PEGA 7 PLATFORM

Pega 7.2.2 High Availability

Administration Guide

Pega®

Trademarks

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. may make improvements and/or changes to the publication at any time.

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: Pega 7.2.2 High Availability Administration User Guide

SOFTWARE VERSION: 7.3

PUBLISHED: Friday, December 09, 2016

Contents

Overview of high availability 3

High availability architecture 4

Privileges and roles for high availability management 4

Checklist for rolling out high availability 5

Configuring a highly available environment 6

Load balancers 6

Configuring session affinity 7

Application tier architecture 7

Single sign-on 8

Shared storage 8

Server redundancy 8

Database tier architecture 8

Split schema database architecture 9

Enterprise configuration of high availability 9

High availability developer environments 10

Configuring ruleset update behavior 10

Managing high availability environment variable settings 11

Updating and upgrading highly available systems 12

High availability update process 12

Setting up split schema 13

Migrating rules to a new schema 13

Migrating auto-generated rules 14

Quiesce 15

Quiesce using slow drain 15

Quiescing a node when using slow drain 15

HA Cluster Management 16

Quiesce using immediate drain 17

On-premises quiesce flow 18

Quiescing a node when using immediate drain 19

HA Cluster Settings 20

Passivation and activation 21

Setting a custom passivation mechanism 21

Quiesce behavior during upgrades 22

Cluster management 23

Cluster management using Autonomic Event Services (AES) 24

Cluster management using the System Management application (SMA) 24

Cluster management using MBeans 25

High availability integration services 26

Listeners 26

Service packages 26

Service request processors 26

Service rules 27

Crash recovery 28

Server crash recovery 28

Browser crash recovery 29

Dynamic containers and HTML5 29

Enabling crash notification 29

Enabling periodic work recording 30

Overview of high availability

High availability is the ability for an application to be accessible with minimal or no downtime, irrespective of various situations that include hardware failure, process crashes due to insufficient resources, memory leaks, maintenance, application upgrade activities, or natural disasters. The Pega 7 Platform provides tools to meet high level production service level agreements for mission critical applications with high availability features.

High availability features are available for use with the Pega 7 Platform. Any applications built in a version prior to the Pega 7 Platform are unable to use high availability features until they are upgraded to a Pega 7 Platform version.

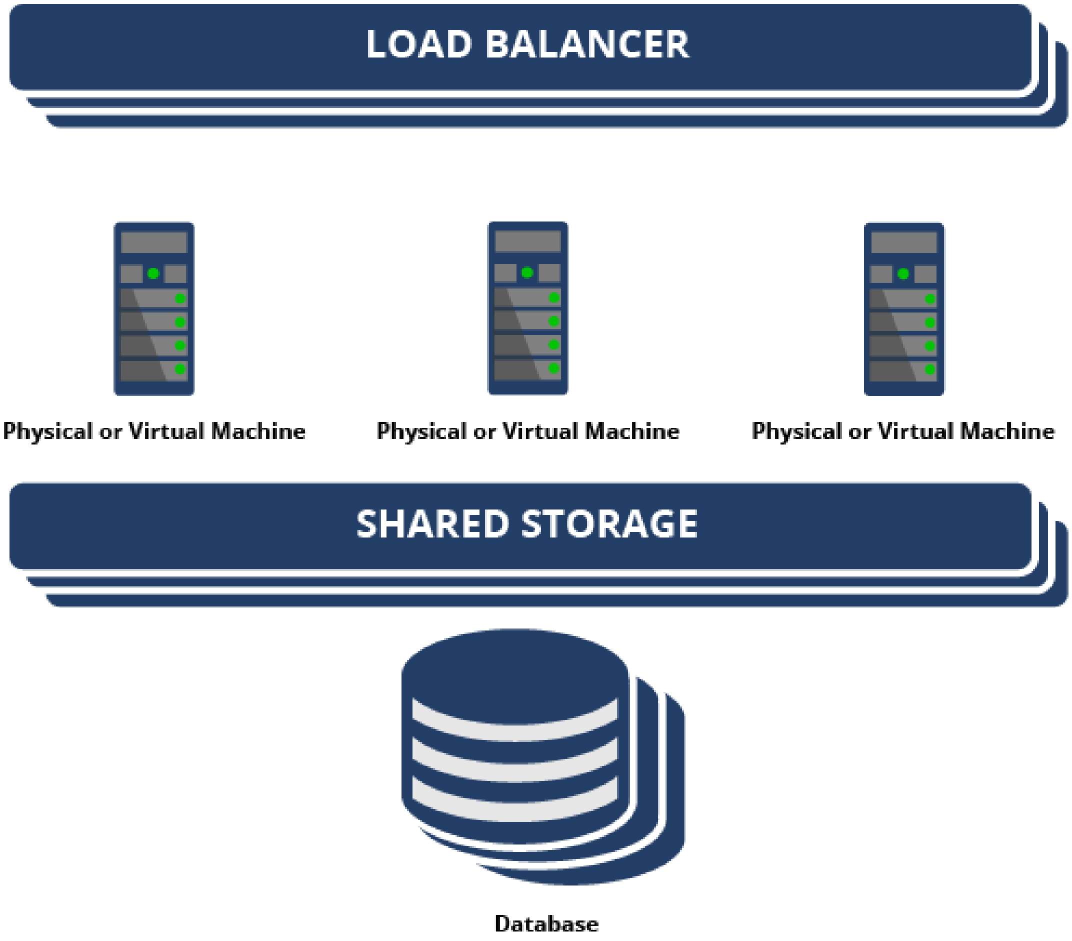
High availability features include:

- Application server maintenance
    - Operations staff can initiate application tier maintenance that is transparent to users.
    - Pega 7 Platform web application servers that require maintenance can be quiesced.
    - Users on a server that has been quiesced are redirected to other servers in a cluster.
    - Rolling server restarts can be used to upgrade Java Virtual Machine (JVM) settings, physical server maintenance, or Pega 7 Platform upgrades.
- Application development guidelines
    - System Architects and developers have guidelines to assist them in developing and maintaining highly available Pega 7 Platform applications and strategic applications.
    - Pega 7 Platform System Administrators can control how rules are introduced into the production environment by locking rulesets.
- Pega 7 Platform updates and upgrades
    - Operations staff can upgrade the Pega 7 Platform without taking production systems offline.
    - Split schema enables administrators to move users on the old rule base to the new rule base with a rolling restart.
- Network Operation Center integration
    - Operations staff can use Autonomic Event Services (AES) or MBeans to integrate high availability features into their operation centers.
- Crash recovery
    - Operations staff can rely on application tier crash recovery to recover the user’s work when they log in to other Pega 7 Platform servers in the cluster after a crash.
    - Administrators can optionally notify users that their session has been recovered.
    - Seamless recovery in the event of a browser crash is provided.

