Perceptive Content Load Balancer

Configuration Guide

Version: Foundation 23.1

Written by: Documentation Team, R&D
Date: June 2023

Documentation Notice

Table of Contents

Overview

Perceptive Content Integration Server and Perceptive Content Server are typically deployed as application pools where many instances of each service are running. Load balancers help distribute the load evenly amongst the available instances and also provide an abstraction to more easily scale Perceptive Content services.

Load balancing Perceptive Content Integration Server

A load balancer is typically placed between the clients and Perceptive Content Integration Server instances. Client socket connections can be ephemeral to the Integration Server instance, but once a Perceptive Content client session is established, it is important that the client maintains session affinity with the Integration Server instance on which the session was initially established and for the lifetime of the session. You should use a Layer 7 load balancer to ensure that once client sessions are established with an Integration Server instance, continuity is maintained for that client. Failure to maintain session affinity with client sessions have an impact on the end user's experience and can cause significant overhead that could potentially degrade system performance. Each client session that is established through an Integration Server instance creates a unique connection to the Perceptive Content Server that is associated with the session hash identifier. If a session is resumed on a different Integration Server instance, a new connection is made to the Perceptive Content Server and the connection previously used for the session is disconnected.

Historically, load balancers have been configured to create sticky persistence state and associate that with the generated Perceptive Content session hash identifier of the established session. Load balancers could then use the persistence state to route subsequent requests to the appropriate Integration Server instance based on the incoming Perceptive Content session hash identifier.

Load balancers must evaluate every request made to Integration Server as clients can potentially use Keep-Alive connection pooling which may result in the same socket handling multiple Integration Server sessions.

In Perceptive Content EP4, a session load balancer cookie setting has been added to Integration Server which sets a cookie that load balancers can use to maintain session affinity with the client. When this setting is enabled, you must update the clients to have a cookie store that is associated with a single Integration Server session and used with all requests for the lifetime of the session.

Integration Server session timeouts

Integration Server has a `session.timeout` setting. The default value for this setting is 60 minutes. This is the amount of time that a unused session is maintained in the Integration Server connection pool. It is important that the client session affinity and persistence state be maintained for at least this amount of time to ensure that requests for a client's session are continuously routed to the originating Integration Server instance. This ensures that sockets are established when the client's session is created and resources are correctly released when the client's session is destroyed. If client persistence is not configured correctly it could result in Integration Server sessions being orphaned across Integration Server instances, and may take up to the configured `session.timeout` value for each affected session to fully close out the orphaned socket and remove the reference from the Integration Server connection pool.

Session load balancer cookie configuration

A session load balancer cookie can be configured so that load balancers can use this cookie to establish persistence and correlate a client's requests with the appropriate Integration Server instance. When a client's session is initially created, the configured session load balancer cookie is set. The client is responsible for ensuring that cookies are persisted and associated with the generated session hash and used for all subsequent requests. To enable session affinity cookies for Integration Server, update the following settings in the **integrationserver.ini** configuration file.

```
session.load.balancer.cookie.enabled=TRUE
session.load.balancer.cookie.name=ISLBPOOL01
```

All instances of Perceptive Content Integration Server within a load balancing pool should use the same cookie name. If an environment contains multiple discrete Integration Server load balancing pools, that can be routed to based on request context, it is important that each Integration Server load balancing pool have a discrete cookie name specified. You should configure the load balancer to use this cookie name for sticky sessions.

Load balancing Perceptive Content Integration Server connections to Perceptive Content Server

A Layer 4 load balancer can be used for socket connections established for client sessions between Perceptive Content Integration Server and Perceptive Content Server. The socket connection between Integration Server and Perceptive Content Server is expected to be maintained and managed by the Perceptive Content Server instance for the client's session. The `inowd.heartbeat.timeout` setting, in the **[Session Management]** group of the **inow.ini** configuration file, controls how long a session's connection is idle before the socket is closed. It is important that socket states are managed by Perceptive Content Server and the load balancer do not preemptively drop idle connections. If possible, set the load balancer's socket idle timeout to ensure that it is greater than the environment's configured `inowd.heartbeat.timeout` value.

Configuring connection socket keepalive packets

In Perceptive Content EP4, a socket keep alive setting has been added to Integration Server which enables the SO_KEEPALIVE flag on sockets established by Integration Server to Perceptive Content Server.

In environments where the load balancer's socket idle timeout is not configurable. Socket keep-alive packets may be necessary to ensure that the load balancer does not preemptively drop client connections prior to reaching the configured `inowd.heartbeat.timeout`. The **integrationserver.ini** configuration file contains a `connection.socket.keep.alive.enabled` setting that enables the SO_KEEPALIVE option on sockets. Additional system wide settings need to be put in place to ensure that the keep alive packets are sent prior to the timeout interval. It is important that the configured keep-alive timeout for the environment is less than the load balancer's idle timeout.

On Linux update the following settings:

- `tcp_keepalive_time` – The interval between the last data packet sent (simple ACKs are not considered data) and the first keepalive probe; after the connection is marked to need keepalive, this counter is not used any further. It is important that this value be less than the load balancer's configured idle timeout period.
- `tcp_keepalive_intvl` – The interval between sequential keepalive probes, regardless of what the connection has exchanged in the meantime.

- `tcp_keepalive_probes` – The number of unacknowledged probes to send before considering the connection dead and notifying the application layer.

On Windows, the default system-wide value of the keep-alive timeout is configurable through the KeepAliveTime registry setting which takes a value in milliseconds. The KeepAliveInterval registry setting is located under HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters.