the multitenant application architecture, which specifies the shared application (or framework) stack, and the means for tenant customization. A multitenant provider might want to control or guide tenant customizations by constructing a customized shared region, especially in situations where tenants require only minor variations of a common business process.
- **System maintenance:** Maintenance activities in multitenancy affect all tenants. For example, when a system patch is applied, all tenants are affected by the patch. Upgrades and patches cannot be applied on a tenant-by-tenant basis.
- **Tenant life cycle:** The number of tenants and the tenant life cycle must be understood to perform capacity planning and to manage disk and database resources.
- **Tenant security:** A multitenant provider administers privileged features and tenant setup. The administrator in the tenant context is an operator that is created at tenant creation time.

Multitenant architecture

Multitenant Edition implementation is achieved completely through architectural means, in contrast to multi-instance or a custom class-based implementation. Its implementation occurs directly in the core engine, database, and application tiers. Tenants access the same database and web application. This fully shared architecture offers potentially significant operational and administrative efficiencies for both multitenant providers and their customers.

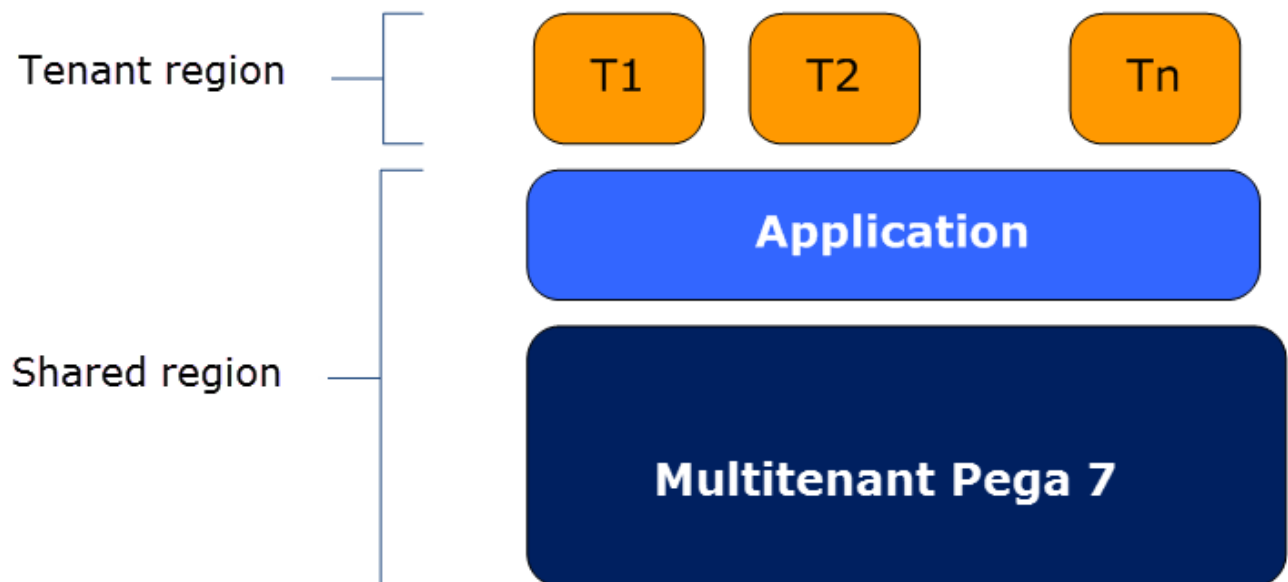
For example, this approach enables tenants to be provisioned when they are created in a matter of seconds or minutes, versus the days or weeks required to deploy a stand-alone system in an enterprise environment.

Multitenancy architecture is guided by these core design objectives:

- Security
- Scalability
- Reuse with the ability to customize shared content and processes per tenant

Multitenant core concepts

Multitenancy achieves its reuse by composing an application into tenant and shared regions. Tenant regions have tenant-specific rules, data, and assets. Shared regions have rules and data that are shared across all tenants.



Tenant representation in multitenancy

Multitenancy uses a tenant ID string to uniquely identify the tenant, its data, and the operations performed by the tenant's operators. The tenant ID is specified when the tenant is created, and it is stored with other tenant metadata in a new instance of the `Data-Admin-Tenant` class.

