Aircraft Situation Display To Industry:

Functional Description and Interface Control Document for the XML Version

Version 1.8

Contract Number: DTFAWA-04-C-00045
CDRL Sequence No:

April 15, 2011

Prepared for:
U.S. Federal Aviation Administration

Prepared by:
Computer Sciences Corporation
Federal Sector – Civil Group
100 Decadon Drive
Egg Harbor Township, NJ 08234

Aircraft Situation Display To Industry:

Functional Description and Interface Control Document for the XML Version

Version 1.8

Contract Number: DTFAWA-04-C-00045
CDRL:

April 15, 2011

Prepared for:
U.S. Federal Aviation Administration

Prepared by:
Computer Sciences Corporation
Federal Sector – Civil Group
100 Decadon Drive
Egg Harbor Township, NJ 08234

Document History Record

Release	Date	Comment
Initial Draft v0.8	March 15, 2007	Initial draft for internal review
Initial Draft v0.9	March 22, 2007	Initial draft – Incorporated comments from first review.
Draft v1.0	March 27, 2007	Draft for Distribution to Vendors/Clients
Draft v1.1	May 14, 2007	Incorporated comments against v1.0, replaced example of the preamble in Section 6 and updated Section 4.4 on security.
Draft v1.2	September 7, 2007	Revised Facility Identifier list in section 3.3 to reflect current operational values; added new section 3.5 for ARTCC codes Updated Contacts Section 1.2
Draft v1.3	November 27, 2007	Revised section 4.1 to more clearly define the XASDI userID/password policy
Draft v1.4	March 17, 2008	Changed FAA contact from Judy Morrill to John McClure in section 1.2. Updated schema example in Appendix A, page A-3 to reflect change in logic, changed `<nxce:assignedAltitude>` to `<nxce:requestedAltitude>` when <nxcm:coordinationTime type="PROPOSED">
Draft v1.5	May 2, 2008	Wording updates to Sections 4.1, 4.5 and 4.6
Draft v1.5	July 29, 2008	Changed contact information in Section 1.2
Draft v1.6	October 31, 2008	Changed all references to Volpe Center to the TFMS Help Desk.
Draft v1.7	May 18, 2009	Added new facility ids in Section 3.3
V1.8	April 15, 2011	Relocated facility ids from Section 3 to the ASDI website at (http://www.fly.faa.goc/ASDI/asdi.html). Modified references from 2 ASDI servers to 1 and corrected registration message format to remove spaces. Editorial changes include; corrected references to ETMS with TFMS, removed references to XASDI and Legacy ASDI feed as this conversion has been completed.

Table of Contents

List of Tables

List of Figures

Foreword

The Aircraft Situation Display to Industry (ASDI) subsystem of the Traffic Flow Management System (TFMS) allows near real-time air traffic data to be disseminated to members of the aviation industry. This document provides a functional description of the ASDI subsystem and gives the interface information that a vendor would need to connect to this subsystem or that a user would need to interpret the ASDI data.

1 Introduction

1.1 Background and Terminology

In 1992 the Federal Aviation Administration (FAA) started providing airlines and other aviation-related organizations with access to near real-time air traffic data from the National Airspace System (NAS). This enabled airlines to use this more accurate positional information in their planning and decision making.

This original data feed was decommissioned in 1999 and was been replaced with an improved feed known as the Aircraft Situation Display to Industry (ASDI) feed. This feed, in turn, was replaced in 2007 with a version of the feed that is in the XML format. This document explains the XML version of the feed.

An organization that receives the ASDI feed directly from the FAA will be termed a *vendor*. An organization that uses the data is called a *user*. It is expected that a typical vendor will re-sell the data to users, though it might be the case that a vendor is also a user.

The purpose of this document is to provide vendors and users with the information that they need to receive and interpret the XML version of the ASDI feed. If a user purchases the feed from a vendor, it might be that this user will require additional information from the vendor.

1.2 Contacts

Throughout this document references are made to FAA contacts. For a complete list of the FAA points of contact and the ASDI Service desk, please refer to the ASDI website located at:

http://www.fly.faa.gov/ASDI/asdi.html

Required Information from Vendors for Data Questions
Each ASDI data packet is comprised of two parts. The first is a fixed length header. The second is the compressed data payload. The header information being requested is for the header of the data packet that contains the message in question.

The following table enumerates the data that vendors need to provide when making queries regarding ASDI data. Given the structure of the code and the corresponding logging that is done by the ASDI Server, all of this information is necessary in order to identify the data in question.

Data	Description	Example
Type of Feed	Determines instance of the ASDI a vendor is connected to	Class 1
Timestamp in header of data packet	The timestamp in the data packet header	20070910205027 (i.e. 2007/09/10 20:50.27)
Packet sequence number of the data packet	The sequence number (in ASCII format) in the data packet header	00016
Message type	Message type in question. For example, if it was in a TZ message, that would be a trackInformation message type.	trackInformation
ACID	Flight's ACID	AAL123
XML tag and value	Each component of the ASDI is encapsulated in an XML container. Need to identify the tag and the associated value of the data in question.	<nxcm:speed> - tag 200 - value

Generally, a vendor should get in touch with the FAA contact to get information about the memorandum of agreement (MOA) that entitles the vendor to get the feed. After a vendor has signed that memorandum of agreement, it should get in touch with the TFMS Help Desk to learn how to get connected, to ask technical questions about the feed, or to find out more about the documents listed in the next section.

1.3 Document Overview

This ICD is organized into the main document sections detailing the features of the ASDI XML data feed and Appendices containing supporting information such as a Glossary and Acronyms.

- Section 1 – Introduction, defines the purpose of this ICD and provides a brief overview of the major sections of the document
- Section 2 – Functional Overview, describes a high-level functional description for the new ASDI XML data feed and explains some of the differences between the old and new feeds.
- Section 3 – Content of the ASDI Data Stream Sent to the Vendors, describes the data content, filtering, message types and other information contained in the ASDI data feed.
- Section 4 – Software Interface, Hardware Interface, and Security, describes the software and hardware interfaces and protocols between the ASDI server and client connections. It also includes specific sub-sections describing software tips for the vendors to follow, recommended bandwidth, a summary of what a vendor needs to do to connect, security requirements and an explanation about the different classes of users.
- Section 5 – Structure of the XML Feed, describes the new structure of an ASDI packet, including the header and compressed payload of data. These details for the uncompressed payload, that is the XML formats, are described later in Section 6.
- Section 6 – Message Formats
- Appendix A – Sample Messages
- Appendix B – Acronyms, defines the acronyms used in this ICD.

1.4 Referenced Materials

http://www.fly.faa.gov/ASDI/asdi.html - This web site contains current information about the ASDI feed. It also includes the XML schema definition files (XSD) and sample ASDI XML messages.

"Memorandum of Agreement for Industry Access to Aircraft Situation Display and National Airspace System Status Information Data," June 1, 2006. This is the MOA

that a vendor must sign in order to get access to the ASDI feed. The current version of this document can be found on the web site referenced above.

http://nasdoc.faa.gov/ This web site contains FAA documentation on the National Aviation System. To find the current versions of the documents below, type "nas-md-311" in the search bar, click enter, and then click on HOST_A5f1.5_CPFSs. This will give a pick list of documents.

National Airspace System En Route Configuration Management Document, Computer Program Functional Specifications: Remote Outputs, *Model A5f1.5,* NAS-MD-315, 4 October 2004, National En Route Automation Division, AOS-300, Federal Aviation Administration, William J. Hughes Technical Center, Atlantic City International Airport, New Jersey 08405. This document describes all the messages that the Host computer can generate; in particular, Section 3 documents messages destined for the Central Flow Automation Facility (CFAF), which are the messages sent to TFMS. From the pick list mentioned above, click on *NAS-MD-315* to open this document.

National Airspace System En Route Configuration Management Document, Computer Program Functional Specifications: Route Conversion and Posting, *Model A5f1.5,* NAS-MD-312, 4 October 2004, National En Route Automation Division, AOS-300, Federal Aviation Administration, William J. Hughes Technical Center, Atlantic City International Airport, New Jersey 08405. This NAS specification describes the syntax of the field 10 (route of flight) that is used on the AF, FZ, and UZ messages. From the pick list mentioned above, click on *NAS-MD-312* to open this document.

National Airspace System En Route Configuration Management Document, Computer Program Functional Specifications: Message Entry and Checking, *Model A5f1.5,* NAS-MD-311, 4 October 2004, National En Route Automation Division, AOS-300, Federal Aviation Administration, William J. Hughes Technical Center, Atlantic City International Airport, New Jersey 08405. Appendix E of this document is the official documentation of the content of each field in the NAS messages. From the pick list mentioned above, click on *NAS-MD-311* to open this document.

Aircraft Situation Display to Industry: Functional Description and Interface Control Document, ver. 5.4, Volpe Center, report no. ASDI-FD-001, 15 November 2005. This document covers the legacy version of the ASDI feed and is replaced by the current document, which covers the XML version of the feed.

