ETMS/ATMS Software Requirements

Name:	Bob Sharick
Date:	11/26/2003
Feature Described:	Early Intent Enhancements
ETMS Version:	1.3
Remarks:	
Reviewers/CSC:	Ken Howard, Ken Drum, Melia Stefanescu, Sylvia Todero
Reviewers/Volpe:	Rick Oiesen
Reviewers/Other:	
Reviewers/FAA:	

Revision History

Revision	Date	Authors Initials	Description of the Change
Version 1.0	April 2, 2003	RS	Initial version of System Requirements
Version 1.1	July 2, 2003	KH	Added fields to the transactions between EDCT, Parser, and FDB. Numerous edits for clarification.
Version 1.2	August 7, 2003	KH	Modified requirements 3.1 – 3.3 to incorporate the use of the NOACK keyword.
Version 1.3	November 26, 2003	RS	Added error returns for not finding a matching record and for ETD out of range. Removed numeric values of error codes since specific values cannot be guaranteed.

1. Overview

The first implementation of early intent messages did not include the sending of a reply back to the airlines to inform them of the success or failure of their messages. To include this functionality in version 7.7, several new messages will need to be passed from FDB to EDCT.

2. Functional Decomposition

ETMS currently does generate reply message to the sender for a normal CDM message. The message flow is as follows:

- Message sent via ARINC or CDMNET.
    - If ARINC, the arinc driver sends it through IFCN.FE to the EDCT driver using NWA.
    - If CDMNET, fd_fe sends it to the EDCT driver using NWA.
- EDCT parses the message packet. If errors are found, EDCT notes them and does not send the erroneous message to FDB.
- EDCT sends any good messages to FDB driver and waits for replies.
- FDB driver processes each message and sends a reply to EDCT. The reply indicates the success or failure of the message. If a failure, FDB sends an error code back to EDCT.
- When EDCT has a result for each message in a packet, EDCT formats a reply for the packet and sends it to the user. EDCT uses the FDB error codes to determine the text error message for the reply.
- EDCT sends the reply through IFCN.FE to the arinc driver, or to fd_fe.

The current EI processing follows a somewhat different path:

- Messages again come via ARINC or CDMNET.
- EDCT does not parse the EI messages, but sends them on to the ETMS parser function using NWA.
- If parser finds an error in an EI message, it discards it.
- If parser is successful, it sends a transaction to FDB driver.
- FDB driver tries to update the database.
- If FDB driver fails to update the database, it discards the transaction.

For consistency and simplicity, the EI replies will be implemented as closely as possible to the CDM message replies. This requires the followings enhancements to the EI processing flow.

- When EDCT sends EI messages off to parser, it will wait for replies from FDB.
- If parser finds an error, it will send a transaction to FDB indicating message and the error reason.
- FDB will always send a reply to EDCT for each EI message. This could be:
    - A “pass-through” error code from parser.
    - An error code from FDB.
    - A success message.

EDCT will need enough information in the reply to correlate the reply with the message that triggered the reply.

The requirements are numbered and shown in boldface. Unbolded text is included for clarification and is not, strictly speaking, part of the requirements.

Following are the specific requirements for each module.

EDCT Requirements

EDCT must wait for a reply from FDB on the FP message in an EI packet just as it waits for a reply on all FC/FM/FX messages in an FD packet. Currently, there is only one FP message per packet.

1. **EDCT shall add the packet ID and message index to each FP message prior to sending the message to parser.**

The packet ID and message index will be passed from EDCT to parser to FDB back to EDCT so that EDCT can be sure to correctly correlate the reply to the message. The message index indicates the number of the message within a packet, and is useful in the case of a packet with multiple messages. Although we are limiting EI packets to one FP, this field has been included to be consistent with the FD processing and to facilitate changes in the future, if needed.