The tenant ID string is used to generate a tenant-specific URL. Tenant operators use the specific URL to access the tenant. This enables the HTTP session to be aware of the tenant context by embedding the tenant ID in each request that is passed to the system. Using this model, the system can track every request by the tenant, thereby providing strict separation of data and rule visibility among tenants.

Multitenancy Installation

The installation of a multitenancy system is supported in the Pega Platform installers (GUI and command-line) that are included with the distribution package. No additional or special configuration is required beyond what is needed for a standard installation or upgrade. See the installation guide for your platform for more details.

If you select the multitenancy option and the installation or upgrade is complete, the Pega Platform instance cannot be reset to perform as a standard Pega Platform or a high-availability instance.

- If you use the Installation and Upgrade Assistant, select the **Pega 7 Multitenant Edition** option on the Product Edition window.
- If you use the command-line tool, specify `multitenant.system=true` in the `setupDatabase.properties` file prior to running the tool.

When you select the multitenancy option, the following configuration elements are added to the standard installation and upgrade:

- Multitenant administrator access role named PegaRULES:MultiTenantAdmin
- Tenant Management landing pages
- `Data-Admin-Tenant` class
- MTManagement service package

For complete details about installation and upgrade for your application server and database combination, see your distribution package or the deployment guides posted on Pega Community.



Note: Only Oracle and Postgres databases are supported for multitenancy.

Data schema update after upgrading

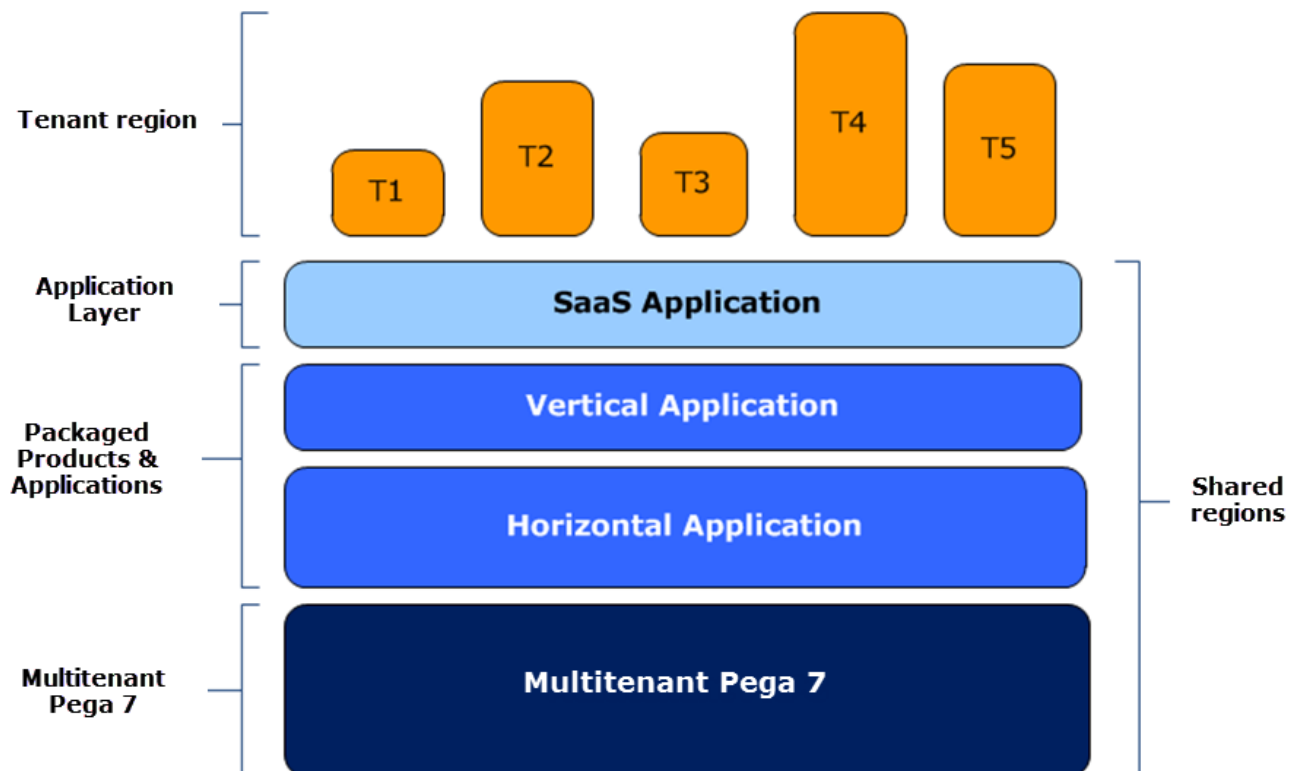
After an out-of-place, multitenant upgrade to Pega 7.2.2 from Pega 7.2 or later, you must update the data schema. After the upgrade, the `PR_DATA_TAG_RELEVANTRECORD` table that is in the existing data schema is not present in the new rules schema. The table must be present in the rules schema for the upgrade to the rules schema to work. Copy the `PR_DATA_TAG_RELEVANTRECORD` table from the existing data schema to the new rules schema, and then upgrade the new rules schema. The `PR_DATA_TAG_RELEVANTRECORD` table is not required to be in the rules schema. After the rules schema upgrade is finished, drop the copy of the table that is in the new rules schema before you upgrade the data schema.