High availability architecture

The Pega 7 Platform supports production service level agreements and emphasizes redundancy as a best practice for physical and software-based highly available production architectures.

In the following diagram, redundancy is applied to the infrastructure across all tiers of the hardware and deployment architecture of Pega 7 Platform servers, represented by the shadowed icons. For example, one implementation can have several load balancers, physical and virtual machines, shared storage repositories, and databases. These principles are general to three-tier architectures and are applicable to private hosting or cloud environments.


LOAD BALANCER
Physical or Virtual Machine
Physical or Virtual Machine
Physical or Virtual Machine
SHARED STORAGE
Database

Privileges and roles for high availability management

Various privileges and roles are used for Pega 7 Platform high availability management, Pega 7 Platform server investigation, and security.

High availability privileges

The following privileges are available:

- pxHighAvailabilityAdmin — Operators with this privilege can access the **HA Cluster Settings** landing page from the Designer Studio menu. The settings are cluster-wide.
- pxHighAvailabilityAdminQuiesce — Operators with this privilege can access the **HA Cluster Management** landing page from the Designer Studio menu. In situations where cluster management is handled by operations centers, this privilege does not need to be granted because a Network Operation center uses AES or MBeans instead of the landing page for high availability administration.

High availability roles

The PegaRULES:HighAvailabilityAdministrator role grants access to high availability cluster administration and quiesce investigation. Cluster administration in this context is access to cluster management and high availability settings that are accessible from the corresponding Pega 7 Platform landing pages. This role has the following privileges:

- pxHighAvailabilityAdmin
- pxHighAvailabilityAdminQuiesce

Checklist for rolling out high availability

The following list outlines the process that project managers should follow when creating a new highly available system using a Pega 7 Platform installation.

- Understand the production requirements
  - Define the service level agreement for the production system.
  - Perform capacity planning to understand the necessary computational requirements to meet the required service level agreement.
- Deploy the physical architecture
  - Review the physical architecture of the system.
  - Verify hardware requirements.
  - Deploy a load balancer.
  - Select a shared storage solution.
  - Install Pega 7 Platform instances on Pega 7 Platform servers using split schema.
- Configure the Pega 7 Platform
  - Configure cookie invalidation.
  - Configure quiesce.
  - Configure crash recovery.
- Build applications or migrate existing applications
- Perform system testing

Configuring a highly available environment

Creating a highly available system typically requires coordination across multiple groups in an organization. Individuals or groups responsible for deploying highly available systems must collectively understand and agree on how to configure the Pega 7 Platform for their organization.

The smallest environment for basic configuration testing requires the following:

- A load balancer
- Two instances of a web application platform of choice
- A database backend
- A shared disk for shared storage

Additionally, as a part of planning for future server maintenance, you must decide whether to use the default slow drain method or configure your system to instead use the immediate drain method for when you perform the quiesce process on a node.

Load balancers

User and service requests are passed through a load balancer to the Pega 7 Platform servers. The Pega 7 Platform server in turn makes requests to a database.

All load balancers, regardless if using an on-premises or cloud Pega 7 Platform installation, use a single quiesce implementation. This allows for a quiesced node to have all users (both new and existing) be passivated immediately and for the node to be removed simultaneously.

To deploy a highly available production system, a load balancer must be used that has the following minimum requirements:

- Session-based affinity support to ensure that requests from one user are directed to a specific application server that maintains the state for that user
- Ability to disable a Pega 7 Platform server or take the Pega 7 Platform server out of the load balancer rotation to facilitate shutdown
- Automatic active or passive Pega 7 Platform server health monitoring for Pega 7 Platform server failure detection
- Scripting to manage cookies, invalidation, and optionally for allocation and deallocation of Pega 7 Platform servers based on production load

Note: Highly available system architectures should support fail-over to a redundant load balancer.

Configuring session affinity

Session affinity is configured with the load balancer. It ensures that all requests from a user are handled by the same Pega 7 Platform server.

Production load balancers offer a range of options for configuring session affinity. The Pega 7 Platform supports cookie-based affinity. Configure cookies for high availability session affinity using the following variables.

session/ha/quiesce/customSessionInvalidationMethod

Note: Session affinity configuration is only applicable when using slow drain for quiesce.

- Value — configurable-cookie-invalidator or fully qualified class name of a class implementing the SessionInvalidationMethod in prpublic
- Functionality — Class name of the invalidation method in use
  - Set to configurable-cookie-invalidator to perform cookie-based invalidation using the value of session/ha/quiesce/cookieToInvalidate as the name of the cookie to invalidate
  - Set to the fully qualified name of a class implementing the interface SessionInvalidationMethod in prpublic

session/ha/quiesce/cookieToInvalidate

Note: Session affinity configuration is only applicable when using slow drain for quiesce.

- Value — Name of the cookie to invalidate
- Functionality — Name of the cookie to invalidate when using the configurable-cookie-invalidator customSessionInvalidationMethod

Application tier architecture

Highly available application tier architectures must have enough computing power to support fail-over, as well as a means to allocate new servers to support increased demands for service.

To deploy a highly available production system, two or more physical or virtual machines are needed. These allow for horizontal cluster scale and provide hardware redundancy to maximize efficiency and offer a solution for crash recovery. Each machine can deploy one or more Pega 7 Platform servers.

Organizations can employ vertical cluster scale by adding multiple Pega 7 Platform server instances to support production load. Determine the number of Pega 7 Platform server instances based on capacity planning and by evaluating operational thresholds.

For information on supported application server platform versions, see the *Platform Support Guide* on the PDN.

Single sign-on

Pega 7 Platform high availability features are secure and require re-authentication in the event of a Pega 7 Platform server crash. The Pega 7 Platform retains operator authorization for redirection of a user during server quiesce. A single sign-on solution, although not required, enables a seamless user experience.

Shared storage

When high availability is enabled, database passivation is used as the default passivation method, in which the database handles all storage requirements.

However, if you decide to instead use filesystem passivation, you must select a shared storage solution and need to implement the shared storage API to integrate with the Pega 7 Platform. This is because application servers require fault-tolerant shared storage in order to facilitate initiated and uninitiated shutdowns when using filesystem passivation. The Pega 7 Platform supports shared storage using Network File Storage (NFS) or shared disk out of the box when using filesystem passivation, but these solutions are not inherently fault tolerant.

- For NFS solutions, the shared storage must point to an NFS location to support access from all nodes.
- For disk-oriented solutions, Pega 7 Platform servers require read/write access to shared storage .

If custom shared storage implementations are required, the following plug-in for shared storage must be used: com.pega.pegarules.pub.session.CustomPassivationMechanism

For details about using this plug-in, see PDN article *Creating a custom passivation method*.

Server redundancy

For consistency, the term "Pega 7 Platform server" is used for each web application server instance (JVM) on a physical or virtual machine. The decision to configure multiple Pega 7 Platform servers on physical or virtual machines must be coupled with redundancy at the machine level.

Pega 7 Platform servers must be designed to support redundancy among Pega 7 Platform various components, such as connectors, services, listeners, and search. The exact configuration varies based on the specifics of the applications in the production environment.

Database tier architecture

Pega 7 Platform high availability is database-agnostic. The recommended database features are:

- Highly available clustered databases or third party or database vendor software that supports this feature

- Database fault tolerance that meets production service level agreement requirements — for example, some databases have add-on products to handle failover of hardware or software sub-systems, enabling another piece to take over

For supported database platform versions, see the *Platform Support Guide* on the PDN.

Split schema database architecture

The Pega 7 Platform supports a split schema database architecture. The split separates rules and data into separate schemas and is essential for enabling minimal to zero down time during Pega 7 Platform application and strategic application upgrades or patch installation.

The use of split schema architecture enables Pega 7 Platform System Administrators to install and upgrade a new rules schema in the production database while the old schema is still in use. Pega 7 Platform high availability features can then be used to migrate users from the old rule schema to the new schema on the production system to complete the upgrade.

Enterprise configuration of high availability

Pega 7 Platform high availability can be configured to run on Java Platform, Enterprise Edition (Java EE) application servers, such as those provided by WebSphere, Weblogic, or JBoss. These enterprise offerings provide services and packaged components that can add other aspects of high availability when properly configured. The primary advantage is that enterprise application servers typically support redundant message queues and buses, which can allow Pega 7 Platform services and listeners to continue during initiated and uninitiated outages.

The configuration and management details for each platform vary, but the concepts remain the same. Pega 7 Platform servers must be configured as a cluster so that there are two or more instances that are eligible to process work from any given queue or bus. During normal operation, only one Pega 7 Platform server should be used to process these requests, and other Pega 7 Platform servers in the cluster should be set up to address the queue or bus.

During failover conditions, the application server management utilities can be used to reconfigure queue destinations and bus listeners.

- If an outage is initiated by an administrator (for example, if it is planned), then the administrator can update the configuration details for the Pega 7 Platform server when modifying the load balancer to remove it from service.
- If the outage is uninitiated, the Pega 7 Platform server that is down no longer processes work while other Pega 7 Platform servers in the cluster continue processing queue items.

In either situation, no loss of work should occur and work continues to be processed (although it might be slower due to a loss of a Pega 7 Platform server). Other bus members might need to be enabled if the system is configured to have a single active Pega 7 Platform server that is processing work.

Pega 7 Platform listeners and agents that use URLs should be configured to use a load-balanced URL. However, listeners that are configured as MBeans need to be managed from the application server. If a

Pega 7 Platform server is to undergo an initiated shutdown, the application server must be configured so that the server does not attempt any MBean invocations on that Pega 7 Platform server. This is because the Pega 7 Platform server is not able to process the request.

If the target server of an MBean invocation has experienced an uninitiated shutdown, the MBean invocation fails. Depending on the application server configuration and the Pega 7 Platform listener configuration, this invocation might either be placed back in the queue or be lost.

Because of the complication in configuring the Pega 7 Platform high availability edition on enterprise application servers, organizations should plan to set up the system in conjunction with Pegasystems Consulting. The Pegasystems Services team has access to resources that can help with configuring specific platforms for maximum availability.

High availability developer environments

Pega 7 Platform developer environments can take partial advantage of high availability features.

Maintenance for application tier physical and virtual machines and Pega 7 Platform server maintenance can be performed in a transparent manner for developers creating frameworks and applications. However, some aspects of development, such as debugging, might be impacted.

Note: Crash recovery and Pega 7 Platform upgrades are currently not supported for developer environments.

Configuring ruleset update behavior

You can update the following prconfig setting to determine when users see new ruleset lists due to changes to Access Groups or Rule-Application instances: either immediately at the start of the next thread (threadset) or at the next session (fixedreq).

Authorization/RSLUpdateBehavior

- Value Type — String
- Functionality — Alters the behavior of ruleset list updates. The value options for this setting are:
  - Immediate
    - The default setting
    - Ruleset lists are updated as application and access group changes are made
    - The shipped behavior of PRPC versions prior to the Pega 7 Platform
  - Threadset
    - Ruleset lists are updated from application and access group changes at thread creation boundaries and stay the same for the entire lifetime of the thread
    - Thread creation boundaries include thread creation and thread switch events

    - Fixedreq
        - A snapshot of the ruleset lists are taken for all available applications at login
        - Fixed throughout the lifetime of the session
        - If a requestor with RSL update behavior set to fixedReq spawns a child requestor, the child requestor gets the most recent context using their parent's access group as a key
- Example: `<env name="Authorization/RSLUpdateBehavior" value= "threadset" />`

Managing high availability environment variable settings

There are multiple ways to set and manage Pega 7 Platform settings. Following are recommended methods listed in terms of precedence:

- Data-Admin-System Settings (DASS) instances for each cluster
- Java Naming and Directory Interface (JNDI) settings for a cluster
- Individual prconfig.xml files for each Pega 7 Platform server

Note: Organizations might have specific preferences that differ from the order listed above.

Using DASS instances

Based on the environment setup, administrators should use Data-Admin-System Settings (DASS) instances for updating cluster-wide settings or for dynamic control of high availability settings. In cases that require finer Pega 7 Platform server control and where restarts are acceptable, administrators should use JNDI or prconfig settings for high availability-specific settings.

Note: Some settings might require a system restart, even if they are configured using DASS instances.

Using JNDI settings

Administrators can use a shared prconfig.xml location by specifying the file URL using JNDI. This can be used for clusters and for Pega 7 Platform server groups.

