NASA / CR-1998-207658

Aircraft / Air Traffic Management Functional Analysis Model, Version 2.0, User's Guide

Melvin Etheridge, Joana Plugge, and Nusrat Retina
Logistics Management Institute, McLean, Virginia

April 1998

NASA / CR-1998-207658

Aircraft / Air Traffic Management Functional Analysis Model, Version 2.0, User’s Guide

Melvin Etheridge, Joana Plugge, and Nusrat Retina
Logistics Management Institute, McLean, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NAS2-14361

April 1998

Available from the following:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 487-4650

Contents

FIGURES

TABLES

Chapter 1
Introduction

The Aircraft/Air Traffic Management Functional Analysis Model, Version 2.0 (FAM 2.0), is a discrete event simulation model designed to support analysis of alternative concepts in air traffic management and control. FAM 2.0 was developed by the Logistics Management Institute (LMI) under task order NS703 of the National Aeronautics and Space Administration (NASA) contract number NAS2-14361. This document provides a guide for using the model in analysis. Those interested in making enhancements or modifications to the model should consult the companion document, *Aircraft/Air Traffic Management Functional Analysis Model, Version 2.0 Technical Description.*

PURPOSE OF THE MODEL

FAM 2.0 is designed to be used by personnel at NASA, the Federal Aviation Administration (FAA), and other organizations and institutions. Those who analyze and decide among competing programs for modernizing air traffic management may find that FAM 2.0 is a useful tool. We intend the model to be usable with little or no instruction by individuals who are unfamiliar with either the model or the host simulation environment. The intended user is the analyst, not the modeler.

FAM 2.0 is designed to provide quantitative time and queuing information about:

- personnel work/task loads,
- equipment demand/utilization, and
- communications channel saturation.

This information is for:

- aircraft,
- air traffic management and control, and
- airline operations centers.

CAPABILITIES

FAM 2.0 provides users the flexibility to define the simulation scenario to address the particular issue or question under analysis. Baseline simulation scenarios come with the model, representing several different 3 hour periods of all flight operations by the Denver Air Route Traffic Control Center (ARTCC), Denver Terminal Radar Approach Control (TRACON), and the Denver International and Colorado Springs Municipal Airports. Users can modify the baseline scenario or load an entirely new scenario if desired. Permissible user modifications include

- adding or deleting scenario events,
- changing the model’s behavior when an event occurs,
- changing the characteristics of simulation objects (i.e., aircraft and ARTCC sectors), and
- defining new simulation objects (i.e., aircraft and ARTCC sectors).

These modifications are made to simple text files. Generally, users make a change once to the appropriate file in the baseline scenario and the model applies that change wherever appropriate in the simulation. Similarly, entirely new files in the appropriate format can be loaded at simulation initialization to replace corresponding parts of the baseline.

FAM 2.0 was developed in the MODSIM III simulation environment hosted on an HP-UNIX platform. Since MODSIM III generates an executable (.exe) file, FAM 2.0 can run on any HP-UNIX platform. It is available from LMI, McLean, Virginia.

Chapter 2

Description of Model Simulation Events

FAM 2.0 is a discrete event simulation model centered around the events associated with a given simulation scenario. Currently, the model replicates the operations of Denver ARTCC, Denver TRACON, and the Denver and Colorado Springs Municpal Airports. All flights handled by one or more of the replicated facilities are in the baseline simulation. Users have the option of modifying one or more parts of the baseline simulation and/or entering an entirely new scenario.

SIMULATION EVENTS

The simulation has two type of events:

- *A priori events.* Events that are known in advance for each flight.
- *Random events.* Events that occur randomly during a flight.

An example of an A priori event would be a handoff of an aircraft from one controller to another. Random events include both routine and unusual or emergency events that occur at irregular intervals, if at all, such as a request for a change of flight level.

Our approach is that each of these primary events, both A priori and random, has a fixed set of associated sub-events. Continuing the previous example, a handoff from one controller to another might be broken down into the following associated sub-events:

- Request from losing to gaining controller to take control
- Acceptance of control by gaining controller to losing controller
- Instructions from losing controller to aircraft to contact gaining controller
  .
  .
  .
- Aircraft “rogers” acknowledgment.

A request for change of flight level might have these associated subevents:

- Aircraft contacts controller
- Controller “rogers”
- Aircraft requests new flight level
  .
  .
  .
- Controller clears aircraft to climb/descend to new flight level or denies request
- Aircraft acknowledges.

There are, then, two levels of events: (1) the primary events; and, (2) for each primary event, a set of associated sub-events. To differentiate between the two, hereinafter **we note a primary Event with a upper case ‘e’ (“Event”) and a associated sub-event with lower case ‘e’ (“event”).**

During the simulation run, whenever a FAM 2.0 primary Event occurs, the model executes the set of associated events. Each of the associated events carries with it personnel task loadings, equipment requirements, and communications channel demands**—**all in units of time.

There can be more than one set of associated events for each A priori Event type. The associated event sets could vary according to the equipment installed on the aircraft or available to the controller. An example could be the use of data link to provide certain communications. The situation could exist where some aircraft had a data link and others did not. Communications with controllers would primarily use the data link, if installed. The model would use different sets of associated events in the simulation for aircraft with and without data link.

There are two sources of primary Events. The A priori Events are contained in a text file that is read by the model at the start of the simulation. Random events are generated by a random event generator inside the model. During the simulation, when an Event occurs, whether from the A priori Event file or originated by the random event generator, the model then executes the appropriate sets of associated events.

With this approach, users only need to change a particular set of associated events once before running the model in order to have the change occur throughout the simulation. If, for example, controller handoffs of aircraft were done automatically via a data link, reducing pilot and controller task loading associated with the handoff, an analyst would make the appropriate changes in the event task loads and (possibly) eliminate the “aircraft changes communications frequency” event.

If desired, users can add or eliminate some A priori Events entirely. In the case of adding Events, users must copy a text file of associated events into the appropriate directory.

Two sets of A priori Events reside in the baseline scenarios. One contains the Events associated with flights as they actually occurred. The flights used the current point-to-point system of air navigation based on ground-based navigational radios. The flights were conducted under positive FAA control by ground-based controllers using conventional voice communications radios. The other set of events contains the wind-corrected great circle flight (so-called “free-flight”) tracks for the same flights. This enables the user to compare current navigation procedures and free-flight procedures.

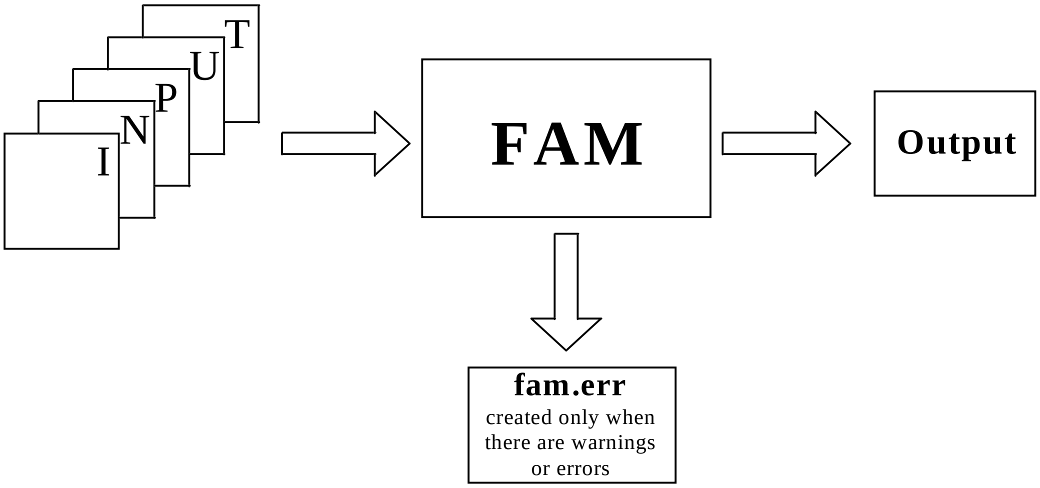
In addition to modifying the events associated with one or more primary Events, and/or changing the primary Event file, users can, if they desire, also read in an entirely new primary Event text file. This would be appropriate if a user wishes to analyze a different set of flights.

Chapter 3

Model Execution

This chapter gives procedures for running FAM 2.0. Figure 3-1 is a diagram of FAM 2.0.

Figure 3-1. Model Diagram



General text and discussion are in this regular font. Since FAM 2.0 is executed at the command line level, this chapter contains specific computer commands using the following typographical conventions. The actual computer commands are given in this font. *Embedded filenames, paths, and flags that are+ set by the user are in italics. If the entry is variable, then it is enclosed in < >.* `This chapter also provides listings of file contents in this font. Again, e`*`mbedded filenames, paths, and flags set by the user within a file listing are in italics.`* You should substitute the appropriate entry for your simulation when you encounter italicized text in a file listing.

