Interface Control Document

Name: R. Sharick / M. Lehky

Date: 10/11/04

Feature Described: CDM Message Protocol

Document Version: 2.1

Remarks: Effective ETMS 8.2

Revision History			
Version	**Date**	**Authors Initials**	**Description of Change**
2.1	10/11/05	RS / ML	• Initial Version of Changes for ETMS 8.2 • Converted Document To ICD • Incorporated Previously Undocumented Messages

In conjunction with the release of version 8.2 of the Enhanced Traffic Management System (ETMS), the Collaborative Decision Making (CDM) Message Protocol is being revised. These changes are primarily driven by the introduction of the Airspace Flow Program (AFP) and Flight Schedule Monitor (FSM) Auto-Monitor functions. This document describes the new format which will be effective upon the operational deployment of ETMS 8.2 in the spring of 2006. At that time, this document will supersede:

- *CDM Message Protocol Version 2.0* dated 27 May 2004

The body of the document has been fully updated to reflect the new protocols. The following is a summary list of the specific protocols that have changed:

- Added three new messages to support the FSM Broadcast functionality.
  - [242] M_AUTO_MONITOR_ REQ
  - [243] M_AUTO_MONITOR_REPLY
  - [244] M_AUTO_MONITOR_MESSAGE
- Added two new messages to support AFP functionality.
  - [245] M_ ADD_ADL_AFP_PARAM
  - [256] M_ DEL_ADL_AFP_PARAM
- Modified one message to support AFP functionality.
  - [201] M_REGISTER
- Deleted nine messages never implemented.
  - [6] M_STATS
  - [7] M_STATS_REPLY
  - [208] M_ADL_ERROR

    - [209] M_NEW_ADL
    - [210] M_ADD_ADL_DATA
    - [211] M_DEL_ADL_DATA
    - [216] M_FADT_REPORT
    - [223] M_ADD_ADL_ADVISORY and/or M_ADD_ADL_GS_REPORT
    - [231] M_DEL_ADL_ADVISORY and/or M_DEL_ADL_GS_REPORT
- Added 21 messages which had previously been undocumented.
    - [10] is standardized as M_HB_REQ
    - [11] is standardized as M_HB_ACK
    - [214] M_REQ_COMMAND
    - [215] M_REQ_REPLY
    - [218] M_ADL_DATA_ACK
    - [219] M_ ADD_ADL_AAR
    - [220] M_ ADD_ADL_ADR
    - [221] M_ ADD_ADL_AAR_GDP
    - [222] M_ ADD_ADL_GDP_PARAM
    - [224] M_ ADD_ADL_COMP_PARAM
    - [225] M_ ADD_ADL_BLANK_PARAM
    - [226] M_ ADD_ADL_GS_PARAM
    - [227] M_ DEL_ADL_AAR
    - [228] M_ DEL_ADL_ADR
    - [229] M_ DEL_ADL_AAR_GDP
    - [230] M_ DEL_ADL_GDP_PARAM
    - [232] M_ DEL_ADL_COMP_PARAM
    - [233] M_ DEL_ADL_BLANK_PARAM
    - [234] M_ DEL_ADL_GS_PARAM
    - [236] M_ WEATHER_COMMAND
    - [237] M_ WEATHER_REPLY
- Clarified numerous sections throughout the document.

1 Introduction

This document describes the protocol for using TCP/IP communications for CDM data exchange between various applications and ETMS hubsite applications. The combined set of networks that provides users with TCP/IP connectivity for CDM are referred to as CDMNET. There are currently three types of application data exchange that occurs over CDMNET:

1. The distribution of Aggregate Demand List (ADL) Files, FSM Broadcast Files, and AFP/Ground Delay Program (GDP) Parameters from the ETMS hub site to users, primarily for viewing via FSM.
2. The feed of flight data messages (FC, FM, FX) from NAS Users' flight data systems to the ETMS hub site.
3. The exchange of AFP/GDP data (slot lists and substitution messages) that is performed under the heading of "Simplified Subs".

This document describes the protocol for each item above.

1.1 Overview

Using the CDMNET, inter-process communications between processes at the various sites (NAS Users, FAA, and Volpe) will be performed in "sessions" through dedicated TCP/IP sockets. In each session, the application running at the user site is considered the client and the application running at the hub is considered the server. The general approach is as follows:

1. A client process opens a socket connection to a server process using a well-known IP address and starts a session.
2. Data is exchanged between the client and server indefinitely.
3. Either the client or server terminates the session and closes the connection.

The session could range from a single message sent and reply returned or months of continuous data exchange.

For flight data messages, the session can be opened and closed simply by opening and closing the socket connection. In other words, the client just opens a socket and begins sending messages. Each message contains client information that is used by the server to validate the connection. Additionally, an optional "connect" message is provided that can be sent by the client at the start of a session to validate the connection prior to submitting any data messages. The use of this message is NOT required.

For ADL distribution, additional messages are used to manage sessions. The client sends an initial connect message identifying itself; the server uses this information to validate the connection. Additional messages are used to notify client/server of a shutdown.

For Simplified Subs, the session will operate similarly to flight data messages with one exception. With simplified subs, the first application message may be from the server to the client. That means that if a client is connecting to monitor the simplified sub messages sent by ETMS, it must send a message after connecting to identify itself. The same connect message used for ADL sessions will be used for this purpose.

Message formats consist of a 24-byte header (binary) optionally followed by a variable number of bytes containing application specific data. Some of the fields in the header are now obsolete, but have been preserved for backwards compatibility.

1.2 Scope

This document covers the protocol for managing socket connections and transferring data. It does not provide descriptions of the application-level data. For example, the flight data feed portion describes how to open a connection, and how to fill the message headers. However, it does not describe how to format an FC, FM, or FZ message and under what circumstances to send them. The application-level data can be found in the following documents:

ADL Feed

- *ADL Format Version 10.*
- *FSM Parameters Formats Version 1.0 (under development).*

Flight Data Feed

- *CDM Message Formats Version 2.1*
- *Early Intent Protocol Version 1.2*

Simplified Subs

- *Simplified Substitution Message Processing Version 1.0*

1.3 Organization

This document is organized into six main sections:

- **1 Introduction** - This section.
- **2 ADL Protocol -** Describes the protocol used to exchange ADL files, FSM Broadcast Files, and Program Parameters with the ETMS Hubsite ADL_FE process. These protocols are generally utilized by the FSM System to interact with ETMS.
- **3 Flight Data Protocol** - Describes the protocol used to exchange CDM Flight Data and CDM Early Intent messages with the ETMS Hubsite FD_FE process. These protocols are generally utilized by CDM data sender processes developed by CDM Participants to interact with ETMS.
- **4 Simplified Substitutions Protocol -** Describes the protocol used to exchange Simplified Substitution messages (including Slot Credit Substitutions) with the FD_FE. These protocols are generally utilized by substations tools developed by CDM Participants to interact with ETMS. Additionally FSM utilized these protocols for some portions of ECR functionality.
- **5 Detailed Message Specifications** - Defines the detailed formats of the messages presented in the previous sessions.
- **6 Message Applicability Summary -** Provides a summary table of the applicability of various messages.

2 ADL Protocol

This section describes the protocol used between clients (typically the FSM Servers) and server (ADL_FE). The ADL_FE connection is utilized to obtain ADL data, to transmit GDP parameters, and to provide access to various ETMS Hubsite commands.