2 Functional Overview

The heart of the ASDI feed is the ASDI Server. These servers run at the Traffic Flow Management (TFM) Production Center located at the William J. Hughes Technical Center in Atlantic City (WJHTC), New Jersey. The main features of the ASDI servers include:

- Obtaining data: TFMS receives data from numerous air traffic control facilities (see the next section for a list). The ASDI Server receives all of these messages from TFMS.
- Processing data: The ASDI Server processes this data by filtering out messages that should not go out into the feed and by removing any fields that should not be sent out; see the next section for more details.
- Providing data: The ASDI Server will accept TCP/IP connections from approved vendors. When the connection is made, the ASDI Server sends to that vendor the messages that the vendor is entitled to receive. Once the vendor has received a message, it is up to the vendor to treat that message as it wishes (as long as it does not violate the memorandum of agreement it has signed with the FAA) and to distribute the data to its users; how the data is distributed by the vendors to the users is not the responsibility of the FAA and is not covered in this document.
- To connect to the ASDI server, a vendor must use a VPN over the Internet. An exception to this is that current vendors who transmit CDM data will be allowed to continue to use dedicated lines.
- The ASDI server provides data in XML format.
- The ASDI server may batch messages in data packets of a configurable size. That is, one data packet may contain 32, 64, 128, or any other amount of ASDI messages. Bandwidth studies indicate that 128 messages per packet to be the most efficient use of available bandwidth and has been set as the defacto size The ASDI server compresses the payload (messages) portion of a data packet; this means that the recipient of the data must decompress it. The structure of an ASDI data packet is further discussed in Section 5.

To get the ASDI XML feed, a vendor must sign the memorandum of agreement (see References) and set up a VPN over the Internet that provides sufficient bandwidth.

To provide reliability, the FAA provides at least two operational ASDI Servers behind a single virtual IP address. The normal condition is that all are running. There are three classes of the ASDI feed. There are two factors used to distinguish these classes

- Does the feed contain the London data?
- Are the messages passed to the vendor immediately or with a delay?

The rationale for these distinctions is explained in the next section. This gives rise to three feeds for non-FAA consumers.

TFMS MOA Class Definations	Class Definition	Corresponding TFMS ASDI Feed Category Types
N/A	Reserved for FAA usage	(1 - Reserved for FAA usage)
Class 1 with London	Undelayed, Filtered with LLON	2
Class 1 without London	Undelayed, Filtered without LLON	3
Class 2	Delayed, Filtered without LLON	4

All vendors who connect to a server for the same class of the feed receive exactly the same data. If two connections are made to two different servers for the same class of the feed, the data will be almost the same but not exactly; for example, messages might be ordered slightly differently in the feeds from the two different servers.

3 Content of the ASDI Data Stream Sent to the Vendors

3.1 Filtering Messages Out of the ASDI Data Stream

The ASDI Server registers with TFMS, which provides the ASDI server with the following messages.

- All of the NAS messages received by TFMS.
- All of the Position Update messages provided by the Surface Movement Advisor program to TFMS.
- All of the oceanic messages generated by TFMS.
- All of the RT messages generated by TFMS.

Before sending these messages over the ASDI feed, the ASDI server filters the messages in various ways.

- Sensitive message types are filtered out. The only message types currently allowed into the ASDI data stream are the following, where the message identifier in the legacy feed is given in brackets:
  - Flight plan [FZ]
  - Flight plan amendment [AF]
  - Departure [DZ]
  - Track information, i.e., a position report [TZ]
  - Boundary crossing [UZ]
  - Flight management information [RT]
  - Oceanic report [TO]
  - Arrival [AZ]
  - Flight plan cancellation [RZ]
  - Heartbeat [HB] (Note: In the XML feed this is represented in the data header. See Section 5 for details).

  See Section 6 for an explanation of these messages. An example of the type of message that is filtered out of the feed is the beacon code [BZ] message.
- Messages on sensitive flights are filtered out. For example, messages on military flights are not allowed into the ASDI data stream.
- Messages from certain facilities are filtered out. For example, currently messages from Mexico are filtered out of the feed. Currently messages are included from facilities in the United States, Canada, and the United Kingdom; . (There are two versions of the feed; one contains messages from the United Kingdom and the other doesn't, as explained below.)

It should be noted that the rules for filtering out sensitive flights are conservative in the sense that some flights that are not truly sensitive are filtered out; for example, most messages on GA flights registered in countries other than the U.S. and Canada are filtered out. If a vendor thinks that certain flights that are filtered from the feed

should be included, that vendor should get in touch with the FAA contact; the ASDI server can be configured to allow data on these flights into the feed.

3.2 Removing Fields From the Allowed ASDI Messages

If a message passes the filtering rules and is eligible to be sent out over the ASDI feed, the ASDI Server removes field 11 (remarks), if present, from this message before it is sent out.

3.3 Facility Identifiers

This section has been removed from the ICD and relocated to the ASDI website. For a list of facilities that send data to the TFMS and are included in the ASDI data feed, refer to the ASDI website located at:

http://www.fly.faa.gov/ASDI/asdi.html

3.4 Variations of TZ Messages

The ASDI feed has four different sources of TZ messages, which give the current position of a flight.

- All of the ARTCCs send NAS TZs (as well as many other message types) directly to TFMS. Each of these ARTCCs has its own facility identifier.
- The ARTSIIIE TRACONs send TZs directly to TFMS, and they send no other message type. Each of these ARTSIIIE TRACONs has its own facility identifier.
- All of the TRACONs that are not ARTSIIIE TRACONs send TZs through the associated ARTCCs to TFMS, and they send no other message types. The messages from these TRACONs have the facility identifier of the ARTCC through which the message was sent.

The TRACONs that are SMA sites send messages directly to TFMS that are translated into TZs. The SMA sites send no other message types.
The first three of these sources send TZs on each flight at roughly one minute intervals. In contrast, the SMA TZs are sent at roughly twenty second intervals.

3.5 ARTCC Codes

This section has been removed from the ICD and relocated to the ASDI website. For a list of facilities and the associated ARTCC Codes that are included in the ASDI data feed, refer to the ASDI website located at:

http://www.fly.faa.gov/ASDI/asdi.html

4 Software Interface, Hardware Interface, and Security

4.1 Software Interface

The interface between the ASDI server running at the William J Hughes Technical Center (WJHTC) and the client software running on the vendor's premises is a standard TCP/IP socket. The ASDI server listens on an assigned IP addresse and port number for clients that request to be registered to receive the data. After the vendor signs the ASDI memorandum of agreement with the FAA, it will be given this IP address and port number that it can use to connect to the ASDI server. The IP addresses from which the vendor will connect must be provided to the TFMS Help Desk so that the Tech Center can configure to allow clients from those addresses to connect to the appropriate servers.

The ASDI server sends identical data to all connected clients (of the same Class type). The ASDI server buffers packets for each connected client. The ASDI server will discard data if the client's buffer overflows; this buffer holds a configurable number of packets per client (currently 16, but this could change). Any write to a client that results in an error condition will cause the ASDI server to terminate the connection. If the connection is terminated, the client must re-establish a connection if it is to receive data. No data will be buffered for a disconnected client. A 'port filled' or 'no room in port' condition is not considered to be an error; if either of these conditions is returned when the ASDI server attempts to write to a socket, then this data is buffered, and later an attempt is made to re-transmit.

The ASDI server uses non-blocking write operations to its clients. The ASDI server sends all packets in sequence. The standard interpretation of any missing packets is that they resulted from a buffer overflow forced by (1) the client's not reading the messages fast enough or (2) the client's communication link not being able to accept the ASDI data feed fast enough.

After a vendor opens a socket, it is given 60 seconds to send a vendor name and password to the ASDI server. The message that contains the vendor name and password is called a registration message; once the ASDI server has received a valid registration message, the client is said to be registered. No data is sent until the ASDI server receives a valid registration message. If 60 seconds after the socket connection is made the ASDI server has not received a legal registration message, the ASDI server will close this socket. The table below specifies the format of the registration message, which contains a vendor name and password. There should be no spaces between each element of this message.

Table 4-1. Registration Message Format

ID	*Keyword specifying id field.*
=	*Field assignment character.*
<vendor name string>	*1 - 80 character alphanumeric string.*
,	*Field separator character.*
PASSWORD	*Keyword specifying password field.*
=	*Field assignment character.*
<vendor password string>	*1 – 12 character alphanumeric string.*

Following is an example of a registration message:

IDIAMAVENDOR,PASSWORDmypassword

1. A vendor is assigned one User ID/password.
2. This User ID/password is valid for one active connection.
3. A vendor is allowed 2 active connections for redundancy.
4. A vendor may therefore have more than assigned User ID/password.
5. Each vendor must determine how to allocate the 2 allowed connections among its authorized IP addresses.

An exception to the two ID/password is as follows: An additional two ID/password will be authorized for an independent data center (limited to only one additional center) and will have access to all operational servers. An additional VPN connection will be required.

The vendor name and password are needed when a vendor connects to an ASDI Server. How a user connects to a vendor to get the feed is beyond the scope of this document; a user should contact its vendor to learn how to connect to receive the feed from the vendor.

4.1.1 Software Tips for Vendors

To provide redundancy, the WJHTC runs multiple ASDI servers and vendors will be given one virtual IP address for access.

If the FAA needs to take down the servers to do maintenance, the FAA may take down one at a time. In a situation like this, no planned outage will be announced.

In short, it is the FAA's responsibility to make sure that at least one ASDI server is providing data.
The FAA strongly recommends that vendor software discard any unknown XML tags. There has been discussion of adding new data to the feed. If the FAA decides

to add additional data, it is required to give at least 60 days notice of a change such as this. Nevertheless, if a vendor's client software discards unknown XML tags, the vendor in this situation can avoid having to develop software against the clock.

4.2 Needed Bandwidth

It is the responsibility of the vendor to make sure that its connection to the Internet has sufficient bandwidth to carry the feed. A frequently asked question asked by vendors is: How much bandwidth is required? Unfortunately, it is not possible to give an entirely satisfactory answer to this question. The following are, however, guidelines.

- As this is written, a 128 kilobits per second (Kbps) line can just barely handle a feed. To prevent this feed from backing up, however, and to allow for future growth of the feed, a minimum bandwidth of 256Kbps is recommended.
- It will sometimes be the case, however, that a vendor will want to get two copies of the feed, e.g., a primary feed and a back-up feed, or a production feed and a test feed. For example, when a change is being made to the feed, sometimes the FAA will make the new feed available on a test basis so that the vendors can test with it before it becomes the production feed. A prudent vendor would want to have enough capacity to connect for both a production feed and a test feed. This means that a vendor might well want to have the capacity for two feeds, which would be 512Kbps. A vendor might want more bandwidth if it needs to get more than one class of the feed.
- A larger capacity might be needed if the vendor plans to use this line for additional types of data. For example, currently Class Two and Three vendors have the option of getting digital Runway Visual Range data, as well as additional types of data that are available under the TFM Data to Industry Program.
- The FAA has plans to increase the amount of data in the feed. No timetable has been announced, but a prudent vendor would purchase a little extra capacity so that it is ready for future additions.

In summary, a prudent vendor will want to be ready for the expected increase in the quantity of data with a line of 256Kbps, 512Kbps, or perhaps more. It should be stressed, however, that the decision on what bandwidth to buy is entirely the responsibility of the vendor; the remarks above are designed only to provide some facts to the vendor who is facing this decision.

4.3 Summary of What Needs To Be Done if a Vendor Is To Receive the Feed

The steps vendors must take to connect to an ASDI Server and get the feed are as follows:

- The vendor must send 2 signed copies of the ASDI memorandum of agreement to the FAA point of contact, along with the completed Vendor Information Form. Both documents can be found on the web site referred to in Section 1.4."
- The FAA point of contact must inform the TFMS Help Desk contact that a vendor has signed the documents required in the first step.
- The vendor must work with the TFMS Help Desk to set up a VPN over the Internet.
- The vendor must provide the TFMS Help Desk contact with the IP address(es) of its VPN Gateway.
- The vendor must provide the TFMS Help Desk contact with the IP address(es) of the clients that will connect to the ASDI Servers.
- The vendor must provide the TFMS Help Desk contact with the vendor name/password pairs that it will use.
- The TFMS Help Desk contact must provide the vendor with the IP addresses/ports on which the ASDI servers listen for connections.

If a vendor wants to receive the London data, it must go through additional steps spelled out in Section 4.6.

4.4 Security

This section deals with general and application security issues.

Please note that the VPN and network security requirements are specified in another document, "XML ASDI via the Internet: VPN End-User Requirements and Guidelines", which is available on the FAA web site. Vendors must meet all requirements specified in that document, as well as those detailed below.

One of the general requirements of the ASDI program is that security be maintained. There are two phases involved in maintaining security:

1. Vendors only engage in permitted activities.
2. Vendors take measures to detect and repel any prohibited activities.

The primary purpose of this section is to spell out which activities are permitted and which are prohibited so that vendors will not be in doubt about what is permitted. The secondary purpose is to state who is responsible when an prohibited action takes place.

Overview

The general principle is: **Any activity that is not explicitly permitted is prohibited.** As this document is written, only the following activities are permitted, and these are only permitted from an approved IP address.

A vendor is only permitted to:

1. Register with the ASDI server to receive ASDI data
2. Manually telnet to addresses and ports to test connectivity
3. Register to get the digital feed of RVR data, if this vendor is eligible under section 11 of the ASDI MOA."
4. Access the TFM Data to Industry Server

Other parties, e.g., organizations that only receive ASDI data from a vendor, are prohibited from all activity, unless this activity is approved in writing by the FAA contact.

It is expected that this list of permitted activities will change over time. Any changes to the list of permitted activities will be documented.

Detailed Guidelines

The following guidelines flesh out the general principle above by pointing out some of the prohibited activities and by specifying who is responsible for a violation of these guidelines. This list is not meant to be an exhaustive list of the security violations; it merely adds detail about some of the more likely violations.

If a prohibited activity is launched from an organization's IP domain, then that organization is responsible for that violation.

For example, if a disgruntled employee engages in a prohibited activity from an authorized IP address, then the vendor is responsible for this violation. For another example, if a hacker uses the Internet to break into a vendor's network and then engages in prohibited activity, then this vendor is responsible for this attack.

A vendor shall only allow the Tech Center to be accessed by parties operating from an approved IP address.

That is, the vendor is responsible for providing network security.

Any attempt to access an IP address other than those that are authorized shall be prohibited.

At the moment, the only authorized IP addresses are those of the various servers specified above. In addition, there might be IP addresses that the FAA does not see, e.g., the IP address that an airline uses to register with a vendor to receive ASDI data.

Any attempt to access a port other than a prescribed, published port shall be prohibited.

For example, if someone tries to access an unused port on the ASDI Server, this will be construed to be an attack.

IP spoofing shall be prohibited.

That is, it is prohibited for one party to attempt to use another's IP address.

An organization shall only capture or read data that it is explicitly authorized to look at.

A vendor is prohibited from using a packet sniffer to capture data that it is not authorized to access.

A password is associated with a set of IP addresses. Use of that password from some other IP address is prohibited.

This means that it is not allowed for one vendor to use another's password.

Summary

These guidelines are not exhaustive in that they do not spell out all of the prohibited activities; they only deal with the more important or the more likely prohibited activities. Again, the general principle is: **Any activity that is not explicitly permitted is prohibited.** If an ASDI vendor engages in a prohibited activity, then under section 12 of the ASDI MOA, the FAA has the right to cut off that vendor's access to the ASDI feed and any other feeds that the vendor is receiving.

4.5 Handling the London Data

British flight data is sent to the TFMS hubsite at the William J. Hughes Tech Center (WJHTC); this data is called the London data. The British authorities have placed on the FAA the requirement that the distribution of this data be restricted. This section discusses this requirement and its implications for ASDI vendors.

4.5.1.1 What Users Can Get the London Data?

The British require that a user only receive the London data if it satisfies one or both of the following conditions.

- The Class One User owns and operates aircraft in Europe, and therefore, pays landing fees and air traffic control fees in Europe.
- The Class One User has a customer who owns and operates aircraft in Europe, and therefore, pays landing fees and air traffic control fees in Europe. Only this customer shall have access to this data; this should be verified during the Class One audit.

If a user satisfies at least one of these conditions, it is said to be an *approved recipient* of the London data. The requirement imposed by the British, then, can be stated as

saying that only approved recipients can receive the data. If a user does not fulfill at least one of these two conditions, then that user is not allowed to receive any of the London data.

4.5.1.2 What Vendors Can Get the London Data?

By default, a vendor does not get the London data. That is, unless a vendor has been given explicit permission to get the London data, WJHTC will only allow that vendor to register to receive the feed that does not contain the London data.

The eligibility of a vendor in receiving the London Data is verified in a yearly audit. A vendor must have a successful audit, including the London eligibility confirmation, on file with the FAA to continue to receive the London data. Once an audit has been completed and submitted to the FAA, WJHTC will allow access to the London data feed.

If at any time the FAA discovers that a vendor is sending London data to an organization that is not an approved recipient, the FAA has the option of immediately cutting off that vendor's access to the feed that contains London data; that vendor would then only be allowed to access the feed that does not contain the London data.

4.5.1.3 How Can the London Data Be Identified?

Each message on the ASDI feed has four characters that indicate the facility that generated the message. These four characters are in bytes 13-16 (where the first byte is numbered 1). If these four bytes contain 'LLON', then this message is from London. If these four characters are anything else, then this message is not from London.

4.6 Class One and Class Two Users

When the ASDI feed was first established, the data was sent out to all vendors and users in near real-time, i.e., with a delay that in practice was a few seconds. The FAA has decided that for security reasons this near real-time feed should only go to those organizations with an established flight dispatch or planning function that requires near real-time data; these organizations are called Class One users. All other users are Class Two users, and these users would receive the feed with a delay. Examples of Class One users would be air carriers, regional air carriers, air taxis, and any organization that provides dispatch or tracking functions for aircraft owners, flight operation centers, government users, and professional flight planning service organizations. Examples of Class Two users would be most general aviation and non-aviation related industries. For a more detailed description of Class One and Class Two users, see Section 5.2 and 5.3 of the Memorandum of Agreement signed between a vendor and the FAA.