FAM 2.0 requires both a scenario file and an output file. These files are mandatory and they must always be provided for FAM 2.0 to run.

RUNNING FAM 2.0

You should be at the command line of the directory containing FAM 2.0 on your HP-UNIX workstation. The command to run FAM 2.0 is

```
$FAM -s <scenariofilename> -o <outputfilename>
```

where:

FAM	- executable model
-s	- scenario file (mandatory)
-o	- output file (mandatory)

INPUT FILES

Input and output files of this model must be in text format. FAM 2.0 requires all input files to be set up properly and contain loads for all activities taking place during simulation. All input files are allowed to have comments as long as there are two forward slashes (//) at the beginning of each comment line. The following subsections describe the formats for each file.

Scenario File

FORMAT

This is a sample scenario file showing its format. Italicized paths and filenames are user inputs (slashes [/], periods [.], and dollar signs [$] are mandatory).

```
[DATA_PATH]
$DATA =./DATA

[WORKING_PATH]
$WORKING =./WORKINGDIRECTORY

[OUTPUT]
stat_start = 0.0
simulation_end = 4000.0

[A PRIORI_EVENT]
A priori_file = $DATA/trig.evt

[RANDOM_EVENT]
random_mode = TRUE
reuse_seed = TRUE
random_file = $DATA/rand.evt
min_inter_random_time = 1.0
max_inter_random_time = 10.0

[EVENT_DICTIONARY]
event_dictionary = $DATA/event.dic

[AIRCRAFT]
aircraft_type = $DATA/aircraft.typ
```

```
[ARTCC]
sector_type = $DATA/sector.typ
sector_dictionary = $DATA/sector.dic

[AOC]
aoc_type = $DATA/aoc.typ
aoc_dictionary = $DATA/aoc.dic

[AIRPORT]
airport_controller_type = $DATA/airport.typ
airport_controller_dictionary = $DATA/airport.dic

[TRACON]
tracon_controller_type = $DATA/tracon.typ
tracon_controller_dictionary = $DATA/tracon.dic
```

All sections and parameters shown above are mandatory except for the random event section ([RANDOM_EVENT]), which is only required if the random_mode is TRUE. Definitions for the file entries are in Table 3-1.

Table 3-1. Scenario File Entry Definitions

File entry	Definition
$DATA	Character variable for directory containing event data files, such as dictionary, type and load files, etc. You may precede any file name with $DATA.
$WORKING	Character variable for directory where the model is executed. You may precede any file name with $WORKING.
stat_start	Numeric variable for time when statistics will begin to be collected. The value of stat_start must be less than simulation_end.
simulation_end	Numeric variable for time when simulation will end.
A priori_file	A character variable for the master event file that drives the simulation and contains different primary events and other relevant information about these events.
random_mode	A logical variable determining whether random events will occur in the simulation. If random_mode is TRUE, then random events will be generated. If random_mode is FALSE, no random events will be generated.
reuse_seed	A logical variable affecting the seed value the model uses in random event generation. If reuse_seed is TRUE, the same seed value is used from run to run and, thus, the same set of random events will be generated from run to run. Otherwise, if reuse_seed is FALSE, different random events will be generated for each run.

Table 3-1. Scenario File Entry Definitions (Continued)

File entry	Definition
random_file	Character variable with the file name for the random event file, which contains names of random events.
min_inter_random_time	Numeric variable with the minimum inter-arrival time for random events. The value of this variable affects the frequency of random event occurrence.
max_inter_random_time	Numeric variable with the maximum inter-arrival time for random events. The value of this variable affects the frequency of random event occurrence.
event_dictionary	Character variable with the name of the event dictionary file. This file contains all A priori event names and their associated event files.
aircraft_type	Character variable with the name of the aircraft type file. This file contains the types of aircraft found in the A priori event file.
sector_type	Character variable with the name of the sector type file. This file contains the types of sectors that found in the sector dictionary file.
sector_dictionary	Character variable with the name of the sector dictionary file. This file contains the sectors used in the simulation.
aoc_type	Character variable with the name of the AOC type file. This file contains the types of AOCs found in the AOC dictionary file.
aoc_dictionary	Character variable with the name of the AOC dictionary file. This file contains the AOCs used in the simulation.
airport_controller_type	Character variable with the name of the airport controller type file. This file contains the types of controllers used in the airports.
airport_controller_dictionary	Character variable with the name of the airport controller dictionary file. This file contains the airport controllers used in the simulation.
tracon_controller_type	Character variable with the name of the TRACON controller type file. This file contains the types of controllers used in the TRACON controller dictionary file.
tracon_controller_dictionary	Character variable with the name of the TRACON controller dictionary file. This file contains TRACON controllers used in the simulation.

Event Files

Event files are the heart of each simulation run. For A priori events, you must specify the A priori event filename, the event dictionary filename, and the associated event filenames. For random events, you must also specify the random event filename. The following subsections describe the event files. In those descriptions, the NULL value in fields (columns) of the A priori event and event dictionary files means that field is not applicable to that event.

A priori Event File

The A priori event file is a master event file that drives the simulation. It contains all the A priori primary Events and other relevant information about each event.

The times for the Events in the A priori file must be in ascending order of time, (i.e., the time of the Event in the second record [row] of the file must not be earlier that that of the first). This is expressed mathematically in Equation 3-1.

$$0 < T\left(event_1\right) \le T\left(event_2\right) \le \ldots \le T\left(event_n\right) \qquad \text{[Eq. 3-1]}$$

where T is the time of the A priori Event.

There can be more than one Event scheduled at a given time in the simulation. If the Events apply to the same simulation objects, they will be queued so that the lowest numbered Event begins executing first.

All of the Events in the A priori file must, of course, be scheduled before simulation end time. The associated events for these primary Events, however, may inadvertently run past simulation end time If this occurs, the events that are already scheduled will be canceled. Similarly, the associated events for a particular aircraft may run past the aircraft deactivation time. If this occurs, FAM 2.0 will generate a warning and readjust the deactivation time to enable all associated events to be completed.

A priori File Format

A sample A priori event file is in Table 3-2, showing the file layout.

Table 3-2. Sample A priori Event File

EVENT[1]	TIME[2]	AL[3]	FN[4]	Type[5]	AltAL[6]	AltFN[7]	SCT1[8]	SCT2[9]	ARPT[10]	TRC[11]	AOC[12]	ACH[13]
ACTIVATE_AC	1000	UA	1707	747	NULL	NULL	NULL	NULL	DEN	NULL	NULL	0.0
SECT_CHG	1000	UA	1707	747	NULL	NULL	4	5	DEN	DEN	NULL	0.0
SECT_CHG	1100	UA	1707	747	NULL	NULL	5	32	DEN	DEN	NULL	0.0
DEACTIVATE	1200	UA	1707	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0.0

Notes:

1. EVENT = Name of the trigger or Primary Event (mandatory).
2. TIME = Absolute time of occurrence (mandatory).
3. AL = Airline of aircraft A1.
4. FN = Flight number of aircraft A1.
5. Type = Type of aircraft A1 (mandatory only for “ACTIVATE_AC”).
6. AltAL = Airline of alternate aircraft A2.
7. AltFN = Flight number of alternate aircraft A2.
8. SCT1 = Losing or Primary Sector’s ID.
9. SCT2 = Gaining or Alternate Sector’s ID.
10. ARPT = Name of Airport.
11. TRC = Name of TRACON.
12. AOC = Name of AOC.
13. ACH = Channel used for aircraft=to=aircraft communication.

The Events described in the A priori event file can each occur many times in a simulation run. Since there is only one associated event file for each of these events, the associated events use keywords to indicate the fields in the A priori event that designate the actual model objects involved in that occurrence of the associated event. These keywords are defined in Table 3-3.