2.1 ADL Session Protocol

2.1.1 Overview

An ADL client/server session will be established and maintained through the following sequence of events:

- A client opens a socket connection to a server using a well-known IP address.
- The client sends a connect message.
- The server validates the connect message, and if valid, sends an accept message to the client. If not valid, the server sends a reject message.
- Various messages are sent between client and server to exchange data.
- When the client wants to verify that the connection is still open, it sends a "keep-alive" message. The server responds with a "keep-alive" acknowledgement.
- When a client wants to terminate a session, it sends a disconnect message to the server and disconnects from the socket.
- When a server wants to terminate a session, it sends a shutdown message to the client(s) and closes the socket(s).

The messages used to initiate, validate, and terminate sessions will be referred to as "session messages".

2.1.2 Security

At the transport level, security will be provided by firewalls. It is the responsibility of each connecting organization to establish whatever firewall they feel will ensure their security. All data exchange between clients and servers will be through a socket connection; no FTP or telnet access is required.

At the application level, Volpe will maintain a table of valid IP client addresses, client names, and client tags. The server at Volpe will validate the client data whenever a client connects, and will reject any connection that is not authorized. The client data is validated when a connect message is sent, and whenever included on a data message (e.g., a flight data message). At any time if the client data is invalid, the server will immediately terminate the session and close the socket. As long as the client data is valid, messages will be allowed to flow freely between the client and server and will be processed as long as the messages are of recognized type and format.

2.1.3 Session Messages

[1] M_ATMS_CONNECT - [client to server]

This is the first message sent to start a session. It identifies the source of the message (e.g., Metron, AAL).

[2] M_ATMS_ACCEPT - [server to client]

Notifies the client that the connection has been accepted. The session is now started.

[3] M_ATMS_REJECT - [server to client]

Notifies the client that the connection has been rejected. Includes an error code.

[4] M_DISCONNECT - [client to server]

Tells the server that the client process is shutting down and going away (i.e., ends the session).

[5] M_SHUTDOWN - [server to client]

Tells the client that server process is shutting down, and gives the client the opportunity to shut down gracefully (i.e., ends the session).

[10] M_HB_REQ - [client to server]

Requests a heartbeat message to confirm that the connection between the client and server is still active.

[11] M_HQ_ACK - [server to client]

Reply to a heartbeat request; this confirms that the connection is still alive.

2.1.4 Error Handling

Loss of Client

A server will consider any of the following events to be a “loss of client”:

- notification that the socket connection to the client has been lost
- notification that a message to the client is undeliverable
- a excessive backup in the sending queue (see below)
- receipt of a disconnect message

When a server detects a loss of client it shall close the connection and remove the client from the registration tables.

Loss of Server

A client will consider any of the following to be a “loss of server”:

- notification that the socket connection to the server has been lost
- notification that a message to the server is undeliverable
- receipt of a shutdown message

When a client detects a loss of server it shall close the connection, notify the user (as appropriate), and try to re-establish the connection.

In either case when a connection is closed the session will be considered to be ended.

Queuing

The server will queue ADL files intended for a client in the event that the socket is not being read as fast as it is being written. A maximum queue of 50 ADL files will be allowed. Once the maximum is exceeded, the server will consider the client to be inaccessible and will terminate the session.

Redundancy

Multiple servers will be provided for obtaining ADL files. A client should have the capability to attempt connections to multiple IP addresses in a round-robin manner. Only one IP address will be active at any given time.

2.2 ADL Application Protocol

2.2.1 Overview

A client running at a user site will get data from the server in the following manner:

- A client first establishes a session using the full protocol described in section 2.
- When a client wants data for an element it sends a register message to the server.
- The server registers that client for that element and starts periodically shipping data files to the client.
- When a client no longer wants data for that element, it sends an un-register message.
- The server stops sending files to that client for that element.
- When a client shuts down it ends the session by sending a disconnect message to server prior to closing the socket connection.
- In the event that server shuts down, it closes its sessions by sending shutdown messages to all clients prior to closing the socket connections.

2.2.2 File Transfer

ADL data files will be transmitted to CDMNET sites in the following manner:

- The server will encrypt the file using Blowfish.
- The server will compress the file using GZIP.
- The server will send the file to the client through the socket until the end of file is reached.
- The client will write the data to a file.
- When the data transmission is complete, the server will notify the client through the socket that the file is ready.
- The client will decrypt, decompress and read the file.

2.2.3 ADL Messages

In addition to the general session messages described in section 2, the ADL protocol will use the following messages:

[201] M_REGISTER - [client to server]

Requests data for an element. Includes location to place the files.

[202] M_REGISTER_ACK - [server to client]

Indicates whether the registration was accepted. Includes an error code if rejected.

[203] M_UN_REGISTER - [client to server]

Tells server to stop sending data for the named element.

[204] M_UN_REGISTER_ACK - [server to client]

Indicates whether the un-registration was accepted. Includes an error code if rejected.

[205] M_START_FILE - [server to client]

Notifies client that a new ADL file is being downloaded through the socket and provides the filename of the ADL. This message can be thought of as a file open. Always contains the sequence number 1.

[206] M_ADL_DATA - [server to client]

Provides a packet of data from the ADL file. This message can be thought of as a file write. Includes sequence numbers 2-N.

[207] M_END_FILE - [server to client]

Notifies client that the download of an ADL file is complete. This message can be thought of as a file close. Includes the sequence number N+1.

[212] M_EDCT_COMMAND - [client to server]

Sends an EDCT command to the server. The message body contains the full text of the command, which is forwarded to EDCT.

[213] M_EDCT_REPLY - [server to client]

Sends the reply generated by EDCT in response to an EDCT command. The message body contains the full text of the reply.

[214] M_REQ_COMMAND - [client to server]

Sends a command line EDCT request to the server. The message body contains the full text of the request, which is forwarded to EDCT.

[215] M_REQ_REPLY - [server to client]

Sends the reply generated by EDCT in response to an EDCT request. The message body contains the full test of the reply.

[218] M_ADL_DATA_ACK - [server to client]

Sends an acknowledgement of the receipt of any type of GDP data.

[219] M_ADD_ADL_AAR - [client to server]

Sends an update of the AAR for an element.

[220] M_ADD_ADL_ADR - [client to server]

Sends an update of the ADR for an airport

[221] M_ADD_ADL_AAR_GDP - [client to server]

Sends the AAR associated with a GDP. This message must precede the GDP parameters.

[222] M_ADD_ ADL_GDP_PARAM - [client to server]

Sends the GDP parameters for an element.

[224] M_ADD_ ADL_COMP_PARAM - [client to server]

Sends the parameters for a compression.

[225] M_ADD_ ADL_BLANK_PARAM - [client to server]

Sends the parameters for a blanket program.

[226] M_ADD_ADL_GS_PARAM - [client to server]

Sends the parameters for a ground stop.

[227] M_DEL_ADL_AAR - [client to server]

Sends a message to delete the AAR for an element.

[228] M_DEL_ADL_ADR - [client to server]

Sends a message to delete the ADR for an airport.

[229] M_DEL_ADL_AAR_GDP - [client to server]

Sends a message to delete the AAR and GDP parameters for an element.

[230] M_DEL_ADL_GDP_PARAM - [client to server]

Sends a message to delete the parameters for a GDP.

[232] M_DEL_ADL_COMP_PARAM - [client to server]

Sends a message to delete the parameters for a compression.

[233] M_DEL_ADL_BLANK_PARAM - [client to server]

Sends a message to delete the parameters for a blanket program.

[234] M_DEL_ADL_GS_PARAM - [client to server]

Sends a message to delete the parameters for a ground stop.

[235] M_UPDATE_ADL_REQ - [client to server]

Requests an updated ADL for an element.

[236] M_WEATHER_COMMAND - [client to server]

Requests METAR and TAF weather data for an airport.

[237] M_WEATHER_REPLY - [server to client]

Sends the reply to a weather command.

[242] M_AUTO_MONITOR_REQ - [client to server]

Requests the current auto-monitor data.

[243] M_AUTO_MONITOR_REPLY - [server to client]

Sends the reply to an auto-monitor request.

[244] M_AUTO_MONITOR_MESSAGE - [server to client]

Sends the current auto-monitor data to any attached client.

[245] M_ADD_ ADL_AFP_PARAM - [client to server]

Sends the AFP parameters for an element.

[246] M_DEL_ADL_AFP_PARAM - [client to server]

Sends a message to delete the parameters for an AFP.

3. Flight Data Protocol

This section describes the protocol used to exchange flight data messages (FC, FM, FX) between client and server.

3.1 Flight Data Session Protocol

3.1.1 Overview

A client/server session may be established and maintained in two different ways depending on the requirements of the client. The simplest session protocol, which was the only option available prior to ETMS 7.9, is conducted through the following sequence of events:

- A client opens a TCP/IP socket connection to the server using a well-known address.
- The server validates the IP address of the connecting client. If invalid, the server closes the connection. If valid, the session for that client has started.
- Flight data messages are sent from client to server. Each message includes client name and tag, as well as the body of the message text.
- The server validates the client name and tag when first flight data message is received. If there is any problem, the server closes the connection. Otherwise this and all subsequent messages are accepted.
- Optionally, replies are sent from server to the client IP or ARINC address.
- When a client wants to terminate a session, it closes the socket connection.
- When a server wants to terminate a session, it closes the socket connection.

A client can, if desired, also use the following additional features. [Note: These are the same messages used for the ADL protocol.]

- The client sends a connect message.
- The server validates the connect message, and if valid, sends an accept message to the client. If not valid, the server sends a reject message.
- When the client wants to verify that the connection is still open, it sends a “keep-alive” message. The server responds with a “keep-alive” acknowledgement.
- When the server terminates, it will send a “shutdown” message before closing the socket.

Additional notes about flight data sessions:

- An NAS User may open and close sessions whenever it wishes; that is, there is not a requirement to keep an open connection with the server at all times, although that is allowed.
- An airline may have multiple open connections if so desired. However, each simultaneous connection must have a unique client tag. Multiple client tags must be assigned for this purpose.
- A client tag may only be associated with one active connection at a time.

3.1.2 Security

At the transport level, security will be provided by firewalls. It is the responsibility of each user to establish whatever firewall they feel will ensure their security. All data exchange between clients and servers will be through a socket connection; no FTP or telnet access is required.

Security will also be provided at the application level. Volpe will maintain a table of valid IP client addresses. The server at Volpe will validate the connecting IP address whenever a client connects, and will reject any connection that is not authorized. Once a session is established, messages will be checked for a valid client tag. If any message is received with an invalid client tag for the sending IP address, the connection will be terminated.

Additionally, the server will maintain a table of which airline codes each client is allowed to send data for, and reject any messages that refer to unauthorized flights. (NOTE: This is the current method of authorizing data on the ARINC network.)

To support the security checking, each airline is required to provide the following parameters:

- IP address where their flight data messages will originate from.
- Three-letter code of sender (e.g. AAL)
- Additional three-letter codes of flights which the sender is authorized to modify (e.g., EGF)

Volpe will provide a client tag for each authorized user connection.

3.1.3 Session Messages

[1] M_ATMS_CONNECT - [client to server]

This is the first message sent to start a session. It identifies the source of the message (e.g., Metron, AAL).

[2] M_ATMS_ACCEPT - [server to client]

Notifies the client that the connection has been accepted. The session is now started.

[3] M_ATMS_REJECT - [server to client]

Notifies the client that the connection has been rejected. Includes an error code.

[5] M_SHUTDOWN - [server to client]

Tells the client that server process is shutting down, and gives the client the opportunity to shut down gracefully (i.e., ends the session).

[10] M_HB_REQ - [client to server]

Requests a reply to confirm that the connection between the client and server is still active.

[11] M_HB_ACK - [server to client]

Reply to a HB_REQ request; this confirms that the connection is still alive.

3.1.4 Error Handling

Loss of Client

A server will consider any of the following events to be a “loss of client”:

- notification that the socket connection to the client has been lost

- notification that a message to the client is undeliverable

When a server detects a loss of client it shall close the connection and remove the client from the registration tables.

Loss of Server

A client will consider any of the following to be a “loss of server”:

- notification that the socket connection to the server has been lost
- notification that a message to the server is undeliverable
- receipt of a shutdown message

When a client detects a loss of server it shall close the connection, notify the user (as appropriate), and try to re-establish the connection.

In either case when a connection is closed the session has ended.

Message Loss

Sequence numbers will be used to track that all flight data messages and replies are being received properly. The client will assign a sequence to each message when it is sent. The server will return the sequence number on the reply for that message (optional). The server will track the sequence numbers and detect any out of sequence occurrences. Sequence errors will be logged and examined periodically to assess the performance of the communications. No recovery processing for individual messages will be implemented at this time.

Redundancy

Volpe will provide multiple servers for flight data. The client should be developed so that it will attempt to connect to different IP addresses. In the case that one connect attempt fails, the client should go on to try the next address. Only one connection should be active from any given client at a time.

3.2 Flight Data Application Protocol

3.2.1 Overview

Once the session is established, the client sends flight data messages and the server optionally sends replies. The airline has control over how their messages are acknowledged. The airline has two choices:

- Whether a reply is sent for every message, or only when errors occur in processing the message.
- Whether the replies go back to the sending address or to an alternate ARINC address.

These choices are controlled by the airlines through the use of keywords in the packet header record, as in the current ARINC-only protocol.

The hub site will generate replies in the following manner by default.

- A reply will be sent for every message.
- The reply will be sent to the address from which the message was received (i.e., either an IP CDMNET address or an ARINC address).

If an airline does NOT wish to get an unconditional reply for a message, it will indicate so by using the NOACK keyword in the packet header line. In this case replies will only be sent when an error is encountered in processing the message.

If an airline does NOT want to get replies at its sending address, it will indicate an alternate address in the message packet header. NOTE: The alternate address can only be an ARINC address!

The use of the NOACK and ARINC address keywords is already accounted for in the CDM message format. This protocol simply preserves what is already being used.

NOTE: There is no way to specify an alternate IP address for replies.

3.2.2 Messages

The following two message types will be used to exchange flight data messages.

[101] M_FLIGHT_DATA_PACKET - [client to server]

Sends flight data from airline to ETMS hub.

[102] M_FLIGHT_DATA_REPLY - [server to client, optional]

Sends an airline the result of the processing of a flight data packet.

[109] M_FPPP_FLIGHT_DATA_PACKET - [client to server]

Sends early intent data from airline to FPPP server which then forwards the message to ETMS.

[110] M_FPPP_FLIGHT_DATA_RESPONSE_PACKET - [server to client]

Sends an airline the result of the processing of an early intent data packet.

[111] M_EI_FLIGHT_DATA_PACKET - [client to server]

Sends early intent data from an airline directly to ETMS. (NOTE: early intent data may also be sent using message type M_FLIGHT_DATA_PACKET. Also, the response will be sent back to the airline under message type M_FLIGHT_DATA_REPLY.)

The flight data messages are sent as a buffer attached to the M_FLIGHT_DATA_PACKET header. The format of the messages themselves is described separately in the *CDM Message Formats* document

4. Simplified Subs Protocol

This section describes the protocol used to exchange simplified subs messages (slot lists and simplified sub packets) between client and server.

