TGF

MISCELLANEOUS

UTILITIES

Last Updated
July 2008

Table of Contents

Aces Geomap to Xml

LOCATION: faa.tg.prep.aces.geomaps

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: The purpose of this script is to convert an Aces Geomap binary file to an Xml file that the PVD will read. The XML file is easier to read and is more uniform with the standard format of TGF files.

HOW TO EXECUTE:

- java faa.tg.prep.aces.geomaps.AcesGeomapToXml -i [input file] –o [output file] -c [center – optional] or –x [filename – optional].
  Parameters:
- Input file: the name of the Aces binary file is required for the conversion and is specified via a command line option -i [or –input] followed by a space and then the full path and file name of the Aces binary file.
- Output file: the command line option –o [or –output] followed by a space and the full path and file name is used to specify the name of the XML file to be created.

Must use one of the following but cannot use both:

- [-c Center] is the 3-letter identifier for a Center. It is used to reference that Center's default point of tangency data and data offsets.

or

- [-x filename] is the name of the Center.Xml file to use.

If user types in java faa.tg.prep.aces.geomaps.AcesGeomapToXml the following will appear:

```
java faa.tg.prep.aces.geomaps.AcesGeomapToXml

Usage: AcesGeomapToXml  [{-i,--input}] [{-o,--output}]
                        [{-c,--center} or {-x,--xml}]

        -i,--input      The file to read ACES data from.(Required)

        -o,--output     The file to write TGF XML to. (Required)

        -c,--center     The center to use for point of tangency acquisition,
                        required if a -x option is not used.

        -x,--xml        The xml file which holds the center PointOfTangency
                        data, required if the -c option is not used.


Example: java  <directory tree>.AcesGeomapToXml -i <filename>(Req) -o
               <filename>(Req) -c <String> or -x <filename>
```

The following is an example of the Center.Xml file that is used with the –x option:

```
<Centers>
  <Center>
    <CenterID>ZCY</CenterID>
     <PointOfTangency>
       <Latitude>37-40-56.000N</Latitude>
       <Longitude>070-43-11.000W</Longitude>
       <EarthRadius Units="375.50 NauticalMiles" />
       <LocX>525.625</LocX>
       <LocY>642.750</LocY>
     </PointOfTangency>
  </Center>
</Centers>
```

AddCvsVersion

NOTE: It is a good idea (but not necessary) to use this utility in conjunction with Add Serial UID

LOCATION: tgf/bin/AddCvsVersion in CVS tree

LANGUAGE: Bash

OPERATING SYSTEM: Linux/UNIX

PURPOSE: The purpose of this script is to add a CVS version ID to all the java files in the current working directory if they don't already have one.

HOW TO EXECUTE:

- AddCvsVersion [-h] - is equivalent to specifying the help option
- AddCvsVersion [-u (pdate)] [filenames]
- Parameters:
    - the command line option [-u] not only finds the files, it updates them. Otherwise, the Java files that need to be modified will be listed but not updated.

    - the command line option [filenames]: limits the number of files checked to the files specified.

Add Serial UID

NOTE: It is a good idea (but not necessary) to use this utility in conjunction with Add CVS Version.

LOCATION: tgf/bin/AddSerialUID in CVS tree

LANGUAGE: Bash

OPERATING SYSTEM: Linux/UNIX

PURPOSE: This script will edit the necessary files in the current directory or a subdirectory and then give them a serial UID. By default, this script runs in testing mode.

HOW TO EXECUTE:

- AddSerialUID [-h] - is equivalent to specifying the help option.
- AddSerialUID [-u(pdate)] – the program will edit the necessary files and insert a serial UID.
- Parameters: - [.jar] is interpreted as CLASSPATH so that a specific serial version UID can be used.
  - [.java] only the specified files will be checked
  - [-f] only Java files from the parameters will be checked
  - [-q] quiet version with reduced messages
  - [-qq] prevents any message output

NOTE: "serialver" may possibly "hang up" when the constructor contains GUI activities. If this happens, kill the latest serialver process that is running and **Add Serial UID** will continue.

Airlines

LOCATION: CVS faa.tg.atvoice.prep

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: this program gets the airlines used in scenario or flight plan. Use this to find out if TGF Airline Hashtable has to be updated, or if a flight plan needs to be changed.

HOW TO EXECUTE:

- java faa.tg.atvoice.prep.Airlines (-i input file) (-o output file)
- Input file which is the flight plan file or scenario directory to look for airlines in. It is required for the conversion and is specified via a command line option **–i.**
- Output file that is the file to write the list of airlines found. It is required and is specified via a command line option **–o.**

If user types in: java faa.tg.atvoice.prep.aces.Airlines the following will appear:

```
java -Xmx1000M faa.tg.atvoice.prep.Airlines

Usage: faa.tg.atvoice.prep.Airlines [{-i,--input}] [{-o,--output}]

        -i,--input              Flight Plan file or directory to look for
Airlines in (Required)

        -o,--output             File to write airlines to (Required)

Example: java <directory tree>.Airlines -i <filename>(Req)
         -o <filename>(Req)
```

Sample output:

```
Airlines

Identifer,"Telephony"
A,"AIR FORCE"
AAL,"AMERICAN"

****************************************

Missing Airline
VVT

****************************************

No Telephony
ATA
```

Airport Xml File Merger

LOCATION: CVS tgf.src.faa.tg.scenario.importer

LANGUAGE: Java

OPERATING SYSTEM: Any system that accepts Java

PURPOSE: This script takes the airport, approach, and runway Xml files and merges them into one file. This keeps all the airport data in one location. (Just to mention that this script is aged but can be implemented if files are found that need to be completed.)

NOTE: User should backup the airport Xml file before executing this utility just as a precaution.

HOW TO EXECUTE:

- Java faa.tg.scenario.importer.AirportXmlFileMerger [-A] [-a] [-r] [-o]
- Input files: the Xml files are required for **–A** <airport Xml file>, **-a** <approach Xml file>, and **-r** <runway Xml file>.
- Output file which is the file to write the TGF Xml to is required and is specified via a command line option **-o**

If user types in: `java faa.tg.scenario.importer.AirportXmlFileMerger` the following will appear:

```
Usage: faa.tg.scenario.importer.AirportXmlFileMerger [{-A,--airport}]
[{-a,--approach}] [{-r,--runway}] [{-o,--output}]

        -A,--airport    The airport XML file (Required)

        -a,--approach   The approach XML file (Required)

        -r,--runway     The runway XML file (Required)

        -o,--output     The XML file to output merged data to (Required)
```

Awy2RouteXml

LOCATION: CVS faa.tg.prep.aces

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: this script converts ACES Awy (air route) data into XML format. The XML file is more uniform with the standard format of TGF files.
Also see: AcesToken.java,AcesParser.java

HOW TO EXECUTE:

- java faa.tg.prep.aces.Awy2RouteXml (-i input file) (-o output file) (-j optional)
- Input file which is the file to read ACES Awy data from is required for the conversion and is specified via a command line option **–i.**
- Output file which is the file to write the TGF Xml to is required and is specified via a command line option **–o.**

If user types in: java faa.tg.prep.aces.Awy2RouteXml the following will appear:

```
java faa.tg.prep.aces.Awy2RouteXml

Usage: Awy2RouteXml [{-i,--input}] [{-o,--outfile}] [{-j,--join}]

        -i,--input     The file to read ACES data from (Required)

        -o,--outfile   The file to write TGF XML to (Required)

        -j,--join      Option to produce join route/fixes.

Example: java <directory tree>.Awy2RouteXml -i <filename>(Req)
         -o <filename>(Req) -j <boolean>(Opt)
```

BareBones

LOCATION: faa.tg.eco.barebones

LANGUAGE: Java

OPERATING SYSTEM: Any system that accepts Java

PURPOSE: This program allows the user to run a simulation without using the graphical user interface (GUI). BareBones is most often used as a script or a Verification and Validation tool (for example during new SimPilot training) as it has a command to terminate the simulation after a given amount of time. In addition, the user can use the BareBones utility to run SimPilot training.

HOW TO EXECUTE:

- java faa.tg.eco.ecogui.BareBones (-p properties file)
- Properties file is a file containing properties used to configure the Simulation. It is specified via the required command line option **-p**.