A vendor that has only Class Two users will only be allowed to get a delayed feed from the William J. Hughes Tech Center (WJHTC). A vendor that has both Class One and Class Two users will be allowed to get both feeds from WJHTC; these vendors are obligated by the Memorandum of Agreement signed with the FAA to provide the undelayed feed only to Class One users. The FAA requires that a vendor that receives an undelayed feed be audited annually at the vendor's expense to verify that the vendor is providing undelayed data only to Class One users.

The FAA retains the option of changing the amount of delay.

4.6.1.1 How Can a Vendor Get the Undelayed Feed

By default, a vendor receives the delayed feed. That is, unless a vendor has been given explicit permission to get the undelayed feed, WJHTC will only allow that vendor to register to get the delayed feed. If a vendor has a Class One customer and wants to get approval to register for the undelayed feed, it must successfully complete a full audit conducted by a Certified Public Accountant (CPA). Once the completed audit is on file at the FAA for the vendor, the WJHTC will allow it to register to receive the undelayed feed.

If at any time the FAA discovers that a vendor is sending undelayed data to an organization that is not a class one user, then the FAA has the option of immediately cutting off that vendor's access to the feed that contains undelayed data; that vendor would then only be allowed to access the delayed feed.

5 Structure of the XML Feed

5.1 Introduction

The XML ASDI data feed accumulates a number of messages, compresses them, puts a header on this compressed data, and then sends out this packet that consists of a header and compressed data. This structure will now be explained.

5.2 Structure of an ASDI Packet

Each data packet that is transmitted to the ASDI vendor will be comprised of a fixed-length header, followed by a variable-sized payload. Following is a description of this format.

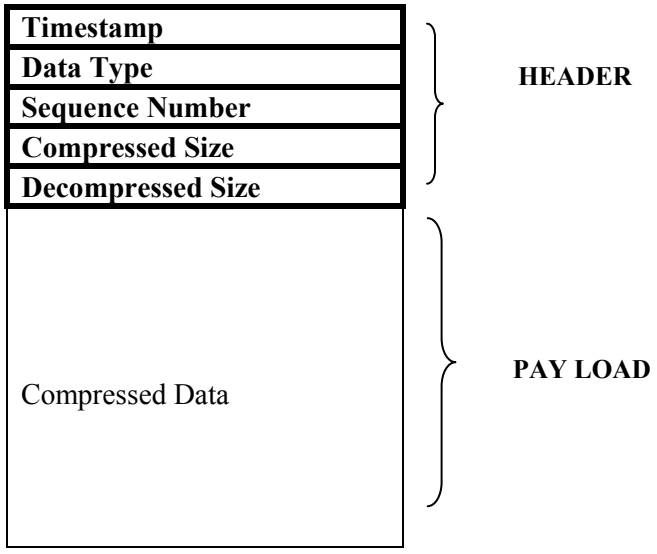


Figure 5-1. ASDI Data Packet

5.3 Structure of the Header

The header consists of the five fields shown in the figure above. The following data structure, written in C, describes the components of this header.

```
struct typedef xml_header_t
{
  char timestamp[16];
  int data_type;
  int sequence_number;
  int compressed_size;
  int decompressed_size;
} xml_header_t;
```

The timestamp header component is 16 bytes long, and the other four fields are 4 bytes long. Therefore, the header is always 32 bytes long. The vendor's client software, therefore, always needs to read exactly 32 bytes to read a header.

The **timestamp header** component is a null terminated ASCII character array in yyyymmddhhmmss format that shows when the packet was transmitted by the ASDI server. 16 bytes are allocated to the timestamp to assure that all header components are word-aligned. The first fourteen bytes of this field is the timestamp; the fifteenth byte is a null character; the last byte is an blank character.

All other components of the header are binary data, represented by 4-byte integer values. These components are converted to network format before being sent to the vendors.

There are two types of packets that are sent. The most common packet type contains flight data messages in XML format. The other packet type is a heartbeat message that indicates to the vendor client software that the ASDI Server is alive. This allows the client software to distinguish the case where the server is down from the case where the server is up but receiving no data from TFMS. The **Data Type** field indicates which type of packet is being sent by using the following values in the data_type field:

```
#define HeartBeatDataType   1
#define XMLDataType   2
```

Each packet is given a **sequence number**. The vendor software can, if desired, keep track of these sequence numbers to see if all packets are being received. The valid range for the sequence number is 0 through 100000, inclusive. However, note that the sequence number is set to 0 only at initial program startup. When the sequence number value reaches 100000, the sequence number is reset to 1. This provides a method for the vendor software to detect when the server has restarted.

As mentioned, this structure will be converted to network format before it is sent to the ASDI vendors.

The **compressed size** gives the size of the compressed data in the payload. This tells the client software how big a buffer is needed to read the payload. The ASDI Server performs in-memory compression using the standard zlib compression library. These are the same compression routines on which the gzip utilities are built. Please refer to www.zlib.net and www.gzip.com for details.

The **decompressed size** gives the size of the data in the payload after it is decompressed. This tells the client software how big a buffer is needed to hold the payload after it is decompressed.

Once the payload is decompressed, the data is in XML. The format of the preamble and messages in the decompressed payload is covered in Section 6.

5.4 Hints to Vendors on Processing Header and Payload

Upon receipt, the ASDI vendor will need to process the header and convert it back to host format. The following is a sample of C code that could be used to perform this conversion. This sample presumes that the data has already been read into buff from the data stream.

```
format_header(char *buff, xml_header_t *header)
{
   char *ptr;

  /* Unpack data stream */
  ptr = buff;
  strcpy(header->timestamp, ptr);
  ptr+= sizeof(header->timestamp);
  memcpy(header->data_type, sizeof(header->data_type));
  ptr+= sizeof(header->data_type);
  memcpy(header->sequence_number, sizeof(header->sequence_number));
  ptr+= sizeof(header->sequence_number);
  memcpy(header->compressed_size, sizeof(header->compressed_size)):
  ptr+= sizeof(header->compressed_size);
  memcpy(header->decompressed_size, sizeof(header->decompressed_size));

  /* Convert to host format */
  header->data_type = ntohl(header->data_type);
  header->sequence_number = ntohl(header->sequence_number);
  header->compressed_size = ntohl(header->compressed_size);
  header->decompressed_size = ntohl(header->decompressed_size);
}
```

Once the header is processed, the vendor would then read the compressed payload, uncompress it, and process the XML data. The sequence number in the header would be used to determine if any data packets were lost during processing.

6 ASDI XML Message Formats

Providing this data in XML format has several advantages to the clients of this feed, mainly validated machine readable data. XML is a well established industry standard and as such tutorials, reference material, documentation, development and test tools and support are readily available.

This section mainly addresses the structure of the XML schemas and formats of the messages populated with those schemas. And to a certain extent will provide a basis of understanding for assisting vendors and clients in making their software changes.

The process of connecting to the feed and the structure the data packets sent over the feed were discussed in previous sections.

