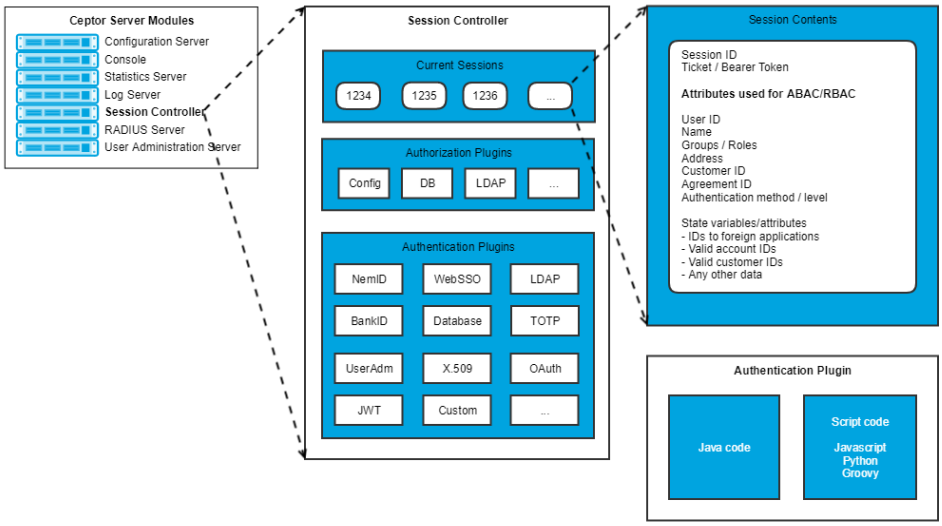
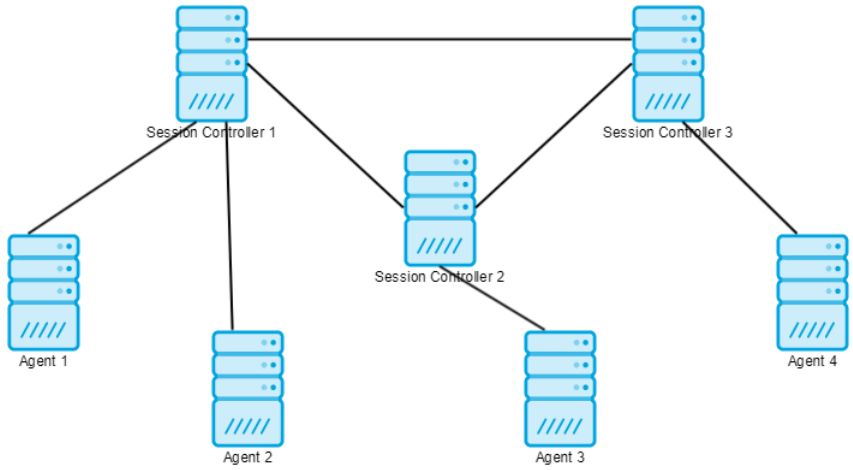
# Ceptor Session Controller

The Session Controller is a primary component in Ceptor – it contains the list of sessions that are currently known in the system, along with the information about each individual session, such as the logged-in flag, userid and password, and access rights. It also contains the authentication and authorization plugins that are responsible for authenticating the user or providing information needed for authorization.




A Session Controller can operate either alone, or in a cluster of multiple Session Controllers, that synchronize sessions information with each other. If they are deployed in a cluster, as SC mirrors, each change in a session, such as session creation, logon, logoff or session removal are broadcasted to the other SC servers, so they all are instantly synchronized.

A number of Agents communicate with one or more SC servers, and each SC server synchronizes changes with the other SC servers like this:




If the Agent 4 asks SC3 to create a new session, SC3 will tell SC1 and SC2 about the new session.

If Agent1 later logs a user in, the other SC servers will be informed about the changes in the session and update their internal tables, as well as send out commands to the Agents that have a copy of the session to tell them to remove that particular one from their caches.

## Clustering

Session Controllers are the most complex part of PortalProtect to cluster. Each individual Session Controller needs to know about all other session controllers, and all session controllers have to be connected to all other session controllers within a cluster.

Each update that an agent does to a session is replicated to all other session controllers, so the more session controllers in a cluster, the longer time it takes to complete an update.

Unlike the configuration servers, there is no master/slave concept in the session controllers – each session controller can listen on a port, and can connect to other session controllers.

In a typical setup with two session controllers, you set one of them to listen on e.g. port 12234, and the other to connect to the first one on port 12234.

With multiple session controllers, each one has to connect to all the others, so with e.g. 3 session controllers, it could look like this.

1 -> 2

1 -> 3

2 -> 3

It does not matter which session controller is listening, and which one is making the call – once their connection is established, they are equal.

Whenever a session needs to be updated (usually because an agent causes a change, such as logon), the session controller doing the update replicates all changes to the other session controllers it itself has a connection to – each of these will then notify the agents to clear their cache, and once that is done, they will return an acknowledge to the originating session controller, which then tells the agent that the change was successful.

It is not recommended to cluster session controllers to obtain better performance, usually having two session controllers for fail-over purposes is more than enough. Testing has proved that there is little or no gain in performance by having more than one session controller. The CPU usage of the PP server is only in very few installations above 20%. So most likely adding additional machines or session controller instances, often only generates more session replication, and does not improve overall performance.

## What Works and What Does Not

First of all, keep things simple – there is no reason to set up 10 different PP servers in a cluster, if 2 is enough – having too many servers just complicates the configuration needlessly, and increases the risk that something will go wrong.

Since updates to a session have to be synchronous (to ensure that consecutive requests will always see the updated session, even if the requests hits other application servers), the more session controllers added to a cluster, the longer time it will take to finish changing the contents of a session.

That being said, the overhead of replicating a session is not large – testing has shown that it can be as little as 1 millisecond, but the more complicated setup, the longer time it takes, and it is highly dependant on network load.

For the session controllers which experience the heaviest use, we recommend keeping the number of machines in a cluster down to 2. Instead of adding more machines to a cluster you can create multiple logically separated clusters, and let each agent connect to multiple clusters. That way the overhead of replicating sessions is kept to a minimum,  but failover still works in case a single machine fails.