4.1 Simplified Subs Session Protocol

4.1.1 Overview

A client/server session may be established and maintained in two different ways depending on the requirements of the client. The simplest session protocol, which was the only option available prior to ETMS 7.9, is conducted through the following sequence of events:

- A client opens a TCP/IP socket connection to the server using a well-known address.
- The server validates the IP address of the connecting client. If invalid, the server closes the connection. If valid, the session for that client has started.
- The client sends a connect message which identifies the client by providing a client name and tag. The client name must be configured as a simplified sub (SS) client by ETMS.
- When a GDP is issued, the server sends the slot list out to any SS clients currently connected.
- Once a GDP is issued, SS packets may be sent from client to server. Each message includes the body of the message text in the buffer.
- Replies are sent from server to the client over the same socket that the message was received on.
- At any time, the SS client can request information about the current GDPs. Replies are sent back over the same socket.
- When the state of a GDP changes (substitutions turned off, substitutions turned on, GDP cancelled), a message is sent out to any currently connected SS clients.
- When an SS client wants to terminate a session, it closes the socket connection.
- When the server wants to terminate a session, it closes the socket connection.

A client can, if desired, also use the following additional features. [Note: These are the same messages used for the ADL protocol.]

- The client sends a connect message.
- The server validates the connect message, and if valid, sends an accept message to the client. If not valid, the server sends a reject message.
- When the client wants to verify that the connection is still open, it sends a “keep-alive” message. The server responds with a “keep-alive” acknowledgement.
- When the server terminates, it will send a “shutdown” message before closing the socket.

Additional notes about simplified subs sessions:

- The simplified subs will use the same port as the flight data message feed. Therefore, an SS client can send regular flight data messages as part of the same session as simplified subs, if desired.
- An airline may have multiple SS clients connected if so desired. These may share the same name, but each concurrent client must have a unique client tag.

4.1.2 Security

At the transport level, security will be provided by firewalls. It is the responsibility of each user to establish whatever firewall they feel will ensure their security. All data exchanges between clients and servers will be through a socket connection; no FTP or telnet access is required.

Security will also be provided at the application level. Volpe will maintain a table of valid IP client addresses. The server at Volpe will validate the connecting IP address whenever a client connects, and will reject any connection that is not authorized. Once a session is established, messages will be checked for a valid client tag. If any message is received with an invalid client tag for the sending IP address, the connection will be terminated.

Additionally, the server will maintain a table of which airline codes each client is allowed to send data for, and reject any messages that refer to unauthorized flights. (NOTE: This is the current method of authorizing data on the ARINC network.)

To support the security checking, each airline is required to provide the following parameters (this data is required for configuration purposes and is not part of the real-time session data):

- IP address from which their flight data messages will originate.
- Three-letter code of sender (e.g. AAL)
- Additional three-letter codes of flights which the sender is authorized to modify (e.g., EGF)

Volpe will provide a client tag for each authorized user connection.

4.1.3 Session Messages

[1] M_ATMS_CONNECT - [client to server]

This is the first message sent to start a session. It identifies the source of the message (e.g., Metron, AAL).

[2] M_ATMS_ACCEPT - [server to client]

Notifies the client that the connection has been accepted. The session is now started.

[3] M_ATMS_REJECT - [server to client]

Notifies the client that the connection has been rejected. Includes an error code.

[5] M_SHUTDOWN - [server to client]

Tells the client that server process is shutting down, and gives the client the opportunity to shut down gracefully (i.e., ends the session).

[10] M_HB_REQ - [client to server]

Requests a reply to confirm that the connection between the client and server is still active.

[11] M_HB_ACK - [server to client]

Reply to a HB_REQ request; this confirms that the connection is still alive.

4.1.4 Error Handling

Loss of Client

A server will consider any of the following events to be a “loss of client”:

- § notification that the socket connection to the client has been lost
- § notification that a message to the client is undeliverable

When a server detects a loss of client it shall close the connection and remove the client from the registration tables.

Loss of Server

A client will consider any of the following to be a “loss of server”:

- § notification that the socket connection to the server has been lost
- § notification that a message to the server is undeliverable
- § receipt of a shutdown message

When a client detects a loss of server it shall close the connection, notify the user (as appropriate), and try to re-establish the connection.

In either case when a connection is closed the session has ended.

Message Loss

Sequence numbers should be used to track that all data messages and replies are being received properly. The client should assign a sequence number to each message when it is sent. The server will return the sequence number on the reply for that message (replies may be optional). The server will track the sequence numbers and detect any out of sequence occurrences. Sequence errors will be logged and examined periodically to assess the performance of the communications. No recovery processing for individual messages will be implemented at this time.

Redundancy

Volpe will provide multiple servers for flight data. The client should be developed so that it will attempt to connect to different IP addresses. In the case that one connect attempt fails, the client should go on to try the next address. Only one connection should be active from any given client at a time.

4.2 Simplified Subs Application Protocol

4.2.1 Overview

Once the session is established, the client sends simplified subs messages and the server sends replies. The airline has control over where the reply goes, but the reply is mandatory. Like the flight data feed, the airline can control:

- § Whether the replies go back to the sending address or to an alternate ARINC address.

This choice is controlled by the airlines through the use of keywords in the packet header record, as in the Flight Data protocol.

The hub site will generate replies in the following manner by default.

- The reply will be sent to the address from which the message was received (i.e., either an IP CDMNET address or an ARINC address).

If an airline does NOT want to get replies at its sending address, it will indicate an alternate address in the message packet header. NOTE: The alternate address can only be an ARINC address!

The use of the ARINC address keywords is already accounted for in the CDM message format. This protocol simply preserves what is already being used.

NOTE: There is no way to specify an alternate IP address for replies.

4.2.2 Messages

The simplified subs protocol will use the following messages:

[102] M_FLIGHT_DATA_REPLY - [server to client]

Sent from ETMS hub to client to indicate the result of the sub processing.

[103] M_SLOT_DATA - [server to client]

Sends slot list data to an airline client as part of a GDP (e.g., initial GDP, compression, revision).

[104] M_GDP_REQ - [client to server]

Sent from client to ETMS hub to request a new slot list. (NOTE: Reply is an M_GDP_REPLY message.)

[105] M_GDP_REPLY - [server to client]

Sent from ETMS to client with the response to the corresponding request.

[106] M_GDP_MESSAGE - [server to client]

Sent from ETMS to client when a GDP is purged (cancelled), when subs are turned off, or when subs are turned on.

[112] M_SS_DATA_PACKET - [client to server]

Sends simplified substitution and slot credit substitution messages from an airline to ETMS hub. (NOTE: beginning with ETMS release 7.9, only M_SS_DATA_PACKET can be used to submit substitution requests.)

The detailed application-level message contents (e.g., slot list format, SS packet format) are described in the ***Simplified Subs Requirements*** document. The following section provides more detail on the message header format.

5. Detailed Message Specifications

Messages consist of a message header, followed by a message body. The message header is fixed length (24 bytes), consisting of six 4-byte integers. The last field of the header specifies the byte count of the optional data buffer. A message contains the following fields:

1) Message Type	Integer value of the message type. [4-byte integer]
2) Message Source	Encoded source identifier assigned by Volpe. Not used for flight data messages. [4-byte integer]
3) Message Destination	Encoded destination identifier assigned by Volpe. Not used for flight data messages. [4-byte integer]
4) Client Tag	Used to identify one of a set of clients (e.g., if there is more than one client running at a site). [4-byte integer]
5) Short Data	32 bits of message-specific data. Included in every message. [4-byte integer]
6) Message Length	Length of the data buffer (may be zero; maximum is 128 Kbytes). [4-byte integer]
7) Data Buffer	An array of bytes that is 'Message Length' bytes long. It is only sent if the Length field in the header is non-zero. Maximum size is 128 Kbytes. Format of the information in the buffer is message-specific.

The remainder of this section describes the detailed contents for each of the message types described above. Literal constants are shown in square brackets; that is, if the field is supposed to contain the integer value 1, it is shown as [1].