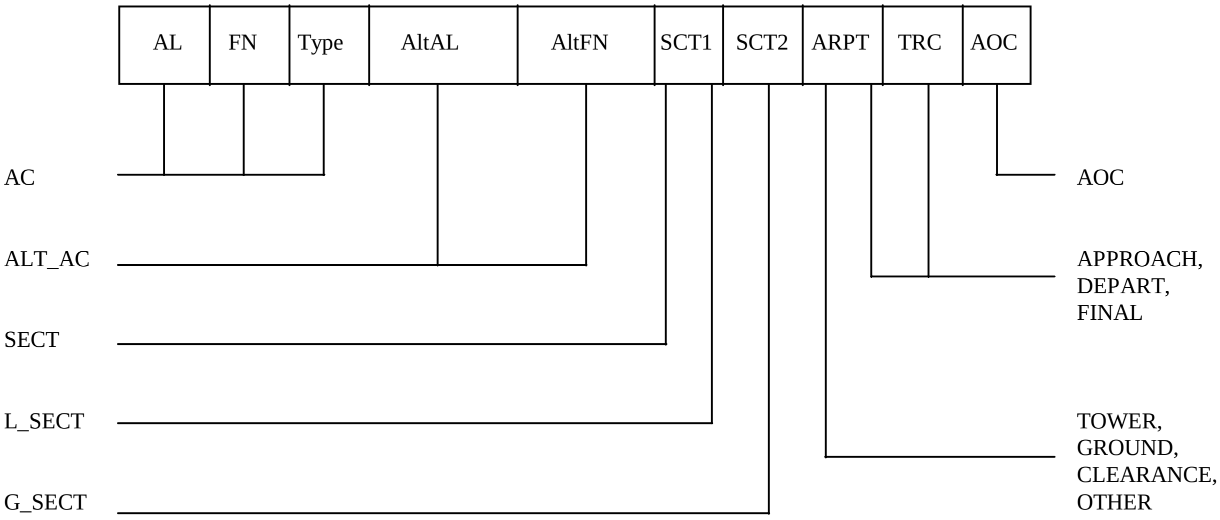
Table 3-3. Associated Event Keywords and Required Fields

Keyword	Required field(s)
AC	Airline, flight number, type of aircraft A1
ALT_AC	Airline, flight number of alternate aircraft A2
SECT	Losing or primary sector’s ID
L_SECT	Losing or primary sector’s ID
G_SECT	Gaining or alternate sector’s ID
AOC	Name of AOC
TOWER, GROUND, CLEARANCE, OTHER	Name of airport
APPROACH, DEPARTURE, FINAL	Name of airport, TRACON

When it encounters a keyword for an event originator or destination, FAM 2.0 looks for the addresses of the actual simulation objects in the A priori file record. Therefore, certain fields must be filled in the A priori file record, as shown in Table 3-3. If you fail to fill in the field, the simulation stops.

Figure 3-2 diagrams the cross-reference between the A priori event record and associated event list keywords.

Figure 3-2. A priori Vector/Keyword Cross-Reference



RANDOM EVENT FILE

The random event file is the master file of the random events to be used in simulation. If the random mode is TRUE (in the scenario file), random events will be generated for a particular aircraft between its activation time and deactivation time. However, if the simulation end time occurs before the aircraft is deactivated, random events will be generated up to the simulation end time.

Random events for aircraft *j* are generated according to Equation 3-2.

$$T\left(ActivateAC_j\right) \leq T\left(RE_1\right) \leq T\left(RE_2\right) \leq \ldots \leq T\left(RE_n\right) < T\left(DeactivateAC_j\right) \quad \text{[Eq. 3-2]}$$

where $T(RE_i)$ is the time of random event *i*.

Random event priorities are specified in the event dictionary file just like A priori events. The number of random events generated depends on the values of the minInterRandomTime and maxInterRandomTime.

FAM 2.0 uses a uniform distribution to generate random events according to Equation 3-3.

$$Frequency\,of\;\; Random\,Events \propto \left[\max InterRandomTime - \min InterRandomTime\right] \quad \text{[Eq. 3-3]}$$

Associated event files for random events are specified in the event dictionary. Figure 3-3 is a sample of a random event file showing the three different random events the simulation would use.

Figure 3-3. Sample Random Event File

```
//RANDOM_EVENT_NAME
ALT_CHG
LOW_FUEL
FIRE_EMERGENCY
```

EVENT DICTIONARY FILE

The event dictionary file lists the names of the files of associated events for each primary Event (A priori or random). All Events in this dictionary file are available for use in simulations. Each primary Event can have more than one associated event list, since the set of associated events for a given primary Event can vary with the type of aircraft and sector participating in the primary Event. Therefore, the event dictionary file contains fields (columns) that identify the types of aircraft and sectors to which the associated event file applies. The dictionary also specifies the priority of the event. The priority determines the rank of service priority when events are queued up for processing at various simulation object, such as sectors, AOCs, airport, and TRACON controllers.

Table 3-4 is a sample event dictionary showing the file format.

In the event dictionary, NULL indicates that the field (column) does not apply to that Event. DEF (default) indicates that the associated event file for that record (row) should be used in all cases unless the type of the objects involved in the actual event are listed in another record. For example, referring to Table 3-4, there are five sector change (SECT_CHG) event records, the first five rows of the table. The first record applies to 747 aircraft where both sectors are type SECTOR_A. Similarly, the second record applies to 747 aircraft where the sector 1 type is SECTOR_A and the sector 2 type is SECTOR_B. The third and fourth records apply when the aircraft is a 777 and the sectors are the types shown. The fifth record, with DEF, applies to all cases not covered by the first four records.

In operation, FAM 2.0 first looks for a record with a match in the appropriate type field(s). If it finds one, it uses the associated event in that record. If no match is found, it looks for DEF and uses that associated event file. If no default row is found, then FAM 2.0 generates an error and quits. For example, if the aircraft type 747 is the primary aircraft participating in an event, FAM 2.0 uses the associated event if it finds a record with 747 in AC1TYP. If FAM 2.0 finds no exact match between 747 and AC1TYP, then it will search for DEF under AC1TYP. If it finds no such record, FAM 2.0 stops and generates an error message.

Table 3-4. Sample Event Dictionary File

EVENT	AC1TYP[1]	AC2TYP[2]	SCT1TYP[3]	SCT2TYP[4]	ASCEVT_FILE[5]	PRIORITY[6]
SECT_CHG	747	NULL	SECTOR_A	SECTOR_A	sector_chg747.evt	5.0
SECT_CHG	747	NULL	SECTOR_A	SECTOR_B	sector_chg747.evt	5.0
SECT_CHG	777	NULL	SECTOR_B	SECTOR_B	sector_chg777.evt	5.0
SECT CHG	777	NULL	SECTOR_B	SECTOR_A	sector_chg777.evt	5.0
SECT_CHG	DEF	NULL	DEF	DEF	sector_chgdef.evt	5.0
DEPART	DEF	NULL	NULL	NULL	departure.evt	5.0
APPROACH	DEF	NULL	NULL	NULL	approach.evt	5.0
ALT_CHG	DEF	NULL	NULL	NULL	altitude_change.evt	10.0
CONFLICT	DEF	NULL	NULL	NULL	conflict.evt	10.0
CATCH_FIRE	DEF	NULL	NULL	NULL	catch_fire.evt	15.0

Notes:

1. Type of primary aircraft (1).
2. Type of second aircraft (2).
3. Type of primary or losing sector (1).
4. Type of secondary or gaining sector (2).
5. Associated event filename.
6. Event priority.

We recommend that you have one record with only DEF value(s) in the appropriate fields (columns) (i.e., a default associated event list,) for each primary event. You may specify other associated event lists in addition to the default case.

If the primary Event involves one aircraft and no sectors, then AC1TYP must contain the type of the aircraft (e.g., 747) or the default value (DEF). If the event involves one aircraft and one sector, then AC1TYP and SCT1TYP must be filled in. In the case of a sector change event, usually an aircraft and two sectors are involved; thus, AC1TYP and both SC1TYP and SCT2TYP must be specified.

ASSOCIATED EVENT FILE

The associated event file contains a list of associated events for a particular primary Event. This is the actual list of activities that will execute when a primary Event (A priori or random) is executed.

Table 3-5 shows a sample associated event file.

Table 3-5. Sample Associated Event File

EVTID[1]	ORG[2]	DST[3]	DLY[4]
REQ_ACC	L_SECT	G_SECT	0.0
ACC_CONT	G_SECT	L_SECT	15.0
INSTR_AC	L_SECT	AC	25.0
TUNE_RAD	AC	AC	35.0
INIT_CALL	AC	G_SECT	45.0

Notes:.

1. EVTID = Identification number (ID) of the associated event.
2. ORG = Originator of the associated event.
3. DST = Destination of the associated event.
4. DLY = Delay in execution time of associated event relative to primary Event time.

The delay time for an associated event is added to the time of the primary Event. The sum is the simulation time when the associated event is executed. As in the A priori event file, the delay times in an associated event file must run sequentially, and they must all be positive values. This is shown in equation 3-4.

$$0 \le DLY\left(AE_1\right) \le DLY\left(AE_2\right) \le \ldots \le DLY\left(AE_n\right) \qquad \text{[Eq. 3-4]}$$

where *DLY(AE$_i$)* is the delay for the ith associated event.

As we discussed earlier, FAM 2.0 uses keywords in the ORG and DST fields of the associated event file to identify the originator and destination of the associated event. FAM 2.0 uses these keywords to tell it which fields in the A priori event record to use to find the address of the actual originator and destination. Table 3-6 lists the definition of these keywords.