Multitenant application development

The approach for multitenant application development with multitenancy, includes application layering, class design, and the multitenant SDLC. The approach and guidelines can be modified to fit the particular needs of the multitenant provider or tenants. Decisions regarding any aspect of development are the responsibility of the multitenant provider.

Application layering in multitenancy

Multitenancy is designed to work with any Pega product and Industry Application, just like a standard (non-multitenant) system. After multitenancy is installed, additional products and Industry Applications can be deployed to create a logical layering as shown in the following figure. These products and Industry Applications are installed by the multitenant provider, and become the shared building blocks for all tenants on the system. Additionally, a multitenant provider can choose to build an application layer that contains the base application that will be customized per tenant.



The tenant region represents tenant-customized content. The level of customization can vary by tenant or by application, ranging from a customized login screen that displays a tenant's company logo to a highly customized business process that is integrated into other customer systems and processes.

An important application design consideration is determining the division of rules and data classes between the shared and tenant regions. For optimal scalability and query performance, minimize the number of custom rules (or data classes) in a tenant, reusing as much of the shared region logic and processes as feasible.

Conceptually, shared and tenant regions are the same except for a few restrictions on administrative operations. The roles and capabilities of the administrative users in the shared and tenant regions are the same, however, their views of Pega Platform differ. Features that can affect an entire system are disabled in Dev Studio for tenant regions.

Tenant life cycle

The tenant life cycle is bounded by the creation and deletion of the tenant. Other events can be added to the tenant life cycle, such as notification of deletions.

Create tenants from the [Tenant management](#) landing page or from the SOAP service. In either case, an organizational structure, application, and access groups must be created manually in the tenant after its creation or programmatically in the setup activity. Similarly, the tenant can be deleted from the UI or the SOAP service.

For additional information about creating tenants, see [Adding tenants and administrators](#).

Self-service systems allow users to sign up for applications from the web. With a multitenant solution, a tenant is created for each user and similarly, depending on the application, automated deletion of the tenant might be required.

One best practice for multitenant applications is to have a defined tenant life cycle. It is necessary for capacity planning for development, test, and production systems.

Multitenant SDLC model

A common development model is to develop and test the applications in the shared region. Iteration over tenant creation, configuration and the tenant user experience provide feedback for the Pega Build for Change model. The base application can be imported from a Standard Edition system, configured for tenant use with a create tenant setup activity, and subsequently tested in the tenant.

Both the nature and customization requirements of the application drive the number of rules and data that are created in the tenant. Applications with small numbers of overrides at the tenant region are most efficient; however, some applications might be required to have more of the application in the tenant region to ensure that each tenant has a sandbox. It is the responsibility of the provider to ensure that these applications do not affect other tenants or the multitenant system as a whole.

Tenant development restrictions

In a multitenant configuration, from the shared region you have the same set of functionality in Dev Studio as in Standard Edition. The subset functionality and underlying restrictions ensure that tenant development has no effect on other tenants.

Tenant system restrictions

Applications developed must be guardrail-compliant.