Individual Pega 7 Platform servers can set specific settings directly using JNDI. See the specific Installation and Upgrade guides for your database and application server for details on how to set Pega 7 Platform settings using JNDI.

Using prconfig files

To make the high availability settings page available and allow the DASS instances to be updated from high availability landing pages, all DASS instances in the cluster must have the following prconfig.xml setting:

```
<env name="initialization/settingsource" value="merged" />
```

Updating and upgrading highly available systems

The Pega 7 Platform supports the separation of rule tables and data tables into separate database schema for the purpose of minimizing the downtime of production during updates of applications, frameworks, and the Pega 7 Platform itself. Use split schema when performing updates that range from staging environments to production environments, as well as to facilitate rolling restarts on production servers.

The schema composition is logically composed of both:

- A rules schema, containing:
  - The rule base
  - System data
- A data schema, containing:
  - Data objects
  - Work objects

If split schema is deployed in production, Pega 7 Platform updates with zero down time are possible. Organizations seeking to use rolling restarts and split schema to perform minimal or no down time upgrades of internal frameworks and applications must plan for production data management.

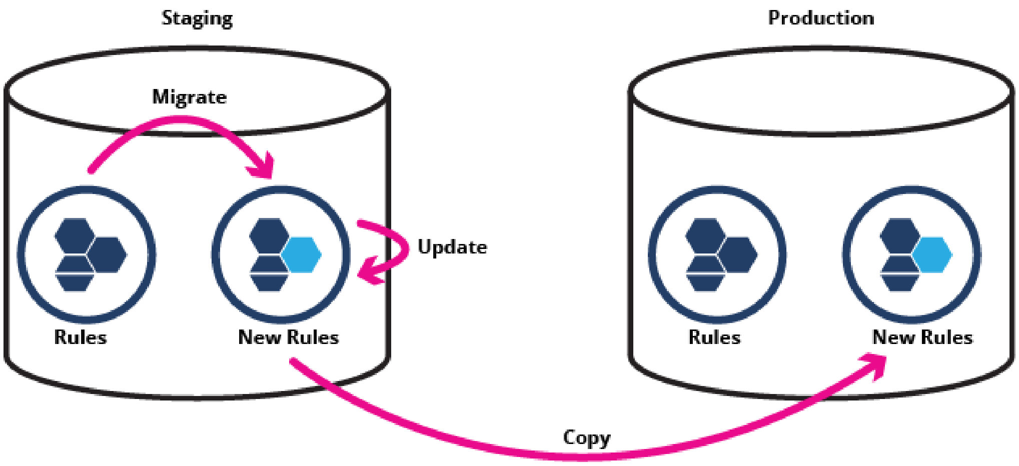
To ensure that data is migrated to the production system:

- If new rules and data are backward compatible, import the application or framework RAP into production after performing the rolling restart. Any new content in existing data tables and changes to workbaskets are visible immediately.
  - To make new rules available upon next login, configure Authorization/RSLUpdateBehavior = fixedreq.
- If new rules and data are not backward compatible, import the application or framework RAP into production after performing the rolling restart at a period of low to no activity. New rules, workbaskets, and data tables are visible immediately upon next login.

High availability update process

The recommended update process prior to production system upgrade includes roll out in development, testing, and acceptance systems. Typically, a company might have a staging system for performing the initial update and transitioning the upgraded rules to production.

On the diagram, you can see an example of a customer application or framework update where the staging system has a copy of the production rules schema. This copy is essential to capture auto-generated rules. The data schema is not shown.



Setting up split schema

Both the **prconfig.xml** and **prbootstrap.properties** file must be updated when using split schema.

1. Add the following environment variables to the **prconfig.xml** file:

   <env name="database/databases/PegaRULES/defaultSchema" value="rulesSchema_Name"/>

   <env name="database/databases/PegaDATA/defaultSchema" value="dataSchema_Name" />

2. For a split schema setup, add the following properties to the **prbootstrap.properties** file:

   com.pega.pegarules.bootstrap.allclasses.schema=" rulesSchema_Name "

   com.pega.pegarules.bootstrap.datatables.schema=" dataSchema_Name "

Note: This functionality is available when you purchase and install the BIX application.

Migrating rules to a new schema

Rules can be migrated to a new schema in a highly available system, which is updated on the staging system and then copied to production. To migrate the rules to a new schema, do the following:

1. Migrate the existing rules into a new rules schema and mark the time.
2. Perform the update on the new rules schema, for example, a framework or application update.
3. Copy the new rules to the production database.
4. Configure the Pega 7 Platform servers to use the new rules schema and perform rolling restarts on any additional production servers.
   - During the rolling restart, users are moved from Pega 7 Platform servers using the old rules schema to Pega 7 Platform servers using the new rules schema.

- The schema names to use are provided through JNDI by using the application server console.

5. Migrate auto-generated rules that were created after step 1.

Migrating auto-generated rules

Some Pega 7 Platform applications have auto-generated rules. These rules are created at runtime during the normal use of a production system.

Note: Auto-generated rules are not present in the upgraded rule base from the time of the initial rule migration and must be migrated manually.

There is a built-in utility to export the auto-generated rules created after a certain time period. Use the prpcUtils.properties utility, located in the coreBuildDistributionImage/scripts/utils directory, to perform the export.

To run the prpcUtils.properties utility, do the following:

1. Create an XML file containing the following classes to export:
   ```
   <?xml version="1.0" encoding="UTF-8" ?>
   <ExportClassList>
   <Database dbname="PegaRULES">
   <Class name="Rule-Obj-Property" column="PXUPDATEDATETIME"/>
   <Class name="Rule-Obj-When" column="PXUPDATEDATETIME"/>
   <Class name="Rule-Obj-FlowAction" column="PXUPDATEDATETIME"/>
   </Database>
   </ExportClassList>
   ```
   - Class name represents either:
     - The name of the class for a rule type that you want to export
     - A wild card pattern, such as parent class name (for example, Rule-), used to export all the rules belonging to the Rule- hierarchy, including its descendants.
   - PXUPDATEDATETIME represents the exposed column, which could be used to match the time stamp in the export.migration.date property in the PRPCUtils.properties file that is set in step 3.
2. Enter the database connection information in the utility by setting the following properties under its Settings for Export Tool section:
   - export.archive.full.path
   - export.migration.date
   - export.migration.xml – the full-path to the XML file containing the classes to export that was created in step 1.
3. Run the prpcUtils.bat file in the same directory as the properties file with the following syntax:
   ```
   C:\coreBuildDistributionImage\scripts\utils>prpcUtils.bat export
   ```

Quiesce

Quiesce provides the ability to gracefully take a Pega 7 Platform server out of service for maintenance or other activities. During quiesce, active users are seamlessly migrated to an active node when the quiescing node is no longer available.