If the user types in faa.tg.eco.ecogui.BareBones the following will appear:

```
java faa.tg.eco.barebones.BareBones

INFO: [faa.tg.eco.barebones.BareBones::main]

option error: 'properties is required!'


Usage: faa/tg/eco/ecogui/BareBones [{-p,--properties}]


    -p,--properties     the properties file as file or URL (Required)
```

Example: java <directory tree>.BareBones -p <filename>(Req)

PROPERTIES FILE:

The properties file must have the following information in it:

Where to get simulation data from
SimDirBase = [Full path to data directory to use]

Port definitions
SpCommPort = 9251
CMRBasePort = 3000
disBasePort = 3200

##What offset to use
BasePortOffset = [A two digit number between 01 and 40]

What address to use to send/broadcast data out on
SimulationBroadcast = [IP Address]

Needed URLS
TgMsgDebugPropertiesURL = [What debug properties to use]
ControllerPropertiesURL = [What controller properties to use]
AircraftFactoryImporterUrl = [A file that contains information about aircraft performance and characteristics]

##Flight plan
FlightImporterUrl = [URL with the full path and file name of the flight plan to use]

##Simulation dir
ScenarioDirectory = [URL with the full path to simulation data]

Example properties file:

An example properties file to use with the BareBones utility:

```
## Where to get simulation data from
SimDirBase              = /tgf/data

## Port definitions
SpCommPort              = 9251
CMRBasePort             = 3000
disBasePort             = 3200
RemoteEcoGUIPort        = 5000

## What offset to use
BasePortOffset          = 34

## Desk
##SimulationBroadcast     = 192.168.64.255
## Lab
SimulationBroadcast     = 192.168.65.255

##Needed URLS
TgMsgDebugPropertiesURL =    file:///tgf/config/NG/c34/debug.properties
ControllerPropertiesURL =    file:///tgf/config/NG/c34/controller.properties
AircraftFactoryImporterUrl = file:///tgf/xml/adm/aircraft_baseline.xml

##BELOW are items that are needed specifically to run the bare bones

##Need flight plan
FlightImporterUrl  =         file:///tgf/data/BigAir_samf/baseline_75A.fp

##Need for SPW manager
SetorImporterUrl =           file:///tgf/data/BigAir_samf/Sector.xml
SimulatorIPaddr = 192.168.65.98

##Need simulation directory
ScenarioDirectory =          file:///tgf/data/BigAir_samf/
```

SIMPILOT TRAINING Using BareBones:

To use BareBones for SimPilot training the User will need include the following in the properties file:

Files needed for SimPilotWorkstation (SPW) software
SetorImporterUrl = [URL with the full path and file name of the Sector file to use]
SpLabLayoutURL = [URL with the full path and file name of the SimPilot Lab Layout to use]

The address of the machine that will be running the BareBones utility
SimulatorIPaddr = [IP Address]

##Audio step up (only needed if the User wish to have the BareBones setup the Audio)
AudioControlURL = [URL with the full path and file name of the Audio file to use]
AudioServerIPaddr = [IP Address of the AudioServer to use]
AudioControlClass = [The class of audio connection to use. Either faa.tg.audio.AudioControlCCCS, faa.tg.audio.AudioControlHfl, or faa.tg.audio.AudioControlErce]

DRA stuff (only needed if the Users wish to collect data on the training run)
RecorderClass = faa.tg.recording.DraRecorder
RecordingDirectory = [Full path to the directory to store the training DRA output in]

Example properties file:

An example properties file to use with the BareBones utility for SimPilot Training:

```
## Where to get simulation data from
SimDirBase              = /tgf/data

## Port definitions
SpCommPort              = 9251
CMRBasePort             = 3000
disBasePort             = 3200

## What offset to use
BasePortOffset          = 34

## Lab
SimulationBroadcast     = 192.168.65.255

##needed URLS
TgMsgDebugPropertiesURL =    file:///tgf/config/NG/c34/debug.properties
ControllerPropertiesURL =    file:///tgf/config/NG/c34/controller.properties
AircraftFactoryImporterUrl = file:///tgf/xml/adm/aircraft_baseline.xml

##BELOW are items that are needed to run the bare bones

##Need flight plan
FlightImporterUrl  =         file:///tgf/data/BigAir_samf/baseline_75A.fp

##Need simulation directory
ScenarioDirectory =          file:///tgf/data/BigAir_samf/

##BELOW are items that are needed to run the bare bones with SimPilotWorkstations (SPW)

##Need for SPW manager
SetorImporterUrl =           file:///tgf/data/BigAir_samf/Sector.xml
SimulatorIPaddr = 192.168.65.98

##Contains SPW step up
SpLabLayoutURL =             file:///tgf/xml/eco/SpLabLayout.xml
XmlClassLoaderURL =          file:///tgf/config/NG/c34/OptionalClasses.xml

##Audio step up (required so that the SPW can talk to controllers and each other)
AudioControlURL =            file:///tgf/data/BigAir_samf/DisplayLabAudioMap.xml
AudioServerIPaddr = 192.168.223.2
AudioControlClass = faa.tg.audio.AudioControlCCCS
```

```
# Dra stuff (only needed if the Users wish to collect data on the training run)
RecorderClass              = faa.tg.recording.DraRecorder
RecordingDirectory         = /recording/BigAirspace_zjx
```

Optional Classes XML file:

The OptionalClasses.xml file will must include the following lines in order to use SimPilot Workstations:

```
 <Class>
  <Load>yes</Load>
  <ClassName>faa.tg.eco.barebones.SimPilotInitiator</ClassName>
  <ConstructArg>
   <ArgName>spwCfgUrl</ArgName>
   <ArgType>java.lang.String</ArgType>
   <ArgValue>[URL with the full path and file name of the SimPilotWorkstation
configuration to use]</ArgValue>
  </ConstructArg>
 </Class>
```

WARNING: The SimPilotInitiator does not prevent SPW assignments from being overwritten. For example if SP09 is assigned to the Ghost frequency on another simulation and the configuration file tells the SimPilotInitiator to assign SP09 to the BAASS frequency the SimPilotInitiator will reassign SP09 without any warning.

An Example Optional Classes XML File:

```
 <Class>
  <Load>yes</Load>
  <ClassName>faa.tg.eco.barebones.SimPilotInitiator</ClassName>
  <ConstructArg>
   <ArgName>spwCfgUrl</ArgName>
   <ArgType>java.lang.String</ArgType>
   <ArgValue>file:///tgf/config/NG/c34/exampleConfig.spcfg</ArgValue>
  </ConstructArg>
 </Class>
```

TERMINATING THE SIMULATION:

To terminate the run the USER can either set up the BareBones to terminate the simulation after a given amount of time or type NIXSIM SimPilot command in to an SPW.

CheckJarsForSerial

LOCATION: tgf/bin

LANGUAGE: Java/Linux script

OPERATING SYSTEM: Any Linux system that accepts Java

PURPOSE: The purpose of this script is to determine the required UID for recording analysis (in a Serializable changed file, if a UID has not been set).
This is used in the event that a recording was made using a class file in which the serial version UID was not set (causing Java to generate one automatically), and later the code for that class file was changed, causing Java to automatically generate a different serial version UID. This would cause an import of the recording file to fail since Java counts them as different versions.
Checking the jar files for the serial version UID makes it easier for the programmer to determine which serial version UID a jar file uses for a specific class and which they would want to use when explicitly setting the serial version UID.

NOTE: When choosing an explicitly set serial version UID for the code:
if 2 or more jar files have a class with different serialVersion UIDs, the UID from the class from the Jar file that has important TGF recordings should be used.

HOW TO EXECUTE:

- checkJarsForSerial faa.tg.list.FlightLevelList *.jar
- Other class file format accepted: faa/tg/list/FlightLevelList
- Parameters: <class file to check>
  <jar file class is in>

It will then print the serial version UID for that class.

CockpitAmmFrame

LOCATION: faa.tg.aircraft.deadreckon.cockpit

LANGUAGE: Java

OPERATING SYSTEM: Any system that accepts Java

PURPOSE: this program sends AMM (airport management module) messages from the simpilot workstation (SPWS) in the TGF lab to the cockpit simulator in the hangar so that the pilot can fly the aircraft in an actual simulator using the same AMM messages. Specifically, this program provides an option that can be used to tunnel the data (updates, epics, etc.) from the SPWS to the SimPilot Multi Cast Address during a simulation. If both of these are running on the same network, Tunneller is not needed. However, if they are not, then Tunneller is required to direct the information to the proper address.

HOW TO EXECUTE:

- faa.tg.aircraft.deadreckon.cockpit.CockpitAmmFrame [-o] [-t]
- the offset to receive the data on is required and is specified via a command line option **–o or --offset.**
- if **–t or --tunnel** is present, than tunneller will be used (optional).

If user types in faa.tg.aircraft.deadreckon.cockpit.CockpitAmmFrame the following will appear (if no parameters are given):

```
java faa.tg.aircraft.deadreckon.cockpit.CockpitAmmFrame

Usage: faa.tg.aircraft.deadreckon.cockpit.CockpitAmmFrame
[{-o,--offset}] [{-t,--tunnel}]

        -o,--offset     The offset to receive data on (Required)

        -t,--tunnel     Whether to use tunneller. If present then tunneller
                        will be used (Optional)
```

CpRecording2PlayBack

LOCATION: CVS faa.tg.aircraft.deadreckon.cockpit

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: This script takes the cockpit concentrator recording file and splits it into separate cockpit playback files according to the number of Cockpit Simulators used in a simulation. These playback files can be used with the fsim program.