[1] M_ATMS_CONNECT

This message is used to tell a server at the ETMS hub site that a new client process is coming on-line. The message identifies where the client is running and includes an encrypted password. The server validates the password and the request. If it is valid, the session starts. The server notifies the client whether the connection was accepted or not. If there is more than one client at this site, the client tag field must contain different invocation numbers for each client.

The detailed field contents are as follows:

1) Message Type -	M_ATMS_CONNECT [1]
2) Message Source -	encoded source identifier [103] for FSM; [0] for flight data or simplified subs client
3) Message Destination -	encoded destination identifier [1 or 0]
4) Client Tag -	client tag assigned by Volpe
5) Short Data -	[0]
6) Message Length -	[0]
7) Data Buffer -	N/A

[2] M_ATMS_ACCEPT

This message from the server tells the client that the socket connection has been set up correctly. The short data field should be zero, and there is no message body.

The detailed field contents are as follows:

1) Message Type - M_ATMS_ACCEPT [2]

2) Message Source - encoded source identifier [1]

3) Message Destination - encoded destination identifier [103]

4) Client Tag - client tag assigned by Volpe

5) Short Data - [0]

6) Message Length - [0]

7) Data Buffer - N/A

[3] M_ATMS_REJECT

This message from the server tells the client that the connection has not been accepted. The short data field contains an error code explaining the cause of the failure. I have started a list of error codes, but we probably will come up with more along the way.

The detailed field contents are as follows:

1) Message Type - M_ATMS_REJECT [3]

2) Message Source - encoded source identifier [1]

3) Message Destination - encoded destination identifier [103]

4) Client Tag - client tag assigned by Volpe

5) Short Data - error code (see following list)

6) Message Length - [0]

7) Data Buffer - N/A

Error codes are as follows:

[1] - UNKNOWN MESSAGE SOURCE

[2] - INVALID PASSWORD

[3] - SERVER PROCESSING CURRENTLY NOT AVAILABLE

[4] - CLIENT WITH SAME CLIENT TAG ALREADY RUNNING

[4] M_DISCONNECT

This message is sent from the client to the server to indicate that the client process is shutting down. The short data field is zero and there is no message body.

The detailed field contents are as follows:

1) Message Type - M_DISCONNECT [4]

2) Message Source - encoded source identifier [103]

3) Message Destination - encoded destination identifier [1]

4) Client Tag - client tag assigned by Volpe

5) Short Data - [0]

6) Message Length - [0]

7) Data Buffer - N/A

[5] M_SHUTDOWN

This message is sent from a server to a client to indicate that the server process is shutting down. The short data field is zero and there is no message body. This gives the client the chance to shut down gracefully.

The detailed field contents are as follows:

1) Message Type - M_SHUTDOWN [5]

2) Message Source - encoded source identifier [1]

3) Message Destination - encoded destination identifier [103]

4) Client Tag - client tag assigned by Volpe

5) Short Data - [0]

6) Message Length - [0]

7) Data Buffer - N/A

[6 through 9] unallocated

[10] M_HB_REQ

This message is sent from a client to the server to request a reply that will confirm that the connection is still alive.

The detailed field contents are as follows:

1) Message Type - M_HB_REQ [10]

2) Message Source - encoded source identifier [103] for FSM; [0] for flight data or simplified subs client

3) Message Destination - encoded destination identifier [1]

4) Client Tag - client tag assigned by Volpe

5) Short Data - sequence number (1 to 9999)

6) Message Length - [0]

7) Data Buffer - N/A

[11] M_HB_ACK

This message is sent from the server to a client as the reply to a M_HB_REQ request. It confirms that the connection is alive.

The detailed field contents are as follows:

1) Message Type - M_HB_ACK [11]

2) Message Source - [1]

3) Message Destination -	encoded source identifier [103] for FSM; [0] for flight data or simplified subs client
4) Client Tag –	client tag assigned by Volpe
5) Short Data –	sequence number from corresponding M_HB_REQ (1 to 9999)
6) Message Length -	[0]
7) Data Buffer -	N/A

[12 through 100] unallocated

[101] M_FLIGHT_DATA_PACKET

This message from a client to the server sends flight data message or simplified subs messages from the airline to the ETMS hub. The data buffer contains the packet formatted as it currently is being sent through ARINC; that is, packet header followed by a variable number of messages terminated by line feeds. The short data field contains a sequence number, which is simply an integer incremented by one for each packet sent; the range of the sequence numbers is 1 to 9999.

The detailed field contents are as follows:

1) Message Type -	M_FLIGHT_DATA_PACKET [101]
2) Message Source -	[0]
3) Message Destination -	[0]
4) Client Tag -	client tag assigned by Volpe
5) Short Data -	sequence number (1 to 9999)
6) Message Length -	total number of bytes in the data buffer
7) Data Buffer -	a message packet containing a header line and a variable number of messages as defined in the simplified subs requirements document

[102] M_FLIGHT_DATA_REPLY

This message from the server to a client sends the replies to the flight data packets or simplified subs packets from the ETMS hub to the airline. The data buffer contains the same reply that is currently being sent through ARINC; that is, an acknowledgment line followed by a variable number of error messages. The same options will exist: NOACK if no “good” acknowledgment is desired and an optional ARINC address for errors. (The FDP server will always send errors back to the FDS client through the socket, but will optionally also send error messages to the ARINC address if provided.) Packet components are not terminated by line feeds, as in the ARINC protocol, but sent as separate strings in the data buffer. The short data field contains the sequence number from the flight data message to which the server is replying.

The detailed field contents are as follows:

1) Message Type -	M_FLIGHT_DATA_REPLY [102]
2) Message Source -	[0]
3) Message Destination -	[0]

4) Client Tag -	client tag assigned by Volpe
5) Short Data -	sequence number (from flight data message)
6) Message Length -	total number of bytes in the data buffer
7) Data Buffer -	a message packet containing an acknowledgment and a variable number of error messages as defined in the simplified subs requirements document

[103] M_SLOT_DATA

This message from the server to the client sends slot lists (as part of a GDP) from the ETMS hub to the airline. The data buffer contains the slot list formatted as a stream of ASCII as specified in the simplified subs requirements documents. The short data field contains a sequence number, which is simply an integer incremented by one for each packet sent; the range of the sequence numbers is 1 to 9999.

The detailed field contents are as follows:

1) Message Type -	M_SLOT_DATA [103]
2) Message Source -	[0]
3) Message Destination -	[0]
4) Client Tag -	client tag assigned by Volpe
5) Short Data -	if in response to an M_LOST_DATA_REQ, the sequence number from the request message; otherwise [0]
6) Message Length -	total number of bytes in the data buffer
7) Data Buffer -	a slot list formatted according to the simplified subs requirements document

[104] M_GDP_REQ

This message from the client to the server sends slot list to the ETMS hub. The data buffer contains the slot list request formatted as a stream of ASCII as specified in the simplified subs requirements documents. The short data field contains a sequence number, which is simply an integer incremented by one for each packet sent; the range of the sequence numbers is 1 to 9999.

The detailed field contents are as follows:

1) Message Type -	M_GDP_REQ [104]
2) Message Source -	[0]
3) Message Destination -	[0]
4) Client Tag -	client tag assigned by Volpe
5) Short Data -	sequence number (1 to 9999)
6) Message Length -	total number of bytes in the data buffer
7) Data Buffer -	a slot list request formatted according to the simplified subs requirements document

[105] M_GDP_REPLY

This message from the server to the client sends the response to a GDP request. The data buffer contains the message formatted as a stream of ASCII as specified in the simplified subs requirements documents.

