ATS and OLDI Message Parser

ATS and OLDI Message Parser API

Flight ATM Systems Ltd.

Table of Contents

List of Tables

List of Figures

Referenced Documents

Referenc	Identification	Name
[1]	Volume II, Communication Procedures including those with PANS status, Sixth Edition October 2001;	Annex 10 to the Convention on International Civil Aviation International Civil Aviation Organization, International Standards and Recommended Practices and Procedures for Air Navigation Services
[2]	ICAO DOC 4444	Procedures for Air Navigation Services, Air Traffic Management, Fourteenth Edition — 2001
[3]	DPS.ET1.ST09-STD-01-01	EUROCONTROL STANDARD DOCUMENT for ATS Data Exchange Presentation (ADEXP) Edition: 2.1 Edition Date: December 2001
[4]	URB/USD/MSG_INTF	EUROCONTROL STANDARD Flight Progress Messages Edition: 1.5 Edition Date: 31 January 2008

1 Document Overview

1.1 Identification

Product:	ATS and OLDI Message Parser
Document Name:	ATS and OLDI Message Parser API
Document Number:	ATS-MSG-API
Revision:	4.0
Revision Date:	Sunday, 30 August 2015
Document Owner:	Peter Venton – Flight ATM Systems Ltd.
File Name:	ATS_Message_Parser_API d0.4.doc

1.2 Purpose

This document provides a description of the Flight ATM Systems Ltd. ATS Message Parser and its API.

1.3 Intended Audience

This document has been produced for programmers using the Flight ATM Systems Ltd. ATS Message Parser in order that they are able to integrate the parser into their own software products. It is expected that a reader have the pre-requisite knowledge in the Java programming language and an understanding of XML and XML Schema definitions.

1.4 Scope

This document describes the ATS Message Parser API and provides a broad overview of the parser functionality. The parser overview has been provided so as to provide background information on the concepts used to implement the parser.

This document is not a requirement document and does not attempt specify topics in an atomic form.

2 Parser Overview

The parser can be configured to process any type of ADEXP message. The ICAO format messages support is fixed and cannot be modified. By default, the parser is delivered with a pre-configured set of ADEXP and ICAO messages.

2.1 Supported Messages

The ATS Message Parser is able to process ATS and OLDI messages in both ICAO and ADEXP formats. The ICAO messages supported are those defined in [2]; these are:

- ARR – Arrival
- CHG – Modification
- CNL – Flight plan cancellation
- CPL – Current flight plan
- DEP – Departure
- DLA – Delay
- EST – Estimate
- FPL – Filed flight plan
- RQP – Request flight plan
- RQS – Request supplementary flight plan
- SPL – Supplementary flight plan

In addition, EUROCONTROL specify an additional set of messages in the ICAO format:

- ACH – Change message
- AFP – ATC flight plan proposal message
- APL – ATC flight plan
- FNM – Flight notification message from Gander
- MFS – Message from an Oceanic centre

Support for messages in the ADEXP format (described in [3]) is also provided; the supported messages can be configured, however, a default set is provided pre-configured with the parser library. The configuration provided with the library is setup to support the following ATS messages in the ADEXP format:

- ACK – Acknowledge (IFPS Operational Reply Message)
- IACH – Change message
- IAFP – ATC flight plan proposal message
- IAPL – ATC flight plan
- IARR – Arrival
- ICHG – Modification
- ICNL – Flight plan cancellation
- IDEP – Departure
- IDLA – Delay
- IFPL – Filed flight plan
- IRQP – Request flight plan
- IRQS – Request supplementary flight plan
- MAN – Manual Processing (IFPS Operational Reply Message)
- REJ – Reject (IFPS Operational Reply Message)

The configuration data provided with the parser library is setup to support the following ETFMS messages in the ADEXP format:

- DES – De-Suspension Message
- ERR – Error (IFPS Operational Reply Message)
- FLS – Flight Suspension Message
- SAM – Slot Allocation Message
- SLC – Slot Cancellation Message
- SMM – Slot Missed message
- SRM – Slot Revision Message

The ATS message parser also supports the following ATFCM messages (defined in [4]) for exchange between AO's and the CFMU ETFMS:

- EFD – ETFMS Flight Data message;
- DPI – Departure Planning Information message;
- FUM – Flight Update Message;

The ATS Parser supports the following OLDI messages, the supported formats is indicated in the table as shown below:

Message Title	Description	ADEXP Format	ICAO Format
ABI	ADVANCE BOUNDARY INFORMATION	✔	✗
ACP	ACCEPTANCE	✔	✔
ACT	ACTIVATION	✔	✔
AMA	Arrival Manager	✔	✔
BFD	BASIC FLIGHT DATA	✔	✗
CDN	CO-ORDINATION	✗	✔
CFD	CHANGE TO FLIGHT DATA	✔	✗
COD	SSR CODE ASSIGNMENT	✔	✔
COF	CHANGE OF FREQUENCY	✔	✗
HOP	HAND-OVER PROPOSAL	✔	✗
INF	INFORMATION	✔	✔
LAM	LOGICAL ACKNOWLEDGEMENT MESSAGE	✔	✔
MAC	MESSAGE FOR ABROGATION OF CO-ORDINATION	✔	✔
MAS	MANUAL ASSUMPTION OF COMMUNICATIONS	✔	✗
OCM	Oceanic Message	✔	✔
PAC	PRELIMINARY ACTIVATION	✔	✔
RAP	REFERRED ACTIVATE PROPOSAL	✔	✔
REV	REVISION	✔	✔
RJC	REJECT CO-ORDINATION	✔	✔
ROF	REQUEST ON FREQUENCY	✔	✗

Message Title	Description	ADEXP Format	ICAO Format
RRV	REFERRED REVISION PROPOSAL	✔	✔
SBY	STAND-BY	✔	✔
SDM	SUPPLEMENTARY DATA	✔	✗
TIM	TRANSFER INITIATION	✔	✗
XAP	CROSSING ALTERNATE PROPOSAL	✔	✗
XCM	CROSSING CANCELLATION	✔	✗
XIN	CROSSING INTENTION NOTIFICATION	✔	✗
XRQ	CROSSING REQUEST	✔	✗

Table 1 - Supported OLDI Messages and Formats

Message Title	Description	ADEXP Format	ICAO Format
CRAM		✔	✗
UUP		✔	✗
AUP		✔	✗

Table 2 - Supported CADF Messages

2.2 Parser overview

The parser consists of two distinct parsers, one processes the ICAO format, the other, the ADEXP format messages. Upon receiving a message, arbitration software determines the message format based on the presence of an open bracket or a hyphen.

If there is a hyphen found before any open bracket then the message is flagged as an ADEXP message, if an open bracket is found before any hyphens then it is flagged as an ICAO format message. Based on this decision process the message is handed off to the appropriate parser.

The ICAO parser is implemented as a finite state machine; this machine is not configurable offline as it was decided during the design phase that the ICAO format is relatively stable and few custom messages are implemented within the industry at large in the ICAO format. However, within the software the parser is highly configurable and can be supplied with custom messages if required.

Contact Flight ATM directly should your company require custom messages in the ICAO format.

The ADEXP parser is implemented as a data driven parser with the supported messages and content defined in a set of offline configuration files. Within the ATM domain, custom messages are often implemented in the ADEXP format; hence at the design stage it was decided that a flexible approach had to be made in order to support custom message definition.

The ADEXP message descriptions are defined in four XML configuration files referred to in this document as the ***descriptor files***.

The parser produces a W3C complaint XML document with its internal structure based on the ICAO fields contained within a message. The ADEXP format contains many fields that have no ICAO equivalent; when the ADEXP parser has completed parsing, the result from the ADEXP parser is ‘copied’ into the W3C output document. The definition of the ADEXP fields to copy to the output document is also defined in the descriptor files. Configuration options exist to copy all ADEXP fields in a message to the output or individually selected fields.

Any errors detected during processing are included in the W3C XML output document.

2.3 Support for FPL 2012

The parser supports processing for messages in the FPL 2012 formats. The parser is able to determine the message 'version' (either pre or post FPL 2012) based on message content. It is important to note that there are situations where the version cannot be determined. The detected version is written out to the XML document; it is a callers responsibility to check the version and take appropriate action as required. The parser includes errors in the output relating to FPL 2012 errors.

2.4 Technical Features

The ATS message parser requires a JRE 1.8.x to run. Other technical features include:

- Written in Java ™;
- Platform independent;
- Output generated in an open standard W3C XML document;

2.5 Software Delivery

The software is delivered in zip files for windows and tar files for Unix based operating systems. The directory structure of the delivered software is as follows:

Directory	Description
lib/	All jar-files are located here
doc/	Location of the ATS Parser documentation (this file)
doc/adexp/flightatm/adexp/xml/descriptors	Root for ADEXP configuration files
doc/ICAOFDR.xsd	XML Schema describing the parser output
doc/api	Java doc for the parser interface
README.txt	Readme file
LICENSE.txt	Licensing agreement

Table 3 - Software Delivery Structure

2.6 ATS Parser Data Flow

The following figure illustrates the data flow through the parser and the association of the configuration data with the various software components.

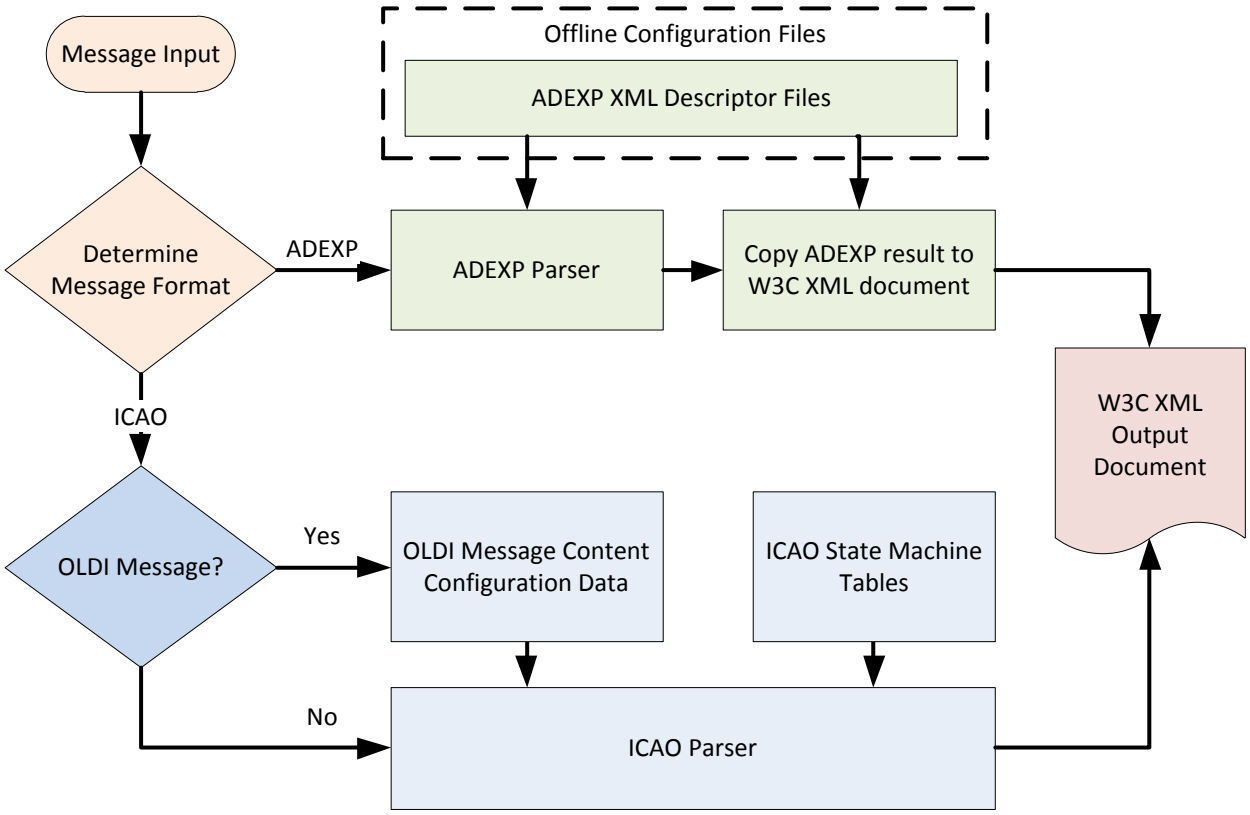

Figure 1 - ATS Message Parser Data Flow

2.7 Error Handling

Errors detected by the parser are included in the W3C XML output document; it is the caller's responsibility to check if errors are present. How this can be checked is explained in section 2.9.1 titled 'Accessing the Output Data'.