You can quiesce a server using one of the following methods:

- Slow drain
- Immediate drain (default)

To switch between slow drain and immediate drain:

1. Update the value of the session/ha/quiesce/strategy prconfig setting to be either "slowDrain" or "immediateDrain".
2. Restart the server.

Note: When using the immediate drain method to perform quiesce on a node, any operator can access a quiesced server for root cause analysis or remediation, regardless of their user role or privileges. For quiesced server access when using the slow drain method, you must still include either the PegaRULES:HighAvailabilityAdministrator or PegaRULES:HighAvailabilityQuiesceInvestigator user role in a user's access group. Administrators using the slow drain method without either of these user roles are exiled from the quiesced node and are redirected to an active node.

Quiesce using slow drain

When using slow drain for quiesce on a highly available system, the following steps occur on a Pega 7 Platform server when the quiesce process is enabled:

1. The default timeout for passivation is reduced, referred to as an accelerated passivation period.
2. Users are placed in a passivation queue.
3. The system stops non-essential agent processing for quiesce and all listeners, with the exception of message-driven bean (MDB) listeners. (The application server should be configured to ensure that no MDB invocations occur.)
   Note: System critical agents remain active during this step.
4. When the active user requestor count drops to 0, the system sets the server state to "Quiesce Complete" and the Pega 7 Platform server is ready for maintenance.

Note: All Pega 7 Platform servers on a physical or virtual machine must be quiesced for maintenance on the machine.

Quiescing a node when using slow drain

When using high availability, you can perform the quiesce process for a node to remove it from a pool of active nodes.

1. Identify the Pega 7 Platform servers that you want to shut down.
2. Configure the load balancer to disable the identified servers.
   Note: This does not allow new connections, but allows existing users and services to complete work.
3. Use the AES, SMA, NOC SMA integration, or high availability landing pages to quiesce the Pega 7 Platform server.

   For more information about these management tools, see Cluster management.
4. Click **Designer Studio > System > High Availability > HA Cluster Management**.
5. Select the check box in front of the node that you want to quiesce.
6. Click **Quiesce**.

HA Cluster Management

The HA Cluster Management page details the current status of any nodes running in high availability mode and is used to perform quiesce actions for a node.

Operators with the following roles can access this page:

- PegaRULES:HighAvailabilityAdministrator
- PegaRULES:HighAvailabilityQuiesceInvestigator

You can access this page by clicking **Designer Studio > System > High Availability > HA Cluster Management**.

Important! For immediate drain, do not use the HA Cluster Management page to quiesce a node. Use the System Management application (SMA) and the JMX APIs instead. See Quiescing a node using immediate drain.

Note: Data is accurate after two system pulse cycles.

The following data is displayed on this page for each node.

Node	*The name of the node*
NodeID	The node ID
Runstate	The status of the node: • Running • Stopped • Quiesce Complete • Quiesce Start
Active Users	The number of active users
Last Pulse	The date and time of the last system pulse
Last Started	The date and time that the node was last started

To perform an action on any of the nodes listed on this page, select the check box next to the node(s) and then click one of the following buttons:

- **Quiesce** – Begin the quiesce process for the selected node(s).
  Note: When using the slow drain method for quiesce, you must take the node(s) out of the load balancer rotation before performing this action.
- **Cancel Quiesce** – Cancel the quiesce process for the selected node(s).
- **Refresh Data** – Refresh the node data listed on the page.

Configuring high availability landing page visibility

You can display or hide high availability landing pages by adjusting the following prconfig setting.

session/ha/Enabled

- Value Type — Boolean
- Value — true or false
- Functionality — Switch to turn on or off the landing page user interface (UI)
- High availability aspect
  - When the Pega 7 Platform is installed, this is set to true.
  - Administrators must choose to use landing pages, AES, a NOC, SMA, or MBeans to manage high availability
  - Landing pages:
    - Are not recommended due to latency of system pulse communication
    - Are useful for testing purposes or Pega 7 Platform installations that do not have Network Operations Centers
    - Do not use for quiesce when using immediate drain
- Can be set through DASS — Yes
- Example: `<env name="session/ha/Enabled" value="true" />`