Table 3-6. Associated Event File Keyword Definitions

Keyword	Definition
AC	Aircraft
ALT_AC	Alternate aircraft
G_SECT	Gaining sector
L_SECT	Losing sector
AOC	Airport Operation Center
TOWER	Tower controller
GROUND	Ground controller
CLEARANCE	Clearance controller
OTHER	Other controller
APPROACH	Position approach
DEPARTURE	Position departure
FINAL	Position file

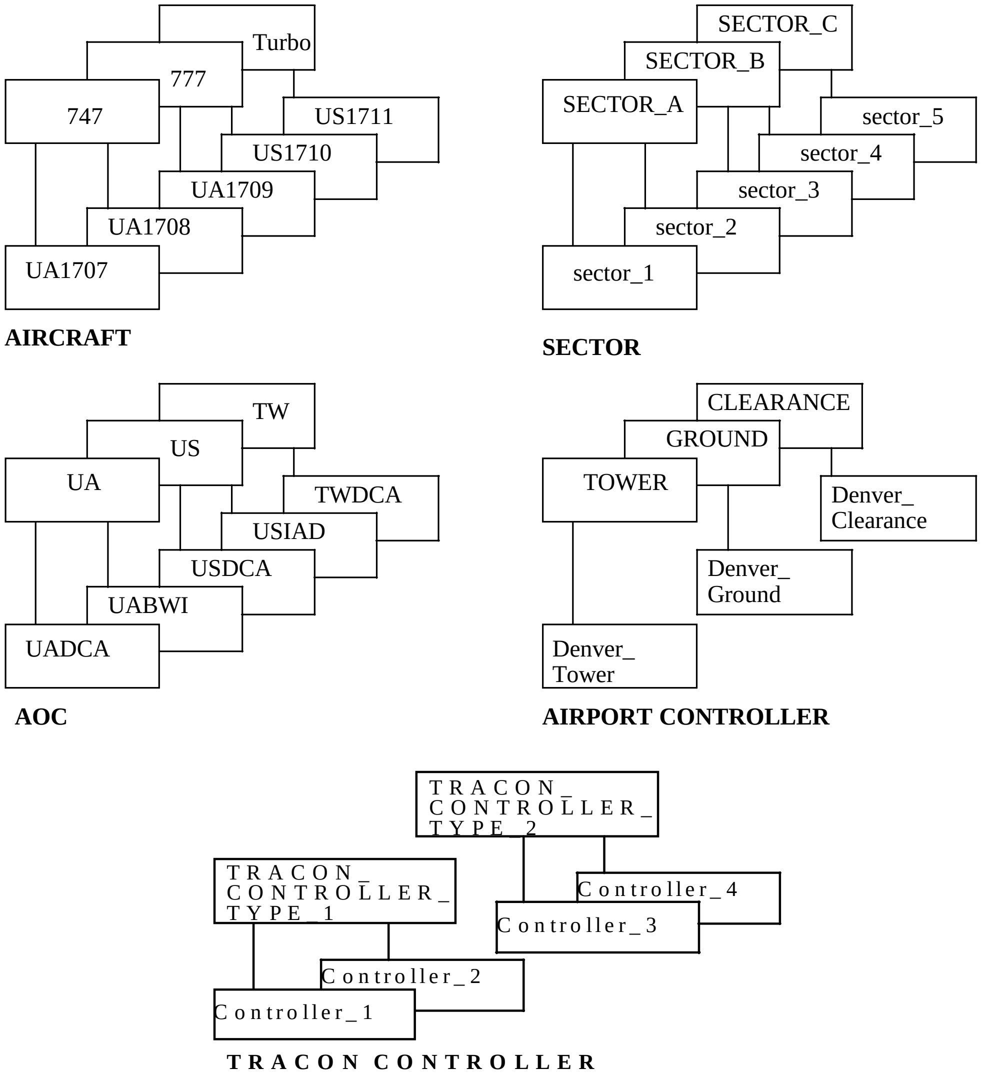
SIMULATION OBJECTS

Simulation Object Types

A simulation object represents a person (e.g., pilot, controller, or dispatcher); communications device (e.g., radio); or equipment system (e.g., radar or computer). A simulation object also can be a collection of other simulation objects, such as an aircraft, which contains simulation objects representing pilots, radios, and equipment. Simulation objects perform activities and/or collect simulation statistical data.

Within each category (e.g., aircraft), there are different types of simulation objects (e.g., 747, 777, or turboprop). Each type of object can have one or more representations. This relationship is shown in Figure 3-4.

Figure 3-4. Relationship Between Simulation Object Types and Representations



Simulation Object Files

Three files define and enumerate simulation objects:

1. *Dictionary file* contains the name, type, and communications channels associated with simulation objects, except aircraft. For aircraft, this information is contained in the A priori aircraft activation Events.

2. *Type file* contains the types[1], paths[2], and filenames of load files for simulation objects

[1] All the types that are used in the event dictionary file, dictionary files, and load files must be defined in the type files. For example, if sector type 'SECTOR_A' is used in the event dictionary file, it must be predefined in the sector type file. Otherwise, if any of the type does not exist, the model will not execute.

[2] You may precede the file name with $DATA to take advantage of the data path set in the scenario file.

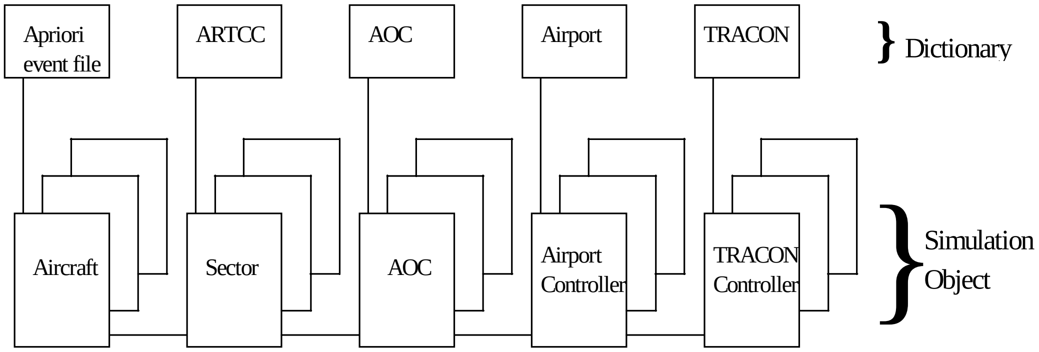
3. *Load file* contains the task loads (for people) and communications channel and equipment utilization for all the activities of the simulation.

Every simulation object must have a dictionary file (except for aircraft), a type file, and a load file.

DICTIONARY FILE

With the exception of aircraft, all simulation objects are listed in one (and only one) of the following dictionaries corresponding to their broad category. This is illustrated in Figure 3-5.

Figure 3-5. Simulation Objects and Dictionaries Relationships



For example, the ARTCC dictionary has a record for every sector in the ARTCC. The fields give the sector's identification number, the type of sector (to allow for different equipage), and the communication channel numbers for up to 10 sector communication devices. Other dictionaries have corresponding fields. Note that there only will be one of each kind of dictionary file in the model.

TYPE FILES

Each simulation object (like aircraft, AOC, sector, etc.) has a type file that contains the different object types and a corresponding load file. The type file usually has the ".typ" suffix in its filename. For example, the aircraft type file is aircraft.typ and the TRACON type file is (tracon.typ). Like dictionary files, there is only one of each kind of type file in the model.

The type files simply cross-reference the type of simulation object to the appropriate load filename. For example, if there are four different types of aircraft in the simulation, there will be one type file listing four load filenames. Figure 3-6 shows a sample aircraft type file.

Figure 3-6. Sample Aircraft Type File (aircraft.typ)

```
//AIRCRAFT_TYPE          AIRCRAFT_LOAD FILENAME
      737                        737.ac
      747                        747.ac
      777                        777.ac
     turbo                      turbo.ac
```

LOAD FILES

Each simulation object type has an associated load file (e.g., 747.ac or sectorA.sct). The load files contain the appropriate task, channel, and equipment loads for each associated event that involves this type of simulation object. There is a record for each event and fields with the event names and corresponding loads. For example, if a particular type of sector has one controller, two radios, and one equipment system of interest, there would be four load fields, one for the controller, two for the radios, and one for the equipment.

In summary, each category of simulation object (except for aircraft) has one dictionary file. There is also one type file for each category of simulation object (including aircraft). There is at least one, and probably several, load files for each category of simulation object. Each individual simulation object is referenced in one and only one dictionary, type, and load file. Each type of file is discussed in more detail in Appendix A. In each load file, there are two additional fields: mode and mate type.

Mode Field

The mode field allows for different loads for an object depending on whether the object is the originator or destination of the event. For example, if two of the same type of sector are involved in an aircraft handoff, the losing sector might have different loads than the gaining sector.

- If the simulation object participates in an activity as the origin, then the mode is ORG.

- If the simulation object participates in an activity as the destination, then the mode is DST.

You may use DEF to specify that the same loads apply regardless of whether the object is the originator or destination. Note also that the same simulation object can be origin and destination for the same activity. This would occur for internal operations, such as aircraft checklists.