HOW TO EXECUTE:

- java faa.tg.aircraft.deadreckon.cockpit.CpRecording2PlayBack –f [file name].
- Input file (which is the .cp file) is required for the translation and is specified via a command line option **–f**.
- The output from this script can be entered into the fsim program.

If user types in java faa.tg.aircraft.deadreckon.cockpit.CpRecording2PlayBack the following will appear:

```
java faa.tg.aircraft.deadreckon.cockpit.CpRecording2PlayBack
option error: 'filename is required!'

Usage: faa.tg.aircraft.deadreckon.cockpit.CpRecording2PlayBack
[{-f,--filename}]

-f,--filename  the cockpit concentrator .cp file to translate(Required)
```

This will remind the user of the format that needs to be used.

Csv2XmlFPConverter

LOCATION: CVS tgf.src.faa.tg.prep.util

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: Converts old comma separated value (CSV) flight plans to XML format. The XML flight plan file is easier to read and is more uniform with the standard format of TGF files.

HOW TO EXECUTE:

- java faa.tg.prep.util.Csv2XmlFpConverter –c (input file) –x (output file – optional)
- Input file: the CSV file is required for the conversion and is specified via a command line option (-c) followed by a space and then the full path and file name of the CSV file.
- Output file: the command line option –x followed by a space and the full path and file name is used to specify the name of the XML file.
- If the user doesn't specify an output file via the –x command line option, then the program will take the name of the file and put an x at the end the extension. (i.e. file.fp becomes file.fpx).

If user types in java faa.tg.prep.util.Csv2XmlFpConverter the following will appear:

```
java faa.tg.prep.util.Csv2XmlFpConverter

Usage: faa.tg.prep.util.Csv2XmlFpConverter [{-c,--csv}] [{-x,--xml}]

        -c,--csv        CSV File to be parsed (Required)

        -x,--xml        File for the XML to be output
```

This will remind the user of the format that needs to be used.

diffAreas

LOCATION: User bin

LANGUAGE: Perl

OPERATING SYSTEM: Linux/UNIX

PURPOSE: To show if there are any differences when comparing files in two directory trees.

For example, if a user makes any variations in a program, the compile won't be the same and it will run differently per user, thus obtaining different results run on the same program. DiffAreas can compare directories and will list differences to explain why the results are not the same (compares your area to someone else's). By default, it compares class, java, properties, and xml files.

HOW TO EXECUTE:

- diffAreas.pl <enter> - is equivalent to specifying the help option
- can input one or two arguments on this line:
  - -If **one directory argument** is used it compares whatever is in the current directory to the directory that you have just input.
  - -If you specify **two directories,** it compares the specified directories.
  - -If given one or more arguments with a hyphen in front, the rest of the argument is used as the extension. Then it will only compare files that have that same extension. For example, if you want to compare only .xml files in the directories, specify –.xml.
- When complete, it shows the number of files compared and the number of differences.

NOTE: Specifying a help option ("diffAreas.pl – help") displays a brief explanation of diffAreas and default files are listed.

OPTIONS FOR THIS UTILITY: you can either do a .java, .properties, .jar, .xml, or .class. You can either specify one of these file types or a directory.

Distance.Java

LOCATION: faa.tg.units

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: This script will decipher and print the distance between two specific points in a particular scenario so that the user does not have to figure it out manually.

NOTE: If an insufficient number of parameters are given, it defaults to the test parameters.

HOW TO EXECUTE:

- java faa.tg.units.Distance [latitude1 longitude1 latitude2 longitude2 [alt]].

For Example:

```
Lat/Lon of p1 is 38-00-00.0N/078-00-00.0W (at alt: 0.0 ft)       Lat/Lon of
p2 is 38-10-00.0N/078-05-00.0W (at alt: 0.0 ft)
Can specify one altitude (alt) for both coordinates
 Distance from p1 to p2  = 10.7407911 nm  = 65262.287097 ft
```

(prints the latitude/longitude coordinates in DD-MM-SS.S
and the distance in nautical miles and feet)

Fix2FixXml

LOCATION: CVS faa.tg.prep.aces

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: This script converts ACES fix data into XML format. The XML file is more uniform with the standard format of TGF files.
Also see: AcesToken.java,AcesParser.java

HOW TO EXECUTE:

- java faa.tg.prep.aces.Fix2FixXml –i (input file) –o (output file)-s (subfx)
- Input file which is the file to read ACES fix data from is required for the conversion and is specified via a command line option **–i.**
- Output file which is the file to write the TGF Xml to is required and is specified via a command line option **–o.**
- Subfx which is the ACES subfx file to use if subfx are requested and is specified via a command line option **–s.**

If user types in java faa.tg.prep.aces.Fix2FixXml the following will appear:

```
Fix2FixXml [{-i,--input}] [{-o,--outfile}] [{-s,--subfx}]

      -i,--input        The file to read ACES fix data from (Required)

      -o,--outfile      The file to write TGF XML to (Required)

      -s,--subfx        The ACES subfx file to use if subfx are requested.
```

FpFixFinder

LOCATION: CVS faa.tg.atvoice.prep

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: This program gets the airlines used in scenario or flight plan. Use this to find out if TGF Airline Hashtable has to be updated, or if a flight plan needs to be changed.

HOW TO EXECUTE:

- java faa.tg.atvoice.prep.Airlines (-i input file) (-o output file)
- Input file that is the scenario directory to look for flight plan fixes in in. It is required for the conversion and is specified via a command line option **–i.**
- Output file that is the file to write the list of flight plan fixes found. It is required and is specified via a command line option **–o.**
- Whether to do strict checking when importing the flight plans is specified via the optional command line parameter **–s**. The default value for strict checking is false.

If user types in: java faa.tg.atvoice.prep.aces.Airlines the following will appear:

```
java -Xmx1000M faa.tg.atvoice.prep.Airlines

Usage: faa.tg.atvoice.prep.Airlines [{-i,--input}] [{-o,--output}] [{-s,--
strict}]

        -i,--input              Directory to look for scenairo info in
(Required)

        -o,--output             File to write fixes to (Required)

        -s,--strict             Whether to do strict flight plan checking.

Example: java <directory tree>.Airlines -i <filename>(Req)
         -o <filename>(Req)  -s
```

Sample output:

```
ABB
ABCOT
ABQ
ACY
AEA
AGARD
AGUNE
AJFEB
AJGON
ALB
ALCOT
ALDAN
ALIXX
ANGEE
APPLE
ARD
ARGAL
ARICE
ASHES
ATL
ATR
```

GenerateSimEDiffFiles

LOCATION: CVS faa.tg.util

LANGUAGE: Java

OPERATING SYSTEM: Linux/Unix Systems with Java installed

PURPOSE: this program generates files that contain information on when the position of an aircraft in a Sget QT message differed more then a given minimum report distance from the position of the aircraft in TGF.

HOW TO EXECUTE:

- java –Xmx1000M faa.tg.util.GenerateSimEDiffFiles (-r TGF input file) (-s SimE input file) (-a Airspace name) (-d optional distance) (-i optional QT Ident output file) (-g optional QT GUI output file) (-x optional QT XML output file)
- The TGF input file is either an AircraftState.csv file or a TGF DRA recording file, which contains the positions of the Aircraft. If a DRA recording file is used then this program generates an AircraftState.csv file in /tmp. To reduce processing time use an AircraftState.csv file. The TGF input file is specified using the **–r** command line option. This program uses the TGF input file to update SimE messages with actual TGF positions. (Note: If you do not already have a TGF input file you may want to run the TGF Eco GUI with the Fast Time Epoch Manager to obtain one.)