Quiesce using immediate drain

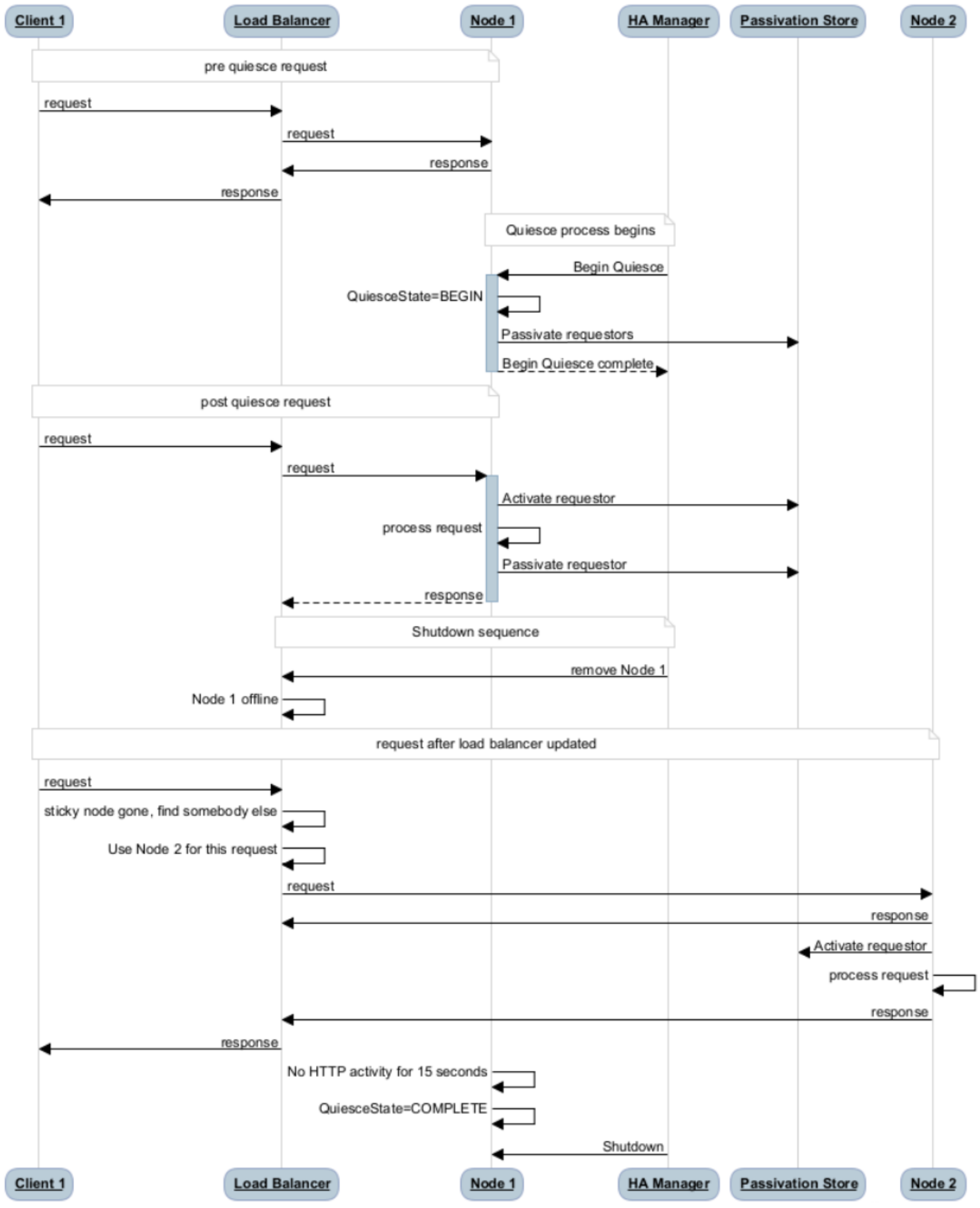
When using immediate drain for quiesce on a highly available system, the following behavior occurs on a Pega 7 Platform server when the quiesce process is enabled:

1. When a node is placed in a quiesce mode, all users (new and existing) are passivated immediately and the node can be removed from the load balancer at the same time. Existing active requestors that are locked are passivated at the end of the current interaction.

2. Access to the quiescing node is allowed for all users until the node is removed from the load balancer.
   - Active requestors can continue sessions on the quiescing node.
   - New sessions can be created on the quiescing node.
   - Requestor data is saved after each interaction and the requestor is removed from memory.
   - For each new interaction, the saved requestor data is used to activate the session on the node selected by the load balancer.

On-premises quiesce flow

The following diagram illustrates the on-premises quiesce flow for highly available Pega 7.2.x systems.


7.2 On-Premise Quiesce Flow

Quiescing a node when using immediate drain

When using high availability, you can perform the quiesce process for a node to remove it from a pool of active nodes.

You can perform this task using a load balancer, such as the F5 management console. Each load balancer has a different user interface. See your load balancer documentation for specific configuration instructions.

1. Identify the node to quiesce.
2. Connect to the System Management application (SMA) for the node.
3. Click **Administration > High Availability Management**.
4. Click **Begin Quiesce**.
5. Identify the Pega 7 Platform servers that you want to shut down.
6. Configure the load balancer to remove the identified servers from the pool. This will prevent any traffic from being directed to the quiescing node.

HA Cluster Settings

The HA Cluster Settings page is used to configure crash recovery and accelerated passivation options.

Operators with the following roles can access this page:

- PegaRULES:HighAvailabilityAdministrator
- PegaRULES:HighAvailabilityQuiesceInvestigator

You can access this page by clicking **Designer Studio > System > High Availability > HA Cluster Settings**.

To save any changes that have been made on this page, click **Submit**.

Cluster Upgrade

Disable the saving of rules when upgrading a cluster by selecting the Cluster Upgrading - Disable saving of rules check box.

Crash Recovery

Enable crash recovery by selecting the Enable server crash recovery on this cluster check box.

Note: You must perform a rolling restart of the server for this cluster wide setting to take effect.

Send messages to users about a crash event by selecting the Enable end user messaging of a crash event check box.

Quiesce

This setting is only available when using slow drain. Modify the time needed in seconds for accelerated passivation to occur by updating the corresponding field.

- The minimum number is 5 and the maximum number is the browser timeout.
- Values larger than the browser timeout are ignored.

Passivation and activation

Passivation

Passivation removes operator data from memory if an operator is not active.

If high availability is enabled, the Pega 7 Platform uses database passivation by default, with filesystem passivation available as an alternate method. If high availability is not enabled, the Pega 7 Platform uses either the configured location (filesystem or database), or, if no location is configured, the temp folder.

Note: The default passivation directory is the operating-specific temporary directory and cannot be used for passivation storage.

To add a process to occur before passivation (for example, saving all call data and closing all tabs in a customer service application), use the pyPrePassivation activity. In order for this activity to be run as expected, you must implement the activity with the correct applies-to class (@baseclass).

Note: The pyPreQuiescePassivation is deprecated.

Activation

If the operator continues working after a period of inactivity, the system retrieves their work from storage and puts it back into memory, referred to as activation.

To add a process to occur after activation (for example, reconnecting a user with a dropped call), use the pyPostActivation activity. In order for this activity to be run as expected, you must implement the activity with the correct applies-to class (@baseclass).

Note: The pyPostQuiesceActivation is deprecated.

Setting a custom passivation mechanism

To set a custom passivation mechanism for high availability, complete the following steps:

1. Set the initialization/persistrequestor/storage environment to custom.
2. Set the initialization/persistrequestor/storage/custom/class value to the fully-qualified class name of a class that implements the custom storage interface.

For more information, see the PDN article *Creating a custom passivation method*.