Mate-Type Field

The mate-type field allows for different loads depending on the type of other object involved in the event. Continuing the sector change example, it is conceivable that routine air-ground communications might be handled by a data link for aircraft and sectors that were so equipped. In this case, a sector would have one set of loads for an event for data-link-equipped aircraft and another set for non-data-link-equipped aircraft.

- If mode is ORG (the simulation object is the event originator), the mate type applies to the destination object.
- If mode is DST (the simulation object is the event destination), the mate type applies to the event originator.

When FAM 2.0 is looking for the load of an activity, it will try to find the exact match for mode (e.g., ORG, DST), and for mate type (e.g., 747 or SECTOR_A). If there is no exact match, the model looks for the default (DEF) value. If there is no default, the model generates an error and quits.

OUTPUT FILES

FAM 2.0 generates an output file and an error file (if there are any warnings or errors) upon completion of the simulation run. The output file will be empty if an error has occurred. You specify the name of the output file at the command line when you run the model. The name of the error file is always *fam.err*. FAM 2.0 creates it in the directory from which the model is executed.

The output file will contain all the statistics of the simulation objects and communication channels. It also tracks the number of aircraft in each sector. Simulation objects with all statistics equal to zero indicate that the simulation objects were not involved in any activity during the simulation. Table 3-7 lists the field descriptions for simulation objects in the output file.

Table 3-7. Simulation Object Output File Field Descriptions

Field	Description
OBJ_ID	Name of the simulation object
START	Time when the simulation object is activated
END	Time when the simulation object is terminated
STAT_OBJ	Controllers/communication devices/equipment of the simulation object
MAX_WAITING	Maximum number of events waiting for a particular STAT_OBJ
AVE_QUE_LEN	Average queue length of a particular STAT_OBJ for events
MAX_WAIT_TIME	Maximum waiting time of a particular STAT_OBJ for events
AVE_WAIT_TIME	Average waiting time of a particular STAT_OBJ for events
NUM_SERVED	Number of the event that a particular STAT_OBJ serves

Table 3-8. Simulation Object Output File Field Descriptions (Continued)

Field	Description
TOTAL_TASK_TIME	Total tasking time of a particular STAT_OBJ
PERC_TASKED	Percentage of simulation time tasked of a particular STAT_OBJ
MAX_CONT_TIME	Longest period of continuous tasking of a particular STAT_OBJ
AVE_CONT_TIME	Average period of continuous tasking of a particular STAT_OBJ

Table 3-8 lists the output file field descriptions for communications channels.

Table 3-9. Communication Channel Output File Field Descriptions

Field	Description
CHANNEL	Channel value
MAX_WAITING	Maximum number of waiting for a particular channel
AVE_QUE_LEN	Average queue length of a particular channel
MAX_WAIT_TIME	Maximum waiting time of a particular channel
AVE_WAIT_TIME	Average waiting time of a particular channel
NUM_SERVED	Number of activities that a particular channel serves
TOTAL_TASK_TIME	Total tasking time of a particular channel
PERC_TASKED	Percentage of simulation time tasked of a particular channel
MAX_CONT_TIME	Longest period of continuous tasking of a particular channel
AVE_CONT_TIME	Average period of continuous tasking of a particular channel

Table 3-9 describes the output file fields for aircraft per sector data.

Table 3-10. Aircraft/Sector Output File Field Descriptions

Field	Description
SECTOR_ID	Identification number of sector that an aircraft is entering or leaving
NUM_AIRCRAFT	Number of aircraft going in and coming out of the sector
MAX_AIRCRAFT	Maximum number of aircraft that is entering the sector
AVE_AIRCRAFT	Average number of aircraft that is entering the sector

ERROR FILE (*FAM.ERR*)

FAM 2.0 only generates the error file, *fam.err*, if there is a warning or error during the model execution. There are two type of messages (warning and error) in *fam.err*. Warning messages mean the problems are minor. In this case, the model will not stop simulation. However, you should be aware that the collected statistics may not be accurate. Error messages mean the problems encountered are critical, and the simulation will stop. In this case, no statistics have been collected.

Examples of problems that will cause a *warning* message are

- load file contains extra data and
- an aircraft is deactivated before its activities are completed.

Examples of problems that will cause *error* messages are

- missing simulation object or a requested object no longer exists and
- a sector change occurs but you did not specify the losing or gaining sector.

Appendix B lists all warning and error messages generated by FAM 2.0. FAM 2.0 generates as many warnings and errors as possible. You should be aware, however, that fixing all the errors listed in the fam.err file does not necessarily mean that the next execution of the model will be successful. FAM 2.0 catches user errors for a particular input file. Therefore, more errors might be trapped during the next execution. You should continue checking for messages in fam.err until it is empty. Additionally, FAM 2.0 also catches run-time errors that could not be trapped during file processing before start of that simulation.

Appendix A

Dictionary, Type, and Load Files for Simulation Objects

This appendix details the file layouts of the dictionary, type, and load files for each category of simulation object.

Aircraft Type Files

FAM 2.0 brings each aircraft into the simulation at its activation time and deletes the aircraft at its deactivation time, which must be strictly greater than its activation time. Times for an aircraft's primary Events may be equal to or later than its activation time and must be earlier than its deactivation time. Equation A-1 shows this relationship.

$$T\left(activate\,aircraft_j\right) \le T\left(event_{ij}\right) < T\left(deactivate\,aircraft_j\right) \qquad \text{[Eq. A-1]}$$

where T_{ij} is the i^{th} event for aircraft j.

Figure A-1 is a sample aircraft type definition file (*aircraft.typ*). The first line, beginning with "//" is a comment line explaining the fields.

Figure A-1. Sample Aircraft Type File

```
//AIRCRAFT_TYPE       AIRCRAFT_LOAD FILENAME
     737                     737.ac
     747                     747.ac
     777                     777.ac
    turbo                   turbo.ac
```

The first field, aircraft_type, contains the names of different types of aircraft found in the simulation. The second field, aircraft_load_filename, contains the names of the load files corresponding to those aircraft types.

Figure A-2 is a sample aircraft load file (*747.ac*).

Figure A-2. Sample Aircraft Load File

```
NUM_PILOTS = 2
NUM_COMMDEVICES = 2
NUM_EQUIPMENT = 2

[LOAD]
//PRIM_ID[1] ASSOC_ID[2]     MODE[3] MATE_TYPE [4]C1[5] C2[5] CD1[5] CD2[5] EQ1[5] EQ2[5]
  SECT_CHG COMP_AC       DST   SECTOR_A   1.0 1.0 1.0  1.0  1.0  1.0
  SECT_CHG COMP_AC       DST   SECTOR_B   1.0 1.0 1.0  1.0  1.0  1.0
  SECT_CHG COMP_NEWSECT  ORG   SECTOR_A   2.0 2.0 2.0  2.0  2.0  2.0
  SECT_CHG COMP_NEWSECT  ORG   SECTOR_B   2.0 2.0 2.0  2.0  2.0  2.0
[LOAD_END]
```

Notes:

1. PRIM_ID = Primary Event ID of the activity and its load vector.
2. ASSOC_ID = Associated event ID of the activity and its load vector.
3. MODE = Mode (origin/destination/default) of the activity and its load vector.
4. MATE_TYPE = Mate type or the type of object (aircraft/sector/aoc/airport_controller/tracon_controller) of the activity and its load vector.
5. C = Controller Load; CD = Communication Device Load; EQ = Equipment Load.

ARTCC DICTIONARY FILES

The ARTCC dictionary is simply a listing of all sectors. Sectors are active throughout simulation. Each sector executes only one associated event at a time. If more than one event becomes scheduled for the same sector FAM 2.0 queues the events and executes them sequentially.

If an activity involves a sector, the sector ID must be provided in the apriori event file. If the sector ID is undefined or not specified, the simulation stops.

Figure A-3 is a sample sector dictionary file (*sector.dic*)

Figure A-3. Sample Sector Dictionary File

```
//                        Communication Devices
// SECTOR_ID  TYPE        1  2  3  4  5  6  7  8  9  10
   1          SECTOR_A    11 25 0  0  0  0  0  0  0  0
   2          SECTOR_B    12 25 0  0  0  0  0  0  0  0
   3          SECTOR_A    13 25 0  0  0  0  0  0  0  0
```

In the simulation using the sector dictionary file in Figure A-3, there are only three sectors, numbered 1, 2, and 3. Sectors 1 and 3 are one type (Sector_A) and sector 2 is another (Sector_B). Each sector has two communications devices (e.g., radio, data link, and telephone) available to it. The first device is used by that sector alone. These are channels 11 to 13. The sectors each have a second communication device, but they all share the same channel, 25. Note that these channels are model assets. It may be convenient to think of them in terms of radio frequencies. For example, sector 1 may communicate with aircraft on channel 11. In this case, the model accumulates usage of channel 11 not only by sector 1, but also by those aircraft (or any other model objects) that are also using channel 11.

