

Using the Simulation Pilot Workstation (SPW) in Java

Prepared for:

Dan Warburton
ACB-860
Simulation Group (ACB-860)
Real & Virtual Division (ACB-800)

Federal Aviation Administration
William J. Hughes Technical Center
Atlantic City, NJ 08405

Prepared by:

Bethany Leffler
Titan Corporation

Under:

Titan Corporation
5218 Atlantic Avenue
Mays Landing, NJ 08330
FAA Prime Contract No. DTFA03-99-D-0017

October 2005

I. Introduction

Significant changes have recently been made to the Simulation Pilot Workstations (SPWs) and the way that the simulator manages these workstations. The purpose of this document is to explain what these changes are and how they affect the simulation process.

The workstations were previously written in TCL-TK and C using the X Windowing System. For compatibility and maintainability, they were rewritten in Java. The simulator had previously managed the SPWs and sent all simulation information directly to each running workstation's IP address. To take the workload of managing these workstations off the dynamics engine, the SimPilotManager was decoupled from the simulator. This also enables the manager to be run from a different machine over the network. Instead of logging which SPWs are running and assigned, the SimPilotManager now listens for messages being multicast from the SPWs and assigns aircraft to them accordingly; this allows the simulator and the SimPilotManager to handle larger studies.

Another simulator redesign advantage is the SimulationActionViewer (SAV), which is an upgrade of the old AircraftDiagnosticViewer (ADV). The ADV was attached to the Exercise Control Operator (ECO) and allowed the user access to data relating to active, pending, and terminated aircraft. The SAV displays all SpCommands that were executed in addition to all of the ADV functionality. The main difference, though is that it can now be used to observe any simulation that is being multicast on the user's network. For more detailed information on this object see the [Target Generation Facility User's Manual](#) section 5.2.

II. The Remote Diagnostic Tool

The RemoteDiagnosticTool (RDT) is used to start the SimulationActionViewer (SAV) on a machine other than the one running the simulator. This class is located in `faa/tg/util/aircraftViewer` and takes one command line arguments, the port offset of the simulation being monitored.

When running the RDT, as with the SPW it is important to run from the same jar file as the simulator. This is to avoid any version discrepancies in the transmitted data. For example, to listen to ECO7's simulation, the user would enter:

```
java -cp /tgf/lib/tgf.jar faa.tg.util.aircraftViewer.RemoteDiagnosticTool -o 7
```

III. Setting up the simulation to use the Java SPW

Before running the simulator using the newly developed SPWs there are several properties that can be specified in the ECO and scenario properties files. First, in the ECO properties file there might be a property named SpCommPort; the default value for it is 9251, which specifies the communication port for the new SPW. However, if the

ECO was run using the old stations, it might be set to 9250. Either change the value to 9251 or remove the line altogether to specify the use of the Java SPW.

Secondly, the simulator has been changed to automatically multicast simulation data and listen for SpCommands. If these actions are not desired, the following lines may be specified in the scenario properties file:

```
SimMessageClass = faa.tg.util.NoAction
SpCmdReceiverClass= faa.tg.util.NoAction
```

There is also a new property associated with the SpwAssigner (see section VIII) which specifies how many seconds the SpwAssigner should wait before assigning an aircraft to a pilot. The default value for this property is two seconds to limit the amount of time that the aircraft remain unpiloted. However, if the scenario has many pilots on the same sector it may be advisable to increase this number so that the SpwAssigner is able to receive responses from more pilots. For example, to change the delay to five seconds insert the following line into the scenario properties file:

```
SpwAssignerDelay=5
```

The last change in running the simulator is the way the simulator is started. The simulator **MUST** be run from a jar with the absolute path specified so the SPW knows the location of the jar to start from (e.g. `java -jar /tgf/lib/tgf.jar -p eco.properties`). This path is then specified as the classpath of the SPW to ensure that the classes encoding and decoding the simulation data over the network are the same version.

IV. Waking up the workstations

Once the “Spw Config” button on the ECO GUI is pressed, the SpSelectionComponent becomes displayed and the SimpilotManager finds out which machines are available. Once the user selects the SPWs to configure and clicks the “Accept Changes” button, the selected workstations are woken up. However, the way the SimpilotManager communicates with the workstations is slightly different.