Quiesce behavior during upgrades

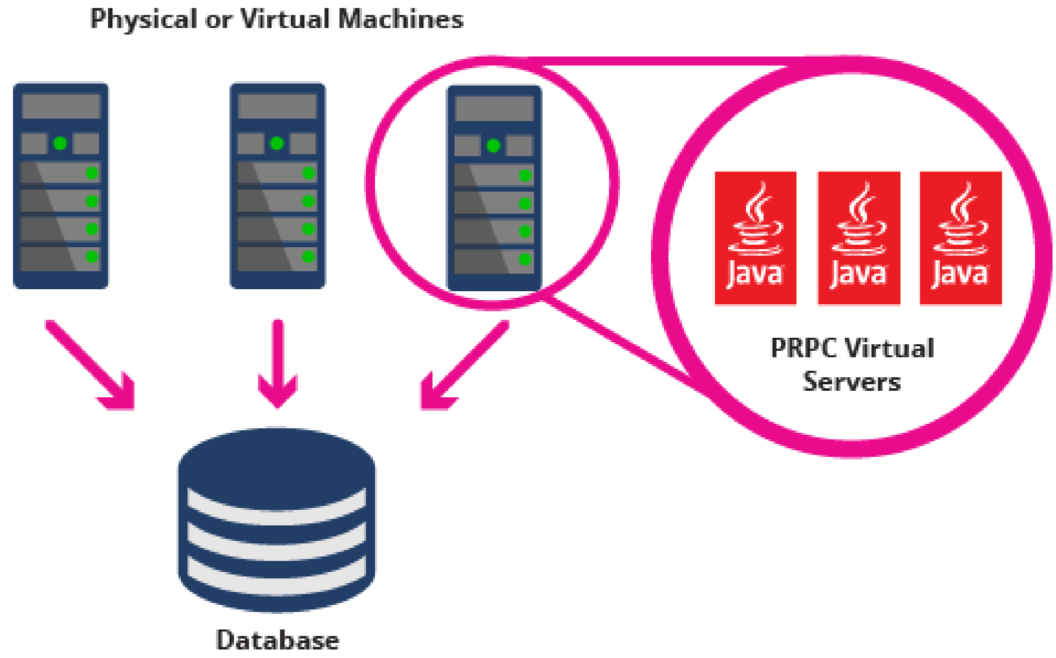
Upgrades of applications or frameworks that include code changes might not work with passivation and activation session recovery.

For example, the Pega 7 Platform identifies that the user is activated on a Pega 7 Platform server with a code set different from the passivated Pega 7 Platform server. If so, a new session must be created and re-authentication forced.

The code set check is automatic and conservative in forcing the recreation of a new session. To the user, this appears as recovery from the Pega 7 Platform server crash.

Cluster management

A typical Pega 7 Platform cluster consists of several physical or virtual machine instances sharing a single database. Machine instances may have multiple web application server instances or Pega 7 Platform servers.



Based on organizational needs, it is important to consider which of the following tools to use for performing system maintenance on Pega 7 Platform servers:

- Incorporate Autonomic Event Services (AES) to integrate this function into network operation centers.
  Tip: It is recommended to deploy AES to manage production systems with multiple clusters.
- Access the System Management Administrator (SMA), which can be used from every Pega 7 Platform server in the system.
- Use high availability MBeans, which can be directly integrated into a custom Pega 7 Platform management console to notify that an upgrade is in process.
  Note: If this process is used, rules that are created at runtime on the old rule schema must be manually moved to the new schema.

If an organization is not managing multiple clusters, you can use high availability landing pages. Because the Pega 7 Platform cluster management pages use the system pulse to communicate, data is accurate after two pulse cycles, and the latency that is introduced when updating the Pega 7 Platform server run state might not be optimal for large organizations.

Cluster management using Autonomic Event Services (AES)

To perform cluster management using Autonomic Event Services (AES), do the following:

1. Identify the Pega 7 Platform server to quiesce on the load balancer and take it out of rotation.
2. Access AES.
3. Open the Quiesce Manager.
4. Select the Pega 7 Platform server and click **Quiesce**.
5. The system run state displays "Quiesce Start." Once the system has been successfully quiesced, it displays "Quiesce Complete."
    - Click **Cancel Quiesce** to cancel quiesce.
    - Click **Refresh** to see the running status.

The Pega 7 Platform server can now be investigated for root cause analysis, managed, or shut down.

Cluster management using the System Management application (SMA)

The System Management application (SMA) can be used to quiesce Pega 7 Platform servers. To access it, click Designer Studio > System > Tools > System Management Application.

The SMA displays the following information for each node that is useful for high availability reference:

- System Run State – Lists the current run state for the system, including the quiesce status.
- Number Active Non Quiesce-Admin Requestors – The number of active users that are not administrators.

Under the Administration section, on the High Availability Management page, the following operations can be performed:

- Click **Begin Quiesce** to start quiesce on a Pega 7 Platform server.
- Click **Cancel Quiesce** to cancel quiesce.
- Click **Get Active Non Quiesce Admin User Count** during the Quiesce Start run state to get a count of users that are still on the Pega 7 Platform server and have not been quiesced yet.

For upgrades to a cluster using rolling restarts, operation teams can notify the cluster that an upgrade is occurring. A flag is set for the cluster that prevents the creation of auto-generated rules. The flag can be reset when the rolling restart has completed.

- Click **Begin Cluster Upgrade** to notify the cluster that a rolling restart is to commence.
- Click **Cluster Upgrade is Complete** to notify the cluster that a rolling restart has completed.

- Click **Get Cluster Upgrade State** to view the upgrade status.

Cluster management using MBeans

MBeans are used to integrate Pega 7 Platform high availability into a Network Operations Center (NOC). Additionally, Autonomic Event Services (AES) uses MBeans to perform Pega 7 Platform high availability functions.

The JConsole can be used to access the following MBeans:

- beginQuiesce
- cancelQuiesce
- getActiveNonQuiesceAdminUserCount
- beginClusterUpgrade
- clusterUpgradeIsComplete
- getClusterUpgradeStatus

Note: The MBean performQuiesceOnDemand is not supported.

High availability integration services

High availability integration services are comprised of several key components. The following artifacts, which are not unique to high availability functionality, should be deployed to support high availability operations:

- Listeners
- Service package data instances
- Service requests
- Service rules

Note: Deployments of Pega 7 Platform integration services that are part of highly available environments require additional configuration that might not be necessary in development or elsewhere.

Listeners

Listeners are background processes in the Pega 7 Platform that wait for inbound network requests or messages. In a high availability environment, listeners should be distributed across hosts and Pega 7 Platform servers to assure redundancy.

The listeners available in the Pega 7 Platform are:

- Email
- File
- Java Message Service (JMS)
- Message Queue for IBM Websphere (MQ)
- JMS Message Driven Bean for JEE (e-tier) deployments (JMS MDB)

Service packages

A service package is a Pega 7 Platform data instance for a collection of services. It controls access to the listeners for the services with defined Pega 7 Platform Access Groups. The service package also defines the service processing mode (stateless or stateful) and pooling options for stateless unauthenticated services.