Figure A-4 shows a sample sector type file (*sector.typ*)

Figure A-4. Sample Sector Type File

```
//   TYPE        LOAD_FILE_NAME
      SECTOR_A   sectorA.sct
      SECTOR_B   sectorB.sct
```

Continuing with the simulation in the previous example, although there were three different sectors, there were only two different sector types (A and B), so the sector type file has only two records, one for each type. The purpose of the file is to provide a pointer for the simulation object type to the load file containing that types task loadings and equipment usage.

Figure A-5 shows a sample sector load file for Sector_A (*sectorA.sct*)

Figure A-5. Sample Sector Load File

```
NUM_CONTROLLER = 1
NUM_COMMDEVICES = 2
NUM_EQUIPMENT = 1
[LOAD]

// PRIM_ID[1]  ASSOC_ID[2]     MODE[3]  MATE_TYPE[4]  C[5]  CD1[5]  CD2[5]  EQ[5]
   SECT_CHG    COMP_AC         ORG      747           9.0   9.0     9.0     9.0
   SECT_CHG    COMP_NEWSECT    DST      747           5.0   5.0     5.0     5.0
   SECT_CHG    REQ_ACC         ORG      SECTOR_A      1.0   1.0     1.0     1.0
   SECT_CHG    REQ_ACC         ORG      SECTOR_B      1.0   1.0     1.0     1.0
   SECT_CHG    REQ_ACC         DST      SECTOR_A      1.0   1.0     1.0     1.0
   SECT_CHG    REQ_ACC         DST      SECTOR_B      1.0   1.0     1.0     1.0
   SECT_CHG    ACC_ACK         ORG      SECTOR_A      2.0   2.0     2.0     2.0
   SECT_CHG    ACC_ACK         ORG      SECTOR_B      2.0   2.0     2.0     2.0
   SECT_CHG    ACC_ACK         DST      SECTOR_A      2.0   2.0     2.0     2.0
   SECT_CHG    ACC_ACK         DST      SECTOR_B      2.0   2.0     2.0     2.0
   SECT_CHG    INSTR_AC        ORG      777           3.0   3.0     3.0     3.0
   SECT_CHG    TUNE_RAD        ORG      777           4.0   4.0     4.0     4.0
   SECT_CHG    TUNE_RAD        DST      777           4.0   4.0     4.0     4.0
   SECT_CHG    INIT_CALL       DST      777           5.0   5.0     5.0     5.0
[LOAD_END]
```

Notes:

1. PRIM_ID = Primary Event ID of the activity and its load vector.
2. ASSOC_ID = Associated event ID of the activity and its load vector.
3. MODE = Mode (origin/destination/default) of the activity and its load vector.
4. MATE_TYPE = Mate type or the type of object. (ac/sector/aoc/airport_controller/tracon_controller) of the activity and its load vector.
5. C = Controller Load; CD = Communication Device Load; EQ = Equipment Load.

AIRLINE OPERATIONS CENTER (AOC) FILES

There is no limit to how many AOCs a user can define. If an AOC is defined, it remains active throughout simulation. If an activity involves an AOC, the AOC name must be provided in the apriori event file. If the AOC name is not specified or not defined, the simulation stops.

Figure A-6 shows a sample AOC dictionary file (*aoc.dic*)

Figure A-6. Sample AOC Dictionary File

```
//                 Communications Devices
// AOC_NAME  TYPE  1      2   3   4   5   6   7   8   9   10
UABWI        UA    101.1  0   0   0   0   0   0   0   0   0
UAIAD        UA    101.1  0   0   0   0   0   0   0   0   0
UADCA        UA    101.1  0   0   0   0   0   0   0   0   0
USBWI        US    102.1  0   0   0   0   0   0   0   0   0
USIAD        US    102.1  0   0   0   0   0   0   0   0   0
USDCA        US    102.1  0   0   0   0   0   0   0   0   0
```

In this simulation, there are six AOCs, three for each of two airlines. Note that an airline can have more than one type of AOC. The term AOC is generic and applies generally to those airline facilities that are involved with air traffic management for that airline. In this simulation, the AOCs use only one communications device. The United AOCs all use frequency 101.1, while the USAirways use frequency 102.1.

Figure A-7 shows a sample AOC type file (*aoc.typ*)

Figure A-7. Sample AOC Type File

```
// TYPE      LOAD_FILENAME
   UA        ua.aoc
   US        us.aoc
```

Figure A-8 shows a sample AOC load file (*ua.aoc*)

Figure A-8. Sample AOC Load File

```
NUM_DISPATCHER = 1
NUM_COMMDEVICES = 2
NUM_EQUIPMENT = 1
[LOAD]
//PRIM_ID[1]  ASSOC_ID[2]  MODE[3]  MATE_TYPE[4]  C[5]    CD1[5]  CD2[5]  EQ[5]
  ONE         ONE          ORG      DEF           1.0     1.0     1.0     1.0
  ONE         ONE          DST      DEF           1.0     1.0     1.0     1.0
  ONE         TWO          ORG      DEF           2.0     2.0     2.0     2.0
  ONE         TWO          DST      DEF           2.0     2.0     2.0     2.0
  ONE         THREE        ORG      DEF           3.0     3.0     3.0     3.0
  ONE         THREE        DST      DEF           4.0     4.0     4.0     4.0
[LOAD_END]
```

Notes:

PRIM_ID = Primary Event ID of the activity and its load vector.

ASSOC_ID = Associated event ID of the activity and its load vector.

MODE = Mode (origin/destination/default) of the activity and its load vector.

MATE_TYPE = Mate type or the type of object (ac/sector/aoc/airport_controller/tracon_controller) of the activity and its load vector.

C = Controller Load; CD = Communication Device Load; EQ = Equipment Load.

AIRPORT FILES

There are four different predefined controller types for each airport. You can use any number of those types to define a particular airport. An airport controller is a collection of personnel, radios, and equipment. The controller types are

- Tower
- Ground
- Clearance
- Other.

An airport controller must be defined before it can be used. Airport controllers remain in the simulation throughout its run. If an activity involves an airport controller (e.g., Tower/Ground/Clearance/Other) in the associated event file, the airport name must be provided in the apriori event file. If the airport name is not specified or not defined, simulation stops.

Figure A-9 shows a sample airport controller dictionary file (*airport.dic*)

Figure A-9. Sample Airport Controller Dictionary File

```
// AIRPORT   CONT_                  Communications Devices
// NAME[1]   NAME[2]   TYPE[3]      1     2  3  4  5  6  7  8  9  10
   Denver    Tower     Tower     101.0 0  0  0  0  0  0  0  0  0
   Denver    Ground    Ground    101.0 0  0  0  0  0  0  0  0  0
   Denver    Clearance Clearance 101.0 0  0  0  0  0  0  0  0  0
   Denver    Other     Other     101.0 0  0  0  0  0  0  0  0  0
   Colorado  Tower     All_Col   102.0 0  0  0  0  0  0  0  0  0
   Colorado  Ground    All_Col   102.0 0  0  0  0  0  0  0  0  0
   Colorado  Clearance All_Col   102.0 0  0  0  0  0  0  0  0  0
```

Notes:

1. AIRPORT_NAME = The unique identifier of the airport to be used by the user in the trigger event file.
2. CONT_NAME = Name of the Airport Controller.
3. TYPE = Type of the Airport Controller.

Figure A-10 shows a sample airport controller type file (*airport.typ*)

Figure A-10. Sample Airport Controller Type File

```
// TYPE         LOAD_FILE_NAME
   Tower        tower.apt
   Ground       ground.apt
   Clearance    clearance.apt
   Other        other.apt
   All_Col      all_colorado.apt
```

Figure A-11 shows a sample airport controller load file (*tower.apt*)

Figure A-11. Sample Airport Controller Load File

```
NUM_CONTROLLER = 1
NUM_COMMDEVICES = 2
NUM_EQUIPMENT = 1

// PRIM_ID[1] ASSOC_ID[2] MODE[3] MATE_TYPE[4]   C[5]   CD1[5] CD2[5] EQ[5]
[LOAD]
   TWO     ONE      ORG   DEF         1.0 1.0  1.0  1.0
   TWO     ONE      DST   DEF         1.0 1.0  1.0  1.0
   TWO     TWO      ORG   DEF         2.0 2.0  2.0  2.0
   TWO     TWO      DST   DEF         2.0 2.0  2.0  2.0
   TWO     THREE    ORG   DEF         3.0 3.0  3.0  3.0
   TWO     THREE    DST   DEF         4.0 4.0  4.0  4.0
[LOAD_END]
```

Notes:

1. PRIM_ID = Primary Event ID of the activity and its load vector.
2. ASSOC_ID = Associated event ID of the activity and its load vector.
3. MODE = Mode (origin/destination/default) of the activity and its load vector.
4. MATE_TYPE = Mate type or the type of object (ac/sector/aoc/airport_controller/tracon_controller) of the activity and its load vector.
5. C = Controller Load; CD = Communication Device Load; EQ = Equipment Load.