The detailed field contents are as follows:

1) Message Type -	M_GDP_REPLY [105]
2) Message Source -	[0]
3) Message Destination -	[0]
4) Client Tag -	client tag assigned by Volpe
5) Short Data -	sequence number from M_GDP_REQ message (1 to 9999)
6) Message Length -	total number of bytes in the data buffer
7) Data Buffer -	a GDP message formatted according to the simplified subs requirements document

[106] M_GDP_MESSAGE

This message from the server to the client sends GDP messages from the ETMS hub to the airline. The data buffer contains the message formatted as a stream of ASCII as specified in the simplified subs requirements documents.

The detailed field contents are as follows:

1) Message Type -	M_GDP_MESSAGE [106]
2) Message Source -	[0]
3) Message Destination -	[0]
4) Client Tag -	client tag assigned by Volpe
5) Short Data -	[0]
6) Message Length -	total number of bytes in the data buffer
7) Data Buffer -	a GDP message formatted according to the simplified subs requirements document

[107 - 108] unallocated

[109] M_FPPP_FLIGHT_DATA_PACKET

This message from a client to the server sends early intent messages from the airline to the FPPP server, which forwards them to the ETMS hub. The data buffer contains the packet header followed by one early intent flight plan.

The detailed field contents are as follows:

1) Message Type -	M_FPPP_FLIGHT_DATA_PACKET [109]
2) Message Source -	[0]
3) Message Destination -	[0]
4) Client Tag -	client tag assigned by Volpe
5) Short Data -	sequence number (1 to 9999)

6) Message Length - total number of bytes in the data buffer

7) Data Buffer - a message packet containing a header line and one early intent message as defined in the early intent protocol document

[110] M_FPPP_FLIGHT_DATA_RESPONSE_PACKET

This message from the server to a client sends early intent messages from the ETMS hub to an airline via the FPPP server. The data buffer contains the packet header followed by the results of one early intent message.

The detailed field contents are as follows:

1) Message Type - M_FPPP_FLIGHT_DATA_RESPONSE_PACKET [110]

2) Message Source - [0]

3) Message Destination - [0]

4) Client Tag - client tag assigned by Volpe

5) Short Data - sequence number (1 to 9999)

6) Message Length - total number of bytes in the data buffer

7) Data Buffer - a message packet containing a header line and the results of one early intent message as defined in the early intent protocol document

[111] M_EI_FLIGHT_DATA_PACKET

This message from a client to the server sends early intent messages from the airline to the ETMS hub. The data buffer contains the packet header followed by one early intent flight plan.

The detailed field contents are as follows:

1) Message Type - M_EI_FLIGHT_DATA_PACKET [111]

2) Message Source - [0]

3) Message Destination - [0]

4) Client Tag - client tag assigned by Volpe

5) Short Data - sequence number (1 to 9999)

6) Message Length - total number of bytes in the data buffer

7) Data Buffer - a message packet containing a header line and one early intent message as defined in the early intent protocol document

[112] M_SS_DATA_PACKET

This message from a client to the server sends simplified subs or slot credit sub messages from the airline to the ETMS hub. The data buffer contains the packet header followed by a variable number of messages terminated by line feeds. The short data field contains a sequence number, which is simply an integer incremented by one for each packet sent; the range of the sequence numbers is 1 to 9999. (NOTE: prior to ETMS version 7.9 both flight data and substitution

messages were sent with M_FLIGHT_DATA_PACKET. Beginning with 7.9, substitution messages must be sent with M_SS_DATA_PACKET. Replies for both will still be sent under M_FLIGHT_DATA_REPLY.)

The detailed field contents are as follows:

1) Message Type -	M_SS_DATA_PACKET [112]
2) Message Source -	[0]
3) Message Destination -	[0]
4) Client Tag -	client tag assigned by Volpe
5) Short Data -	sequence number (1 to 9999)
6) Message Length -	total number of bytes in the data buffer
7) Data Buffer -	a message packet containing a header line and a variable number of messages as defined in the simplified subs and slot credit subs requirements documents

[113 through 200] unallocated

[201] M_REGISTER

This message from the client tells server that this server wants data for a particular airport. The short data field tells the server what type of site the client is (CDMNET or ETMS). The message body should contain the name of the airport and the pathname where the file will go. For ETMS sites, two pathnames will be sent: the pathname that the server should use to FTP the file to the user site, and the pathname that the server will use to access the file. The server will construct full pathnames for the files. For CDMNET sites, server will send this pathname to client in the M_START_ADL message.

The detailed field contents are as follows:

1) Message Type -	M_REGISTER [201]
2) Message Source -	encoded source identifier [103]
3) Message Destination -	encoded destination identifier [1]
4) Client Tag -	client tag assigned by Volpe
5) Short Data -	ADL Type Requested;
	0 for an Airport ADL with Arrivals Only
	1 for an Airport ADL with Departures Only
	2 for an Airport Arrivals and Departures
	3 for an FEA ADL
	7 for an FCA ADL
6) Message Length -	total number of bytes in data buffer
7) Data Buffer -	client name, element name, client pathname, and client version; fields are space delimited

[202] M_REGISTER_ACK

This message from the server to the client tells client whether the registration was accepted or not. The short data field is 0 if the registration was accepted, or contains an error code is the registration was not accepted. The data buffer contains the name of the element for which the registration was requested.

The detailed field contents are as follows:

1) Message Type -	M_REGISTER_ACK [202]
2) Message Source -	encoded source identifier [1]
3) Message Destination -	encoded destination identifier [103]
4) Client Tag -	client tag assigned by Volpe
5) Short Data -	0 if OK; error code if rejected
6) Message Length -	total number of bytes in data buffer
7) Data Buffer -	element name

Error codes are as follows:

1 - UNKNOWN ELEMENT

2 - ELEMENT ALREADY REGISTERED

3 - MAXIMUM NUMBER OF ELEMENTS EXCEEDED

[203] M_UN_REGISTER

This message from the client tells server to stop collecting data at a particular element for this client. If this is the last client attached to that element, then the ADL Data Distributor may stop collecting data completely for that element. The short data field should be zero. The message body should contain the element name, or the keyword "ALL". If "ALL" is specified, then the ADL Data Distributor should stop collecting data for all elements that have been previously registered by this client.

If a client is shutting down, it should send a un-register for all elements currently being collected prior to sending an M_DISCONNECT message.

The detailed field contents are as follows:

1) Message Type -	M_UN_REGISTER [203]
2) Message Source -	encoded source identifier [103]
3) Message Destination -	encoded destination identifier [1]
4) Client Tag -	client tag assigned by Volpe
5) Short Data -	[0]
6) Message Length -	total number of bytes in data buffer
7) Data Buffer -	element name or “ALL”

[204] M_UN_REGISTER_ACK

This message from the server to the client tells client whether the un-registration was accepted or not. The short data field is 0 if the un-registration was accepted, or contains an error code is the

un-registration was not accepted. The data buffer contains the name of the element for which the un-registration was requested.

The detailed field contents are as follows:

1) Message Type - M_UN_REGISTER_ACK [204]
2) Message Source - encoded source identifier [1]
3) Message Destination - encoded destination identifier [103]
4) Client Tag - client tag assigned by Volpe
5) Short Data - 0 if OK; error code if rejected
6) Message Length - total number of bytes in data buffer
7) Data Buffer - element name

Error codes are as follows:

1 - ELEMENT NOT REGISTERED
2 - NO ELEMENTS REGISTERED

[205] M_START_ADL

This message from the server to client tells client that server is beginning to send a new ADL file. The data buffer contains the pathname of the file. The client opens the file and waits for data packets, which it will write to the file. The client also starts a sequence counter for the file packets. The short data field contains the first sequence number that is always one.