2. **After sending an FP message to parser, EDCT shall wait for a response from the FDB as to the success or failure of each early intent message.**
3. **EDCT shall accept a message type FDB_EI_MSG_STATUS in response to an FP message.**
   - **3.1. If the return type is fd_msg_ok and the "NOACK" keyword was not present on the message, EDCT shall send a message back to the airline indicating the EI packet was successfully processed.**
   - **3.2. If the return type is ei_match_add, EDCT shall send a message back to the airline indicating the EI packet was used to add a new entry to the database. [NOTE: This warning is sent regardless of whether "NOACK" keyword was present on the message.]**
   - **3.3. If the return type indicates an error, EDCT shall send one of the following messages back to the airline. [NOTE: These errors are sent regardless of whether "NOACK" keyword was present on the message.]**
     - **3.3.1. ei_no_spd – CRUISING SPEED MISSING.**
     - **3.3.2. ei_no_actype – AIRCRAFT TYPE MISSING.**
     - **3.3.3. ei_no_orig– DEPARTURE AIRPORT MISSING.**
     - **3.3.4. ei_no_dest – ARRIVAL AIRPORT MISSING.**
     - **3.3.5. ei_no_ptime – DEPARTURE TIME MISSING.**
     - **3.3.6. ei_no_coord_fix – COORDINATION FIX MISSING.**
     - **3.3.7. ei_no_alt – ALTITUDE MISSING.**

3.3.8. ei_bad_rte – ROUTE SYNTAX ERROR.

3.3.9. ei_fz_rcvd – FLIGHT PLAN ALREADY PROCESSED.

3.3.10. ei_act – FLIGHT ACTIVE.

3.3.11. ei_comp – FLIGHT COMPLETED.

3.3.12. ei_ptime_chg – ETD OUT OF RANGE.

3.3.13. ei_database_error – ETMS DATABASE ERROR.

3.3.14. ei_no_match – FP MESSAGE CANNOT MATCH ANY FLIGHT

3.4. If EDCT does not get a reply back within 30 seconds, it shall send back the following message to the user "ETMS INTERNAL ERROR"

Parser Requirements

4. If the Parser finds an error in its message processing, it shall send a message to FDB of type EI_PARSER_ERROR with an indicator of the specific error. The error codes used by parser are:

4.1. No cruising speed – ei_no_spd.

4.2. No aircraft type – ei_no_actype.

4.3. No origin airport – ei_no_orig

4.4. No destination airport – ei_no_destination

4.5. No departure time – ei_no_ptime.

4.6. No coordination fix – ei_no_coord

4.7. No altitude – ei_no_alt.

4.8. Route could not be parsed – ei_bad_rte.

FDB Requirements

FDB sets a global variable, airline_message_status, with the return code that it will send to EDCT. This contains either fd_msg_ok, which is 0, or an error value. The values are defined in edct.fdb_status.h. FDB negates the value before sending the reply with the function send_edct_return. The message type is always FD_FDB_MSG_STATUS.

5. If FDB successfully processes the early intent message, it shall send back a message to EDCT of type FD_FDB_MSG_STATUS with a return type of fd_msg_ok.

6. If FDB receives a message of type EI_PARSER_ERROR from the PARSER, it shall send a message of type FDB_EI_MSG_STATUS to EDCT to indicate that the early intent flight plan failed to parse. FDB shall include the specific error code returned from parser on the message to EDCT.

7. If FDB itself finds an error with an early intent packet, it must send back a message to EDCT of type FD_FDB_MSG_STATUS. The possible errors and their return types are:

7.1. FZ already received for flight – ei_fz_rcvd. This covers any time the flight has already received an FZ with a status of controlled, cancelled or filed.

7.2. Flight is active – ei_act.

7.3. Flight is complete - ei_comp.

7.4. The flight-id in the message failed to match any record – ei_no_match. A flight cannot be created from an FP message.

7.5. Database cannot be updated for any other reason – ei_database_error.

3. External Systems

8. External clients shall be able to accept replies for every early intent message.

4. Databases

None.

5. Data Interfaces

9. The FP message format passed from EDCT to parser shall be modified to include the packet ID and message index at the end of the message. The format shall be:

<FP as received> # <packet ID as received> <message index>

where:

- **the packet ID is the 16 character string provided by the sender to be the unique packet identifier (example: COA0408123745.01)**
- **the message index is an integer assigned by EDCT to indicate the position of the message within a packet (example: 1).**

Example:

0408123745XFP COA1282 B737 SEA P1345 370
SEA.J90.MWH..BIL..RWF..CRL.J584.SLT.FQM1.EWR/1752 # COA0408123745.01 1

10. The parser/FDB transaction shall be modified to include the packet ID, message index, and status.

6. User Interfaces

None.

7. Implementation Issues/Suggestions

None.