2.8 Configuration Files

There are four configuration files that define the supported ADEXP message titles and their content. These files are delivered with the parser software and can be found in the following directory:

`doc/adexp/flightatm/adexp/xml/descriptors`

These files can be used to configure custom ADEXP messages.[1] By default the ATS Parser uses the class loader to read the ADEXP configuration files. The name space used is:

`flightatm.adexp.xml.descriptors`

There is also a fifth XML file in the 'descriptors' directory that contains the text for the global errors reported by the parser, further information about these error messages can be found in section 2.8.5 titled 'Global Errors'.

The default ADEXP configuration files are packaged into the **`adexp-configuration-X.jar`**.

Setting the system property (Java –d) **`flightatm.adexp.config.dir`** tells the ATS Parser to load the ADEXP configuration files from the directory corresponding to the value of this property.

An overview of the ADEXP parser configuration files is provided in the following table, a more detailed description of each file is provided in the following sub-sections:

File Name	Description
`ADEXP_Supported_Messages.xml`	Contains a list of the ADEXP message titles and for each title, the ADEXP fields that make up the message;
`ADEXP_primary_fields.xml`	Contains a list of ADEXP primary fields that are currently specified by EUROCONTROL;
`ADEXP_subfields.xml`	Contains a list of ADEXP sub-fields that are currently specified by EUROCONTROL;
`ADEXP_auxiliary_terms.xml`	Contains a list of ADEXP auxiliary terms that are currently specified by EUROCONTROL;
`Global_Error_Text.xml`	Contains global errors reported by the parser;

Table 4 - Descriptor Files Overview

[1] Extreme caution must be taken when altering these files, as inconsistencies in the data will lead to run time exceptions being raised in the parser.

The relationship between these files is shown in the following diagram.

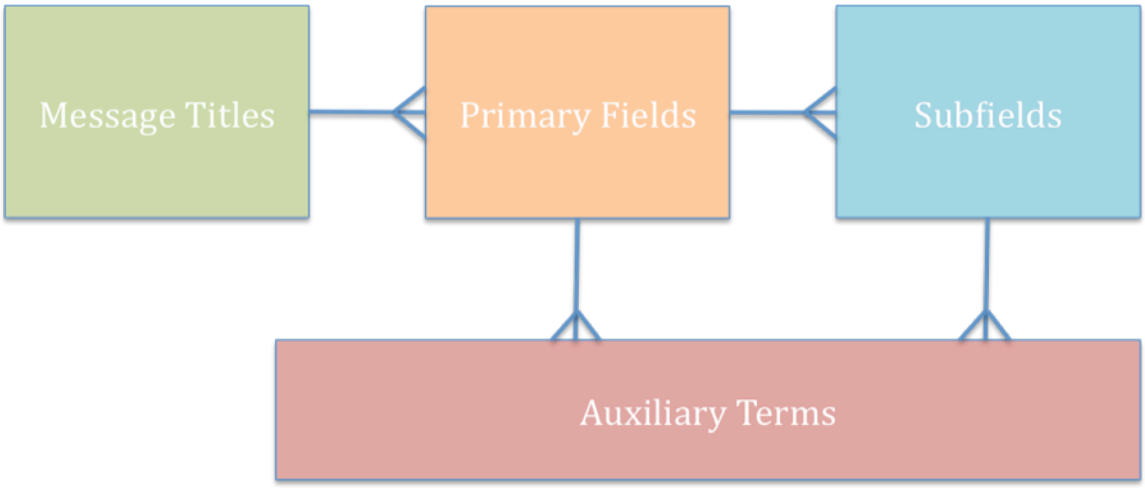

Figure 2 - Descriptor Files Relationship

2.8.1 Supported Messages File

This file specifies the message titles to be supported by the ADEXP parser. For each message title a list of primary fields is specified from which a message is comprised.

This file has the structure shown in the following figure.

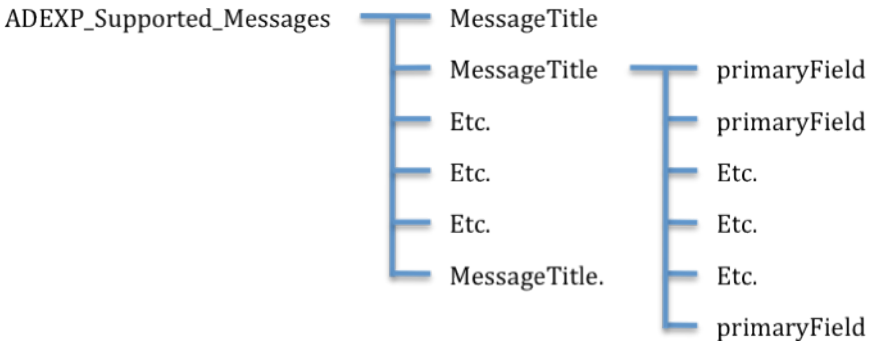

Figure 3 - Message Title Descriptor File Structure

Each of the primary fields specified must have a corresponding entry in the **`ADEXP_primary_fields.xml`** descriptor file.

The order in which the fields appear is not important, although for clarity it is recommended that they be entered in alphabetical order. Some of the messages contain many fields and it is easier to locate a field when sorted alphabetically. A number of attributes can also be specified on the XML nodes, these are described in the following table.

Node Name	Attributes	Description	Valid Values
MessageTitle	name	The name of a message title such as ‘ACK’ or ‘IFPL’.	The name of a message title as a string;
	adexpOutput	Informs the parser to output all field of the ADEXP message to the W3C XML output. If this is ‘false’ then ADEXP fields that do not have an ICAO equivalent field name, are ignored and will not appear in the W3C XML output.	True or False

Node Name	Attributes	Description	Valid Values
	isOLDImessage	Specifies the message type, true indicates that the message is an OLDI message, false indicates an ATS message. If the attribute is omitted, false is assumed and the message is flagged as an ATS message. This attribute is optional.	True or False
	icao	Provides the ICAO message title, IFPS ATS messages in ADEXP format typically have an 'I' preceding the ICAO message title, i.e. FPL is IFPL. This attribute provides a means to specify the ICAO name for any IFPS ATS message. This attribute is optional.	An ICAO message title
	oldiversion	Specifies the OLDI version of a message. Functionality to process this attribute is currently not implemented, the attribute is reserved for later use and definition. Any OLDI message omitting this attribute is considered to be OLDI 2.x.	An OLDI version number given as 4.x.
primaryField	name	The name of a primary field; this name must be defined in the descriptor file containing the primary field descriptions. If the primary field is a primary list field the 'BEGIN' keyword must not be entered, the parser knows which primary fields are list fields, (more about this is described in the primary field file description).	Any primary field name that exists in the Primary Field descriptor file.
	min	The minimum number of occurrences that this field is allowed to appear in a message, typical values are: 0 – Indicates that the field is optional; 1 – Indicates that the field is compulsory and must appear at least once, if it is missing the parser will report an error. If the 'max' attribute is set to 1 then this implies a single compulsory occurrence of the field is required;	Integers in the range 0…n
	max	The maximum number of occurrences that this field is allowed to appear in a message, typical values are: 1 – The field cannot occur more then once, if the 'min' attribute is also set to 1 then this implies a single compulsory occurrence of the field is required; -1 – The field has no maximum occurrence limit and can appear any number of times;	Integers in the range -1, 0…n

Node Name	Attributes	Description	Valid Values
mutuallyExclusive	rule	Specifies fields that are mutually exclusive; such fields must be defined as optional and then be listed in this section as mutually exclusive. Any message definition can optionally include an additional tag called <mutuallyExlusive> with an embedded tag <rule> used to specify one or more mutually exclusive fields in a message. For example, if a field is mutually exclusive with another field, i.e. MACH | SPEED then the message definition should include: <mutuallyExlusive> <rule>SPEED | MACH</rule> </mutuallyExlusive> The <rule> tag can be repeated to specify more than one set of mutually exclusive fields. Any field specified in a <rule> tag must be specified with min=0 amd max=1 or max=-1 (i.e. as an optional field, one to 'n' times).	A logically 'or' set of primary field names.

Table 5 - Supported Messages File Nodes and Attributes

2.8.2 Primary Fields

This file specifies the primary fields supported by the ADEXP parser. The primary fields come in 3 different types:

- Basic – Key value pair fields, this is the simplest type of field;
- List – Begins with the 'BEGIN' keyword and ends with the 'END' keyword. The order of the items contained within the BEGIN/END is important and must be maintained. The fields specified within a BEGIN/END block can be any primary and/or subfields; this includes nested list fields, (recursive).
- Structured – Contains an un-ordered set of sub-fields

This file is structured in three sections to reflect the 3 different types of primary fields. The basic fields are the simplest, represented as key-value pairs in a message. These are defined in this file with one or more pointers to the auxiliary terms file. The auxiliary term file specifies the regular expression with which to parse the value part of a key-value pair; typically only one term is specified, but in some cases an 'option' is provided to cover more than one possible definition.

The list and structured fields contain pointers to the fields that they themselves contain. It is to be noted that recursive structures can be defined where a list or structured field my include another list or structured field. This implementation is according to the EUROCONTROL ADEXP specification, which does specify nested levels of field descriptions.

This file has the structure shown in the following figure.