Note: Stateless services are recommended for high availability.

Service request processors

Service request processors are also Pega 7 Platform data instances that provide configuration options for asynchronous processing.

Service rules

The processing of service requests is handled by service rules as appropriate for each service type.

There are some configuration aspects common across all Pega 7 Platform integration services. This includes ensuring proper distribution of listeners across hosts and Pega 7 Platform servers for the required level of redundancy. An unexpected failure may result in the loss of in-flight transactions and possibly the re-processing of messages.

- Email service and listener
  - For unexpected shutdowns, it is possible for email to have been processed (read) by the Pega 7 Platform but not marked accordingly in the email server. This might result in duplicate processing of email messages either when a Pega 7 Platform server is restored, or immediately if the message is processed by a listener on another Pega 7 Platform server. In this scenario, logs should be reviewed for such duplication as required.
  - For planned shutdowns, email listeners are marked for stopping and terminate when all inflight messages are processed. The time it takes for an email listener to stop depends on several factors, including the number of messages that the listener is configured to process at a time.
- File service and listener
  - Unexpected shutdowns are handled with built-in recovery features.
  - For high availability, select the attempt recovery and lock temporary file names options during configuration. Additionally, file source locations should be common across Pega 7 Platform servers, for example, on a redundant network drive.
- JMS service and listener - Enable durable subscriber for JMS services and use after message processing for acknowledgments.
- MQ service and listener - Use transacted messaging for IBM Websphere MQ services.
- JMS MDB service - Use for e-tier deployments.
  - Be sure to use container-managed transactions and durable subscribers.
  - Consult the appropriate web application server vendor documentation for proper configuration of the JEE container for high availability.
- HTTP, REST, SAP, SAPJco, SOAP, and Java services - Support the ability to perform asynchronous processing on child threads.
  - For unexpected shutdowns, there is a slight possibility of unpredictable message processing.
  - For planned shutdowns, route traffic away from the Pega 7 Platform server and monitor requestor activity on servlets, including any batch requestors that might be in use.

Crash recovery

The Pega 7 Platform provides crash recovery for browser and servers. With crash recovery enabled, the Pega 7 Platform saves the structure of the user interface and relevant work metadata to the database on user interface events.

The following matrix details the events that occur if a browser, Pega 7 Platform server, or machine crashes.

Events	Browser crash	Pega 7 Platform server crash
Pega 7 Platform application user interface is redrawn	Yes	Yes
User must re-authenticate	• No, if redirected to the same Pega 7 Platform server • Yes, if authentication cookie was lost	• No, with SSO • Yes, without SSO
Data entry loss	• No, if redirected to the same Pega 7 Platform server • Data not committed is lost if authentication cookie was lost	Data not committed is lost

Additionally, after a crash event, users must:

- Reacquire work object locks.
- Manually reopen assignments.

Server crash recovery

Two settings are required to enable Pega 7 Platform server crash recovery:

- storage/class/passivation/rootpath must be set to shared storage that is available to all servers in the cluster.
  - Depending on the operating system, the details of the configuration will vary.
  - Shared storage should be deployed so it is not a single point of failure.
  - Shared storage itself should have a failover solution.
  - Server restart is required to change the location of shared storage in the Pega 7 Platform.
- session/ha/crash/RecordWorkInProgress=true indicates to the Pega 7 Platform that user interface metadata will be stored to the share file system.
  - This setting can be changed on the high availability landing page, in DASS, or using prconfig.xml settings, depending on requirements.
  - A server restart is required for changes to take effect.

Pega 7 Platform server failover only works if the Pega 7 Platform server that fails is taken out of service from the load balancer. Requests that were serviced from the crashed Pega 7 Platform server are redirected to new Pega 7 Platform servers. This implies that a production class load balancer is employed, as well as passive or active monitoring of the application.

There are two steps in the recovery:

- On redirection to a new Pega 7 Platform server, the user must re-authenticate. The high availability best practice is to enable single sign on to avoid user interruption.
- When the server processes the request, it detects that there has been a crash event and uses the user interface metadata to reconstruct the user interface. Since the user's clipboard is not preserved from the crash, data that has been entered but not committed on assign, perform, and confirm harnesses is lost.

Browser crash recovery

Browser crash recovery is provided as a part of the Pega 7 Platform. The state of a Pega 7 Platform application is recovered without loss. The clipboard preserves both metadata for the user interface and any data entered on screens and submitted.

When the browser is terminated or crashes, the next user is connected to the correct server based on session affinity. The user interface metadata and clipboard are used to reconstruct the Pega 7 Platform application state without loss.

Dynamic containers and HTML5

In order to recover work after a browser crash, the Pega 7 Platform application must be HTML5 ready so that it can use the Dynamic Container feature that enables application tab recovery. To do so, use the HTML5 Application Readiness wizard by clicking Designer Studio > User Interface > HTML5 Application Readiness.

Once the Dynamic Container is working for the application, retest the browser and the Pega 7 Platform server crash scenarios. For more information, see PDN article *Upgrading an application to render in HTML5 Document Type*.

Enabling crash notification

In the event of a server or browser crash, you can configure a notification to display to users. This notification is configured by modifying session/ha/crash/EnableUserNotification:

- Value — True or false
- Functionality
  - Turns on the ability to notify a user if there has been a high availability crash event.
  - The message that displays to the user is:

"Something appears to have gone wrong. Your session has been recovered."

  - To modify this message, override the pyHANotifyMessage value.

- High availability aspect — This is available as the **Enable end user messaging of a crash event** check box on the HA Cluster Settings landing page and is required for high availability.
- Can be set through DASS — Yes

Enabling periodic work recording

To recover from server or browser crashes, the Pega 7 Platform can be configured to periodically record work in progress if the **RecordWorkInProgress** variable is set to true. This saves metadata about the state of the user interface interactions, tabs, and screens. This is configured by modifying session/ha/crash/RecordWorkInProgress:

- Value — For high availability, should be set to true
- Functionality
  - Enables the persistence of all user UI states (dynamic containers) in a cluster and restores the UI state to the last persisted state after a crash.
  - This requires a system restart if changed.
- High availability aspect — Required for crash recovery
- Can be set through DASS — Yes
- Example: `<env name="session/ha/crash/RecordWorkInProgress" value="true" />`