- Java development is not allowed in a tenant. Java activity step and JSP development in stream HTML rules are not allowed.
- Custom SQL is prohibited. Any SQL must be reviewed and approved by the multitenant provider prior to deployment. For custom SQL, the multitenant provider should also review plans for SQL queries, because problematic SQL can cause performance issues that affect all tenants.

Dev Studio differences from Standard Edition

- All tenant users are completely blocked from creating rules in shared rulesets.
- Tenants cannot view/browse/modify instances of the `Data-Admin-Tenant` class.
- Tenant users cannot create a new Database Table to Class Mapping.

Rules restricted in tenant context

Restricted rule types can be saved unless they use restricted features such as Java steps, @Java expressions, and custom HTML that has not been autogenerated.

 **Caution:** You can loosen the restrictions on restricted rule types by using the `pxAccessToMTRestrictedRules` privilege that is part of the `PegaRULES:AppArchitect` role that is assigned to the tenant administrator operator by default. Be aware that if you allow access to these rules, tenant developers might be able to access other tenants' data or destabilize the system for all users.

- Rule-Access-When
- Rule-Alias-Function
- Rule-Corr-Fragment
- Rule-Declare-CaseMatch
- Rule-Declare-DecisionTree
- Rule-Declare-Expressions
- Rule-HTML-Paragraph
- Rule-HTML-Property
- Rule-HTML-Section
- Rule-Obj-Corr
- Rule-Obj-Model
- Rule-Obj-When
- Rule-Obj-Validate

Rules blocked in tenant context

Blocked rules cannot be saved or accessed.

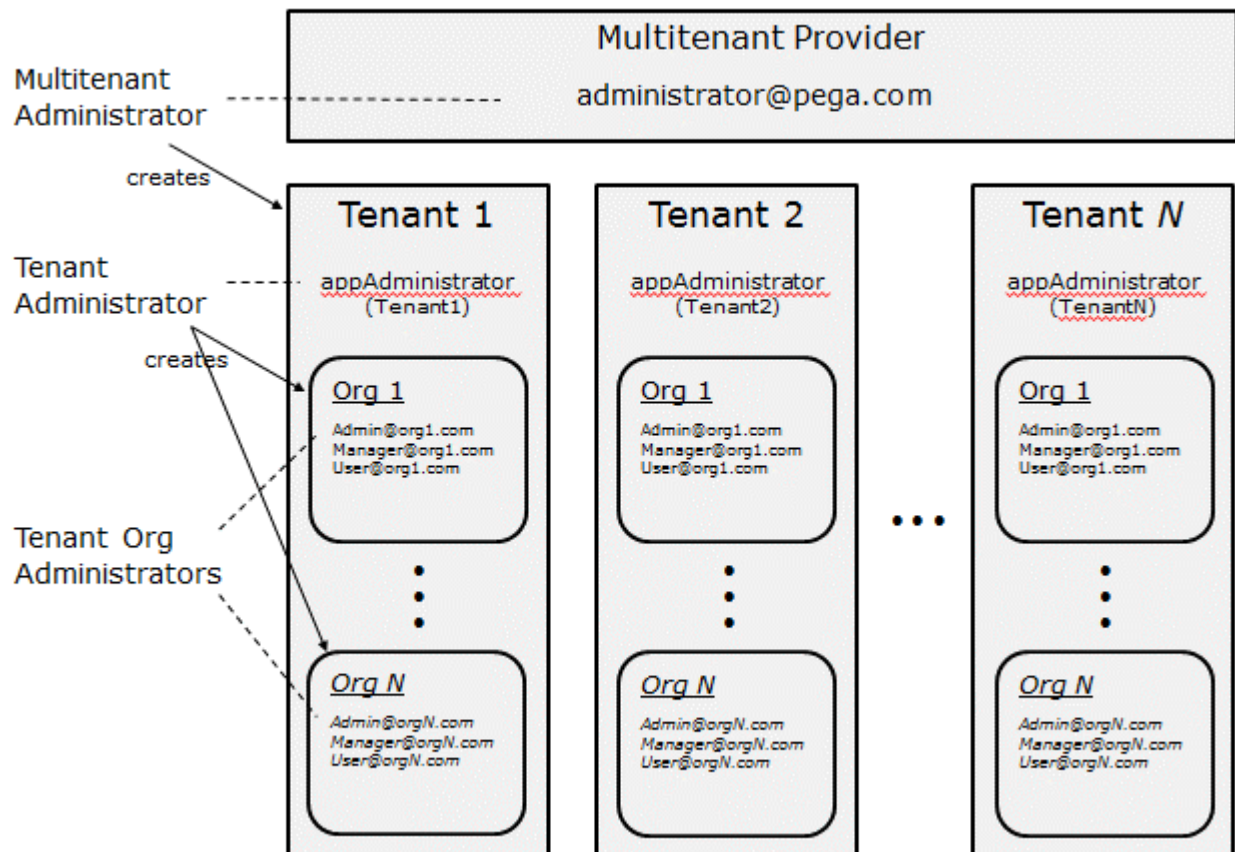
- Data-Admin-Connect-AtomServer
- Data-Admin-Connect-EmailListener
- Data-Admin-Connect-FileListener
- Data-Admin-Connect-FTPServer
- Data-Admin-Connect-JMSListener
- Data-Admin-Connect-JMSMDBListener