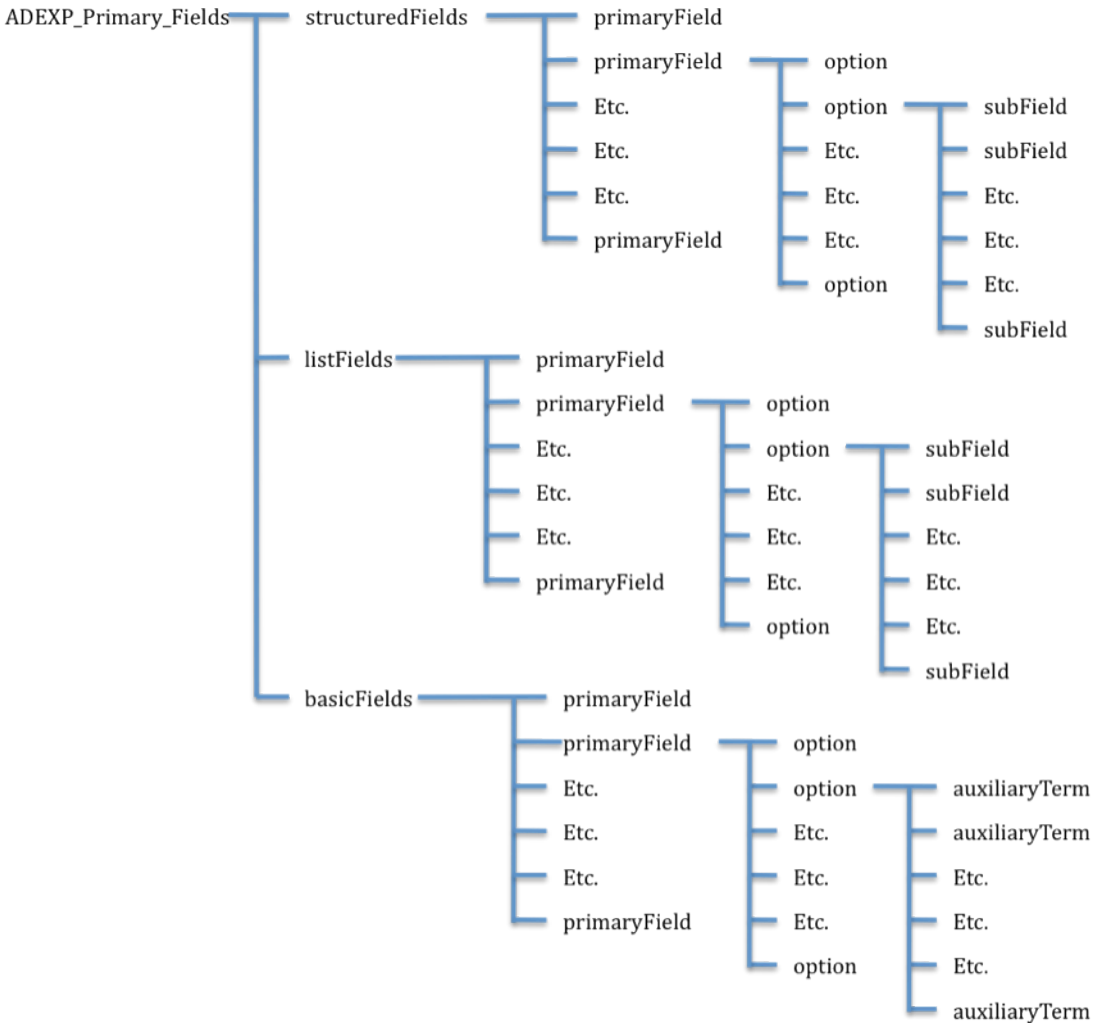

Figure 4 - Primary Field Descriptor File Structure

Each of the subfields specified must have a corresponding entry in the **`ADEXP_subfields.xml`** descriptor file. Each of the auxiliary terms specified must have a corresponding entry in the **`ADEXP_auxilliary_terms.xml`** descriptor file.

The order in which the fields appear is not important, although for clarity it is recommended that they be entered in alphabetical order to make it easier to locate a field. A number of attributes can also be specified on the XML nodes, these are described in the following table.

Node Name	Attributes	Description	Valid Values
structuredFields	None	This is a single node that contains all the primary structured fields as child nodes.	N/A
listFields	None	This is a single node that contains all the primary list fields as child nodes.	N/A
basicFields	None	This is a single node that contains all the primary basic fields as child nodes.	N/A

Node Name	Attributes	Description	Valid Values
primaryField	name	The name of a primary field; If the primary field is a primary list field the 'BEGIN' keyword must not be entered, the parser knows which primary fields are list fields because of its placement within the 'structuredFields' or 'listFields' nodes. The primary field nodes contain one or more option nodes, which in turn contain either a subfield or auxiliary term. The option nodes provide a means to: - Offer one or more alternative combination of subfields (this is particularly true for the structured fields as the ADEXP specification often describes several variations on the content of a structured field). - Offer one or more alternative auxiliary terms.	Any field name specified as: [A-Z0-9]+
	icaoname	If this attribute is present, the parser will copy the data from the primary field to the W3C output document to a tag with the name as entered for this attribute. For example, if the attribute is set to 'icaoname="F13,a"' then the value is copied to the 'fields/F13/a' node in the output document. By default the parser is delivered with this attribute set on all ADEXP primary fields that have an equivalent ICAO field to ensure that irrespective of the format of a message, that the ICAO content is contained in the output document. If no ICAO subfield is specified (the comma and last letter(s) is not present) then a node is created as a child of the 'fields' node. This can be a useful mechanism to copy data from ADEXP fields that have no equivalent ICAO field. For example, the CTOT field from an ETFMS message may be wanted in the W3C output document. If the attribute 'icaoname="CTOT"' is set for the CTOT primary field then a node 'fields/CTOT' is created in the output document containing the data received for the CTOT field. The 'icaoname' attribute has no effect when it is defined on structured or list fields. If all ADEXP fields are required in the W3C output then the 'adexpOutput' attribute defined for the message title should be used as opposed to labelling each primary field with this attribute. ***Note: If a basic primary field is modified to include the 'icaoname' attribute the XSD must be updated to ensure the schema remains valid; this is necessary as a new node is being added to the output document and the schema must also reflect this change.***	The name for an XML node that will appear as a child node of the 'fields' node.

Node Name	Attributes	Description	Valid Values
option	None	These nodes contain one or more sets of subfields (for list and structured fields) or one or more auxiliary terms for the basic fields. Each set contained within one option node represents a particular combination of subfields or auxiliary terms that are valid for the primary field. This concept has been implemented to support the multiple combinations specified by the ADEXP standard for the list and structured fields. Typically for the auxiliary terms, only one definition is defined. However, there are a few instances where a field's possible semantics cannot be defined in a single regular expression, in such cases a list of one or more possibilities can be defined.	N/A
subField	name	The name of the subfield; the subfield must be present in the Subfield descriptor file.	Any field name that exists in the Subfield descriptor file.
	optional	Indicates if a subfield is optional or compulsory, the definition is 'yes' or 'no';	'yes' or 'no' in lower case letters.
	multiple	Indicates that the subfield may occur more than once, the definition is '-1' or a positive integer value. If '-1' then the field can occur 1 to 'n' times, if a positive integer value is defined then the field must occur the exact number of times specified.	'-1' or 0..n
auxilliaryTerm	term	The name of the auxiliary term that defines the syntax and possibly the semantics of the data associated with a basic field. The name must be a name defined in the auxiliary term descriptor file.	Any auxiliary term name that exists in the Auxiliary Term descriptor file.

Table 6 - Primary Field File Nodes and Attributes

'Appendix B – ADEXP to ICAO Field Mapping' contains a complete list of the ADEXP fields mapped to the equivalent ICAO field using the 'icaoname' attribute as delivered with the parser.

2.8.3 Sub-fields

This file specifies the subfields supported by the ADEXP parser. The subfields, similar to the primary fields, come in 3 different types:

- Basic – Key value pair fields, this is the simplest type of field;
- List – Begins with the 'BEGIN' keyword and ends with the 'END' keyword. The order of the items contained within the BEGIN/END is important and must be maintained. The fields specified within a BEGIN/END block can be any subfield; this includes nested list fields, (recursive).
- Structured – Contains an un-ordered set of subfields

This file is structured in three sections to reflect the 3 different types of subfields fields. The basic fields are the simplest, represented as key-value pairs in a message. These are defined in this file with one or more pointers to the auxiliary terms file. The auxiliary term file specifies the regular expression with which to parse the value part of a key-value pair; typically only one term is specified, but in some cases an 'option' is provided to cover more than one possible definition.

The list and structured subfields contain pointers to the subfields that they themselves contain. It is to be noted that recursive structures can be defined where a list or structured subfield my include another list or structured subfield. This implementation is according to the EUROCONTROL ADEXP specification, which does specify nested levels of subfield descriptions.

This file has the structure shown in the following figure.

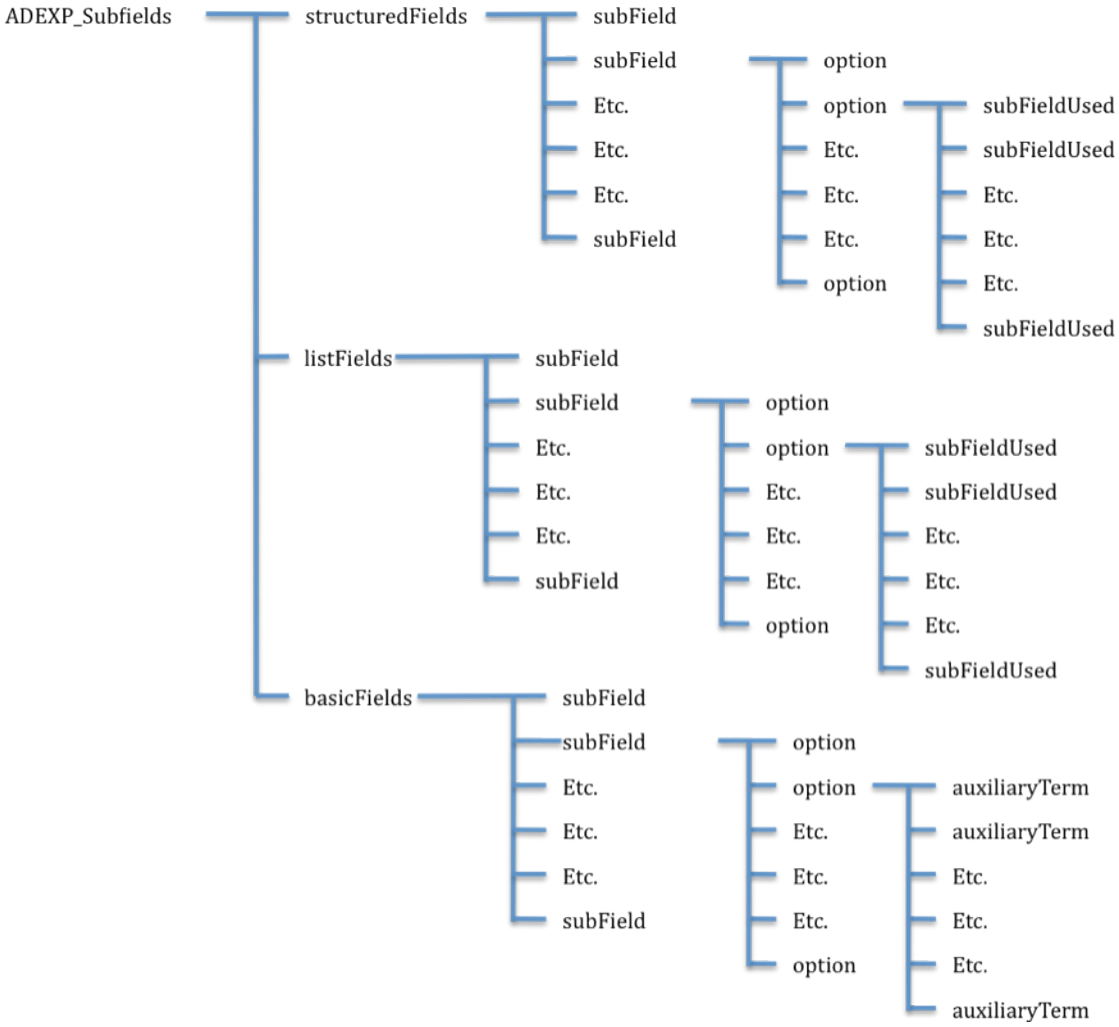

Figure 5 - Subfield Descriptor File Structure

Each of the subfields specified must have a corresponding entry in the **`ADEXP_subfields.xml`** descriptor file. Each of the auxiliary terms specified must have a corresponding entry in the **`ADEXP_auxilliary_terms.xml`** descriptor file.

The order in which the fields appear is not important, although for clarity it is recommended that they be entered in alphabetical order to make it easier to locate a field when sorted alphabetically. A number of attributes can also be specified on the XML nodes, these are described in the following table.