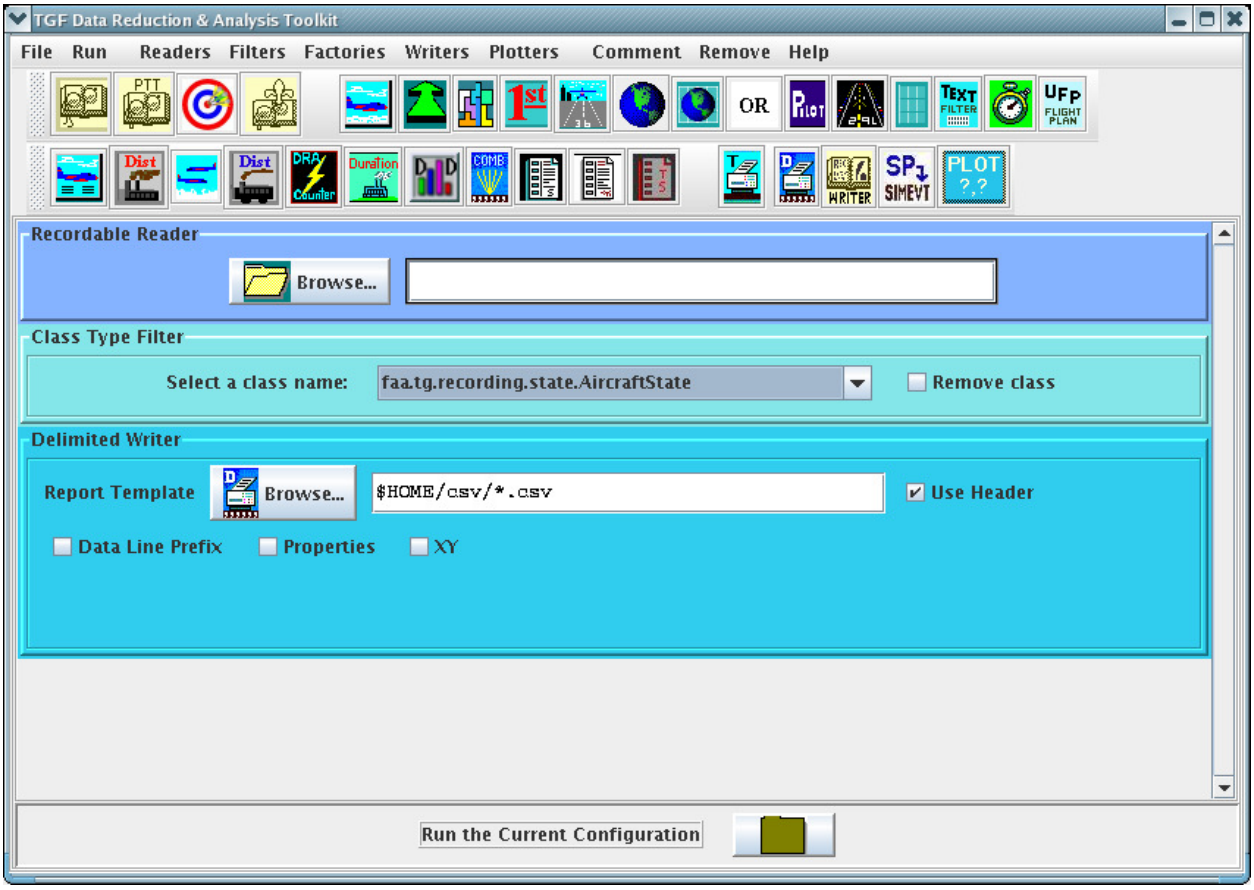
    - ***How to use the Fast Time Epoch Manager with TGF's Eco GUI***
        - Make sure that the OptionalClasses.xml file used by the scenario you are running contains the following:

```
<Class>
  <Load>yes</Load>

    <ClassName>faa.tg.executive.FastTimeEpochManager
    </ClassName>
  <ConstructArg>
    <ArgValue>7300</ArgValue>
  </ConstructArg>
</Class>
```

        - To determine the Optional Classes XML file that your scenario will load check your properties files. Typically, the Optional Classes Xml File to load is specified the **scenario.properties** file located in your scenario's data directory. If your properties file has **XmlClassLoaderURL** then the file pointed to by that property will be used. Otherwise, the default is **OptionalClasses.xml** in the scenario's data directory.
        - Please see "TGF User's Manual" at http://public.tgf.tc.faa.gov/documentation/eco/ecomanual.html for more information on either how to run the TGF Eco GUI or the OptionalClasses.xml file.

- ***How to get AircraftState.csv file from a TGF DRA recording***
  - After making a DRA recording file open a terminal window and type **java –Xmx1000M faa.tg.dra.gui.Drat**
- ***How to get AircraftState.csv file from a TGF DRA recording***
  - After making a DRA recording file open a terminal window and type **java –Xmx1000M faa.tg.dra.gui.Drat**
  - DRAT should look like the following picture after selecting the following:
    - The Recordable Reader from the Readers menu
    - The Class Type Filter from the Filters menu
      - On the Class Type Filter select faa.tg.recording.state.AircraftState
    - The Delimited Writer from the Writers menu



  - Save the configuration using the Save option on the File menu. The next time you run DRAT load the configuration using by selecting the Open Recent option on the File menu and selecting the configuration file you saved.
  - Fill in the name of the recording file to process in the recordable reader
  - Fill in the name of the output file in the Delimited writer. (Suggest leaving the *.csv at the end of the file.)

    - Please see the "TGF Data Reduction and Analysis Toolkit (DRAT)" manual at http://public.tgf.tc.faa.gov/documentation/drat/dratmanual.htm for more information on how to run DRAT.
- The SimE input file or directory contains Sget SimE messages, which this program updates to use actual TGF positions. This file or directory is specified using the required command line option **–s**. If a directory is specified than the any files that end in .rsi* and .<Airspace name> are processed. For example if the Airspace name is zdc than files ending in *.ZDC are processed. (Note: Check that you have the latest SimE message files).
- The Airspace name is the name of Airspace it is required in order to find files to process in a directory. The Airspace name is specified using the **–a** command line option.
- Distance is the minimum distance in nautical miles at which to report a discrepancy between where an Aircraft is in TGF and where an SIME messages says the aircraft is. The default distance is five nm. The distance is specified using the **–d** command line option.
- The QT Ident output file contains an Sim Event XML file that has TGF issue a ID Sp Command at the time the Aircraft gets a QT message if the difference between the TGF position of the Aircraft and the position of the Aircraft in QT is greater than or equal to the report distance. This output file is optional and is specified using the **–i** command line option. (Note: the name of the SimE input used is appended to the end of the file name.)
- The QT GUI output file contains an original SimE QT messages if the difference between the TGF position of the Aircraft and the position of the Aircraft in QT is greater than or equal to the report distance. This output file is optional and is specified using the **–g** command line option. (Note: the name of the SimE input used is appended to the end of the file name.)
- The QT XML output file contains information about an original QT message in XML format if the difference between the TGF position of the Aircraft and the position of the Aircraft in QT is greater than or equal to the report distance. This output file is optional and is specified using the **–x** command line option. (Note: the name of the SimE input used is appended to the end of the file name.)

If user types in: java –Xmx1000M faa.tg.util.GenerateSimEDiffFiles the following will appear:

```
java -Xmx1000M faa.tg.util.GenerateSimEDiffFiles

Usage: faa.tg.util.GenerateSimEDiffFiles [{-r,--rec}] [{-s,--sime}] [{-a,--
airspace}] [{-d,--dist}] [{-i,--identqt}] [{-g,--guiqt}] [{-x,--xmlqt}]

      -r,--rec                TGF recording (or AircraftState.csv file for
faster processing) (Required)

      -s,--sime               an SimE input file or a directory where to
find Sime input files (Required)

      -a,--airspace           Airspace to used to find files to process
(Required)

      -d,--dist               Minimum distance to report discrepancies on
(defaults to 5[nm])

      -i,--identqt            Output QT ident file with modified locations

      -g,--guiqt              QT GUI output file to create

      -x,--xmlqt              Output QT XML file with modified locations
```

Example: java -Xmx1000M faa.tg.util.GenerateSimEDiffFiles
-r AircraftState.csv -d 0.5 -g gui_ -x xml_ -a zdc -i ident_
-s TC4FS10_v2.ZDC

This program creates a log file called SimEDiffFiles.log, which contains any error or warning messages the program issued.

LocationModSimE

LOCATION: CVS faa.tg.util

LANGUAGE: Java

OPERATING SYSTEM: Linux/Unix Systems with Java installed

PURPOSE: this program replaces Sget positions and velocity vectors in QT, TI, and TU messages with positions of a TGF aircraft

HOW TO EXECUTE:

