State Detail

EndState *EndState1*

User does not have permissions to create a schedule, so the state becomes an inactive state (where user cannot perform actions in the scope of this diagram) even though the user is not logged out of the system. Essentially the system is left in a semi-active state until user logs off.

StartState *existState*

This state is how the system is before any modified states are introduced. Theoretically, it represents an idle server containing the system. An inactive system implies that a user is attempting to use it.

EndState *finishedWorkState*

User has finished activities in the system and returns the system to an inactive state, or one before they logged on.

EndState *loggedOutState*

User has requested that the system return to an inactive state. From there, system can adopt the 'not-in-use' state where system on the server is idle and has no connections.

State *LogOut*

System begins to return to an inactive state.

State *NoCreateSchedPerms*

The state of permissions is absent and the system cannot deal with scheduling.

State *SystemActive*

The system is now active and supports different smaller states of creating and modifying a schedule. Deleting is not covered, but would be part of modifying the schedule.

Substates

State *CanCreateNewSchedule*

System state allows for creation of schedules.

State *ConstraintsViolated*

If objects in the schedule are illegal, the schedule will refuse to save the schedule until the constraints are met.

State *DoNotSaveSchedule*

System enters a state where it is free of temporary objects; any that are present are discarded.

State *GenerateNewScheduleTemplate*

When a schedule is generated, the state now supports a temporary object that is modified before it is officialized into the backend.