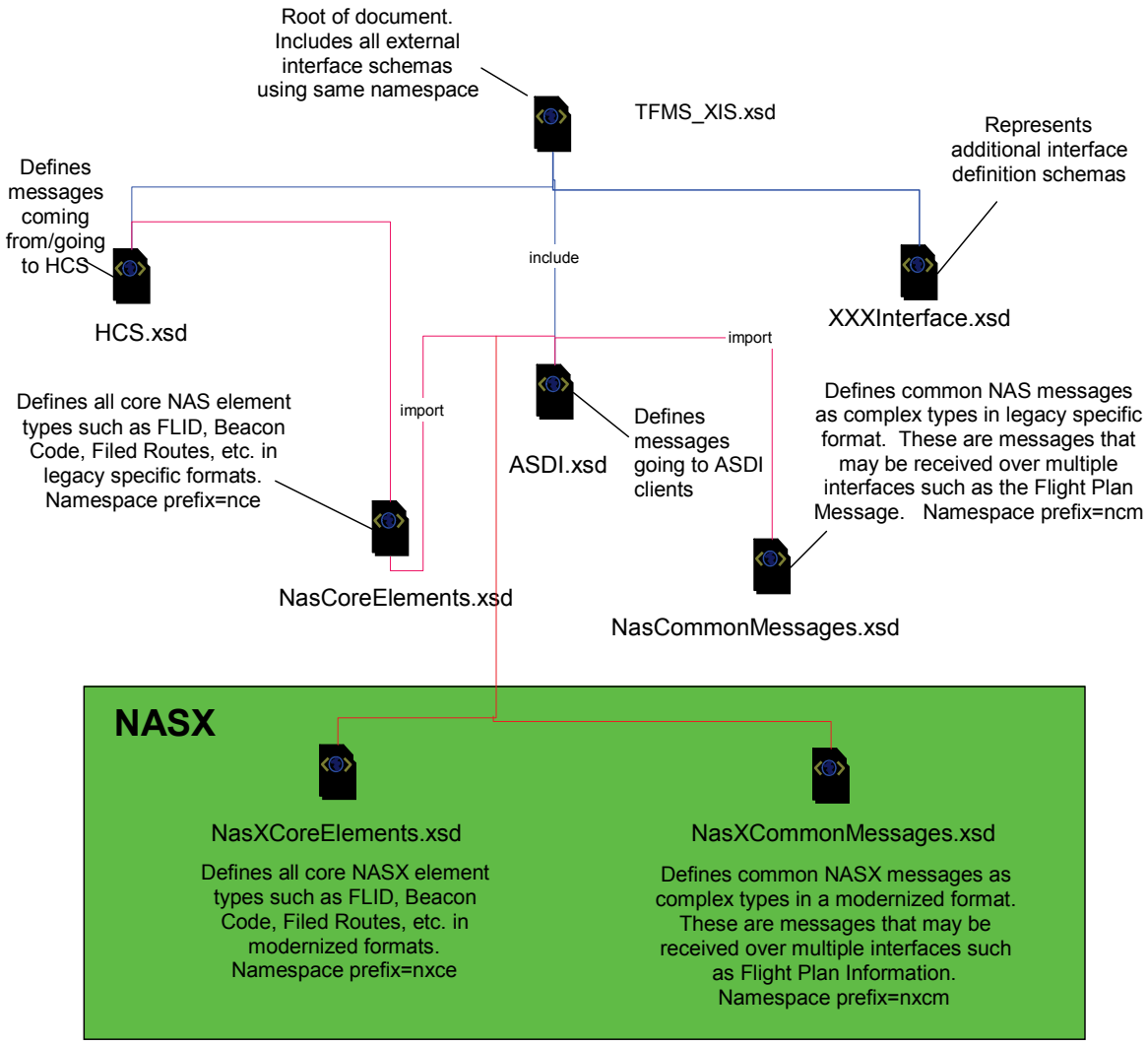
6.1 The Design of the ASDI Schema

The ASDI schema was designed and developed as part of the TFM Modernization (TFM-M) effort. The approach taken was to develop a comprehensive XML schema that is descriptive and extensible and to include all data and types that is available from the TFM-M system. The schema design used a hierarchal approach starting with a root schema representing the namespaces (schemas) needed to interpret the data content of the XML feed. Lower level schemas are then imported into this root level schema as needed by specific XML data feeds. The next level of the schema design hierarchy contains the specific interfaces or data feeds providing some uniqueness in content, for example the ASDI data feed. This level of the schema design represents the external client/vendor interfaces for both incoming and outgoing data. The lowest level of the schema design hierarchy contains commons elements, messages and types shared by higher level schemas, such as ASDI.

As the TFM-M development effort progresses other parts of the TFM-M schema will be utilized, however for the purpose of this ICD the focus will be on the elements, attributes and types required for the current release of the TFMS ASDI XML data feed.

Figure1 illustrates the hierarchal schema design used to develop the XML schemas for TFM-M and the ASDI XML data feed.

Figure 6-1. TFM-M XML Schema Design



6.1.1 Required ASDI Schema Definition Files and Locations

The XML developed for the ASDI data feed is defined by XML Schema Definition (XSD) files. These XSD files are used to describe and validate the elements used in the XML documents sent over the ASDI data feed. The following schema definition files are needed to validate, parse and process ASDI XML documents, they are;

1. **TFMS_XIS.xsd** – This is the root TFMS schema definition which includes all external interface schemas using the same name space.

2. **ASDI.xsd** – This is the schema definition used to represent all inbound and outbound data transmitted over the ASDI interface.

3. **MessageMetaData.xsd** – This is the schema definition used for all TFMS external interface data to represent common header type information.

4. **NasXCommonMessages.xsd** – This schema definition describes common messages sent and received over multiple TFMS interfaces such as ASDI. These messages are built from the data types defined in the NasXCoreElements schema which make use of standard XML schema data types defined by the World Wide Web Consortium (W3C).

5. **NasXCoreElements.xsd** – This schema definition describes all common element types used by NasXCommonMessages.xsd.

These XSD files were developed as part of the TFM-M program and as such contain additional element/attribute definitions supporting multiple TFM-M data interfaces and in some cases data is represented that will be available in future TFM-M releases. Appendix B provides a reference to identify the elements/attributes not used in the initial TFMS ASDI XML data feed.

The ASDI XML documents formed using these schema definitions are contained in the compressed payload portion of the ASDI data packet, this compressed payload will contain multiple batched messages as discussed in prior sections.

6.2 Structure of an ASDI XML Document

As discussed in a prior section an ASDI data packet contains two parts, an uncompressed fixed length ASCII header and a variable length compressed "payload" of data. The ASCII header is described in Section 5. The compressed payload portion of the ASDI data packet contains the XML formatted messages. This portion of the data packet must be uncompressed prior to processing ASDI XML messages.

Once uncompressed, the XML payload is comprised of a document preamble followed by multiple ASDI XML formatted messages. The number of messages

following the preamble is determined by a configuration value established prior to activating the ASDI feed.

6.2.1 ASDI XML Preamble

The preamble included at the beginning of every ASDI XML document lists the namespaces used in forming the ASDI messages contained in the document. The following is an example of the ASDI XML preamble segment of the document.

```
<?xml version="1.0"?>
<asdiOutput xmlns="http://tfm.faa.gov/tfms/TFMS_XIS"
xmlns:nxce="http://tfm.faa.gov/tfms/NasXCoreElements"
xmlns:mmd="http://tfm.faa.gov/tfms/MessageMetaData"
xmlns:nxcm="http://tfm.faa.gov/tfms/NasXCommonMessages"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://tfm.faa.gov/tfms/TFMS_XIS
http://www.fly.faa.gov/ASDI/asdidocs/TFMS_XIS.xsd"
timestamp="2007-02-05T21:02:24.0Z">
```

6.2.2 ASDI XML Messages

Appendix A provides actual sample ASDI messages in the current format and the corresponding XML format. These sample messages are provided as reference to help guide vendors/users with writing software to parse and validate the ASDI XML documents.

A larger sampling of ASDI XML documents is available at:

http://www.fly.faa.gov/ASDI/asdi.html

6.3 Development, Test and Validation tools

The XML and XSD files developed for the ASDI data feed adhere to standards and recommendations defined by the W3C, any XML development tool based on these standards and recommendations can be used for developing software to process ASDI XML documents and messages.

However the recommended tool for developing and validating ASDI XML documents and XSD schema definitions is Altova® XMLSpy®. The version currently in use by the TFMS development team is; Enterprise Edition version 2006 rel. 3 sp1.

Additionally, the Java programming language provides a number of built-in Java classes that can be used by ASDI clients to validate an ASDI XML document from within Java source code. An example of how to use these Java classes for validation is available at:

http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/validation/package-summary.html

There are two levels of ASDI XML document validation. The first level of validation is the determination as to whether or not there are any errors in the raw NAS input received by the ASDI XML process. If such an error is detected, an error will be logged and the message will not be sent out in the ASDI XML feed. For example, suppose a NAS message is received containing an invalid value for MACH speed, as specified by the schema. In the case the ASDI XML process would detect this error, log the message and not send it out in the feed.

The second level of validation is the detection of any errors relative to the construction and format of the document. This level of validation ensures that the document was constructed in the format defined in the various XSD schema definition files. An ASDI client should never receive an ASDI XML document that fails this level of validation as that would imply a software bug in the ASDI distribution software that constructed the document.

6.4 Hints to Vendors to help parse ASDI XML documents

The Java programming language provides a built-in Java class (the SAXParser) that ASDI clients can utilize from within Java source code to parse individual NAS data elements from an ASDI XML document. The SAXParser API can be found at:
http://java.sun.com/j2se/1.5.0/docs/api/javax/xml/parsers/SAXParser.html

And a SAXParser tutorial is available at:
http://java.sun.com/webservices/jaxp/dist/1.1/docs/tutorial/sax/index.html

The Java programming language also provides a framework (JAXB) that enables a user to create a Java object from an XML document resulting in a Java object representation of the data contained in the XML document. JAXB information is available at:
http://java.sun.com/webservices/jaxb/

A freely available C language source library that provides functionality similar to the Java SAXParser is available at:
http://directory.fsf.org/expat.html

A freely available C++ language source library that provides XML validation and parsing functionality is available at:
http://xml.apache.org/xerces-c/

Appendix A Sample ASDI Messages

This section contains one example of each type of message in the XML format. The next section shows these same messages in the new XML format.