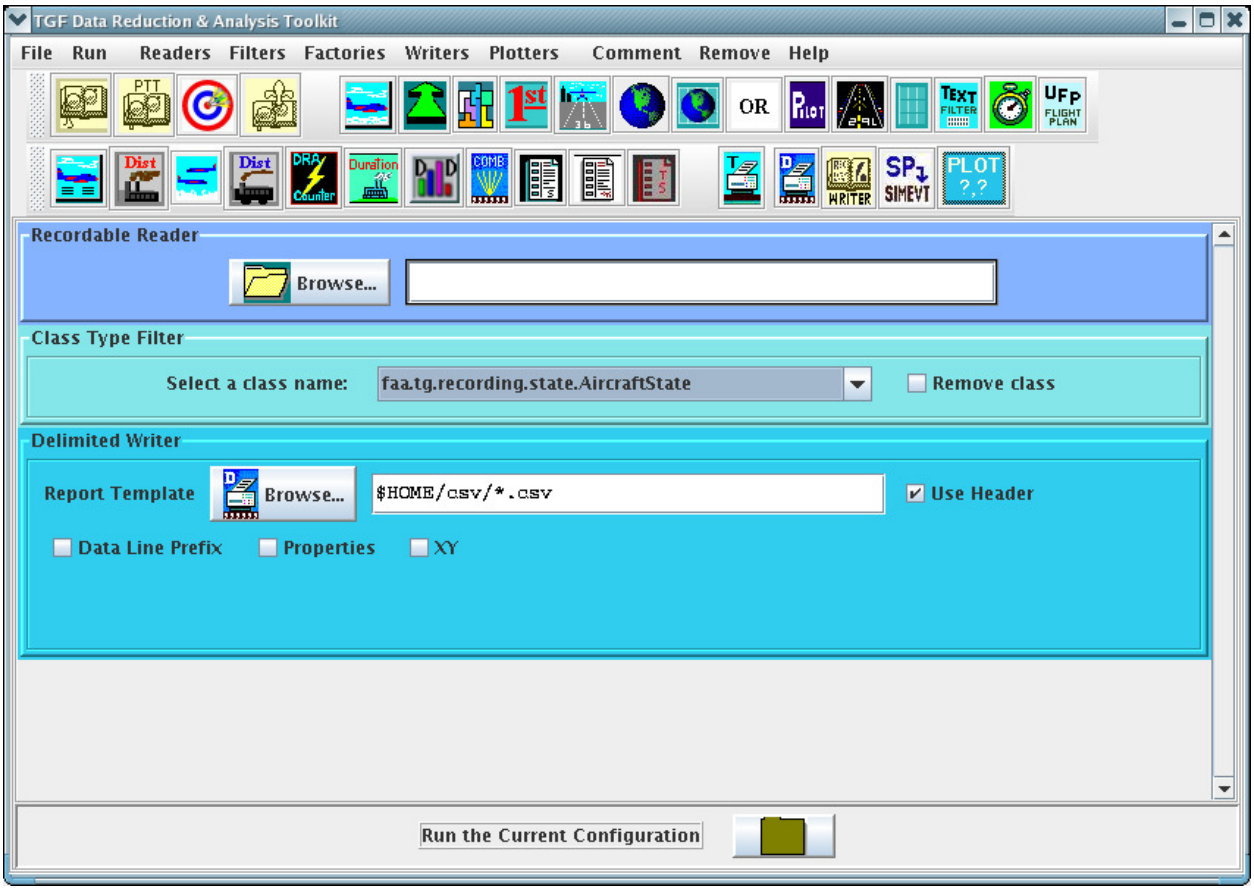
- java –Xmx1000M faa.tg.util.LocationModSimE (-r TGF input file) (-s SimE input file) (-ea ERAM ARTS XML input file) (-a Airspace name) (-d optional distance) (-alt optional altitude) (-o optional modified SimE output file)
- The TGF input file is either an AircraftState.csv file or a TGF DRA recording file, which contains the positions of the Aircraft. If a DRA recording file is used then this program generates an AircraftState.csv file in /tmp. To reduce processing time use an AircraftState.csv file. The TGF input file is specified using the **–r** command line option. This program uses the TGF input file to update SimE messages with actual TGF positions. (Note: If you do not already have a TGF input file you may want to run the TGF Eco GUI with the Fast Time Epoch Manager to obtain one.)
-
    - ***How to use the Fast Time Epoch Manager with TGF's Eco GUI***
        - Make sure that the OptionalClasses.xml file used by the scenario you are running contains the following:

```
<Class>
  <Load>yes</Load>

    <ClassName>faa.tg.executive.FastTimeEpochManager
    </ClassName>
  <ConstructArg>
    <ArgValue>7300</ArgValue>
  </ConstructArg>
</Class>
```

        - To determine the Optional Classes XML file that your scenario will load check your properties files. Typically, the Optional Classes Xml File to load is specified the **scenario.properties** file located in your scenario's data directory. If your properties file has **XmlClassLoaderURL** then the file pointed to by that property will be used. Otherwise, the default is **OptionalClasses.xml** in the scenario's data directory.
        - Please see "TGF User's Manual" at http://public.tgf.tc.faa.gov/documentation/eco/ecomanual.html for more information on either how to run the TGF Eco GUI or the OptionalClasses.xml file.
    - ***How to get AircraftState.csv file from a TGF DRA recording***

- After making a DRA recording file open a terminal window and type **java –Xmx1000M faa.tg.dra.gui.Drat**
- DRAT should look like the following picture after selecting the following:
    - The Recordable Reader from the Readers menu
    - The Class Type Filter from the Filters menu
        - On the Class Type Filter select faa.tg.recording.state.AircraftState
    - The Delimited Writer from the Writers menu



- Save the configuration using the Save option on the File menu. The next time you run DRAT load the configuration using by selecting the Open Recent option on the File menu and selecting the configuration file you saved.
- Fill in the name of the recording file to process in the recordable reader
- Fill in the name of the output file in the Delimited writer. (Suggest leaving the *.csv at the end of the file.)
- Please see the "TGF Data Reduction and Analysis Toolkit (DRAT)" manual at http://public.tgf.tc.faa.gov/documentation/drat/dratmanual.htm for more information on how to run DRAT.

- The SimE input file or directory contains Sget SimE messages, which this program updates to use actual TGF positions. This file or directory is specified using the required command line option **–s**. If a directory is specified than the any files that end in .rsi* and .<Airspace name> are processed. For example if the Airspace name is zdc then files ending in *.ZDC are processed. (Note: Check that you have the latest SimE message files).
- The ERAM ARTS XML input file is required in order to calculate x/y positions for TI and TU messages. The file is specified using the **–ea** command line option/
- The Airspace name is the name of Airspace it is required in order to calculate an x/y map coordinates. The Airspace name is specified using the **–a** command line option.
- Distance is the minimum distance in nautical miles at which to report a discrepancy between where an Aircraft is in TGF and where an SIME messages says the aircraft is. The default distance is five nm. The distance is specified using the **–d** command line option.
- Altitude is the minimum altitude in feet at which to report a discrepancy between what altitude an Aircraft's is at in TGF and what altitude an SIME TI messages says the aircraft is. The default altitude is eight hundred ft. The altitude is specified using the **–alt** command line option.
- The SimE output file contains SimE messages in which TGF positions for an aircraft are used. This output file is optional and is specified using the **–o** command line option. (Note: the name of the SimE input used is appended to the end of the file name.) If this option is not used then the default output file name is out_.

If user types in: java –Xmx1000M faa.tg.util.LocationModSimE the following will appear:

```
java -Xmx1000M faa.tg.util.LocationModSimE

Usage: faa.tg.util.LocationModSimE [{-r,--rec}] [{-s,--sime}] [{-ea,--
eramArts}] [{-a,--airspace}] [{-d,--dist}] [{-alt, --altitude}] [{-o,--
output}]

      -r,--rec                TGF recording (or AircraftState.csv file for
faster processing) (Required)

      -s,--sime               an SimE input file or a directory where to
find Sime input files (Required)

      -ea,--eramArts          The ERAM ARTS Xml File to import (Required)

      -a,--airspace           Airspace to use to calculate the record x/y
map coordinates (Required)

      -d,--dist               Minimum distance to report discrepancies on
(defaults to 5[nm])

      -alt,--altitude         Minimum altitude to report discrepancies on
(defaults to 800 [ft])

      -o,--output             Output SimE file with modified locations
```

Example: `java -Xmx1000M faa.tg.util.LocationModSimE -r AircraftState.csv -ea ARTS.xml -d 0.5 -o out_ -a zdc -s TC4FS10_v2.ZDC -alt 600`

This program creates a log file called simeReplace.log, which contains any error or warning messages the program issued.

makeTgfJar.pl

LOCATION: CVS tgf/bin

LANGUAGE: Perl

OPERATING SYSTEM: Linux

PURPOSE: This script is used to make a JAR file that can run either the TGF simulator (ECO) or DRAT (if the –drat option is specified). The JAR file is derived from the class files in the TGF tree that the user is working from. If the simulation recordings are to be kept, the CVS tree should be tagged. Otherwise, the JAR file is created for developmental purposes only (which is the default).

HOW TO EXECUTE:

- makeTgfJar.pl [-v] [-drat] [-noci | -batch] [-name[JARNAME] | [-dev | -src | -tag [CVSTAG] | -rel [CVSTAG]]

Parameters: (options)

-v verbose (gives messages on what its doing on each file).
-drat creates a JAR that starts DRAT instead of the TGF simulator(ECO); the default JAR name becomes “drat<dateString>. Jar”.
-noci prevents the log file from being checked into CVS.
-batch development purposes only (no CVS check in and no logging).
-name<JARNAME> names the output JAR to the name specified - will append .jar if necessary. FOR EXAMPLE: the JAR file name is tgf1.jar if –nametgf1 is an option.
-dev (default) creates a JAR file not intended for saving recordings; the CVS tree is not tagged. This option will request specific log file information beforehand in order to include the log in the new JAR file. Also, this option will check the log file into CVS as a permanent record.
-**src** creates a JAR file that includes source (.java) files; the CVS tree is not tagged.
-**tag** adds tagging to JAR process: the user’s last check out of theCVS tree is tagged with “JAR_<filename>”, and the JAR filenamewill be “tgftag<dateString>”. Since the tag is applied to the last version of the tree that you checked out/in, CHECK IN DESIREDCHANGES FIRST FOR THE TAGS TO BE CORRECT! The source code can then be reconstructed using: “cvs upd –r <tagname>”
-tag<CVSTAG> same as “-tag” except that the CVS tree is tagged with the given <TAGNAME> instead of “JAR_<filename>”

FOR EXAMPLE: the CVS tag is Sats2005 using –tagSats2005 as an option.