- Data-Admin-Connect-JNDIServer
- Data-Admin-Connect-MQListener
- Data-Admin-Connect-MQServer
- Data-Admin-Connect-SOAPServer
- Rule-Agent-Queue
- Rule-Connect-Cassandra
- Rule-Connect-CMIS
- Rule-Connect-dotNet
- Rule-Connect-EJB
- Rule-Connect-File
- Rule-Connect-HBase
- Rule-Connect-HTTP
- Rule-Connect-Java
- Rule-Connect-JCA
- Rule-Connect-JMS
- Rule-Connect-MQ
- Rule-Connect-SAP
- Rule-Connect-SAPJCo
- Rule-Connect-SQL
- Rule-Edit-Input
- Rule-Edit-Validate
- Rule-File-Form
- Rule-File-Text
- Rule-HTML-FixedList
- Rule-HTML-Fragment
- Rule-Obj-HTML
- Rule-Obj-JSP
- Rule-RDB-SQL
- Rule-Report-RuleBase
- Rule-Service-BPEL
- Rule-Service-COM
- Rule-Service-dotNet
- Rule-Service-EJB
- Rule-Service-Email
- Rule-Service-File
- Rule-Service-HTTP
- Rule-Service-Java

- Rule-Service-JMS
- Rule-Service-JSR94
- Rule-Service-MQ
- Rule-Service-Portlet
- Rule-Service-SAP
- Rule-Service-SAPJCo
- Rule-Utility-Function
- Rule-Utility-Library

Multitenant security

Standard and Multitenant Editions use the same security model. A shared region operator requires the PegaRULES:MultiTenantAdmin role to access tenant administration, product installations and upgrades, and to apply patches. After a tenant region is created, the administrator of the tenant is created with the PegaRULES:AppArchitect role. The tenant application administrator can then create applications, organizations, and operators within the tenant.



Roles, access groups, and privileges

The roles and access groups relevant to a multitenant system are summarized in the following table.

Scope	Role / actor	Access group	Pega Platform roles
Multitenant provider	Multitenant administrator	PRPC:Administrators	PegaRULES:SysAdm4 PegaRULES:MultiTenantAdmin PegaRULES:SecurityAdministrator PegaRULES:HighAvailabilityAdministrator

Scope	Role / actor	Access group	Pega Platform roles
Tenant	Administrator	[TenantName]:Administrators	PegaRULES:AppArchitect PegaRULES:SysOpsObserver

File system temporary directory

A tenant cannot access another tenant's files with the Pega-provided PFile APIs for file access. Each tenant is allocated a separate temporary directory that is named with the tenant's ID under the base temporary directory. Therefore, temporary files such as static content cache files, diagnostic trace files, exported files, and so on, that are created during the tenant application processing will be available in the tenant's temporary directory. Unlike rules, shared region files are not accessible to a tenant.

Multitenant system deployment planning

The following considerations are applicable when planning for a multitenant-enabled system.