Node Name	Attributes	Description	Valid Values
structuredFields	None	This is a single node that contains all the structured subfields as child nodes.	N/A
listFields	None	This is a single node that contains all the list subfields as child nodes.	N/A
basicFields	None	This is a single node that contains all the basic subfields as child nodes.	N/A

Node Name	Attributes	Description	Valid Values
subField	name	The name of a subfield; this name must be defined in the descriptor file containing the subfield descriptions. If the field is a primary list subfield the 'BEGIN' keyword must not be entered, the parser knows which subfields are list subfields because of its placement within the 'structuredFields' or 'listFields' nodes. The subfield nodes contain one or more option nodes, which in turn contain either a subfield or auxiliary term. The option nodes provide a means to: - Offer one or more alternative combination of subfields (this is particularly true for the structured fields as the ADEXP specification often describes several variations on the content of a structured field). - Offer one or more alternative auxiliary terms.	Any subfield name that exists in the Subfield descriptor file given as: [A-Z0-9]+
option	None	These nodes contain one or more sets of subfields (for list and structured subfields) or one or more auxiliary terms for the basic subfields. Each set contained within one option node represents a particular combination of subfields or auxiliary terms that are valid for the subfield. This concept has been implemented to support the multiple combinations specified by the ADEXP standard for the list and structured subfields. Typically for the auxiliary terms, only one definition is defined. However, there are a few instances where a subfields possible semantics cannot be defined in a single regular expression, in such cases a list of one or more possibilities can be defined.	N/A
subFieldUsed	name	The name of the subfield; the subfield must be present in the Subfield descriptor file.	Any subfield name that exists in the Subfield descriptor file.
	optional	Indicates if a subfield is optional or compulsory, the definition is 'yes' or 'no';	'yes' or 'no' in lower case letters.
auxiliaryTerm	term	The name of the auxiliary term that defines the syntax and possibly the semantics of the data associated with a basic subfield. The name must be a name defined in the auxiliary term descriptor file.	Any auxiliary term name that exists in the Auxiliary Term descriptor file.

Table 7 - Subfield File Nodes and Attributes

2.8.4 Auxiliary Terms

The auxiliary terms file is a 'flat' file that contains a list of auxiliary terms where each term has a name and a number of other attributes. The terms in this file are referenced from the primary and subfield descriptor files. Any term referenced from the primary and subfield files must be present in the auxiliary term file. If this is not the case then the parser with fail with an exception when it tries to initialise its internal data storage.

The parser software is delivered with all auxiliary terms currently defined by EUROCONTROL in the ADEXP standards given in [3].

The nodes and attributes in the auxiliary term file are described in the following table.

Node Name	Attributes	Description	Valid Values
auxilliaryTerm	name	The name of an auxiliary term;	A name comprised of the following characters: [a-zA-Z0-9]+ They are typically all lower case letters.
	regexp	The regular expression that defines a given 'value' of a key-value pair. For example "[A-Z]{4}" is the regular expression that defines valid syntax for an aerodrome location indicator.	Any regular expression
	errmsg	The text to be displayed in an error message if a value does not match the specified regular expression, (see section 2.8.5 also).	Text describing the error.
	multi	Indicates that the multiple auxiliary terms are present, i.e. ADES requires that only a single data item/auxiliary term is present, but the ROUTE field contains a complete ICAO field 15 route description and hence contains more than one term. Such fields must be specified with 'yes'. Omitting this field is the same as specifying 'No' (the default value)	'Yes' or 'No'

Table 8 - Auxiliary Term File Node and Attribute Descriptions

2.8.4.1 Auxiliary Term FPL 2012 Impact

The FPL 2012 changes have imposed a number of limitations on the ability to express the syntax and semantics in the auxiliary term file due to the syntax complexity of a number of the FPL 2012 fields. The following auxiliary terms have additional hard coded syntax and semantic checks performed that cannot be controlled from the auxiliary terms file:

- aidequipment
- ssrequipment
- pbn

The regular expressions for these fields are deliberately lax to only check the allowable characters for each field. The ICAO syntax and semantic rules for these fields is hardcoded in the software implementation. The syntax and semantic checks are carried out according to the ICAO FPL 2012 changes.

2.8.5 Global Errors

A fifth XML file is delivered with the software that contains ‘global’ errors that may reported for any field, i.e. if a field contains too many auxiliary terms then this error is not related to any specific field or subfield, it’s a ‘global’ error that is applicable for any field or subfield.

Field specific errors relating to syntax and semantics are defined in the auxiliary term file, (described in the previous section).

The **`GlobalErrorText.xml`** file contains tag names that are referenced from the software and must not be changed. This XML file is provided in order to support internationalisation and is not foreseen to be modified; hence no further description of this files content are contained in this document, although the structure of the file is simple enough to understand if it is felt that the error messages need to be changed.

2.8.5.1 OLDI Configuration Data

When parsing OLDI ICAO format messages, the message content is typically defined between two adjacent units, hence the message structure, i.e. the ICAO fields included in a message, are dependant on the adjacent unit from which an OLDI message is received.

In order to support the OLDI message content definition, the message content for each supported adjacent unit is defined in a Java property file. When an OLDI message is received by the parser, the parser analysis ICAO field 3 to obtain the sending unit identifier. The sending unit identifier is then used to obtain the field list for the message. The field list is used by the ICAO message parser to validate the message content. OLDI ADEXP messages are treated as per the definition in the ADEXP configuration data.

The ICAO OLDI content is specified in terms of ICAO field numbers, for example F13 is ICAO field 13 'a' (ADEP) and 'b' (EOBT).

If only part of an ICAO field is required then this must be specified as such, for example F16A implies only the ADES is included as a field.

With the OLDI property file, an adjacent unit name is prefixed to the message title separated by a dot ('.'), for example BU.ACT=... where 'BU' is the adjacent unit name as it appears in the OLDI message title field 3 'b' and/or 'c'. If no adjacent unit name prefixes a message title then such an occurrence of a message title is considered the default occurrence.

Each message also contains a bilaterally agreed field 22 content that has to be specified in a similar manner. The F22 content is only used when building a message for output, the ICAO OLDI message parser processes any and all F22 amendment fields received in a message.

The default specifications contain all the compulsory fields for the ICAO format messages as specified in the OLDI 4.x standard.

The following is an example for the default and adjacent units ‘AA’ and ‘BB’ configuration data for the OLDI ACT message as it appears in the Java property file:

```
! ********* ACT START *********************
ACT=F3,F7,F13a,F14,F16a,F22
ACTf22=F9,F80,F81

AA.ACT=F3,F7,F13a,F14,F16a,F22
AA.ACTf22=F8a,F9,F15,F18,F80,F81

BB.ACT=F3,F7,F13a,F14A,F16a,F22
BB.ACTf22=F8a,F9,F80,F81
! ********* ACT END ***********************
```

One section as shown above is defined for each supported ICAO format OLDI message title. There is always a default definition and any number of specific adjacent unit definitions.

2.9 W3C XML Output Document Structure

The software is provided with an XSD document that describes, in detail, the structure and content of the W3C XML output document. This section provides a simplified overview of the output documents content without going into the level of detail provided in the XSD.

The XSD is also delivered with the parser software and can be found in the following directory:

`.doc/ICAOFDR.xsd`

The overall structure is shown in the following figure, the names used for the various nodes depicted in the diagram are the names used for the XML tags in the output document.

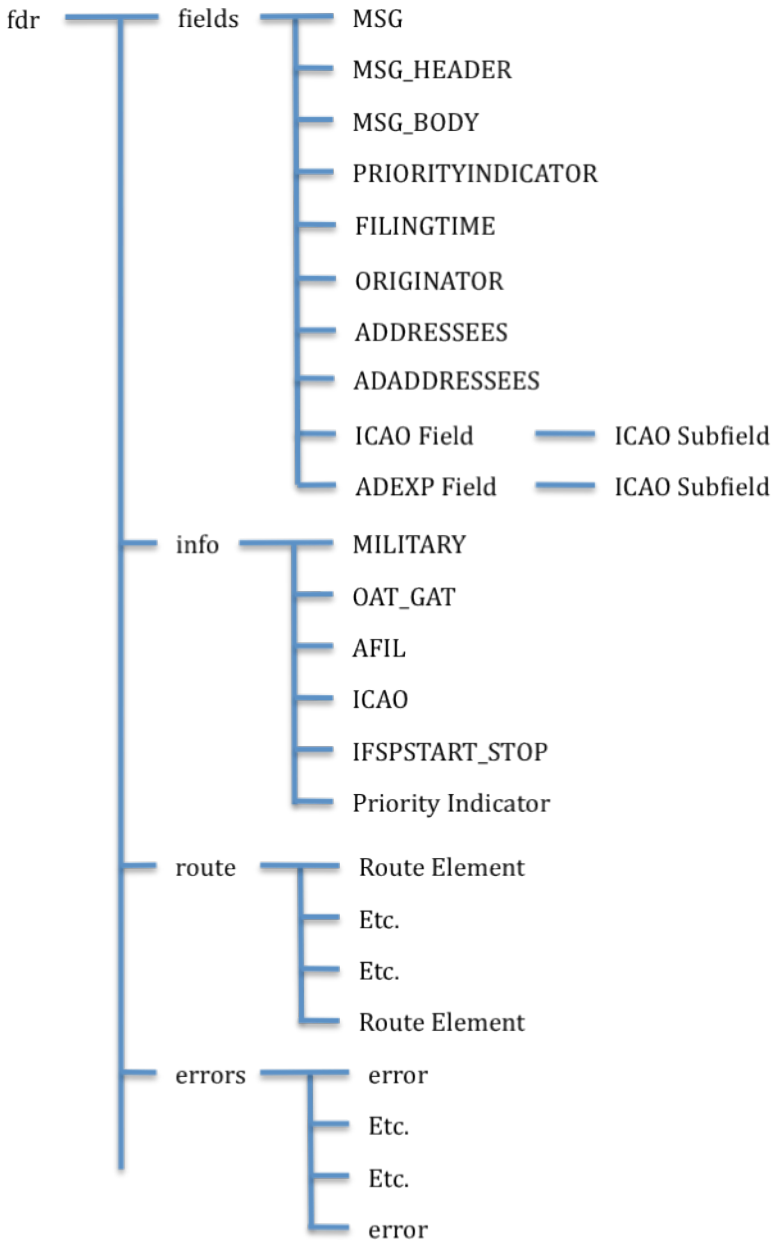

Figure 6 - W3C Output Document File Structure

The node names shown in the figure are described in the following table. One way with which to access the data in the W3C output document would be to use Xpath with the path names as described in the XSD.

Node Name	Description
fdr	The root node of the document;

Node Name	Description
fields	This node has one or more child nodes, where each child node contains the ICAO field data as extracted from a message; the node naming conventions are given in the XSD.
info	The child nodes in this node contain global data applicable to the message as a whole;
errors	This node contains one or more error nodes, one for each error generated by the parser. If no errors are present this node has no children. Each error node is in itself a complex structure that contains additional information about an error, this includes: - The token causing the error; - The start position of the erroneous token in a given field; - The end position of the erroneous token in a given field; - The ICAO name of the field with the error; - The offset of the erroneous field with respect to the start of the message This structure is not described further in this document. If an index to an erroneous token is needed then the start position in a field must be added to the field offset, this yields a zero based index of the first character of the erroneous token within the message as a whole.
route	This node contains the extracted route derived from field 15. The node ‘fields/F15’ contains the route data as supplied in field 15 of a message and never contains any child nodes as for the ‘ICAO Field’ node. The extracted route is output to a dedicated structure as two or more ‘Route Element’ child nodes to the ‘route’ node. This document does not describe this structure further here and the reader is advised to examine the XSD for more details. The sequence always starts and ends with the ADEP and ADES with any number of intermediate items in between. A ‘Route Element’ item is created for each token in field 15 apart from the speed/level token, these have their values ‘applied’ to a previous point token. Note that the parser does not contain any AIP data with which to complete the extracted route sequence; hence the extracted route is an initial extracted route that requires further processing. However, the fields set aside in this output document are sufficient to represent a complete trajectory should further processing be carried out.
ICAO Field	This node contains the actual field as extracted from a message; the syntax is always a capital ‘F’ followed by the ICAO number, e.g. F13. The data contained in these nodes is the actual data present in a field such as F13, e.g. LOWW0800. This node also contains child nodes where a given ICAO field is decomposed into its subfields. This however, does not apply for field 15, which is handled a little differently, (refer to the ‘route’ node description above).
ADEXP Field	If a basic ADEXP primary field has been configured in the descriptor file ‘Primary Field’ to be copied to the W3C output document, then the ADEXP field will also appear as a child node of the ‘fields’ nodes The node name assigned will be as specified in the ‘Primary Field’ descriptor file.