-rel for official TGF releases: the latest version of the CVS treeis tagged with "JAR_<filename>", and the JAR filename will be tgfrel<dateString>". This is intended for very important versionsof the CVS tree.

-rel<CVSTAG> same as "-rel" except that the CVS tree is taggedwith the given <TAGNAME> instead of "JAR_<filename>".

FOR EXAMPLE: CVS tag is Sats2005-04_FormalRun when using tagSats2005-04_FormalRun as an option.

plot

LOCATION: CVS tgf/bin

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: This script provides a shortcut to run the Ptolemy Plotter application that is capable of interpreting PlotML format XML files. If multiple files are specified, the plot will overlay each successive plot file over the prior plot files in the order specified allowing the user to compare plots.

HOW TO EXECUTE:

- plot [options] [file(s)]
- Parameters: - **height** (in pixels). Defaults to 1000
  - **width** (in pixels). Defaults to 1400
  - **help** prints help message
  - **test** brings up plot window for a few seconds then closes it
  - **version** displays plot version information

NOTE: Information on the Ptolemy Plotter can be found on the internet at:
http://ptolemy.eecs.berkeley.edu/java/ptplot/
The Ptolemy Plotter version that TGF uses has been modified by Mike Ross.

ReCompile.pl

LOCATION: CVS tgf/bin

LANGUAGE: Perl

OPERATING SYSTEM: Linux/UNIX

PURPOSE: This script does a Java compile for the current directory and its subdirectories. It is a quick way to recompile a small subset of files that have been changed since the last compile.
(This particular script is the same as ReCompile.sh but runs faster.)

For example, if three files in one directory have been edited since they were last compiled, the user could run ReCompile.pl in or above that directory to update the class files.

HOW TO EXECUTE:

- ReCompile.pl <enter> will automatically search for uncompiled files and then compile them.
- Parameters: **-v** makes it verbose (gives messages on what its doing on each file)
  -q makes it quiet (gives no messages)
  -cp specifies a class path (must be the last argument)
  -debug or **–g** specifies compiling with the debug option (gives extra messages in case something goes wrong)
  -all or **–a** recompiles everything in the subdirectory whether or not its been edited

NOTE: There is also a build.xml file for compiling TGF with ANT, which will build all of the class much faster.

RemoteDiagnosticTool

LOCATION: faa.tg.util.aircraftViewer

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: This program will bring up the Simulation Action Viewer on a different machine, independent of the ECO (Exercise Control Operator). It allows the user to view and customize any information, prompts, command input/output, etc. on active, pending or terminated a/c during a particular simulation. For more detailed information, refer to the TGF User's Manual Section 5.2 The Simulation Action Viewer.

HOW TO EXECUTE:

- java faa/tg/util/aircraftViewer/RemoteDiagnosticTool [-o]
- the offset of the simulation is required and is specified via a command line option **–o**

If user types in java faa/tg/util/aircraftViewer/RemoteDiagnosticTool the following will appear:

```
java faa/tg/util/aircraftViewer/RemoteDiagnosticTool

Usage: faa.tg.util.aircraftViewer.RemoteDiagnosticTool
            [{-o,--offset}]

        -o,--offset      The port offset of the simulation (Required)
```

SimDiff

NOTE: This is one of an array of V&V (Verification and Validation) tools that are used to compare two TGF recordings with the same setup parameters. Each tool, using its own set of criteria, determines differences in the two simulations to see if the TGF simulator will behave the same (or improve) when code changes occur. The developer can use these tools to examine the effects of code changes before checking them in. Simply put, each V & V tool compares two recordings, but each compares different things about the two recordings.

LOCATION: CVS tgf.src.faa.tg.dra.tools

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: Basically, this tool compares two TGF recordings (creating text output) to determine what parts of the TGF recordings differ. SimDiff looks for textual disparities in simulator outputs. It provides comprehensive to very brief information on a wide array of user specifiable alterations. The classes to incorporate in the comparisons must be specified to the constructor via a properties file containing the class names or the package names (package names must end with a ".") to generate a compareSet of classes to evaluate. SimDiff compares events to each other (using the TextMatchCriteria) to determine if there are any disparities in the simulations. This can be a stringent test, depending on the classes selected, given that any fractional discrepancy that is displayed in the text output (of matching objects that implement TextWritable) will be deemed a difference between the simulations.

HOW TO EXECUTE:

- java tgf.src.faa.tg.dra.tools.SimDiff [-noinfo] baseFile testFile [outfile (default: "dif.txt") [propertiesFile]
- **-q** prevents messages to standard output
- **-noinfo** prevents an information message containing metrics from being fired.
- If no properties are specified, **default properties** (files or packages to compare text output of) are:

  faa.tg.sp.SpCommandResult: true
  faa.tg.recording.state.FlightTerminatedEventState: true
  faa.tg.aircraft.FlyingStatusChangeEvent: true
  faa.tg.nas: true
  faa.tg.recording.state.FlightActivatedEventState: true

Examples are as follows:

Examples of the output in the "dif.txt" file:

```
BAD MATCH.  Criteria: faa.tg.dra.tools.TextMatchCriteria
  Reason: Text for objects does not match.
   Old AIRPORT: MMU  Mag dec: -13.0 deg  Rwys: 23
   Test AIRPORT: MMU  Mag dec: -0.0 deg  Rwys: 23

BAD MATCH.  Criteria: faa.tg.dra.tools.TextMatchCriteria
  Reason: Test object had no matching base object.
   Old: null
   Test RUNWAY: 23 Heading: 216.0 deg Length: 5999.0 ft Width: 150.0 ft
   Threshold:(Lat: 40-48-23.052N Long: 074-24-23.301W Alt: 183 ft)
```

SimValidator

NOTE: This is one of an array of V&V (Verification and Validation) tools that are used to compare two TGF recordings with the same setup parameters. Each tool, using its own set of criteria, determines differences in the two simulations to see if the TGF simulator will behave the same (or improve) when code changes occur. The developer can use these tools to examine the effects of code changes before checking them in. Simply put, each V & V tool compares two recordings, but each compares different things about the two recordings.

LOCATION: CVS tgf.src.faa.tg.dra.tools

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: SimValidator provides a more targeted simulation comparison than SimDiff, only reporting on recordable objects that have been defined in the code to be comparable types by implementing SimComparable. It will, by default, generate a detail file (val.txt) and a summary file (val.dur.txt).
Currently, the tests that occur are a location test for aircraft state and a test for new error messages generated during the simulation.

Programmers Note: SimValidator extends SimDiff but does not use the SimDiff compareSet. This is because the compareSet consists only of the objects that implement Java class SimComparable, indicating that they have a self-defined criteria to compare objects between two simulations to verify that they are equivalent.

HOW TO EXECUTE:

- java faa.tg.dra.tools.SimValidator [-q] [-noinfo] baseDra testDra [(outfile)]
- **-q** prevents messages to standard output
- **-noinfo** prevents a final information message containing bad match counts and file information from being sent to the text file.

NOTE: User must specify recording files (required) and can specify an outfile name (optional). If an outfile name is specified, the summary file will have a .dur added to it. If not, then the default is val.txt and val.dur.txt.

Examples are as follows:

Example of the output in the "val.txt" file:

```
BAD MATCH.  Criteria: faa.tg.dra.tools.PositionCriteria  Time: 00:03:11
  Reason: Distance is greater than 1.0 ft apart: 23.26 ft
 Old PER SEC: AAL8117  Time: 00:03:11  AC type: MD80  Start time: 00:00:01
  Bcn: 4502  Sector: TERM_1  Freq: 127.300  SP: NO_SPW  FlStat: ON_ROUTE
  Lat: 40-23-38.633N  Long: 075-55-04.733W  Alt: 27587 ft  Hdg: 226.06373 deg
  TAS: 435.108  IAS: 289.988 kts  Alt rate: 1194.924 ft/m  TurnR: 0.0003deg/s
  Filed: LRP059046.LRP
 Test PER SEC: AAL8117  Time: 00:03:11  AC type: MD80  Start time: 00:00:01
  Bcn: 4502  Sector: TERM_1  Freq: 127.300  SP: NO_SPW  FlStat: ON_ROUTE
  Lat: 40-23-38.653N  Long: 075-55-04.705W  Alt: 27610.1 ft  Hdg: 226.06383 deg
  TAS: 434.344  IAS: 289.327 kts  Alt rate: 1761.722 ft/m  TurnR: 0.0003deg/s
  Filed: LRP059046.LRP
```