- Number of tenants and tenant load profile – The number of tenants and tenant load profile influence two key areas: the infrastructure architecture, such as the number of JVMs required to host all tenants, and the operations or system maintenance plan, which will need to accommodate increasing support demands from the tenant base as more and more tenants are added.
- Tenant customization requirements – The multitenant provider builds the “shared” stack with a design that facilitates tenant customizations. In cases where tenants can make their own modifications, the multitenant provider is responsible for any effect that tenant modifications might have on the system.
 - 📌 **Note:** To maximize performance, minimize the number of customized rules and data classes in the tenant region.
- Centralized management requirements – The number of tenants created and the tenant life cycle will drive the need for a self-service application or manual use of tenant management SOAP calls.
- System utilization and monitoring – Before deploying a multitenant system, the multitenant provider determines how the system will be monitored. Monitoring is particularly important on a multitenant system because system resources (CPU, memory, disk) are shared and not allocated by the tenant.
- Maintenance and communication plan – With multitenancy, some system management and maintenance activities will affect all tenants, in particular upgrades and hotfixes. The multitenant provider ensures that such activities are communicated to all tenants in advance.

Tenant life cycle administration

A multitenant administrator can create and maintain the tenants on a multitenant system either by using the Tenant Management landing page or by calling a SOAP service for remote system management.

For step-by-step details, see the following tutorials on Pega Community:

- *Quick Start: Logging in as the multitenant administrator*
- *Quick Start: Creating a tenant*
- *Quick Start: Logging in as the tenant administrator*
- *Quick Start: Generating stubs for multitenant SOAP services*
- *Quick Start: Creating a tenant by using the SOAP service operation*

Tenant management

The Tenant Management landing page is visible when the instance has been enabled as a multitenant system during installation. It is accessible only to the multitenant administrator who uses it to manage the tenants in the system.

To access the landing page from the header of Dev Studio, click **Configure > System > Tenant Management**.

Viewing tenant information

The multitenant administrator can view a read-only list of active system tenants and information about each of them.

- **Tenant Name** – The tenant name.
- **Description** – Useful for systems with multiple shared-region applications.
- **Contact Email** – Email address of the tenant creator.
- **Created On** – The date and time the tenant was created.
- **Last Sign On** – The date and time a user last logged in to this tenant.

To view additional details about a specific tenant, select a row and click next to the row number to expand the tenant. In addition to the tenant name, the administrator ID, and the login URL of the tenant are displayed.

Adding tenants and administrators

The multitenant administrator can add tenants and their administrator to the system. When the tenant is created, the tenant organization, division, and unit, as well as a `Data-Admin-Tenant` class instance for the tenant, are automatically created.

1. In the **Tenant Name** field, enter the name of the system tenant. Systems for public signup will auto generate the name.
2. In the **Description** field, enter a short (64 char) description useful for systems with multiple shared-region applications.
3. In the **Contact Full Name** field, enter the name of the person who is the contact for this tenant.
4. In the **Contact Email** field, enter the email address of the tenant creator.

5. In the **Setup Activity** field, enter the activity that is invoked in the tenant region after the tenant record has been created. In the user interface, this activity is invoked with the access group of the logged-in user.



Note: The default is to not select an activity. Add activities to the `Data-Admin-Tenant` class to have them accessible when you create a tenant.

6. Optional: Select the `ImportRAPs` value if you want to launch a sample setup.
 - a) Create all RAP files (both `.jar` and `.zip` files are supported) that the user would like to pass to the `ImportRAPs SetupActivity`.
 - b) Upload each RAP file into its own `Rule-File-Binary` instance in the shared region. The application specified for the `Rule-File-Binary` instances must be `webwb`.
 - c) Navigate to the **Create** tab of the Tenant Management landing page and fill in all required fields.
 - d) Select **ImportRAPs** from the **SetupActivities** list.
 - e) In the **FileList** parameter input field that is now visible, enter a comma-separated list of the names of the `Rule-File-Binary`, for example, `abc.jar, def.zip`, or if importing from only one `Rule-File-Binary` instance, for example, `abc.jar`.
 - f) Click **Create Tenant** and wait for the process to complete.

The preceding steps imply that the operator who creates the `Rule-File-Binary` also creates the tenant. However, a different operator can create the tenant. In this case, the operator who creates the tenant must have access to the `Rule-File-Binary` through the operator's primary access group.