TERMINAL RADAR APPROACH CONTROL (TRACON) FILES

A TRACON is a collection of personnel (controllers), radios, and equipment. Each TRACON controller may serve one or more positions and one or more airports. The dictionary file for TRACONs has an additional section entitled [AIRPORT_POSITION] that identifies the airport and position served by each TRACON controller.

There are three predefined TRACON controller positions:

- Approach
- Departure
- Final.

Each TRACON controller must be defined and assigned an airport name and position before it can be used. Therefore, whenever a TRACON controller identifier is used in the associated event file, you must specify both the *airport name* and the *tracon name* in the apriori event file.

A TRACON controller may serve any of the following:

- *All three positions for the same airport*. For example, a single controller could serve approach, final, and departure controller positions at the Colorado Springs Municipal Airport.

- *Different airports and different positions.* For example, a single controller could serve approach for Centennial Airport and departure for Buckley Airport (Denver metropolitan area airports that are not included in the baseline simulation).

- *Different airport and the same position.* For example, a single controller could serve as departure controller for both Denver International and Centennial airports.

If the TRACON controller of the given TRACON name does not serve the given airport name and position, simulation terminates. Because of this situation, the TRACON controller dictionary file contains a unique block of text that defines the positions served by each controller.

Figure A-12 shows a sample TRACON controller dictionary file (*tracon.dic*).

Figure A-12. Sample TRACON Controller Dictionary File

```
// TRACON     CONT_               Communications Device[4]
// NAME[1]    NAME[2]  TYPE[3]    1    2  3  4  5  6  7  8  9  10
DENVER_AREA   ONE    DEN_TRACON   10.0 0  0  0  0  0  0  0  0  0
DENVER_AREA   TWO    DEN_TRACON   10.0 0  0  0  0  0  0  0  0  0
DENVER_AREA   THREE  DEN_TRACON   20.0 0  0  0  0  0  0  0  0  0
//Airport and Position assignment block[5]
//TRACON_NAME    CONT_NAME   AIRPORT    POSITION
[AIRPORT_POSITION]
DENVER_AREA      ONE         DENVER     APPROACH
DENVER_AREA      ONE         DENVER     DEPARTURE
DENVER_AREA      TWO         COLORADO   APPROACH
DENVER_AREA      TWO         COLORADO   DEPARTURE
DENVER_AREA      THREE       DENVER     FINAL
DENVER_AREA      THREE       COLORADO   FINAL
[AIRPORT_POSITION_END]
```

Notes:

1. TRACON_NAME = the name of the TRACON (to be used in the Trigger Event File to identify TRACON).
2. CONT_NAME = the identifier for the controller inside each TRACON. Each TRACON Controller is modeled at the level of an aircraft.
3. TYPE = the type of the TRACON Controller being used. The type defines a particular configuration.
4. Communications Devices 1 to 10 = the maximum 10 channels that may be assigned to each TRACON Controller.
5. Each TRACON Controller can be assigned multiple airports and positions. This block makes the airport-position assignment to each TRACON Controller.

Figure A-13 shows a sample TRACON controller type file (*tracon.typ*)

Figure A-13. Sample TRACON Controller Type File

```
// TYPE    LOAD_FILENAME
   DENVER  denver.tr
```

Figure A-14 shows a sample TRACON controller load file (denver.tr).

Figure A-14. Sample TRACON Controller Load File

```
NUM_CONTROLLER = 1
NUM_COMMDEVICES = 1
NUM_EQUIPMENT = 1

// PRIM_ID[1] ASSOC_ID[2] MODE[3] MATE_TYPE[4] C[5]  CD[5] EQ[5]
[LOAD]
   THREE     ONE       ORG    DEF         1.0   1.0   1.0
   THREE     ONE       DST    DEF         1.0   1.0   1.0
   THREE     TWO       ORG    DEF         2.0   2.0   2.0
   THREE     TWO       DST    DEF         2.0   2.0   2.0
   THREE     THREE     ORG    DEF         3.0   3.0   3.0
   THREE     THREE     DST    DEF         4.0   4.0   4.0
[LOAD_END]
```

Notes:

1. PRIM_ID = Primary Event ID of the activity and its load vector.
2. ASSOC_ID = Associated event ID of the activity and its load vector.
3. MODE = Mode (origin/destination/default) of the activity and its load vector.
4. MATE_TYPE = Mate type or the type of object (ac/sector/aoc/airport_controller/tracon_controller) of the activity and its load vector.
5. C = Controller Load; CD = Communication Device Load; EQ = Equipment Load.

Appendix B
FAM 2.0 Error & Warning Messages

“Error: number of pilots must be one or more.”

“Error: number of communication devices must be zero or more.”

“Error: number of equipment must be zero or more.”

“Error: Random event *‘random event name’* does not exist.”

“Error: There is no AOC *‘aoc name’* for primary event *‘primary event name’* and associated event *‘associated event name’* at time *‘t’* with delay *‘delay’* [MODE=ORG]”

“Error: There is no tower controller for airport *‘airport name’* (primary event *‘primary event name’* and associated event *‘associated event name’* at time *‘t’* with delay *‘delay’*) [MODE=ORG]”

“Error: There is no ground controller for airport *‘airport name’* (primary event *‘primary event name’* and associated event *‘associated event name’* at time *‘t’* with delay *‘delay’*) [MODE=ORG]”

“Error: There is no clearance controller for airport *‘airport name’* (primary event *‘primary event name’* and associated event *‘associated event name’* at time *‘t’* with delay *‘delay’*) [MODE=ORG]”

“Error: There is no other controller for airport *‘airport name’* (primary event *‘primary event name’* and associated event *‘associated event name’* at time *‘t’* with delay *‘delay’*) [MODE=ORG]”

“Error: There is no TRACON controller for approach with TRACON *‘TRACON name’* and airport *‘airport name’* (primary event *‘primary event name’* and associated event *‘associated event name’* at time *‘t’* with delay *‘delay’*) [MODE=ORG]”

“Error: There is no TRACON controller for departure with TRACON *‘TRACON name’* and airport *‘airport name’* (primary event *‘primary event name’* and associated event *‘associated event name’* at time *‘t’* with delay *‘delay’*) [MODE=ORG]”

“Error: There is no TRACON controller for final with TRACON *‘TRACON name’* and airport *‘airport name’* (primary event *‘primary event name’* and associated event ‘*associated event name*’ at time ‘*t*’ with delay ‘*delay*’) [MODE=ORG]”

“Error: There is no AOC ‘*aoc name*’ for primary event ‘*primary event name*’ and associated event ‘*associated event name*’ at time ‘*t*’ with delay ‘*delay*’ [MODE=DST]”

“Error: There is no tower controller for airport ‘*airport name*’ (primary event ‘*primary event name*’ and associated event ‘*associated event name*’ at time ‘*t*’ with delay ‘*delay*’) [MODE=DST]”

“Error: There is no ground controller for airport ‘*airport name*’ (primary event ‘*primary event name*’ and associated event ‘*associated event name*’ at time ‘*t*’ with delay ‘*delay*’) [MODE=DST]”

“Error: There is no clearance controller for airport ‘*airport name*’ (primary event ‘*primary event name*’ and associated event ‘*associated event name*’ at time ‘*t*’ with delay ‘*delay*’) [MODE=DST]”

“Error: There is no other controller for airport ‘*airport name*’ (primary event ‘*primary event name*’ and associated event ‘*associated event name*’ at time ‘*t*’ with delay ‘*delay*’) [MODE=DST]”

“Error: There is no TRACON controller for approach with TRACON ‘*TRACON name*’ and airport ‘*airport name*’ (primary event ‘*primary event name*’ and associated event ‘*associated event name*’ at time ‘*t*’ with delay ‘*delay*’) [MODE=DST]”

“Error: There is no TRACON controller for departure with TRACON ‘*TRACON name*’ and airport ‘*airport name*’ (primary event ‘*primary event name*’ and associated event ‘*associated event name*’ at time ‘*t*’ with delay ‘*delay*’) [MODE=DST]”

“Error: There is no TRACON controller for final with TRACON ‘*TRACON name*’ and airport ‘*airport name*’ (primary event ‘*primary event name*’ and associated event ‘*associated event name*’ at time ‘*t*’ with delay ‘*delay*’) [MODE=DST]”

“Error: number of controllers must be one or more.”

“Error: number of dispatchers must be one or more.”

“Error: File ‘*filename*’ does not exist.”

“Error: Error occurs while opening file ‘*filename*’.”

“Error: Expected ‘*token*’ (Error occurs in file ‘*filename*’ at line # and column #)”

“Error: Expected ‘=‘(Error occurs in file *‘filename’* at line # and column #)”

“Error: *‘token’* is not a valid real number; a real value is expected.”

“Error: Expected a real value.”