Example of the output in the "val.dur.txt" file:

```
DURATION: 99  Start time: 00:03:09  End time: 00:04:47  Name: AAL8117_Dista  1ST BAD
MATCH.  Criteria: faa.tg.dra.tools.PositionCriteria  Time: 00:03:09
  Reason: Distance is greater than 1.0 ft apart: 4.8 ft
 Old PER SEC: AAL8117  Time: 00:03:09  AC type: MD80  Start time: 00:00:01
  Bcn: 4502  Sector: TERM_1  Freq: 127.300  SP: NO_SPW  FlStat: ON_ROUTE
  Lat: 40-23-48.684N  Long: 075-54-51.013W  Alt: 27547.1 ft  Hdg: 226.063 deg
  TAS: 434.838  IAS: 289.988 kts  Alt rate: 1200.944 ft/m  TurnR: 0.0004deg/s
  Filed: LRP059046.LRP
 Test PER SEC: AAL8117  Time: 00:03:09  AC type: MD80  Start time: 00:00:01
  Bcn: 4502  Sector: TERM_1  Freq: 127.300  SP: NO_SPW  FlStat: ON_ROUTE
  Lat: 40-23-48.687N  Long: 075-54-51.008W  Alt: 27551.9 ft  Hdg: 226.06306 deg
  TAS: 434.622  IAS: 289.809 kts  Alt rate: 1521.144 ft/m  TurnR: 0.0004deg/s
  Filed: LRP059046.LRP
 LAST BAD MATCH.  Criteria: faa.tg.dra.tools.PositionCriteria  Time: 00:04:47
  Reason: Distance is greater than 1.0 ft apart: 23.77 ft
 Old PER SEC: AAL8117  Time: 00:04:47  AC type: MD80  Start time: 00:00:01
  Bcn: 4502  Sector: TERM_1  Freq: 127.300  SP: NO_SPW  FlStat: ON_ROUTE
  Lat: 40-15-32.935N  Long: 076-06-07.126W  Alt: 28000 ft  Hdg: 226.06395 deg
  TAS: 437.999  IAS: 290.057 kts  Alt rate: 0.033 ft/m  TurnR: -0.0deg/s
  Filed: LRP059046.LRP
 Test PER SEC: AAL8117  Time: 00:04:47  AC type: MD80  Start time: 00:00:01
  Bcn: 4502  Sector: TERM_1  Freq: 127.300  SP: NO_SPW  FlStat: ON_ROUTE
  Lat: 40-15-33.098N  Long: 076-06-06.905W  Alt: 27999.8 ft  Hdg: 226.06398 deg
  TAS: 437.999  IAS: 290.057 kts  Alt rate: 1.123 ft/m  TurnR: -0.0deg/s
  Filed: LRP059046.LRP
```

SimValueDiff

NOTE: This is one of an array of V&V (Verification and Validation) tools that are used to compare two TGF recordings with the same setup parameters. Each tool, using its own set of criteria, determines differences in the two simulations to see if the TGF simulator will behave the same (or improve) when code changes occur. The developer can use these tools to examine the effects of code changes before checking them in. Simply put, each V & V tool compares two recordings, but each compares different things about the two recordings.

LOCATION: CVS faa.tg.dra.tools

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: SimValueDiff compares base and test files. It looks for disparities in fields that can be accessed by getMethodMap() in tagged DelimWritable objects, which includes aircraft state and most aircraft change of state objects. This is less sensitive to accumulated differences of aircraft position than SimDiff and SimValidator. It will, by default, generate a detail file (value.txt) and a summary file (value.dur.txt).

This tool requires a properties file to be specified. An example of a set of properties is below. Also, a properties file with a detailed description of the value encoding is in faa/tg/dra/tools/SimValueDiff.properties. Creating a new properties file requires knowledge of the class of DelimWritable objects being targeted.

Programmer's Note: SimValueDiff uses the keys in the properties file as keys to look for in the MethodMap (from the getMethodMap() implemented by DelimWritable objects). If the MethodMap contains a matching field name key, then the method (or method chain) specified by that key is called. The properties file value encoding is then used to test the values between the simulation objects. If the difference amount from the properties file is exceeded, a BadMatch with a description of the difference is sent. SimValueDiff extends SimDiff.

HOW TO EXECUTE:

- java faa.tg.dra.tools.SimValueDiff [-q] [-noinfo] baseDra testDra propertiesFile [outfile (default: "value.txt")]
- **-q** prevents messages to standard output
- **-noinfo** prevents an information message containing bad match counts and file information from being sent to the text file.

NOTE: User must specify recording files (required) and can specify an outfile name (optional). If an outfile name is specified, the summary file will have a .dur added to it. If not, then the default is value.txt and value.dur.txt.

Examples are as follows:

Example of Properties File:

```
This set of properties has been shown to find differences between
simulations without being overly strict:
  DesAlt= +0.1  (bad match if desired altitudes differ by more than 1)
  DesHdg= +0.1  (bad match if desired headings differ by more than 1)
  DesIAS= +0.1 (bad match if desired indicated airspeeds differ by 1+)
  DesRoll= +0.1  (bad match if desired roll angles differ by 1+)

  Hdg= +1, *0.01 (bad match if actual headings differ by 1+ AND 1% or more)
  TAS= +1, *0.01  (bad match if true airspeeds differ by 1+ AND 1% or more)
  Alt= +10, *0.01   (bad match if altitudes differ by 10+ AND 1% or more)
  AltRate= +10, *0.01  (bad match if altitude rates differ by 10+ AND 1% or more)
```

Example of the output in the "value.txt" file (tested for altitude rate difference of more than 10ft/min or of more than 1%):

```
BAD MATCH.  Criteria: .dra.tools.ValueCriteria:AltRate  Time: 00:00:06
  Reason: AltRate_maxDiff_maxPerctDiff  Base=-2493.44; Test=-2607.14
  Reason detail:  Diff>10: 113.705  DiffPerct>1: 4.56%
 Old AIRCRAFT: A310  Time: 00:00:06  AC type: A310  Start time: 00:00:05
  Bcn: 1013  Sector: DEPARTURE  Freq: 122.000  SP: n/a  FlStat: ON_ROUTE
  Lat: 39-51-40.414N  Long: 075-08-03.534W  Alt: 40917.1 ft  Hdg: 24.5438 deg
  TAS: 459.197  IAS: 237.574 kts  Alt rate: -2493.436 ft/m  TurnR: -0.111 deg/s
  Filed: NAVYY.ARD
 Test AIRCRAFT: A310  Time: 00:00:06  AC type: A310  Start time: 00:00:05
  Bcn: 1013  Sector: DEPARTURE  Freq: 122.000  SP: n/a  FlStat: ON_ROUTE
  Lat: 39-51-40.451N  Long: 075-08-03.642W  Alt: 40916.4 ft  Hdg: 23.55211 deg
  TAS: 459.207  IAS: 237.584 kts  Alt rate: -2607.141 ft/m  TurnR: -1.365 deg/s
  Filed: NAVYY.ARD
```

Example of the output in the "value.dur.txt" file (tested for altitude rate difference of more than 10ft/min or of more than 1%):

```
DURATION: 5  Start time: 00:00:06  End time: 00:00:10  Name: A310_AltRate_ma  1ST
BAD MATCH.  Criteria: .dra.tools.ValueCriteria:AltRate  Time: 00:00:06
  Reason: AltRate_maxDiff_maxPerctDiff  Base=-2493.44; Test=-2607.14
  Reason detail:  Diff>10: 113.705  DiffPerct>1: 4.56%
 Old AIRCRAFT: A310  Time: 00:00:06  AC type: A310  Start time: 00:00:05
  Bcn: 1013  Sector: DEPARTURE  Freq: 122.000  SP: n/a  FlStat: ON_ROUTE
  Lat: 39-51-40.414N  Long: 075-08-03.534W  Alt: 40917.1 ft  Hdg: 24.5438 deg
  TAS: 459.197  IAS: 237.574 kts  Alt rate: -2493.436 ft/m  TurnR: -0.111 deg/s
  Filed: NAVYY.ARD
 Test AIRCRAFT: A310  Time: 00:00:06  AC type: A310  Start time: 00:00:05
  Bcn: 1013  Sector: DEPARTURE  Freq: 122.000  SP: n/a  FlStat: ON_ROUTE
  Lat: 39-51-40.451N  Long: 075-08-03.642W  Alt: 40916.4 ft  Hdg: 23.55211 deg
  TAS: 459.207  IAS: 237.584 kts  Alt rate: -2607.141 ft/m  TurnR: -1.365 deg/s
  Filed: NAVYY.ARD
 LAST BAD MATCH.  Criteria: .dra.tools.ValueCriteria:AltRate  Time: 00:00:10
  Reason: AltRate_maxDiff_maxPerctDiff  Base=-2426.28; Test=-2458.83
  Reason detail:  Diff>10: 32.549  DiffPerct>1: 1.34%
 Old AIRCRAFT: A310  Time: 00:00:10  AC type: A310  Start time: 00:00:05
  Bcn: 1013  Sector: DEPARTURE  Freq: 122.000  SP: n/a  FlStat: ON_ROUTE
  Lat: 39-52-08.215N  Long: 075-07-47.056W  Alt: 40753.9 ft  Hdg: 24.38248 deg
  TAS: 459.166  IAS: 238.46 kts  Alt rate: -2426.277 ft/m  TurnR: -0.026 deg/s
  Filed: NAVYY.ARD
 Test AIRCRAFT: A310  Time: 00:00:10  AC type: A310  Start time: 00:00:05
  Bcn: 1013  Sector: DEPARTURE  Freq: 122.000  SP: n/a  FlStat: ON_ROUTE
  Lat: 39-52-08.613N  Long: 075-07-48.251W  Alt: 40744.2 ft  Hdg: 22.8108 deg
  TAS: 459.204  IAS: 238.536 kts  Alt rate: -2458.827 ft/m  TurnR: 0.054 deg/s
  Filed: NAVYY.ARD
```

StarXml2Xpvd

LOCATION: CVS faa.tg.prep.util.