7. In the **Administrator** field, enter the user ID of the tenant administrator.
8. In the **Password** field, enter the password of the tenant's administrator.
9. Click **Create Tenant** to create the tenant and the organization. This creates an instance of `Data-Admin-Tenant`.

Managing tenants

The multitenant administrator can update the password of a tenant's administrator and delete a tenant from the **Create** tab of the System-Tenant landing page.

1. Select the tenant that you want to manage. Use the Auto-complete control to enter starting text, and then press the arrow key to see a list of existing tenants in the system.
2. In the **Update Administrator** section of the tab, you can update the password of the administrator for the selected tenant.

Deleting tenants

In the **Delete Tenant** section of the tab, you can delete the selected tenant. When you delete a tenant, its rules, data, work objects, users, and configuration are permanently deleted from the application.



Note: After you start this process, it cannot be canceled or undone. Optionally, a Post Delete activity, a custom activity defined in `MTManagement`, will run after the tenant has been deleted.

Multitenancy SOAP services

Multitenancy includes several SOAP services operations that allow remote administration of the system. You can use these services to create and delete tenants, or return metadata about existing tenants, for example, the last login time. The services are implemented in the `MTManagement` service package.

WSDL is available from:

[app server] /prweb/PRSOAPServlet/shared/SOAP/MTManagement/Service?WSDL

Methods

Service type

Rule-Service-SOAP ▾

Service type	Class name	Method name	Ruleset
Rule-Service-SOAP Service		CreateTenant	Pega-LP-SystemSettings:07-10-11
Rule-Service-SOAP Service		DeleteTenants	Pega-LP-SystemSettings:07-10-11
Rule-Service-SOAP Service		GetDeleteTenantsStatus	Pega-LP-SystemSettings:07-10-11
Rule-Service-SOAP Service		GetTenantDetails	Pega-LP-SystemSettings:07-10-11
Rule-Service-SOAP Service		GetTenantStatus	Pega-LP-SystemSettings:07-10-11

CreateTenant service

Use this service to create a new tenant. The service has all of the same functionality that is available on the Tenant Creation landing page, and adds additional capabilities that are available only in the service implementation. Specifically, the service allows custom parameters to be passed to the post-setup activity allowing client applications to vary the configuration of a tenant. For example, the sample ImportRAPs activity has been enhanced to load different RAP files from different directories based on specific parameter values.

The service takes the following parameters as arguments:

- Tenant ID
- Description
- Contact Name
- Contact Email
- Activity Name (optional)
- AccessGroup (if an access group different from the caller is required)
- Param Name (optional)
- Param Value (optional)
- Administrator
 - Username
 - Password
- Organization Name (for backward compatibility)
- Web Domain Type (for backward compatibility)

The service returns a status message of success or failure.

- If the service request is successful, the tenant's generated URL is returned in the response.
- If the service fails, the service response contains an error code or error message.

- If an optional post-setup activity is specified, the service returns a job ID. If a job ID is returned, it can be used by the `getTenantStatus` service to determine the status of the tenant creation process.

[Tenant life cycle administration](#)

[Managing tenants](#)

DeleteTenants service

Use this service to delete one or more tenants. It takes the Tenant ID parameter as an argument.

One or more tenants can be specified as an enumerated list. The service returns a job ID string. This string can be passed to the `GetTenantDeletesStatus` service operation to determine if the delete operation was successful.

GetDeleteTenantsStatus service

Use this service to delete one or more tenants. It takes the Job ID parameter as an argument.

The service returns the status of the tenant delete operation for each tenant associated with the job ID.



Note: This service is used with the `DeleteTenants` service, which is usually called first. The `DeleteTenants` service operation returns a Job ID, which is used as input for the `GetDeleteTenantsStatus` service operation.

GetTenantDetails service

This service returns metadata about a tenant, including tenant creation time, and the last login time. It takes the Tenant ID parameter as an argument.

GetTenantStatus service

This service returns the status of the tenant creation process. It takes the Job ID parameter as an argument.

Use this service to check on the status of any tenant creation process that optionally specifies a post-setup activity.



Note: This service is used with the `CreateTenant` service, which is usually called first. The `CreateTenant` service operation returns a Job ID, which is used as input for the `GetTenantStatus` service operation.