“Error: Expected a string.”

“Error: Mode *‘type name’* is invalid.”

“Error: Type *‘type name’* does not exist.”

“Error: Loads expected in file *‘filename’*.”

“Error: *‘token’* is not a valid integer number; an integer value is expected.”

“Error: Expected integer value.”

“Warning: There are pending activities at time *‘simulation time’*. “

“Error: index value *‘value of index’* is out of range.”

“Error: Time of a-priori event must be in increasing order.”

“Error: Time of activity for aircraft with airline *‘airline’* and flight number *‘flightnumber’* is less than its activation time.”

“Error: Time of activity for aircraft with airline *‘airline’* and flight number *‘flightnumber*” is greater than or equal to its deactivation time.”

“Error: Primary event *‘primary event name’* is not in the event dictionary.”

“Error: Sector type *‘type name’* is not in the sector dictionary.”

“Error: AOC type *‘type name’* is not in the AOC dictionary.”

“Error: Airport controller type *‘type name’* is not in the airport controller dictionary.”

“Error: TRACON controller type *‘type name’* is not in the TRACON controller dictionary.”

“Error: There is no TRACON controller for TRACON named *‘TRACON name’* and controller named *‘controller name’*.”

“Error: Airport position must be defined in the TRACON dictionary file.”

“Error: No random event was specified.”

“Error: stat_start must be: stat_start <= simulation_end”

“Error: Random mode must be TRUE or FALSE.”

“Error: Reuse seed mode must be TRUE or FALSE.”

“Error: *‘tag name’* is an invalid tag.”

“Error: tag *‘tage name’* is already defined.”

“Error: *‘token’* must be preceded by a proper tag.”

“Error: AOC *‘aoc name’* does not exist.”

“Error: Airport *‘airport name’* does not exist.”

“Error: TRACON *‘TRACON name’* does not exist.”

“Warning: Trigger event *‘trigger name’* and associated event *‘associated event name’* at time *‘simulation time’* cannot be processed because the simulationEnd time has occurred.”

“Error: There is no default load for primary event *‘primary event name’*, associated event *‘associated event name’*, mode *‘mode name’* and type *‘type name’*.”

“Error: Previous activity ended after current time.”

“Error: ACTIVATE_AC at time *‘time’* fails; aircraft type *‘type name’* is not in the aircraft dictionary.”

“Warning: You are trying to deactivate aircraft *‘aircraft name’* at time *‘simulation time’*, and there are pending activities. Adjust its deactivation time.”

“Error: DEACTIVATE_AC at time *‘time’* fails; aircraft *‘aircraft name’* does not exist.”

“Error: Primary event *‘primary event name’* at time *‘time’* fails; aircraft *‘aircraft name’* does not exist.”

“Error: Primary event at time *‘time’* fails; sector # *‘sector number’* does not exist.”

“Error: There is no associated event list for primary event *‘primary event name’*, aircraft 1 type *‘type name’*, aircraft 2 type *‘type name’*, sector 1 type *‘type name’*, sector 2 type *‘type name’* at time *‘time’*. “

"Error: There is no AOC *'aoc name'* for primary event *'primary event name'* and associated event *'associated event name*" at time *'time'* with delay *'delay'* [MODE=ORG]."

"Error: There is no tower controller for airport *'airport name'* (primary event *'primary event name'* and associated event *'associated event name'* at time *'time'* with delay *'delay'*) [MODE=ORG]"

"Error: There is no ground controller for airport *'airport name'* (primary event *'primary event name'* and associated event *'associated event name'* at time *'time'* with delay *'delay'*) [MODE=ORG]"

"Error: There is no clearance controller for airport *'airport name'* (primary event *'primary event name'* and associated event *'associated event name'* at time *'time'* with delay *'delay'*) [MODE=ORG]"

"Error: There is no other controller for airport *'airport name'* (primary event *'primary event name'* and associated event *'associated event name'* at time *'time'* with delay *'delay'*) [MODE=ORG]"

"Error: There is no TRACON controller for approach with TRACON *'TRACON name'* and airport *'airport name'* (primary event *'primary event name'* and associated event *'associated event name'* at time *'time'* with delay *'delay'*) [MODE=ORG]."

"Error: There is no TRACON controller for departure with TRACON *'TRACON name'* and airport *'airport name'* (primary event *'primary event name'* and associated event *'associated event name'* at time *'time'* with delay *'delay'*) [MODE=ORG]."

"Error: There is no TRACON controller for final with TRACON *'TRACON name'* and airport *'airport name'* (primary event "primary event name" and associated event *'associated event name'* at time *'time'* with delay *'delay'*) [MODE=ORG]."

"Error: There is no AOC *'aoc name'* for primary event *'primary event name'* and associated event *'associated event name*" at time *'time'* with delay *'delay'* [MODE=DST]."

"Error: There is no tower controller for airport *'airport name'* (primary event *'primary event name'* and associated event *'associated event name'* at time *'time'* with delay *'delay'*) [MODE=DST]"

"Error: There is no ground controller for airport *'airport name'* (primary event *'primary event name'* and associated event *'associated event name'* at time *'time'* with delay *'delay'*) [MODE=DST]"

“Error: There is no clearance controller for airport *‘airport name’* (primary event *‘primary event name’* and associated event *‘associated event name’* at time *‘time’* with delay *‘delay’*) [MODE=DST]”

“Error: There is no other controller for airport *‘airport name’* (primary event *‘primary event name’* and associated event *‘associated event name’* at time *‘time’* with delay *‘delay’*) [MODE=DST]”

“Error: There is no TRACON controller for approach with TRACON *‘TRACON name’* and airport *‘airport name’* (primary event “primary event name” and associated event *‘associated event name’* at time *‘time’* with delay *‘delay’*) [MODE=DST]”

“Error: There is no TRACON controller for departure with TRACON *‘TRACON name’* and airport *‘airport name’* (primary event “primary event name” and associated event *‘associated event name’* at time *‘time’* with delay *‘delay’*) [MODE=DST]”

“Error: There is no TRACON controller for final with TRACON *‘TRACON name’* and airport *‘airport name’* (primary event “primary event name” and associated event *‘associated event name’* at time *‘time’* with delay *‘delay’*) [MODE=DST]”

“Error: Expected tag *‘tag name’*.”

“Error: Sector *‘sector ID’* does not exist.”

“Error: Channel *‘channel value’* could not be found.”

“Error: Aircraft *‘aircraft name’* does not exist; primary event *‘primary event name’* and associated event *‘associated event name*” fails at time ‘*t*’; you must specify an aircraft when an associated event involves AC.”

“Error: Aircraft *‘aircraft name’* does not exist; primary event *‘primary event name’* and associated event *‘associated event name*” fails at time ‘*t*’; you must specify an aircraft when an associated event involves ALT_AC.”

“Error: There is no sector ID *‘sector ID’* for primary event *‘primary event name’* and associated event *‘associated event name’;* you must specify the sector ID when an associated event involves SECT.”

“Error: There is no sector ID *‘sector ID’* for primary event *‘primary event name’* and associated event *‘associated event name’;* you must specify the sector ID when an associated event involves L_SECT.”

"Error: There is no sector ID *'sector ID'* for primary event *'primary event name'* and associated event *'associated event name';* you must specify the sector ID when an associated event involves G_SECT."

"Error: scenario file *'filename'* does not exist."

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY ***(Leave blank)***

2. REPORT DATE
April 1998

3. REPORT TYPE AND DATES COVERED
Contractor Report

4. TITLE AND SUBTITLE
Aircraft/Air Traffic Management Functional Analysis Model, Version 2.0, User's Guide

5. FUNDING NUMBERS
C NAS2-14361
Task 97-03

WU 538-04-14-02

6. AUTHOR(S)
Melvin Etheridge, Joana Plugge, Nusrat Retina

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Logisitics Management Institute
2000 Corporation Ridge
McLean, Virginia 22102-7805

8. PERFORMING ORGANIZATION REPORT NUMBER
NS703S1

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
NASA/CR-1998-207658

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Robert E. Yackovetsky
Final Report

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Unclassified - Unlimited

Subject Category 01 Distribution: Nonstandard
Availability: CASI (301) 621-0390

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

The Aircraft/Air Traffic Management Functional Analysis Model, Version 2.0 (FAM 2.0), is a discrete event simulation model designed to support analysis of alternative concepts in air traffic management and control. FAM 2.0 was developed by the Logistics Management Institute (LMI) under task order NS703 of the National Aeronautics and Space Administration (NASA) contract number NAS2-14361. This document provides a guide for using the model in analysis. Those interested in making enhancements or modification to the model should consult the companion document, Aircraft/Air Traffic Management Functional Analysis Model, Version 2.0 Technical Description.

14. SUBJECT TERMS
airport
air traffice management
air transportation

15. NUMBER OF PAGES
44

16. PRICE CODE
A03

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	Unlimited