A.1 ASDI Messages in the Original Format (This section has been removed)

(This section has been removed)

A.2 ASDI Messages in XML Format

This subsection contains an uncompressed payload from one XML data packet that contains the preamble to a packet of messages and then contains one sample of each type of message. For a much longer collection of sample messages, refer to the file on the website given in Section 1.3.

```
<?xml version="1.0"?>
<asdiOutput xmlns="http://tfm.faa.gov/tfms/TFMS_XIS"
xmlns:nxce="http://tfm.faa.gov/tfms/NasXCoreElements"
xmlns:mmd="http://tfm.faa.gov/tfms/MessageMetaData"
xmlns:nxcm="http://tfm.faa.gov/tfms/NasXCommonMessages"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://tfm.faa.gov/tfms/TFMS_XIS TFMS_XIS.xsd"
timestamp="2007-02-08T20:28:49.0Z">
  <asdiMessage sourceFacility="KZME" sourceTimeStamp="2007-01-22T21:59:42.0Z">
    <flightPlanAmendmentInformation>
      <nxcm:qualifiedAircraftId>
        <nxce:aircraftId>AAL1765</nxce:aircraftId>
        <nxce:departurePoint>
          <nxce:fix>
            <nxce:namedFix>STL</nxce:namedFix>
          </nxce:fix>
        </nxce:departurePoint>
        <nxce:arrivalPoint>
          <nxce:fix>
            <nxce:namedFix>DAL</nxce:namedFix>
          </nxce:fix>
        </nxce:arrivalPoint>
      </nxcm:qualifiedAircraftId>
      <nxcm:amendmentData>
        <nxcm:newCoordinationPoint>
          <nxce:latLong>
            <nxce:latitude>
              <nxce:latitudeDMS degrees="36" minutes="42" direction="NORTH"/>
```

```
            </nxce:latitude>
            <nxce:longitude>
              <nxce:longitudeDMS degrees="092" minutes="52" direction="WEST"/>
            </nxce:longitude>
          </nxce:latLong>
        </nxcm:newCoordinationPoint>
        <nxcm:newCoordinationTime type="ESTIMATED">2007-02-
08T21:59:00.0Z</nxcm:newCoordinationTime>
        <nxcm:newRouteOfFlight
legacyFormat="STL./.RZC062066..FINGR.FINGR3.DAL"/>
      </nxcm:amendmentData>
    </flightPlanAmendmentInformation>
  </asdiMessage>
  <asdiMessage sourceFacility="KZME" sourceTimeStamp="2007-01-22T21:59:43.0Z">
    <trackInformation>
      <nxcm:aircraftId>N51CT</nxcm:aircraftId>
      <nxcm:computerId>
        <nxce:idNumber>565</nxce:idNumber>
      </nxcm:computerId>
      <nxcm:speed>197</nxcm:speed>
      <nxcm:reportedAltitude>
        <nxce:assignedAltitude>
          <nxce:simpleAltitude>220C</nxce:simpleAltitude>
        </nxce:assignedAltitude>
      </nxcm:reportedAltitude>
      <nxcm:position>
        <nxce:latitude>
          <nxce:latitudeDMS degrees="34" minutes="46" direction="NORTH"/>
        </nxce:latitude>
        <nxce:longitude>
          <nxce:longitudeDMS degrees="085" minutes="02" direction="WEST"/>
        </nxce:longitude>
      </nxcm:position>
    </trackInformation>
  </asdiMessage>
  <asdiMessage sourceFacility="KZME" sourceTimeStamp="2007-01-22T21:59:52.0Z">
    <flightPlanInformation>
      <nxcm:aircraftId>SWA158</nxcm:aircraftId>
      <nxcm:computerId>
        <nxce:idNumber>913</nxce:idNumber>
      </nxcm:computerId>
      <nxcm:flightAircraftSpecs equipmentQualifier="J">B733</nxcm:flightAircraftSpecs>
      <nxcm:speed>
        <nxce:filedTrueAirSpeed>0430</nxce:filedTrueAirSpeed>
      </nxcm:speed>
      <nxcm:coordinationPoint>
        <nxce:namedFix>BNA</nxce:namedFix>
      </nxcm:coordinationPoint>
      <nxcm:coordinationTime type="PROPOSED">2007-02-
08T23:56:00.0Z</nxcm:coordinationTime>
      <nxcm:altitude>
```

```
        <nxce:requestedAltitude>
          <nxce:simpleAltitude>300</nxce:simpleAltitude>
        </nxce:requestedAltitude>
      </nxcm:altitude>
        <nxcm:routeOfFlight
legacyFormat="BNA..BNA.J46.ARG..SGF.TYGER5.MCI/0127"/>
      </flightPlanInformation>
    </asdiMessage>
    <asdiMessage sourceFacility="KZME" sourceTimeStamp="2007-01-22T21:59:54.0Z">
      <boundaryCrossingUpdate>
        <nxcm:aircraftId>TRS145</nxcm:aircraftId>
        <nxcm:flightAircraftSpecs
equipmentQualifier="Q">B712</nxcm:flightAircraftSpecs>
        <nxcm:speed>
          <nxce:filedTrueAirSpeed>0453</nxce:filedTrueAirSpeed>
        </nxcm:speed>
        <nxcm:boundaryPosition boundaryCrossingTime="2007-02-08T22:06:00.0Z">
          <nxce:latitude>
            <nxce:latitudeDMS degrees="37" minutes="18" direction="NORTH"/>
          </nxce:latitude>
          <nxce:longitude>
            <nxce:longitudeDMS degrees="086" minutes="10" direction="WEST"/>
          </nxce:longitude>
        </nxcm:boundaryPosition>
        <nxcm:reportedAltitude>
          <nxce:assignedAltitude>
            <nxce:simpleAltitude>370</nxce:simpleAltitude>
          </nxce:assignedAltitude>
        </nxcm:reportedAltitude>
        <nxcm:routeOfFlight
legacyFormat="DTW./.IIU..BWG.BWGTRANS.RMG.ERLIN2.ATL/2252"/>
      </boundaryCrossingUpdate>
    </asdiMessage>
    <asdiMessage sourceFacility="KZOA" sourceTimeStamp="2007-01-22T21:59:58.0Z">
      <beaconCodeInformation>
        <nxcm:qualifiedAircraftId>
          <nxce:aircraftId>SWA1953</nxce:aircraftId>
          <nxce:computerId>
            <nxce:idNumber>883</nxce:idNumber>
          </nxce:computerId>
          <nxce:departurePoint>
            <nxce:fix>
              <nxce:namedFix>BUR</nxce:namedFix>
            </nxce:fix>
          </nxce:departurePoint>
          <nxce:arrivalPoint>
            <nxce:fix>
              <nxce:namedFix>SJC</nxce:namedFix>
            </nxce:fix>
          </nxce:arrivalPoint>
        </nxcm:qualifiedAircraftId>
```

```
        <nxcm:beaconCode>2046</nxcm:beaconCode>
      </beaconCodeInformation>
    </asdiMessage>
    <asdiMessage sourceFacility="KZNY" sourceTimeStamp="2007-01-22T21:59:45.0Z">
      <departureInformation>
        <nxcm:qualifiedAircraftId>
          <nxce:aircraftId>EGF875</nxce:aircraftId>
          <nxce:computerId>
            <nxce:idNumber>069</nxce:idNumber>
          </nxce:computerId>
          <nxce:departurePoint>
            <nxce:fix>
              <nxce:namedFix>LGA</nxce:namedFix>
            </nxce:fix>
          </nxce:departurePoint>
          <nxce:arrivalPoint>
            <nxce:fix>
              <nxce:namedFix>CLE</nxce:namedFix>
            </nxce:fix>
          </nxce:arrivalPoint>
        </nxcm:qualifiedAircraftId>
        <nxcm:flightAircraftSpecs
equipmentQualifier="Q">E135</nxcm:flightAircraftSpecs>
        <nxcm:timeOfDeparture estimated="false">2007-02-
08T22:00:00.0Z</nxcm:timeOfDeparture>
        <nxcm:timeOfArrival estimated="true">2007-02-
08T23:45:00.0Z</nxcm:timeOfArrival>
      </departureInformation>
    </asdiMessage>
    <asdiMessage sourceFacility="ETMS" sourceTimeStamp="2007-01-22T21:59:38.0Z"
trigger="FA">
      <flightManagementInformation>
        <nxcm:qualifiedAircraftId>
          <nxce:aircraftId>EVA607</nxce:aircraftId>
          <nxce:computerId>
            <nxce:idNumber>897</nxce:idNumber>
          </nxce:computerId>
          <nxce:departurePoint facilityId="ZLA">
            <nxce:airport>LAX</nxce:airport>
          </nxce:departurePoint>
          <nxce:arrivalPoint>
            <nxce:airport>RCTP</nxce:airport>
          </nxce:arrivalPoint>
        </nxcm:qualifiedAircraftId>
        <nxcm:aircraftInformation>
          <nxce:flightStatus>ACTIVE</nxce:flightStatus>
          <nxce:aircraftCategory>JET</nxce:aircraftCategory>
          <nxce:userCategory>COMMERCIAL</nxce:userCategory>
        </nxcm:aircraftInformation>
```

```
        <nxcm:flightTimeData estimatedDeparture="2007-01-22T21:54:00.0Z"
estimatedArrival="2007-01-23T11:46:00.0Z" originalDeparture="2007-01-22T21:26:00.0Z"
originalArrival="2007-01-23T10:26:00.0Z"/>
        <nxcm:flightTraversalData2>
          <nxce:fix sequenceNumber="1">VTU</nxce:fix>
          <nxce:fix sequenceNumber="2">LAX</nxce:fix>
          <nxce:fix sequenceNumber="3">LIBBO</nxce:fix>
          <nxce:fix sequenceNumber="4">BRINY</nxce:fix>
          <nxce:fix sequenceNumber="5">BOARS</nxce:fix>
          <nxce:fix sequenceNumber="6">AMAKR</nxce:fix>
          <nxce:fix sequenceNumber="7">LINUZ</nxce:fix>
          <nxce:fix sequenceNumber="8">NATTE</nxce:fix>
          <nxce:fix sequenceNumber="9">ZANNG</nxce:fix>
          <nxce:fix sequenceNumber="10">51140</nxce:fix>