The detailed field contents are as follows:

1) Message Type - M_START_ADL [205]
2) Message Source - encoded source identifier [1]
3) Message Destination - encoded destination identifier [103]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [1]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - pathname of file

[206] M_ADL_DATA

This message is used to send a piece of an ADL file from server to client. The data buffer contains the data. The client will extract the data from the buffer and write to the file that is already open. The short data field contains the packet sequence number. The client checks the sequence number and notifies server if there is an error.

The detailed field contents are as follows:

1) Message Type - M_ADL_DATA [206]
2) Message Source - encoded source identifier [1]
3) Message Destination - encoded destination identifier [103]
4) Client Tag - client tag assigned by Volpe

5) Short Data - sequence number

6) Message Length - total number of bytes in data buffer

7) Data Buffer - data

[207] M_END_ADL

This message from the server to client tells client that the ADL file is complete. The data buffer contains the pathname of the file. The short data field contains the last sequence number. The client checks it and if all is well, closes the file.

The detailed field contents are as follows:

1) Message Type - M_END_ADL [207]

2) Message Source - encoded source identifier [1]

3) Message Destination - encoded destination identifier [103]

4) Client Tag - client tag assigned by Volpe

5) Short Data - sequence number

6) Message Length - total number of bytes in data buffer

7) Data Buffer - pathname of file

[208 through 211] unallocated

[212] M_EDCT_COMMAND

This message from the client to the server is an EDCT command. The message body contains the full text of the command, which is forwarded to EDCT.

The detailed field contents are as follows:

1) Message Type - M_EDCT_COMMAND [212]

2) Message Source - encoded source identifier [103]

3) Message Destination - encoded destination identifier [1]

4) Client Tag - client tag assigned by Volpe

5) Short Data - sequence number (1 to 9999)

6) Message Length - total number of bytes in data buffer

7) Data Buffer - EDCT UPDATE request

[213] M_EDCT_REPLY

This message from the server to the client is the reply generated by EDCT in response to an EDCT command. The short data field is zero. The message body contains the full text of the reply.

The detailed field contents are as follows:

1) Message Type - M_EDCT_REPLY [213]

2) Message Source - encoded source identifier [1]

3) Message Destination - encoded destination identifier [103]

4) Client Tag - client tag assigned by Volpe

5) Short Data - sequence number from M_EDCT_COMMAND message (1 to 9999)

6) Message Length - total number of bytes in data buffer

7) Data Buffer - text of reply

[214] M_REQ_COMMAND

This message from the client to the server is an EDCT command line request. The message body contains the full text of the command, which is forwarded to EDCT.

The detailed field contents are as follows:

1) Message Type - M_REQ_COMMAND [214]

2) Message Source - encoded source identifier [103]

3) Message Destination - encoded destination identifier [1]

4) Client Tag - client tag assigned by Volpe

5) Short Data - sequence number (1 to 9999)

6) Message Length - total number of bytes in data buffer

7) Data Buffer - request

[215] M_REQ_REPLY

This message from the server to the client is the reply generated by EDCT in response to an EDCT command line request. The short data field is zero. The message body contains the full text of the reply.

The detailed field contents are as follows:

1) Message Type - M_REQ_REPLY [215]

2) Message Source - encoded source identifier [1]

3) Message Destination - encoded destination identifier [103]

4) Client Tag - client tag assigned by Volpe

5) Short Data - sequence number from M_REQ_COMMAND message (1 to 9999)

6) Message Length - total number of bytes in data buffer

7) Data Buffer - text of reply

[216 through 217] unallocated

[218] M_ADL_DATA_ACK

This message from the server to the client tells the client that the previous GDP data message was received. The short data field equals the value of the short data field of the GDP message.

The detailed field contents are as follows:

1) Message Type - M_ADL_DATA_ACK [218]
2) Message Source - encoded source identifier [1]
3) Message Destination - encoded destination identifier [103]
4) Client Tag - client tag assigned by Volpe
5) Short Data - value of previous message
6) Message Length - [0]
7) Data Buffer - N/A

[219] M_ADD_ADL_AAR

This message from a client to the server submits an updated AAR for an element.

The detailed field contents are as follows:

1) Message Type - M_ADD_ADL_AAR [219]
2) Message Source - encoded source identifier [103]
3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the element and the new rates

[220] M_ADD_ADL_ADR

This message from a client to the server submits an updated ADR for an airport.

The detailed field contents are as follows:

1) Message Type - M_ ADD_ADL_ADR [220]
2) Message Source - encoded source identifier [103]
3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the airport and the new rates

[221] M_ADD_ADL_AAR_GDP

This message from a client to the server submits the AAR associated with a GDP for an element. This message must be sent prior to message M_ADD_ADL_GDP_PARAM.

The detailed field contents are as follows:

1) Message Type - M_ADD_ADL_AAR_GDP [221]
2) Message Source - encoded source identifier [103]
3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the element and the new rates

[222] M_ADD_ADL_GDP_PARAM

This message from a client to the server submits the parameters for a GDP.

The detailed field contents are as follows:

1) Message Type - M_ADD_ADL_GDP_PARAM [222]
2) Message Source - encoded source identifier [103]
3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the airport and the parameters

[223] unallocated

[224] M_ADD_ADL_COMP_PARAM

This message from a client to the server submits the parameters for a compression.

The detailed field contents are as follows:

1) Message Type - M_ADD_ADL_COMP_PARAM [224]
2) Message Source - encoded source identifier [103]
3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the element and the parameters

[225] M_ADD_ADL_BLANK_PARAM

This message from a client to the server submits the parameters for a blanket program.

The detailed field contents are as follows:

1) Message Type - M_ADD_ADL_BLANK_PARAM [225]
2) Message Source - encoded source identifier [103]
3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the airport and the parameters

[226] M_ADD_ADL_GS_PARAM

This message from a client to the server submits the parameters for a ground stop.

The detailed field contents are as follows:

1) Message Type - M_ADD_ADL_GS_PARAM [226]
2) Message Source - encoded source identifier [103]
3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the airport and the parameters

[227] M_DEL_ADL_AAR

This message from a client to the server deletes the AAR for an element.

The detailed field contents are as follows:

1) Message Type - M_DEL_ADL_AAR [227]
2) Message Source - encoded source identifier [103]
3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the element

[228] M_DEL_ADL_ADR

This message from a client to the server deletes the ADR for an airport.

The detailed field contents are as follows:

1) Message Type - M_ DEL_ADL_ADR [228]
2) Message Source - encoded source identifier [103]

3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the airport

[229] M_DEL_ADL_AAR_GDP

This message from a client to the server deletes the AAR and GDP for an element.
The detailed field contents are as follows:

1) Message Type - M_DEL_ADL_AAR_GDP [229]
2) Message Source - encoded source identifier [103]
3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the element

[230] M_DEL_ADL_GDP_PARAM

This message from a client to the server deletes the parameters for a GDP.
The detailed field contents are as follows:

1) Message Type - M_DEL_ADL_GDP_PARAM [230]
2) Message Source - encoded source identifier [103]
3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the airport

[231] unallocated

[232] M_DEL_ADL_COMP_PARAM

This message from a client to the server deletes the parameters for a compression.
The detailed field contents are as follows:

1) Message Type - M_DEL_ADL_COMP_PARAM [232]
2) Message Source - encoded source identifier [103]

3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the airport

[233] M_DEL_ADL_BLANK_PARAM

This message from a client to the server deletes the parameters for a blanket program.

The detailed field contents are as follows:

1) Message Type - M_DEL_ADL_BLANK_PARAM [233]
2) Message Source - encoded source identifier [103]
3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the airport

[234] M_DEL_ADL_GS_PARAM

This message from a client to the server deletes the parameters for a ground stop.