Node Name	Description
ICAO Subfield	This node contains an ICAO subfield such as the EOBT from F13. The syntax is simply the letter assigned by ICAO. For example, F13 has an ‘a’ and a ‘b’ subfield so the child nodes for F13 would be ‘a’ and ‘b’ where ‘a’ = LOWW and ‘b’ = 0800 if F13 contained LOWW0800. For field 18 and 19 the subfield names are used, e.g. if a message had a field 18 containing ‘RMK/THIS IS A REMARK’ the ‘fields’ node would have a child node called F18 with the data ‘THIS IS A REMARK’ and a child node called ‘rmk’ also containing ‘THIS IS A REMARK’. For ADEXP fields the subfield node name assigned will be as specified in the ‘Primary Field’ descriptor file.
MILITARY	Set to ‘true’ if ICAO field 8b = ‘M’, false otherwise;
OAT_GAT	Set to ‘true’ if field 15 contains the tokens OAT/GAT in the route description, if not present in field 15 then this node is not present.
AFIL	Set to ‘true’ if ICAO field 13a = ‘AFIL’, false otherwise;
IFPSTART_STOP	Set to ‘true’ if field 15 contains the tokens IFPSTOP/IFPSTART in the route description, if not present in field 15 then this node is not present.
ICAO	Set to ‘true’ if a message was processed in ICAO format and false if the message was in ADEXP format;
ADJ_UNIT_NAME	The name of the sending unit if a message contained F3b – this is used for OLDI message processing.
ATS_MESSAGE	Indicates if the message is an ATS or OLDI message, True for ATS, false if an OLDI message.
F10a_VERSION	Indicates if the ICAO field 10a is pre or post FPL 2012; the content is an enumeration (FPL_2012_VERSION) defined at the end of this table.
F10b_VERSION	Indicates if the ICAO field 10b is pre or post FPL 2012; the content is an enumeration (FPL_2012_VERSION) defined at the end of this table.
F10_VERSION	Indicates if the ICAO field 10 is pre or post FPL 2012; the content is an enumeration (FPL_2012_VERSION) defined at the end of this table.
F18_VERSION	Indicates if the ICAO field 18 is pre or post FPL 2012; the content is an enumeration (FPL_2012_VERSION) defined at the end of this table.
Priority Indicator	This node name is the priority indicator value, which can be one of the following when the priority indicator is a recognised value – ‘DD’, ‘EE’, ‘FF’, ‘GG’ or ‘SS’. If the priority indicator is unknown ‘__UNKNOWN__’ is inserted, the value is always ‘true’.

Node Name	Description
Primary_Basic_Fields	Contains one or more tags with names equal to the ADEXP basic primary field in a message. Each tag has three attributes: - data – Contains the data provided with the ADEXP field key value pair; - start – A zero based index where the ADEXP and data tokens start in the message. - end – A zero based index where the ADEXP and data tokens end in the message. This tag is only present if the message has been flagged for ADEXP output using the attribute 'adexpOutput' in the supported message title configuration file.
Primary_List_Fields	Contains a list of ADEXP list fields with the BEGIN and END field delimiting a given list structure. Within in each list field, there is one or more list, basic and/or structured fields. This tag is only present if the message has been flagged for ADEXP output using the attribute 'adexpOutput' in the supported message title configuration file.
Primary_Structured_Fields	Contains a list of ADEXP structured fields with field name delimiting a given structure. Within in each structured field, there is one or more basic and/or structured fields. This tag is only present if the message has been flagged for ADEXP output using the attribute 'adexpOutput' in the supported message title configuration file.
Undefined_for_this_Message	This tag contains a list of fields that exist in the parser configuration data, (i.e. they are 'known' to the parser) but are not defined for a given message title. The ADEXP specification states that unknown fields are to be ignored and should not generate an error. This tag is only present if the message has been flagged for ADEXP output using the attribute 'adexpOutput' in the supported message title configuration file.
Unknown_Content	This tag contains a list of fields that are not defined in the parser configuration data, (i.e. they are 'unknown' to the parser). The ADEXP specification states that unknown fields are to be ignored and should not generate an error. This tag is only present if the message has been flagged for ADEXP output using the attribute 'adexpOutput' in the supported message title configuration file.

The version of a message with respect to the FPL 2012 specification can fall into one of four categories as defined by the enumeration type FPL_2012_VERSION:

- PRE_FPL_2012 – Indicates that a field is in the format of a pre FPL 2012 message;
- POST_FPL_2012 – Indicates that a field is in the format of a post FPL 2012 message;
- PRE_AND_POST_FPL_2012 – Indicates that a field contains data that is both pre and post FPL 2012 format;
- UNKNOWN_FPL_2012 – Indicates that the FPL 2012 version cannot be determined from the data in the field, i.e. the data is allowed in both pre and post FPL 2012 messages.

It is a callers responsibility to decide on how to process messages that indicate mixed versions. When the parser detected FPL 2012 related errors, these are included in the XML output. There is

no ‘global’ flag indicating if a message is in pre or post FPL 2012 format. It is recommended that the F10_VERSION and F18_VERSION tags be used to ascertain if a message is pre or post FPL 2012. Note that one or both of these tags may be absent if they were not included in a message or an error occurs that prevents these fields from being created.

If the parser detected that there are mixed pre and post data in the F10 and F18 fields an error is raised and included in the output.

2.9.1 Accessing the Output Data

This section provides guidance on how to retrieve data from the W3C document and some additional caveats to be observed when accessing the output document.

If a field is not present in a received message then the corresponding node will be missing from the W3C output document. Hence node retrieval can return null and should be allowed for in the code accessing the W3C output document.

One method to access data in the W3C output document is to use Xpath queries. Most of the data is available in single child nodes, for example F13 ‘a’ only occurs once, hence a ‘node’ retrieval for this field can be made with an Xpath path:

```
/fdr/fields/F13/a
```

This returns a single node, which can be interrogated for its data.

A similar strategy can be used for accessing the ICAO fields themselves:

```
/fdr/fields/F13
```

This example would recover the F13 data that includes a concatenated F13 ‘a’ & ‘b’, e.g. LOWW0300.

Some of the data items in the W3C output document contain more than one child node, these are:

- The ‘errors’ node, which may contain zero or more ‘error’ nodes;
- The ‘route’ node contains at least two nodes (the ADEP and ADES for a pure VFR flight) with ‘n’ nodes that represent an extracted route if the route contains IFR route descriptions;
- The field 18 RMK and STS nodes, (there may be others if the XSD is so configured, the parser is delivered with these two F18 subfields specified as ‘0..n’ occurrences);

Some of the field 18 subfields may appear more than once. Typically the STS and RMK subfields are included more than once in received messages. When accessing the child nodes of field 18, the programmer should be aware that not all the F18 child nodes are single nodes and that code accessing the F18 child nodes may have to provide for accessing multiple nodes for some of the F18 subfields, as already noted, this includes the RMK and STS subfields.

To retrieve data from those fields containing multiple nodes, a ‘Node List’ must be recovered and iterated over to retrieve the individual child nodes.

‘Appendix C – Example W3C Access Source Code’ includes a sample Java source code file that can be used to access the various nodes in the W3C output document.

3 The API

The API for the parser is very simple with 3 public methods available:

- ***parseMessageWithHeader*** – Parses a complete message with an AFTN header;
- ***parseMessageNoHeader*** – Parses a message without the AFTN header;
- ***main*** – A convenience method provided to run the parser from the command line

3.1 Method Synopsis

3.1.1 Parse Message With Header

This method parses a complete ATS message including the AFTN message header. The message format can be either ICAO or ADEXP. An example of the expected input is shown below:

```
FF AAAAAAAA
201043 ORGNORGN
AD OOOOOOOO
(ARR-ARR002/A7654-EDDW1200-EFHK0145-ZZZZ0200 AERODROME NAME)
```

This method returns a W3C XML document containing the fields of a message as described in the XSD. Refer to the XSD document for a description of the schema.

Any errors that are detected are included in the XML document.

It is the caller's responsibility to test for errors in the XML output document by examining if the 'fields/errors' node is empty or not.

Method Signature

```
public org.w3c.dom.Document parseMessageWithHeader(final String message)
```

Input Parameters

A string containing the message;

Output Parameters

The W3C XML document containing the parsed message;

3.1.2 Parse Message Without Header

This method parses a complete ATS message without the AFTN message header. The message format can be either ICAO or ADEXP. An example of the expected input is shown below:

```
(ARR-ARR002/A7654-EDDW1200-EFHK0145-ZZZZ0200 AERODROME NAME)
```

This method returns a W3C XML document containing the fields of a message as described in the XSD. Refer to the XSD document for a description of the schema.

Any errors that are detected are included in the XML document.

It is the caller's responsibility to test for errors in the XML output document by examining if the 'fields/errors' node is empty or not.

Method Signature

```
public org.w3c.dom.Document parseMessageNoHeader(final String message)
```

Input Parameters

A string containing the message;

Output Parameters

The W3C XML document containing the parsed message;

3.1.3 Main

This method has been provided as a convenience method to parse a single message from the command line. A command line switch determines if the message is to be parsed with or without a header. The result is output to the terminal window.

Command line arguments are

- h – Invoke the help for this command;
- f <filespec> - A file containing a message to parse;
- i – If present, indicates that the message includes the AFTN header;
- x – Outputs the XSD

Method Signature

```
public static void main(String[] args)
```

Input Parameters

A space separated list of command line switches and the message to be parsed;

Output Parameters

None

Command line Invocation

```
java –classpath {all jars in ./lib} com.flightatm.ats.parser.Parser –h
```

Example

If running from directory '`foo`' with '`lib`' as a sub-directory:

```
java –cp "./lib/*" com.flightatm.ats.parser.Parser –f "testmessage.fpm"
```

4 Appendix A – Acronyms

	Acronym	Description
A		
	ABI	Advanced Boundary Information message (OLDI)
	ACH	ATC Change Message (ICAO format, CFMU special)
	ACK	EUROCONTROL Operational Reply Message indicating a message was successfully accepted by EUROCONTROL and automatically processed
	ACP	Acceptance message (OLDI)
	ACT	Activation message (OLDI)
	ADEP	Departure Aerodrome
	ADES	Destination Aerodrome
	ADEXP	ATS Data Exchange Presentation
	AFIL	Air Filed Flight Plan
	AFP	ATC Flight Plan proposal Message (ICAO)
	AFTN	Aeronautical Fixed Telecommunications Network
	AIP	Aeronautical Information Publication
	ALR	ATC Alerting Message (ICAO)
	ALTNZ	ICAO F18 subfield, this is the ADEXP equivalent to the F18 subfield ALTN;
	AMA	Arrival Manager Constraint message (OLDI)
	AO	Airline Operator
	API	Application Programming Interface
	APL	ATC Flight Plan Message (ICAO)
	ARCID	Aircraft Identification (callsign)
	ARR	ICAO ATS Arrival Message
	ATA	Actual Time of Arrival
	ATC	Air Traffic Control
	ATD	Actual Time of Departure
	ATFCM	Air Traffic Flow and Capacity Management
	ATM	Air Traffic Management
	ATOT	Actual Take Off Time
	ATS	Air Traffic Service
	AUP	Airspace Use Plan
	AWR	ICAO F18 subfield, indicates AO What-if Re-routing
B		
	BFD	Basic Flight Data (DFS version of an IFPL)
C		
	CADF	Centralised Airspace Data Function
	CDN	Coordination Message (OLDI)
	CFD	Change to Flight Data (DFS version of ICHG)
	CFL	Cleared Flight Level