          <nxce:fix sequenceNumber="11">55150</nxce:fix>
          <nxce:fix sequenceNumber="12">OGGOE</nxce:fix>
          <nxce:fix sequenceNumber="13">OFORD</nxce:fix>
          <nxce:fix sequenceNumber="14">ONEIL</nxce:fix>
          <nxce:fix sequenceNumber="15">OPAKE</nxce:fix>
          <nxce:fix sequenceNumber="16">OLCOT</nxce:fix>
          <nxce:fix sequenceNumber="17">OPHET</nxce:fix>
          <nxce:fix sequenceNumber="18">OGDEN</nxce:fix>
          <nxce:fix sequenceNumber="19">OMOTO</nxce:fix>
          <nxce:fix sequenceNumber="20">OPULO</nxce:fix>
          <nxce:fix sequenceNumber="21">ONEMU</nxce:fix>
          <nxce:fix sequenceNumber="22">OATIS</nxce:fix>
          <nxce:fix sequenceNumber="23">GOC</nxce:fix>
          <nxce:fix sequenceNumber="24">NAKTU</nxce:fix>
          <nxce:fix sequenceNumber="25">XMC</nxce:fix>
          <nxce:fix sequenceNumber="26">KEC</nxce:fix>
          <nxce:fix sequenceNumber="27">APU</nxce:fix>
          <nxce:waypoint sequenceNumber="1" latitudeDecimal="2037"
longitudeDecimal="7104"/>
          <nxce:waypoint sequenceNumber="2" latitudeDecimal="2160"
longitudeDecimal="7310"/>
          <nxce:waypoint sequenceNumber="3" latitudeDecimal="2238"
longitudeDecimal="7360"/>
          <nxce:waypoint sequenceNumber="4" latitudeDecimal="2294"
longitudeDecimal="7388"/>
          <nxce:waypoint sequenceNumber="5" latitudeDecimal="2340"
longitudeDecimal="7425"/>
          <nxce:waypoint sequenceNumber="6" latitudeDecimal="2499"
longitudeDecimal="7590"/>
          <nxce:waypoint sequenceNumber="7" latitudeDecimal="2514"
longitudeDecimal="7609"/>
          <nxce:waypoint sequenceNumber="8" latitudeDecimal="2646"
longitudeDecimal="7799"/>
          <nxce:waypoint sequenceNumber="9" latitudeDecimal="3060"
longitudeDecimal="8400"/>
          <nxce:waypoint sequenceNumber="10" latitudeDecimal="3300"
longitudeDecimal="9000"/>
```

```
        <nxce:waypoint sequenceNumber="11" latitudeDecimal="3420"
longitudeDecimal="9600"/>
        <nxce:waypoint sequenceNumber="12" latitudeDecimal="3497"
longitudeDecimal="10415"/>
        <nxce:waypoint sequenceNumber="13" latitudeDecimal="3452"
longitudeDecimal="10608"/>
        <nxce:waypoint sequenceNumber="14" latitudeDecimal="3402"
longitudeDecimal="-10800"/>
        <nxce:waypoint sequenceNumber="15" latitudeDecimal="3252"
longitudeDecimal="-10361"/>
        <nxce:waypoint sequenceNumber="16" latitudeDecimal="3180"
longitudeDecimal="-10194"/>
        <nxce:waypoint sequenceNumber="17" latitudeDecimal="3086"
longitudeDecimal="-9933"/>
        <nxce:waypoint sequenceNumber="18" latitudeDecimal="3058"
longitudeDecimal="-9863"/>
        <nxce:waypoint sequenceNumber="19" latitudeDecimal="2969"
longitudeDecimal="-9662"/>
        <nxce:waypoint sequenceNumber="20" latitudeDecimal="2940"
longitudeDecimal="-9601"/>
        <nxce:waypoint sequenceNumber="21" latitudeDecimal="2732"
longitudeDecimal="-9223"/>
        <nxce:waypoint sequenceNumber="22" latitudeDecimal="2504"
longitudeDecimal="-8893"/>
        <nxce:waypoint sequenceNumber="23" latitudeDecimal="2269"
longitudeDecimal="-8612"/>
        <nxce:waypoint sequenceNumber="24" latitudeDecimal="2205"
longitudeDecimal="-8421"/>
        <nxce:waypoint sequenceNumber="25" latitudeDecimal="2128"
longitudeDecimal="-8248"/>
        <nxce:waypoint sequenceNumber="26" latitudeDecimal="2082"
longitudeDecimal="-8218"/>
        <nxce:waypoint sequenceNumber="27" latitudeDecimal="2007"
longitudeDecimal="-8148"/>
        <nxce:waypoint sequenceNumber="28" latitudeDecimal="1511"
longitudeDecimal="-7291"/>
        <nxce:waypoint sequenceNumber="29" latitudeDecimal="1505"
longitudeDecimal="-7273"/>
        <nxce:airway sequenceNumber="1">V299</nxce:airway>
        <nxce:airway sequenceNumber="2">G469</nxce:airway>
        <nxce:airway sequenceNumber="3">R591</nxce:airway>
        <nxce:airway sequenceNumber="4">R341</nxce:airway>
        <nxce:airway sequenceNumber="5">R580</nxce:airway>
        <nxce:center sequenceNumber="1">ZLA</nxce:center>
        <nxce:center sequenceNumber="2">ZOA</nxce:center>
        <nxce:center sequenceNumber="3">ZSE</nxce:center>
        <nxce:center sequenceNumber="4">ZAN</nxce:center>
        <nxce:center sequenceNumber="5">ZPA</nxce:center>
        <nxce:sector sequenceNumber="1">ZLALAX</nxce:sector>
      </nxcm:flightTraversalData2>
```

```
        <nxcm:routeOfFlight
legacyFormat="KLAX./.RZS122061..VTU200020..LIBBO..BRINY..BOARS..AMAKR..LI
NUZ..NATTE..ZANNG..5100N/14000W..5500N/15000W..5700N/16000W..OGGOE..OFO
RD..ONEIL..OPAKE..OLCOT..OPHET..OGDEN..OMOTO.R580.OATIS.OTR3.GOC.W1
8.NAKTU.V58.XMC.V52.KEC.A1.APU.AU1A.RCTP/1220"/>
      </flightManagementInformation>
    </asdiMessage>
    <asdiMessage sourceFacility="KOOA" sourceTimeStamp="2007-01-22T21:59:48.0Z">
      <oceanicReport>
        <nxcm:qualifiedAircraftId>
          <nxce:aircraftId>UAL893</nxce:aircraftId>
          <nxce:departurePoint>
            <nxce:airport>KSFO</nxce:airport>
          </nxce:departurePoint>
          <nxce:arrivalPoint>
            <nxce:airport>RKSI</nxce:airport>
          </nxce:arrivalPoint>
        </nxcm:qualifiedAircraftId>
        <nxcm:speed>471</nxcm:speed>
        <nxcm:plannedPositionData planNumber="1">
          <nxce:position>
            <nxce:latitude>
              <nxce:latitudeDMS degrees="42" minutes="00" direction="NORTH"/>
            </nxce:latitude>
            <nxce:longitude>
              <nxce:longitudeDMS degrees="130" minutes="00" direction="WEST"/>
            </nxce:longitude>
          </nxce:position>
          <nxce:altitude>320</nxce:altitude>
          <nxce:time>2007-01-22T22:21:00.0Z</nxce:time>
        </nxcm:plannedPositionData>
        <nxcm:plannedPositionData planNumber="2">
          <nxce:position>
            <nxce:latitude>
              <nxce:latitudeDMS degrees="47" minutes="00" direction="NORTH"/>
            </nxce:latitude>
            <nxce:longitude>
              <nxce:longitudeDMS degrees="140" minutes="00" direction="WEST"/>
            </nxce:longitude>
          </nxce:position>
          <nxce:altitude>320</nxce:altitude>
          <nxce:time>2007-01-22T23:29:00.0Z</nxce:time>
        </nxcm:plannedPositionData>
        <nxcm:reportedPositionData>
          <nxce:position>
            <nxce:latitude>
              <nxce:latitudeDMS degrees="40" minutes="19" direction="NORTH"/>
            </nxce:latitude>
            <nxce:longitude>
              <nxce:longitudeDMS degrees="127" minutes="02" direction="WEST"/>
            </nxce:longitude>
```

```
          </nxce:position>
          <nxce:altitude>320</nxce:altitude>
          <nxce:time>2007-01-22T22:00:00.0Z</nxce:time>
        </nxcm:reportedPositionData>
      </oceanicReport>
    </asdiMessage>
    <asdiMessage sourceFacility="KZJX" sourceTimeStamp="2007-01-22T21:59:47.0Z">
      <arrivalInformation>
        <nxcm:qualifiedAircraftId>
          <nxce:aircraftId>PDT4212</nxce:aircraftId>
          <nxce:departurePoint>
            <nxce:fix>
              <nxce:namedFix>CLT</nxce:namedFix>
            </nxce:fix>
          </nxce:departurePoint>
          <nxce:arrivalPoint>
            <nxce:fix>
              <nxce:namedFix>HXD</nxce:namedFix>
            </nxce:fix>
          </nxce:arrivalPoint>
        </nxcm:qualifiedAircraftId>
        <nxcm:timeOfArrival estimated="false">2007-02-
08T22:00:00.0Z</nxcm:timeOfArrival>
      </arrivalInformation>
    </asdiMessage>
    <asdiMessage sourceFacility="KZAU" sourceTimeStamp="2007-01-22T21:59:57.0Z">
      <flightPlanCancellation>
        <nxcm:qualifiedAircraftId>
          <nxce:aircraftId>N1017L</nxce:aircraftId>
          <nxce:computerId>
            <nxce:idNumber>072</nxce:idNumber>
          </nxce:computerId>
          <nxce:departurePoint>
            <nxce:fix>
              <nxce:namedFix>MSN</nxce:namedFix>
            </nxce:fix>
          </nxce:departurePoint>
          <nxce:arrivalPoint>
            <nxce:fix>
              <nxce:namedFix>IMT</nxce:namedFix>
            </nxce:fix>
          </nxce:arrivalPoint>
        </nxcm:qualifiedAircraftId>
      </flightPlanCancellation>
    </asdiMessage>
</asdiOutput>
```