The NewSpInitMessage, which includes the absolute path of the jar being run, the sector name, the frequency, the base port offset, the PVD to run, and the PVD map file, is broadcast to the selected workstations by the SimpilotManager over the SpCommPort specified. If the SpCommPort is set to 9251 the workstation then runs the `/tgf/user/sp/startjavasp` script that starts KDE and calls Java to start the SPW class using the specified jar. Once the software finishes loading, a SpStatusChangeMsg is multicast on base port 3800 to the SimpilotManager stating that the workstation is ready to receive simulation data.

In addition, the SPW can be run from the command line. To start the workstation from the command line there are several command line arguments that need to be specified: sector name, frequency, and port offset. Two other optional arguments are the name of the PVD program to use (XPVD, QPVD, or JPVD), and file for the PVD to load. If no PVD program is specified, no PVD will be displayed.

Here is an example of running a pilot on the Ghost sector in the genera scenario run on ECO7:

```
java -cp /tgf/lib/tgf.jar faa.tg.sp.gui.SimPilotWorkstation -s GHOST -f 129.990 -o 7  
-p XPVD -m /tgf/xpvd/cntl/genera
```

Again, the jar used should be the same version as the simulator.

V. The multicasting of simulation data

During the simulation the ECO multicasts several types of data over 225.225.225.225, base port 3600. This data include ADMAircraftTransferMedium, DeadReconAircraftTransferMedium, EndOfEpochTransferMedium, FlightActivatedEvent, FlightPendingEvent, FlightTerminatedEvent, SpCmdResultTransferMedium and TerminateEvent. All of these objects implements TransmittedSimData.

The transfer mediums are created by the SimMessageCenter to minimize the size of the transmitted packets. They are then multicast to be read by the SimDataReceiver, which sends the queue of TransmittedSimData to the SimDataFactory where the data is put back into its original format. The ADMAircraftTransferMedium, DeadReconAircraftTransferMedium, SpCmdResultTransferMedium, and EndOfEpochTransferMedium objects are created from UpdateCompleteEvent, SpCmdResult, and EndOfEpochEvent objects respectively.

VI. Which aircraft the workstations display

While a SPW receives the multicast data for all aircraft, (which includes both DeadRecon and Aircraft Dynamics Model (ADM) aircraft) it will display only the data for ADM aircraft in its sector. If an ADM aircraft is in a pilot's sector, but is assigned to another pilot, then it will go in the "Other Aircraft" tab. Data for all ADM aircraft that are assigned to the pilot are displayed in the "My Aircraft" tab. The PVD will only show datablocks for ADM aircraft assigned to this pilot.

VII. Preset Assignment Files

The user can specify which aircraft get assigned to what SPWs by creating a preset assignment file and giving it to the SimPilotManager. This is desirable when training a pilot or trying to recreate old simulations. An example of this file is specified below:

```
<Preset_Assignment>  
  <SpAssignment acid="UAL543" spid="sp42" />  
  <SpAssignment acid="BLR8793" spid="sp42" />  
  <SpAssignment acid="AAL4409" spid="sp42" />  
  <SpAssignment acid="AFL1089" spid="sp40" />  
  <SpAssignment acid="UAL543" spid="sp35" />  
</Preset_Assignment>
```

This file assumes that sp35 and sp42 are in different sectors. If they are not, the aircraft will be assigned to the first pilot station listed in that sector.

VIII. Assigning aircraft to pilots

The way that aircraft are assigned to pilots was changed on the new SimpilotManager. Unlike the old manager, the new system does not store data on each pilot station. The SPWs can be shut down or started by something other than the SimpilotManager (e.g. by the user) and still receive aircraft and simulation data. The simulator is the only object that tracks which aircraft is assigned to which workstation. This is done by storing a SpID of the assigned workstation name in each Flight object. If no pilot is assigned to the aircraft, this field (spId) is set to SpID.NONE.

Once an aircraft's data is multicast, the workstation checks the aircraft's sector and spId. If the aircraft is in same sector and its spId equals SimPilotWorkstation.NONE, the SPW multicasts a LoadRpt on base port 3800. This report consists of the assigned workstation name, the number of aircraft currently assigned to it, and the ID of the aircraft the workstation is responding to.