	Acronym	Description
	CFMU	Central Flow Management Unit
	CHG	ICAO ATS Change Message
	CNL	ICAO ATS Cancel Message
	COD	SSR Code Assignment Message (OLDI)
	CODE	ICAO F18 subfield, specifies the aircraft code
	COF	Change of Frequency (OLDI)
	COM	ICAO F18 sub-field containing information about communication equipment
	CPL	ICAO ATS Current Flight Plan message title
	CRAM	Conditional Route Availability Message
	CTOT	Calculated Take Off Time
D		
	DAT	ICAO F18 subfield, indicating the data link capability of an aircraft.
	DB	Database
	DEP	ICAO ATS Departure Message
	DEPZ	ICAO F18 sub-field, ADEP plain text name
	DES	De-Suspension Message (ETFMS)
	DESTZ	ICAO F18 sub-field, ADES plain text name
	DLA	ICAO ATS Delay Message
	DLE	ICAO F18 subfield, indicates a delay on a point of the route;
	DPI	ETFMS AO Departure Information Message
	DPS	Data Playback System
E		
	EET	Estimated Elapsed Time
	EFD	ETFMS AO Estimated Flight Departure Message
	ELDT	Estimated Landing Time
	EOBD	Estimated Off Block Date
	EOBT	Estimated Off Block Time
	ERR	ETFMS Operational Reply Message
	EST	ICAO ATS Estimate Message
	ETFMS	Enhanced Tactical Flow Management System
	EUROCONTROL	European Organisation for the Safety of Air Navigation
F		
	FL	Flight Level
	FLS	Flight Suspension Message (ETFMS)
	FNM	ETFMS Flight Notification Message
	FUM	ETFMS Flight Update Message
G		
	GAT	General Aviation Traffic

	Acronym	Description
H		
	HOP	Hand Over Proposal
I		
	IACH	EUROCONTROL ATC Change Message
	IAFP	EUROCONTROL ATC Flight Plan Proposal Message
	IAPL	EUROCONTROL ATC Flight Plan Message
	IARR	EUROCONTROL ATS Arrival Message
	ICAO	International Civil Aviation Organisation
	ICHG	EUROCONTROL ATS Change Message
	ICNL	EUROCONTROL ATS Cancel Message
	ID	Identifier
	IDEP	EUROCONTROL ATS Departure Message
	IDLA	EUROCONTROL ATS Delay Message
	IFP	ICAO F18 sub-field, Indication of known errors within a FPL
	IFPL	EUROCONTROL ATS Flight Plan Message
	IFPLID	Unique Flight Plan Identifier originated by the CFMU
	IFPS	Integrated Initial Flight Plan Processing System
	IFPSTART	Indicates the start of part of field 15 that the IFPS will ignore for processing.
	IFPSTOP	Indicates the end of part of field 15 that the IFPS will ignore for processing.
	IFR	Instrument Flight Rules
	INF	Information Message (OLDI)
	IRQP	EUROCONTROL Request Flight Plan Message
	IRQS	EUROCONTROL Request Supplementary Information
J		
	JRE	Jave Run Time Environment
L		
	LAM	Logical Acknowledgement Message (OLDI)
	LOWW	ICAO Location Indicator for Vienna Airport
M		
	MAC	Message for Abrogation of Coordination (OLDI) or Media Access Control
	MAN	EUROCONTROL Operational Reply Message indicating a message received by EUROCONTROL required manual processing
	MAS	Manual Assumption of Communication Message (OLDI)
	MSG	Message
N		
	NAV	ICAO F18 sub-field, contains additional information about navigation equipment on board an aircraft;
O		
	OAT	Operational Air Traffic
	OCM	ATS Oceanic Message

	Acronym	Description
	OLDI	On-Line Data Interchange
	OPR	ICAO F18 sub-field, contains name of the airline operator
	ORGN	ICAO F18 sub-field, contains AFTN originator address
P		
	PAC	Preliminary Activation Message (OLDI)
	PANS	Procedures for Air Navigation Services
	PBN	ICAO F18 subfield, used to specify the ICAO code for RNAV and RNP capabilities;
	PER	ICAO F18 sub-field, contains aircraft performance data
R		
	RALT	ICAO F18 sub-field, used to specify the name of en-route alternative aerodromes;
	RAP	Referred Activate Proposal Message (OLDI)
	REG	ICAO F18 sub-field, contains aircraft registration markings
	REJ	EUROCONTROL Operational Reply Message indicating a message was rejected by EUROCONTROL and returned to sender
	REV	Revision Message (OLDI)
	RFP	ICAO F18 sub-field, repetitive flight plan indicator
	RIF	ICAO F18 sub-field, used to specify a revised route subject to clearance in flight, and terminating with the ICAO designator of the revised aerodrome of destination.
	RJC	Reject Coordination Message (OLDI)
	RMK	ICAO F18 sub-field, contains free text remarks
	ROF	Request on Frequency Message (OLDI)
	RQP	Request Flight Plan (ICAO)
	RQS	Request Supplementary Information (ICAO)
	RRV	Referred Revision Proposal Message (OLDI)
	RVR	Runway Visual Range
S		
	SAM	Slot Allocation Message (ETFMS)
	SBY	Standby Message (OLDI)
	SDM	Supplementary Data Message (OLDI)
	SEL	ICAO F18 sub-field, used to specify a SELCAL code.
	SFL	Supplementary Flight Level
	SITA	Société Internationale de Télécommunications Aéronautiques
	SLC	Slot Requirement Cancellation Message (ETFMS)
	SMM	Sector Monitor Message (RMCDE)
	SPL	ICAO ATS Supplementary Flight Plan Message
	SRC	ICAO F18 subfield, specifies the source of a message
	SRM	Slot Revision Message (ETFMS)
	SSR	Secondary Surveillance Radar
	STAYINFO	A token in ICAO field 15 specifying holding in a given airspace;
	STS	ICAO F18 sub-field, contains free text (priority flights)

	Acronym	Description
	SUR	ICAO F18 sub-field, specifies surveillance applications or capabilities not in SEQPT
T		
	TALT	ICAO F18 sub-field, contains the name of alternate take-off aerodromes;
	TFL	Coordinated entry level
	TIM	Transfer Initiation Message (OLDI)
	TM	Trajectory Management
	TYPZ	ICAO F18 sub-field, contains the type of aircraft when no ICAO code exists.
U		
	URB	User Requirement Brief
	UUP	Updated Airspace Use Plan
V		
	VFR	Visual Flight Rules
W		
	W3C	World Wide Web Consortium
	WTC	Wake Turbulence Category
X		
	XAP	Crossing Alternate Proposal Message (OLDI)
	XCM	Crossing Cancellation Message (OLDI)
	XIN	Crossing Intention Notification Message (OLDI)
	XML	Extensible Mark up Language
	XRQ	Crossing Clearance Request Message (OLDI)
	XSD	XML Schema Description
Z		

5 Appendix B – ADEXP to ICAO Field Mapping

The Flight ATM Parser is delivered with all the ADEXP fields that have an equivalent ICAO field, mapped to the relevant ICAO field. In addition the ETFMS ADEXP field CTOT and ADEXP fields IFPLID and ELDT are included as child nodes of the 'fields' node.

When mapping ADEXP fields to an ICAO field, the ICAO field numbers must be expressed as 'F[0-9]{1,2}'; the subfields must also be as used by ICAO, e.g. '[a-e]{1}'. If the fields and/or subfields are named according to a different schema they will not be copied from the ADEXP field to the ICAO output document. F18 and F19 use the equivalent subfield names as described by ICAO in all cases apart from F19, (syntax shown in the following table).

The table shown below defines:

- The ADEXP field name mapped to its ICAO equivalent field;
- The ICAO field name and subfield syntax;
- The Xpath path to use in order to access a given field;

ADEXP Field	ICAO Field	ICAO Subfield	Xpath Path
TITLE	F3	'a'	fdr/fields/F3/a
ARCID	F7	'a'	fdr/fields/F7/a
SSRCODE	F7	'c'	fdr/fields/F7/c
FLTRUL	F8	'a'	fdr/fields/F8/a
FLTTYP	F8	'b'	fdr/fields/F8/b
NBARC	F9	'a'	fdr/fields/F9/a
ARCTYP	F9	'b'	fdr/fields/F9/b
WKTRC	F9	'c'	fdr/fields/F9/c
CEQPT	F10	'a'	fdr/fields/F10/a
SEQPT	F10	'b'	fdr/fields/F10/b
ADEP	F13	'a'	fdr/fields/F13/a
ATD, ATOT, EOBT	F13	'b'	fdr/fields/F13/b
COORDATA basic subfield PTID	F14	'a'	fdr/fields/F14/a
COORDATA basic subfield TO	F14	'b'	fdr/fields/F14/b
COORDATA basic subfield STO	F14	'b'	fdr/fields/F14/b
COORDATA basic subfield TFL	F14	'c'	fdr/fields/F14/c
COORDATA basic subfield SFL	F14	'd'	fdr/fields/F14/d
ROUTE	F15	None[2]	fdr/fields/F15
ADES	F16	'a'	fdr/fields/F16/a
TTLEET	F16	'b'	fdr/fields/F16/b
ALTRNT1	F16	'ca'	fdr/fields/F16/ca
ALTRNT2	F16	'cb'	fdr/fields/F16/cb
ATA	F17	'b'	fdr/fields/F17/b
ALTNZ	F18	'altn'	fdr/fields/F18/altn
AWR	F18	'awr'	fdr/fields/F18/awr

[2] The extracted route is placed in the 'route' node – refer to the XSD for details;

ADEXP Field	ICAO Field	ICAO Subfield	Xpath Path
ARCADDR	F18	'code'	fdr/fields/F18/code
COM	F18	'com'	fdr/fields/F18/com
DAT	F18	'dat'	fdr/fields/F18/dat
DEPZ	F18	'dep'	fdr/fields/F18/dep
DESTZ	F18	'dest'	fdr/fields/F18/dest
DLE	F18	'dof'	fdr/fields/F18/dle
EETFIR	F18	'eet'	fdr/fields/F18/eet
EOBD	F18	'dof'	fdr/fields/F18/dof
EUR	F18	'eur'	fdr/fields/F18/eur
IFP	F18	'ifp'	fdr/fields/F18/ifp
NAV	F18	'nav'	fdr/fields/F18/nav
OPR	F18	'opr'	fdr/fields/F18/opr
ORGN	F18	'orgn'	fdr/fields/F18/orgn
PBN	F18	'pbn'	fdr/fields/F18/pbn
PER	F18	'per'	fdr/fields/F18/per
RALT	F18	'ralt'	fdr/fields/F18/ralt
REG	F18	'reg'	fdr/fields/F18/reg
RFP	F18	'rfp'	fdr/fields/F18/rfp
RIF	F18	'rif'	fdr/fields/F18/rif
RMK[3]	F18	'rmk'	fdr/fields/F18/rmk
RVR	F18	'rvr'	fdr/fields/F18/rvr
SEL	F18	'sel'	fdr/fields/F18/sel
SRC	F18	'src'	fdr/fields/F18/src
STAYINFO[1-9]	F18	'stayinfo[1-9]'	fdr/fields/F18stayinfo[1-9]
STS[4]	F18	'sts'	fdr/fields/F18/sts
SUR	F18	'sur'	fdr/fields/F18/sur
TALT	F18	'talt'	fdr/fields/F18/talt
TYPZ	F18	'typ'	fdr/fields/F18/typ
SPLA	F19	'a'	fdr/fields/F19/a
SPLC	F19	'c'	fdr/fields/F19/c
SPLDCAP[5]	F19	'd2'	fdr/fields/F19/d2
SPLDCOL[6]	F19	'd4'	fdr/fields/F19/d4
SPLDCOV[7]	F19	'd3'	fdr/fields/F19/d3

[3] This subfield can occur more than once;
[4] This subfield can occur more than once;
[5] This field is part of ICAO F19 'd' but is split into 4 component parts in ADEXP; if any of these subfields are found they are concatenated and 'put back together' as field 'd' and can be accessed using 'fdr/fields/F19/d'.
[6] This field is part of ICAO F19 'd' but is split into 4 component parts in ADEXP; if any of these subfields are found they are concatenated and 'put back together' as field 'd' and can be accessed using 'fdr/fields/F19/d'.