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: This script converts a STAR (Standard Terminal Arrival Route) Xml file into Xpvd format.

HOW TO EXECUTE:

- java faa.tg.prep.util.StarXml2Xpvd –i(input file)-o(output file)
- Input file which is the STAR Xml file is required for the conversion and is specified via a command line option **–i**
- Output file which is the file to write the XPVD to is required and is specified via a command line option **-o**

Tunneller

LOCATION: faa.tg.net.tunneller

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: This script creates a tunnel socket to pass data from one wire to another. Tunneller can be used with or without a firewall present.

HOW TO EXECUTE:

- java faa.tg.net.tunneller.Tunneller
- Parameters: **-c** <server ip:port>client machine to connect to after starting the server.
  -r causes the tunneller to read all data from the specified address/port and send it through the tunnel to the other side.
  -w specifies the socket/address where the data that is passed through the tunnel is to be written.
  -s server, **default** listening port for client connections/tunnel data.
  -t <port> server application – but use the specified port to hear client connections/tunnel data.

NOTE: a server/client can function as a reader or a writer.

What follows are examples of the various methods for utilizing Tunneller.

A running client program that reads data and sends it to a server.

java.faa.tg.net.tunneller.Tunneller –c 172.26.65.98 –r 172.26.64.255:3224

Translation: run a client application and connect to the server located at 172.26.65.98. The client will read data from 172.26.64.255:3224 and send it through the tunnel to the server.

java fa.tg.net.tunneller.Tunneller –s –r 255.255.255.255:3624

Translation: run a server application and read data from 255.255.255.255:3624 – sending it through the tunnel to the client.

(W)rite takes data received from the tunnel and writes it out to a wire.

java.faa.tg.net.tunneller.Tunneller –s –w 172.26.65.255:3223

Translation: run a server application, all data received through the tunnel will be written to the 172.26.65.255 address on port 3223.

Firewall location decides who the **C**lient is and who the **S**erver is. Whoever wants to receive the data is the **W**riter. Whoever wants to send the data is the **R**eader.

Example: firewall

TGF	Hangar
Server / Reader	(firewall) | Client / Writer

User wants to send data from the TGF sim lab to the Hangar. But, the Hangar has a firewall. Tunneller will create a socket to bypass the firewall. The TGF would become the Server and the Hangar is the Client. Since the user is sending the data from TGF to the Hangar, the TGF is also the reader and the Hangar is the writer.

Example: firewall

TGF	Hangar
Client / Writerl (firewall)	Server / Reader

User wants to receive data from the Hangar to the TGF lab. But, the TGF has a firewall. Tunneller will create a socket to bypass the firewall. The Hangar would become the Server and the TGF would become the Client. Since the user is receiving data from the Hangar to the TGF, the TGF is the writer and the Hangar is the reader.

Example: no firewall

TGF	Hangar
Client / Reader	Server / Writer

In some cases, a firewall is not present yet data cannot get through for some reason. In this circumstance, Tunneller can be used to transmit the data. Tunneller will take the data (in one protocol) to be transmitted, wrap it in another protocol, send it across the wire, unwrap it on the other side and guarantee that all of the data will get to the Writer on the other side in proper sequence. In this example, the user wants to send data from the Hangar to the TGF lab but for some reason it won't go through. Using Tunneller, the data from the Hangar is enveloped in Transmission Connection Protocol (TCP), guaranteeing that all information will get across in sequence, and sent through the tunnel to the TGF Lab. It will then remove the TCP casing on the TGF side. The user should know what format the data to be transmitted is using before sending it through a tunnel in order to ensure that the Client will be able to read the data once it is unwrapped.

Xml File Creator

LOCATION: CVS faa.tg.prep.xmleditor.gui.XmlFileCreator

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: This Java class expedites the process of creating a new Xml file by using the appropriate schema. It fills in the minimum number of elements possible to be valid and passes that information to the Xml Editor. The user then fills in the rest of the values using the Xml Editor in order to complete the Xml file according to the schema.

HOW TO EXECUTE:

- java faa.tg.prep.xmleditor.gui.XmlFileCreator
- FileCreator asks which schema is to be used and where it is to be saved.
- GUI appears asking for: -the schema to use
  -where to save the Xml file
  -the primary key
  -the primary key value
- Then, open the file (which opens the Xml Editor).
- The Xml Editor requires the repeated element.
- The first error that Editor comes to will cause a message to appear stating that the file is not valid.
- Xml Editor will then red highlight (invalidate) the file errors.
- User will then fill in the correct information to validate (white highlight) the Xml file.
- The file can now be saved and placed wherever the user needs it.

XpvdFpa2Xml

LOCATION: CVS tgf/src/faa/tg/prep/util

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: translates old Xpvd Fpa text file into XML format. The XML file is more uniform with the standard format of TGF files. The Fpa connects the nodes that define an airspace sector and gives it an Fpa ID and sector name.

HOW TO EXECUTE:

- java faa.tg.prep.util.XpvdFpa2Xml -i (input file) -o (output file)
- Input file which is the Xpvd Fpa file is required for the conversion and is specified via a command line option **-i**
- Output file which is the file to write the TGF Xml to is required and is specified via a command line option **-o**

NOTE: This utility is usually used in conjunction with XpvdNode2Xml.

If user types in java faa.tg.prep.util.XpvdFpa2Xml the following will appear:

```
java faa.tg.prep.util.XpvdFpa2Xml

Usage: faa.tg.prep.util.XpvdFpa2Xml [{-i,--input}] [{-o,--outfile}]

      -i,--input        The Xpvd Fpa file (Required)

      -o,--outfile      The file to write TGF XML to. (Required)
```

This will remind the user of the format that needs to be used.

XpvdLinesToAcesGeoMapXml

LOCATION: CVS faa.tg.atcview.XpvdLinesToAcesGeoMapXml

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: This translates an Xpvd Line text file into an Aces Geo Map Xml. The Xml is more uniform with the standard format of TGF files. In addition, the Xpvd Line text file can only be read by the Xpvd program whereas the Xml file can be imported and displayed by the Qpvd, Xpvd, and Jpvd programs.

HOW TO EXECUTE:

- java faa.tg.atcview.XpvdLinesToAcesGeoMapXml –i [input file] –o [output file].
- The input file is the Xpvd lines file that is required for the conversion and is specified via a command line option **–i.**
- The output file is the file to write the TGF Xml to is required and specified via a command line option **–o**.

If user types in java faa.tg.atcview.XpvdLinesToAcesGeoMapXml the following will appear:

```
java faa.tg.atcview.XpvdLinesToAcesGeoMapXml

Usage: faa.tg.atcview.XpvdLinesToAcesGeoMapXml [{-i,--input}]
[{-o,--outfile}]

            -i,--input        The Xpvd lines file (Required)

            -o,--outfile      The file to write TGF XML to. (Required)
```

This will remind the user of the format that needs to be used.

XpvdNode2Xml

LOCATION: CVS tgf.src.faa.tg.prep.util

LANGUAGE: Java

OPERATING SYSTEM: any system that accepts Java

PURPOSE: translates old Xpvd node text file into Xml format. The Xml file is more uniform with the standard format of TGF files. The Node file defines the points to be connected for a particular airspace sector.

HOW TO EXECUTE:

- Java faa.tg.prep.util.XpvdNode2Xml –i (input file) –o (output file)
- Input file which is the Xpvd Node file is required for the conversion and is specified via a command line option **-i**
- Output file which is the file to write the TGF Xml to is required and is specified via a command line option **-o**

NOTE: This utility is usually used in conjunction with XpvdFpa2Xml.

If user types in java faa.th.prep.util.XpvdNode2Xml the following will appear:

```
java faa.tg.prep.util.XpvdNode2Xml

Usage: faa.tg.prep.util.XpvdNode2Xml [{-i,--input}] [{-o,--outfile}]

      -i,--input        The Xpvd Node file (Required)

      -o,--outfile      The file to write TGF XML to. (Required)
```

This will remind the user of the format that needs to be used.