Appendix B ASDI Schema Elements

B.1 Table Descriptions

For each XSD noted above in Section 6, there is a corresponding table if that schema has any element/attribute that is either not used in the initial versions or if the definition of the element/attribute is a superset of what will actually be included in the initial versions of the ASDI XML feed. This appendix is provided as guide to vendor/users to identify areas that do not need to be coded for in the initial release. Following describes the various columns of the tables:

- **Element/Attribute** – Identifies the element/attribute.
- **Description** - Provides a high level description of meaning of the specified element/attribute. The content is largely derived from the annotation in the corresponding schema for the given element or attribute.
- **Status** - Indicates whether the specified element/attribute is not used nor has a superset definition. If the latter case, then the Status field also qualifies the superset definition.
- **Comments** - Provided for any optional comment/feedback.

B.1.1 ASDI.xsd

ASDI.xsd is the top-level schema that describes the overall organization of the XML data. The following table enumerates those components of this schema that either will not be in the initial versions of the TFMS ASDI XML Server, or qualifies the representation of the element/attribute.

Element/Attribute	Description	Status	Comments
AsdiInput	Root node for all elements received by TFMS.	Not used in the current version	
AsdiOutput	Root node for all elements sent by TFMS to ASDI clients.	Only the *timestamp* attribute of this element will be present. None of the other attributes of this element will be included.	
triggerType	Attribute of the *asdiMessage* element. Simple enumerated type used to describe what triggered an ASDI message or event.	Only included for message type *flightInformationMessageType* (i.e. RT messages). Will never be present for any other message type.	

B.1.2 NasCommon Elements.xsd

NasCommonElements.xsd is the schema that contains the high-level element descriptions; such as the description of all the fields in a given message type. The following table enumerates those components of this schema that either will not be in the initial versions of the ASDI XML Server, or qualifies the representation of the element/attribute.

Element/Attribute	Description	Status	Comments
FlightPlanInformation	Complex type describing the components of a FlightPlanInformation element (represents a FZ message).	The optional child element, *BeaconCode* is never used .	

B.1.3 NasCoreElements.xsd

NasCoreElements.xsd is the schema that contains the majority of the detailed element descriptions. The following table enumerates those components of this schema that either will not be in the initial versions of the ASDI XML Server, or qualifies the representation of the element/attribute.

Element/Attribute	Description	Status	Comments
adaptedRouteSegment	Simple type describing part of a flight's adapted route consisting of two fixes and the route between them.	Not used in the current version	
AirlineIdType	Simple type defining 3 uppercase letters that prefix an airline designation.	Not used in the current version	
airwayNameType	Simple type defining a valid format of 2-8 alphanumerics	Not used in the current version	
assignedAltitudeType	Complex type defining the altitude assigned to this flight by Air Traffic Control.	If the child element of this element is *simpleAltitude* and the optional attribute, *above* is defined, it will always be set to “true”.	
codedRouteNameType	Simple type defining a valid format of 2 to 8 alphanumerics with the exception of "XXX", "VFR" and "DVFR”.	Not used in the current version	
computerSystemIdType	Simple enumerated type used to predefine the identification of systems which the TFMS exchanges data with.	Not used in the current version	
coordinationTimeType		Optional attribute, *delayTime*, will not be used	
delayDataType	Simply type defining delay data from field 10 in the format of "/D(d)d+dd where the first two digits(hours) must not exceed 21 and the last two digits(minutes) must not exceed 59	Not used in the current version	
distanceType	Simple type defining a distance in the range of 0 thru 999	Not used in the current version	
enRouteElementType	Complex type defining an enRoute element	Not used in the current version	

Element/Attribute	Description	Status	Comments
flightClass	Simple enumerated type defining the flight class of the aircraft which is determined by the aircraft call sign that is in the ASDI feed.	Not used in the current version	
flightComputerId	Complex type used to uniquely identify a flight across systems.	This element will only be present if the CID of a flight is in the message. The only child element currently used is the *idNumber* , which represents the CID. The other two optional child elements, *facilityIdentifier* and *computerSystemId*, are not currently used.	
flightPlanRemarks	Complex type defining comments related to a flight plan	Not used in the current version	
flightTraversalData	Complex type defining a collection of airspace the referent flight traverses. Within each unit of airspace, the fixes the flight traverses as well as the position data for those fixes. If the route of flight contains an adapted route, the route name is given, and those fixes which make up the segment of the route in the given airspace is defined.	Not used in the current version	
fpaNumberType	Simple type used to identify an FPA number	Not used in the current version	
hoursMinutesTimeType	Simple type describing time format currently used in ASDI fields	Not used in the current version	
latitudeType	Complex type	Only *latitudeDMS* is currently used. *latitideDecimal* and *latitudeRadians* representations are not currently used.	
longtitudeType	Complex type	Only *longitudeDMS* is currently used. *longtitudeDecimal* and *longitudeRadians* representations are not currently used.	
loopReEntryType	Simple type defining a coded route with a re-entry option adapted; which includes an adapted entry/exit fix.	Not used in the current version	

Element/Attribute	Description	Status	Comments
qualifiedAircraftIdType	Complex type defining the combination of elements contained within a flight plan, which together can be used to more accurately identify a particular flight.	The optional attribute, *flightClass*, is not currently used. The optional attribute, *facilityId*, of the child element arrivalPoint is not currently used. The optional attribute, *facilityId*, of the child element departurePoint may only be present in the flightManagementInformationType (i.e. RT) message. The optional child element, *fix*, of the child element arrivalPoint is not currently used in an oceanicReportType message (i.e. TO). The optional child element, *fix*, of the child element departurePoint is not currently used in an oceanicReportType message (i.e. TO) or a flightManagementInformationType (i.e. RT) message. The optional child element, *airpor*t, of the child element arrivalPoint may only be present in an oceanicReportType (i.e. TO) or a flightManagementInformationType (i.e. RT) message; it will never be in any other message type. The optional child element, *airpor*t, of the child element departurelPoint may only be present in an oceanicReportType (i.e. TO) or a flightManagementInformationType (i.e. RT) message; it will never be in any other message type.	
qualifiedAirspaceType	Complex type defining the structure and constraints for describing a qualified airspace.	Not used in the current version	

Element/Attribute	Description	Status	Comments
reportedAltitudeType	Complex type defining the structure for describing the elements necessary to provide a flight's reported altitude.	Not used in the current version	
requestedAltitudeType	Complex type defining the structure for describing the elements necessary to provide a flight's requested altitude.	If the child element of this element is *simpleAltitude* and the optional attribute, *above* is defined, it will always be set to "true".	
routeOfFlightType	Complex type defining the structure for describing a flight's route of flight.	Only the attribute, *legacyFormat,* will be present. None of the other attributes or child elements are currently used.	
routeTypesType	Simple enumerated type defining the structure for describing the elements necessary to define the identity of a flight's route type.	Not used in the current version	
sectorNumberType	Simple type used to place constraints on the sector identification number.	Not used in the current version	
simpleAltitudeType		If present, the optional attribute, *above* , will always be set to "true".	
standardArrivalRouteNameType	Simple type used to place constraints on a flight's standard arrival route name.	Not used in the current version	
standardInstrumentDepartureType	Complex type defining the structure for describing a standard Instrument Departure (SID) type.	Not used in the current version	
standardInstrumentDepartureRouteNameType	Simple type defining the structure and constraints for a standard Instrument Departure (SID) Route Name type.	Not used in the current version	
visualFlightRulesType	Complex type defining the structure for describing a flight's visual flight rules type.	If present, the optional attribute, *vfrOnTop* , will always be set to "true".	