ADEXP Field	ICAO Field	ICAO Subfield	Xpath Path
SPLDNB[8]	F19	‘d1’	fdr/fields/F19/d1
SPLE	F19	‘e’	fdr/fields/F19/e
SPLJ	F19	‘j’	fdr/fields/F19/j
SPLN	F19	‘n’	fdr/fields/F19/n
SPLP	F19	‘p’	fdr/fields/F19/p
SPLR	F19	‘r’	fdr/fields/F19/r
SPLS	F19	‘s’	fdr/fields/F19/s

Table 9 - ADEXP to ICAO Field Mapping

The following table lists the ADEXP fields that have no ICAO equivalent that are also included in the W3C output file.

ADEXP Field	Xpath path
CTOT	fdr/fields/CTOT
ELDT	fdr/fields/ELDT
IFPLID	fdr/fields/IFPLID
CFL	fdr/fields/CFL

Table 10 - Additional ADEXP Fields

[7] This field is part of ICAO F19 ‘d’ but is split into 4 component parts in ADEXP; if any of these subfields are found they are concatenated and ‘put back together’ as field ‘d’ and can be accessed using ‘fdr/fields/F19/d’.
[8] This field is part of ICAO F19 ‘d’ but is split into 4 component parts in ADEXP; if any of these subfields are found they are concatenated and ‘put back together’ as field ‘d’ and can be accessed using ‘fdr/fields/F19/d’.

6 Appendix C – Example W3C Access Source Code

This section provides a simple example for accessing the W3C output document.

```
package yout.package.header.in.here;

import java.util.Vector;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathConstants;
import javax.xml.xpath.XPathExpression;
import javax.xml.xpath.XPathExpressionException;
import javax.xml.xpath.XPathFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

public class XPathHelper {

	public XPathHelper(){}

	public String getSubFieldData(Document document, String fieldName, String subfieldName) {
		return getSubFieldDataByNodeName(document, "fields", fieldName, subfieldName);
	}

	public String getFieldData(Document document, String fieldName) {
		return getFieldDataByNodeName(document, "fields", fieldName);
	}

	public String getInfoFieldData(Document document, String fieldName) {
		return getFieldDataByNodeName(document, "info", fieldName);
	}
```

```
private String getSubFieldDataByNodeName(
                              Document document, String nodeName, String fieldName, String subfieldName){
      String retVal = "";
      XPathFactory factory = XPathFactory.newInstance();
      XPath xpath = factory.newXPath();
      try {
            XPathExpression expr = xpath.compile("/fdr/" + nodeName + "/" + fieldName + "/" + subfieldName);
            Object result = expr.evaluate(document, XPathConstants.NODE);
            retVal = getTextFromTextNode((Node) result);
      } catch (XPathExpressionException e) {
            e.printStackTrace();
      }
      return retVal;
}

private String getFieldDataByNodeName(Document document, String nodeName, String fieldName) {
      String retVal = "";
      XPathFactory factory = XPathFactory.newInstance();
      XPath xpath = factory.newXPath();
      try {
            XPathExpression expr = xpath.compile("/fdr/" + nodeName + "/" + fieldName);
            Object result = expr.evaluate(document, XPathConstants.NODE);
            retVal = getTextFromTextNode((Node) result);
      } catch (XPathExpressionException e) {
            e.printStackTrace();
      }
      return retVal;
}
```

```
	private Vector<String> getErrorFieldData(Document document) {
		XPathFactory factory = XPathFactory.newInstance();
		XPath xpath = factory.newXPath();
		Vector<String> errMsgs = new Vector<String>();
		try {
			XPathExpression expr = xpath.compile("/fdr/errors/*");
			Object result = expr.evaluate(document, XPathConstants.NODESET);
			NodeList nodelist = (NodeList) result;
			for(int i=0; i<nodelist.getLength(); i++){
				errMsgs.add(getTextFromTextNode(nodelist.item(i)));
			}
		} catch (XPathExpressionException e) {
			e.printStackTrace();
		}
		return errMsgs;
	}

	private String getTextFromTextNode(Node node){
		String retVal = "";
		NodeList nodeList = node.getChildNodes();
		for(int i=0; i<nodeList.getLength(); i++){
			node = nodeList.item(i);
			if(node.getNodeType() == Node.TEXT_NODE){
				return retVal = node.getNodeValue();
			}
		}
		return retVal;
	}

}
```