An object called SpwAssigner is created on SimpilotManager start up. This object's purpose is to listen for LoadRpts, calculate which pilot should receive the aircraft, and assign the aircraft to that pilot. There is a slight delay between when the first LoadRpt for an aircraft is received and when the aircraft is assigned. This delay is set to two seconds by default, but can be specified in the properties file as SpwAssignerDelay. This is done to ensure that the assigner receives LoadRpts from all of the pilots on a sector.

Once the assignment starts, the SpwAssigner checks to see if a preset assignment file was passed in from the SimpilotManager. If it was, the SpwAssigner checks for the aircraft ID in the current sector to see if there is a designated assignment. Otherwise, it compares each of the reports received and assigns the aircraft to the station with the smallest current workload. If there are two or more pilot stations with the same number of aircraft, it looks in the queue of recently assigned pilot stations to see which station is due. This way, several aircraft can be bid on at the same time. Without a queue, it is possible that the same station can be assigned several aircraft in a row due to equal workloads at the time the LoadRpts were sent. Once the desired pilot station is found, the assigner multicasts an AssignCmd to the simulator and all the workload reports for that aircraft are removed.

If a SPW crashes or is manually brought down, its aircraft are reassigned among the other SPW's on the same sector as described above.

IX. SpCommands

The SpwAssigner, SPW, and SAV multicast SpCommands on base port 3700. The SpCmdReceiver, which is started when the simulation is loaded, listens on this port and

executes any commands received. These commands are multicast using the Light-weight Reliable Multicast Protocol rather than the default multicast protocol to ensure that every command is received by the simulator. More information on the LRMP is located at <http://webcanal.inria.fr/lrmp>.

X. Shutting down the workstations

When the “Terminate” button on the ECO is pressed, a `TerminateEvent` is multicast on the base port 3600. Once the workstations receive this message, a dialog is displayed giving the user the option to keep the workstation up. After five seconds, if the “Shut Down” button is pressed or the “Leave Up” button is not pressed, the dialog will disappear, a `SpStatusChangeMsg` will be multicast stating that the current workstation is going down, and the entire `SimPilotWorkstation` will be terminated.

XI. Changes in the pilot prompting

In order to work with the new bidding system and the way the `SimPilotWorkstation` receives simulation data, some changes were made to the way the prompt messages were handled. Since the simulator is no longer connected to any of the workstations, there is no way for the simulator to directly prompt them. Instead, the `PromptCmd` is always considered successful and the SPWs check every `SpCmdResult` to see if it is a `PromptCmd`. If it is, then it is displayed as a prompt instead of as a `SpCmdResult`.

The other change made was the way the prompts are handled if an aircraft is not yet assigned a SPW. This is especially a problem with “Contact Controller” messages since they can be sent out within a second of the aircraft changing sectors. This is handled by having the `Aircraft.prompt` method check to see if the aircraft is piloted. If it is, then the prompt is sent out as expected. If it is not, then the aircraft creates a `PilotPrompt SimEvent` to occur in the future. The time is determined by adding four seconds plus the set delay of the assigner (`SpwAssignerDelay` in the properties file) to the current time. If the `PilotPrompt` occurs and there is still no pilot, then nothing happens, otherwise `Aircraft.prompt` is called again.

Base Ports Used:

All of the base ports used are specified as static variables in SimPilotWorkstation as the following:

SIM_DATA_PORT	3600	//base port for all simulation data such as aircraft //data and termination events
SP_CMD_PORT	3700	//base port for all SpCommands
SP_LOAD_RPT_PORT	3800	//base port for all LoadRpts

Useful Files to look at:

Faa.tg.sp.gui.SimPilotWorkstation
Faa.tg.opt.dataStream.SimMessageCenter
Faa.tg.opt.dataStream.SimDataFactory
Faa.tg.eco.ecogui.spwcfg.EcoAcceptSpwCfgComponent
Faa.tg.eco.ecogui.spwcfg.SimpilotManager
Faa.tg.eco.ecogui.spwcfg.SpwAssigner