The detailed field contents are as follows:

1) Message Type - M_DEL_ADL_GS_PARAM [234]
2) Message Source - encoded source identifier [103]
3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the airport

[235] M_UPDATE_ADL_REQ

This message from a client to the server requests an updated ADL for an element.

The detailed field contents are as follows:

1) Message Type - M_UPDATE_ADL_REQ [235]
2) Message Source - encoded source identifier [103]
3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe

5) Short Data - [0]

6) Message Length - total number of bytes in data buffer

7) Data Buffer - name of the element

[236] M_WEATHER_COMMAND

This message from a client to the server requests a weather report for an airport.

The detailed field contents are as follows:

1) Message Type - M_WEATHER_COMMAND [236]

2) Message Source - encoded source identifier [103]

3) Message Destination - encoded destination identifier [1]

4) Client Tag - client tag assigned by Volpe

5) Short Data - sequence number (1 to 9999)

6) Message Length - total number of bytes in data buffer

7) Data Buffer - name of the airport

[237] M_WEATHER_REPLY

This message from a server to the client contains the weather report that was requested for an airport.

The detailed field contents are as follows:

1) Message Type - M_WEATHER_REPLY [237]

2) Message Source - encoded source identifier [1]

3) Message Destination - encoded destination identifier [103]

4) Client Tag - client tag assigned by Volpe

5) Short Data - sequence number from corresponding M_WEATHER_COMMAND (1 to 9999)

6) Message Length - total number of bytes in data buffer

7) Data Buffer - text of reply

[238 through 241] unallocated

[242] M_AUTO_MONITOR_ REQ

This message from a client to the server requests an auto-monitor report. This report contains the current traffic management initiatives that are in place, are proposed, or have been purged and the current FEAs and FCAs available for monitoring with FSM.

The detailed field contents are as follows:

1) Message Type - M_AUTO_MONITOR_REQ [242]

2) Message Source - encoded source identifier [103]

3) Message Destination - encoded destination identifier [1]

4) Client Tag - client tag assigned by Volpe

5) Short Data - [0]

6) Message Length - [0]

7) Data Buffer - N/A

[243] M_AUTO_MONITOR_REPLY

This message from the server to the client sends the auto-monitor report from the ETMS hub to the client that made the request. The data buffer contains the message formatted as a stream of ASCII as specified in *ADL Data Format document, version 10*.

The detailed field contents are as follows:

1) Message Type - M_AUTO_MONITOR_REPLY [243]

2) Message Source - [1]

3) Message Destination - [103]

4) Client Tag - client tag assigned by Volpe

5) Short Data - [0]

6) Message Length - total number of bytes in the data buffer

7) Data Buffer - an auto-monitor message formatted according to the *ADL Data Format document, version 10*

[244] M_AUTO_MONITOR_MESSAGE

This message from the server to the client sends the auto-monitor report from the ETMS hub to any connected client. The data buffer contains the message formatted as a stream of ASCII as specified in the *ADL Data Format document, version 10*.

The detailed field contents are as follows:

1) Message Type - M_AUTO_MONITOR_MESSAGE [244]

2) Message Source - [1]

3) Message Destination - [103]

4) Client Tag - client tag assigned by Volpe

5) Short Data - [0]

6) Message Length - total number of bytes in the data buffer

7) Data Buffer - an auto-monitor message formatted according to the *ADL Data Format document, version 10*

[245] M_ADD_ADL_AFP_PARAM

This message from a client to the server submits the parameters for an AFP.

The detailed field contents are as follows:

1) Message Type - M_ADD_ADL_AFP_PARAM [245]
2) Message Source - encoded source identifier [103]
3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the element and the parameters

[246] M_DEL_ADL_AFP_PARAM

This message from a client to the server deletes the parameters for an AFP.

The detailed field contents are as follows:

1) Message Type - M_DEL_ADL_AFP_PARAM [246]
2) Message Source - encoded source identifier [103]
3) Message Destination - encoded destination identifier [1]
4) Client Tag - client tag assigned by Volpe
5) Short Data - [0]
6) Message Length - total number of bytes in data buffer
7) Data Buffer - name of the element

6. Message Applicability

This section provides a summary list of all CDM messages and which connection they are applicable to.

			Applicable Protocols		
Message Number	**Message Type**	**Message Direction**	**ADL / GDP**	**Flight Data**	**Simplified Subs**
1	M_ATMS_CONNECT	C > S	X	X	X
2	M_ATMS_ACCEPT	S > C	X	X	X
3	M_ATMS_REJECT	S > C	X	X	X
4	M_DISCONNECT	C > S	X		
5	M_SHUTDOWN	S > C	X	X	X
6 - 9	unallocated				
10	M_HB_REQ	C > S	X	X	X
11	M_HB_ACK	S > C	X	X	X
12 – 100	unallocated				
101	M_FLIGHT_DATA_PACKET	C > S		X	
102	M_FLIGHT_DATA_REPLY	S > C		X	X
103	M_SLOT_DATA	C > S			X
104	M_GDP_REQ	C > S			X
105	M_GDP_REPLY	S > C			X
106	M_GDP_MESSAGE	S > C			X
107 - 108	unallocated				
109	M_FPPP_FLIGHT_DATA_PACKET	C > S		X	
110	M_FPPP_FLIGHT_DATA_RESPONSE_PACKET	S > C		X	
111	M_EI_FLIGHT_DATA_PACKET	C > S		X	
112	M_SS_DATA_PACKET	C > S			X
113 – 200	Unallocated				
201	M_REGISTER	C > S	X		
202	M_REGISTER_ACK	S > C	X		
203	M_UN_REGISTER	C > S	X		
204	M_UN_REGISTER_ACK	S > C	X		
205	M_START_FILE / M_START_ADL	S > C	X		
206	M_ADL_DATA	S > C	X		
207	M_END_FILE	S > C	X		

			Applicable Protocols		
Message Number	**Message Type**	**Message Direction**	**ADL / GDP**	**Flight Data**	**Simplified Subs**
208 - 211	unallocated				
212	M_EDCT_COMMAND	C > S	X		
213	M_EDCT_REPLY	S > C	X		
214	M_REQ_COMMAND	C > S	X		
215	M_REQ_REPLY	S > C	X		
216 – 217	unallocated				
218	M_ADL_DATA_ACK	S > C	X		
219	M_ADD_ADL_AAR	C > S	X		
220	M_ADD_ADL_ADR	C > S	X		
221	M_ADD_ADL_AAR_GDP	C > S	X		
222	M_ADD_ADL_GDP_PARAM	C > S	X		
223	unallocated				
224	M_ADD_ ADL_COMP_PARAM	C > S	X		
225	M_ADD_ ADL_BLANK_PARAM	C > S	X		
226	M_ADD_ADL_GS_PARAM	C > S	X		
227	M_DEL_ADL_AAR	C > S	X		
228	M_DEL_ADL_ADR	C > S	X		
229	M_DEL_ADL_AAR_GDP	C > S	X		
230	M_DEL_ADL_GDP_PARAM	C > S	X		
231	unallocated				
232	M_DEL_ADL_COMP_PARAM	C > S	X		
233	M_DEL_ADL_BLANK_PARAM	C > S	X		
234	M_DEL_ADL_GS_PARAM	C > S	X		
235	M_UPDATE_ADL_REQ	C > S	X		
236	M_WEATHER_COMMAND	C > S	X		
237	M_WEATHER_REPLY	S > C	X		
238 – 241	unallocated				
242	M_AUTO_MONITOR_REQ	C > S	X		
243	M_AUTO_MONITOR_REPLY	C > S	X		
244	M_AUTO_MONITOR_MESSAGE	S > C	X		
245	M_ADD_ADL_AFP_PARAM	C > S	X		
246	M_DEL_ADL_AFP_PARAM	C > S	X		