7 Appendix D – XSD For W3C Output Document

This section contains the XSD used to describe the W3C output document produced by the Flight ATM Systems Ltd. ATS Message Parser.

```
<?xml version="1.0" encoding="utf-8"?>
<!--Copyright 2010 Flight ATM Systems Ltd. * * This file and its contents
      both textual and intellectual remains the * sole and exclusive property of
      Flight ATM Systems Ltd. No part of it * may be copied, or disclosed by the
      recipient to third persons, without * the prior written consent of Flight
      ATM Systems Ltd.; nor shall it be * used for any purpose other than in connection
      with an agreement or * proposed agreement with Flight ATM Systems Ltd. *
      * Flight ATM Systems Ltd. Registration number 5625816 * * http://www.flightatm.com -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified">
      <xs:complexType name="F3" mixed="true">
            <xs:all>
                  <xs:annotation>
                        <xs:documentation>F3 - Message type (Message Title), number and
                              reference data</xs:documentation>
                  </xs:annotation>
                  <xs:element name="a" type="xs:string" minOccurs="1"
                        maxOccurs="1">
                        <xs:annotation>
                              <xs:documentation>F3a - Message Title</xs:documentation>
                        </xs:annotation>
                  </xs:element>
                  <xs:element name="bpt1" type="xs:string" minOccurs="0"
                        maxOccurs="1">
                        <xs:annotation>
                              <xs:documentation>The bpt and cpt fields are rarely used and need
                                    not be part of any further processing</xs:documentation>
                              <xs:documentation>They do not hold any application related data in
                                    any case.</xs:documentation>
                        </xs:annotation>
                  </xs:element>
                  <xs:element name="bpt2" type="xs:string" minOccurs="0"
```

```
                  maxOccurs="1" />
            <xs:element name="bpt3" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="cpt1" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="cpt2" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="cpt3" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
      </xs:all>
</xs:complexType>
<xs:complexType name="F5" mixed="true">
      <xs:all>
            <xs:annotation>
                  <xs:documentation>Description of emergency</xs:documentation>
            </xs:annotation>
            <xs:element name="a" type="xs:string" minOccurs="0"
                  maxOccurs="1">                        <xs:annotation>
<xs:documentation>F5a - Phase of emergency</xs:documentation>                        </xs:annotation>
</xs:element>
            <xs:element name="b" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F5b - Originator of message</xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="c" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F5c - Nature of emergency</xs:documentation>
                  </xs:annotation>
            </xs:element>
      </xs:all>
</xs:complexType>
<xs:complexType name="F7" mixed="true">
      <xs:all>
```

```
            <xs:annotation>
                  <xs:documentation>F7 - Aircraft identification and SSR Mode and Code
                  </xs:documentation>
            </xs:annotation>
            <xs:element name="a" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F7a - Aircraft identification (Callsign)
                        </xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="b" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F7b - SSR Mode</xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="c" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F7c - SSR Code</xs:documentation>
                  </xs:annotation>
            </xs:element>
      </xs:all>
</xs:complexType>
<xs:complexType name="F8" mixed="true">
      <xs:all>
            <xs:annotation>
                  <xs:documentation>F8 - Flight rules and type of flight
                  </xs:documentation>
            </xs:annotation>
            <xs:element name="a" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F8a - Flight rules</xs:documentation>
                  </xs:annotation>
```

```
            </xs:element>
            <xs:element name="b" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F8b - Type of flight</xs:documentation>
                  </xs:annotation>
            </xs:element>
      </xs:all>
</xs:complexType>
<xs:complexType name="F9" mixed="true">
      <xs:all>
            <xs:annotation>
                  <xs:documentation>F9 - Number and type of aircraft and wake
                        turbulence category</xs:documentation>
            </xs:annotation>
            <xs:element name="a" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F9a - Number of aircraft</xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="b" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F9b - Type of aircraft</xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="c" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F9c - Wake turbulence category (WTC)
                        </xs:documentation>
                  </xs:annotation>
            </xs:element>
      </xs:all>
</xs:complexType>
```

```
<xs:complexType name="F10" mixed="true">
      <xs:all>
            <xs:annotation>
                  <xs:documentation>F10 - Equipment</xs:documentation>
            </xs:annotation>
            <xs:element name="a" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F10a - Equipment, Radio Communication, Navigation
                              and Approach Aid Equipment</xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="b" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F10b - Surveillance Equipment</xs:documentation>
                  </xs:annotation>
            </xs:element>
      </xs:all>
</xs:complexType>
<xs:complexType name="F13A" mixed="true">
      <xs:annotation>
            <xs:documentation>Abbreviated F13 that contains only F13a, used for
                  some messages where ICAO do not require an associated F13b
            </xs:documentation>
            <xs:documentation>This field may be terminated in message types CHG,
                  CNL, ARR, CPL, EST, CDN, ACP and RQS and message type RQP if the
                  estimated off-block time is not known.</xs:documentation>
      </xs:annotation>
      <xs:all>
            <xs:element name="a" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F13a - Departure aerodrome (ADEP)
                        </xs:documentation>
                  </xs:annotation>
```

```
            </xs:element>
      </xs:all>
</xs:complexType>
<xs:complexType name="F13" mixed="true">
      <xs:all>
            <xs:annotation>
                  <xs:documentation>F13 - Departure aerodrome and time
                  </xs:documentation>
            </xs:annotation>
            <xs:element name="a" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F13a - Departure aerodrome (ADEP)
                        </xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="b" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F13b - Estimated of block time (EOBT)
                        </xs:documentation>
                  </xs:annotation>
            </xs:element>
      </xs:all>
</xs:complexType>
<xs:complexType name="F14" mixed="true">
      <xs:all>
            <xs:annotation>
                  <xs:documentation>F14 - Estimate data</xs:documentation>
            </xs:annotation>
            <xs:element name="a" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F14a - Boundary point</xs:documentation>
                  </xs:annotation>
            </xs:element>
```

```
            <xs:element name="b" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F14b - Time at Boundary point</xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="c" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F14c - Cleared level at Boundary point
                        </xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="d" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F14d - Supplementary crossing data
                        </xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="e" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F14e - Crossing condition</xs:documentation>
                  </xs:annotation>
            </xs:element>
      </xs:all>
</xs:complexType>
<xs:complexType name="F15" mixed="true">
      <xs:annotation>
            <xs:documentation>F15 - Route</xs:documentation>
      </xs:annotation>
      <xs:simpleContent>
            <xs:extension base="xs:string" />
      </xs:simpleContent>
</xs:complexType>
```

```
<xs:complexType name="F16A" mixed="true">
      <xs:annotation>
            <xs:documentation>Abbreviated F16 that contains only F16a, used for
                  some messages where ICAO do not require an associated F16b
            </xs:documentation>
            <xs:documentation>This field is terminated in all message types other
                  than ALR, FPL and SPL.</xs:documentation>
      </xs:annotation>
      <xs:all>
            <xs:element name="a" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F16a - Destination aerodrome (ADES)
                        </xs:documentation>
                  </xs:annotation>
            </xs:element>
      </xs:all>
</xs:complexType>
<xs:complexType name="F16" mixed="true">
      <xs:all>
            <xs:annotation>
                  <xs:documentation>F16 - Destination aerodrome and total estimated
                        elapsed time, alternate aerodrome(s)</xs:documentation>
            </xs:annotation>
            <xs:element name="a" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F16a - Destination aerodrome (ADES)
                        </xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="b" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F16b - Total estimated elapsed time, (EET)
                        </xs:documentation>
```

```
                  </xs:annotation>
            </xs:element>
            <xs:element name="ca" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F16c - Alternate aerodrome one(ALTN1)
                        </xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="cb" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F16c - Alternate aerodrome two(ALTN2)
                        </xs:documentation>
                  </xs:annotation>
            </xs:element>
      </xs:all>
</xs:complexType>
<xs:complexType name="F17" mixed="true">
      <xs:all>
            <xs:annotation>
                  <xs:documentation>F17 - Arrival aerodrome and time
                  </xs:documentation>
            </xs:annotation>
            <xs:element name="a" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F17a - Arrival aerodrome</xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="b" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F17b - Time of arrival</xs:documentation>
                  </xs:annotation>
            </xs:element>
```

```
            <xs:element name="c" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F17c - Plain text name of arrival aerodrome
                        </xs:documentation>
                  </xs:annotation>
            </xs:element>
      </xs:all>
</xs:complexType>
<xs:complexType name="F1819subfield">
      <xs:simpleContent>
            <xs:extension base="xs:string">
                  <xs:attribute name="TYPE" type="xs:string" use="optional" />
                  <xs:attribute name="SUBTYPE" type="xs:string" use="optional" />
                  <xs:attribute name="START_INDEX" type="xs:string" use="optional" />
                  <xs:attribute name="END_INDEX" type="xs:string" use="optional" />
                  <xs:attribute name="SUBTYPE.class" type="xs:string"
                        use="optional" />
                  <xs:attribute name="TYPE.class" type="xs:string" use="optional" />
            </xs:extension>
            <!--xs:attribute name="DATASTART" type="xs:string" use="optional"/ -->
      </xs:simpleContent>
</xs:complexType>

<xs:complexType name="F18" mixed="true">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
                  <xs:documentation>F18 - Other information</xs:documentation>
                  <xs:documentation>F18 subfields can occur in any order any number of
                        times</xs:documentation>
            </xs:annotation>
            <xs:element name="altn" type="F1819subfield" />
            <xs:element name="awr" type="F1819subfield" maxOccurs="1" />
            <xs:element name="code" type="F1819subfield" maxOccurs="1" />
            <xs:element name="com" type="F1819subfield" maxOccurs="1" />
            <xs:element name="dat" type="F1819subfield" maxOccurs="1" />
```

```
<xs:element name="dep" type="F1819subfield" maxOccurs="1" />
<xs:element name="dest" type="F1819subfield" maxOccurs="1" />
<xs:element name="dle" type="F1819subfield" maxOccurs="1" />
<xs:element name="dof" type="F1819subfield" maxOccurs="1" />
<xs:element name="eet" type="F1819subfield" maxOccurs="1" />
<xs:element name="est" type="F1819subfield" maxOccurs="1" />
<xs:element name="eur" type="F1819subfield" maxOccurs="1" />
<xs:element name="ifp" type="F1819subfield" maxOccurs="1" />
<xs:element name="nav" type="F1819subfield" maxOccurs="1" />
<xs:element name="opr" type="F1819subfield" maxOccurs="1" />
<xs:element name="orgn" type="F1819subfield" maxOccurs="1" />
<xs:element name="pbn" type="F1819subfield" maxOccurs="1" />
<xs:element name="per" type="F1819subfield" maxOccurs="1" />
<xs:element name="ralt" type="F1819subfield" maxOccurs="1" />
<xs:element name="reg" type="F1819subfield" maxOccurs="1" />
<xs:element name="rfp" type="F1819subfield" maxOccurs="1" />
<xs:element name="rif" type="F1819subfield" maxOccurs="1" />
<xs:element name="rmk" type="F1819subfield" />
<xs:element name="rvr" type="F1819subfield" maxOccurs="1" />
<xs:element name="sel" type="F1819subfield" maxOccurs="1" />
<xs:element name="src" type="F1819subfield" maxOccurs="1" />
<xs:element name="stayinfo1" type="F1819subfield"
      maxOccurs="1" />
<xs:element name="stayinfo2" type="F1819subfield"
      maxOccurs="1" />
<xs:element name="stayinfo3" type="F1819subfield"
      maxOccurs="1" />
<xs:element name="stayinfo4" type="F1819subfield"
      maxOccurs="1" />
<xs:element name="stayinfo5" type="F1819subfield"
      maxOccurs="1" />
<xs:element name="stayinfo6" type="F1819subfield"
      maxOccurs="1" />
<xs:element name="stayinfo7" type="F1819subfield"
      maxOccurs="1" />
<xs:element name="stayinfo8" type="F1819subfield"
```

```
                  maxOccurs="1" />
            <xs:element name="stayinfo9" type="F1819subfield"
                  maxOccurs="1" />
            <xs:element name="sts" type="F1819subfield" />
            <xs:element name="sur" type="F1819subfield" maxOccurs="1" />
            <xs:element name="talt" type="F1819subfield" maxOccurs="1" />
            <xs:element name="typ" type="F1819subfield" maxOccurs="1" />
      </xs:choice>
</xs:complexType>

<xs:complexType name="F19" mixed="true">
      <xs:all>
            <xs:annotation>
                  <xs:documentation>F19 - Supplementary information</xs:documentation>
            </xs:annotation>
            <xs:element name="a" type="F1819subfield" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="c" type="F1819subfield" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="d" type="F1819subfield" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="e" type="F1819subfield" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="j" type="F1819subfield" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="n" type="F1819subfield" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="p" type="F1819subfield" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="r" type="F1819subfield" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="s" type="F1819subfield" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="d1" type="F1819subfield" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
```

```
                        <xs:documentation>The d1, d2, d3 and d4 fields are component parts
                              of the d field, ADEXP splits this filed up.</xs:documentation>
                        <xs:documentation>They do not need to be processed further as the
                              parser always re-assembles them to field d.</xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="d2" type="F1819subfield" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="d3" type="F1819subfield" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="d4" type="F1819subfield" minOccurs="0"
                  maxOccurs="1" />
      </xs:all>
</xs:complexType>

<xs:complexType name="F22" mixed="true">
      <xs:annotation>
            <xs:documentation>F22 - Amendment field list</xs:documentation>
      </xs:annotation>
      <xs:all>
            <xs:element name="f3a" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="f3b" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="f5a" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="f5b" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="f5c" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="f7a" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="f7b" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="f7c" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
```

```
<xs:element name="f8a" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f8b" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f9a" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f9b" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f9c" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f10a" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f10b" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f13a" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f13b" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f14a" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f14b" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f14c" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f14d" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f14e" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f15" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f16a" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f16b" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f16ca" type="xs:string" minOccurs="0"
      maxOccurs="1" />
```

```
<xs:element name="f16cb" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f17a" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f17b" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f17c" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f17d" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f17e" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18altn" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18awr" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18code" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18com" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18dat" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18dep" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18dest" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18dle" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18dof" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18eet" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18est" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18eur" type="xs:string" minOccurs="0"
      maxOccurs="1" />
```

```
<xs:element name="f18ifp" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18nav" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18opr" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18orgn" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18pbn" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18per" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18ralt" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18reg" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18rfp" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18rif" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18rmk" type="xs:string" minOccurs="0" />
<xs:element name="f18rvr" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18sel" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18src" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18stayinfo1" type="xs:string"
      minOccurs="0" maxOccurs="1" />
<xs:element name="f18stayinfo2" type="xs:string"
      minOccurs="0" maxOccurs="1" />
<xs:element name="f18stayinfo3" type="xs:string"
      minOccurs="0" maxOccurs="1" />
<xs:element name="f18stayinfo4" type="xs:string"
      minOccurs="0" maxOccurs="1" />
<xs:element name="f18stayinfo5" type="xs:string"
```

```
      minOccurs="0" maxOccurs="1" />
<xs:element name="f18stayinfo6" type="xs:string"
      minOccurs="0" maxOccurs="1" />
<xs:element name="f18stayinfo7" type="xs:string"
      minOccurs="0" maxOccurs="1" />
<xs:element name="f18stayinfo8" type="xs:string"
      minOccurs="0" maxOccurs="1" />
<xs:element name="f18stayinfo9" type="xs:string"
      minOccurs="0" maxOccurs="1" />
<xs:element name="f18sts" type="xs:string" minOccurs="0" />
<xs:element name="f18sur" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18talt" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f18typ" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f19a" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f19c" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f19d" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f19e" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f19j" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f19n" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f19p" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f19r" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f19s" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80a" type="xs:string" minOccurs="0"
      maxOccurs="1" />
```

```
<xs:element name="f80b" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80c" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80d" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80e1" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80e2" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80e3" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80f" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80g" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80h" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80i" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80j1" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80j2" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80j3" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80j4" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80j5" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80j6" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80j7" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80k" type="xs:string" minOccurs="0"
      maxOccurs="1" />
```

```
<xs:element name="f80l" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80m1" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80m2" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80m3" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80o" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80p1" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80p2" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80p3" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80p4" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80p5" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80p6" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80p7" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80p8" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80p9" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80r" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80s" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80t" type="xs:string" minOccurs="0"
      maxOccurs="1" />
<xs:element name="f80u" type="xs:string" minOccurs="0"
      maxOccurs="1" />
```

```
            <xs:element name="f80v" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="f80w" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="f80x" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="f80y" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="f80z" type="xs:string" minOccurs="0"
                  maxOccurs="1" />

            <xs:element name="f81a" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="f81s" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="f81adsb" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="f81adsc" type="xs:string" minOccurs="0"
                  maxOccurs="1" />
      </xs:all>
</xs:complexType>
<xs:complexType name="F14A" mixed="true">
      <xs:all>
            <xs:element name="a" type="xs:string" minOccurs="0"
                  maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>F16a - Destination aerodrome (ADES)
                        </xs:documentation>
                  </xs:annotation>
            </xs:element>
      </xs:all>
</xs:complexType>
<xs:simpleType name="PRIORITYINDICATOR">
      <xs:restriction base="xs:string">
            <xs:enumeration value="DD" />
            <xs:enumeration value="EE" />
```

```
            <xs:enumeration value="FF" />
            <xs:enumeration value="GG" />
            <xs:enumeration value="SS" />
            <xs:enumeration value="XX" />
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ADDRESS">
      <xs:annotation>
            <xs:documentation>AFTN or SITA address</xs:documentation>
      </xs:annotation>
      <xs:restriction base="xs:string">
            <xs:maxLength value="8" />
      </xs:restriction>
</xs:simpleType>
<xs:complexType name="ADDRESSEES" mixed="true">
      <xs:annotation>
            <xs:documentation>Header addresses, from the AD address line (these
                  are any 'extra' addressees in a message)</xs:documentation>
      </xs:annotation>
      <xs:sequence>
            <xs:element name="addressa" type="ADDRESS" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="addressb" type="ADDRESS" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="addressc" type="ADDRESS" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="addressd" type="ADDRESS" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="addresse" type="ADDRESS" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="addressf" type="ADDRESS" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="addressg" type="ADDRESS" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="addressh" type="ADDRESS" minOccurs="0"
                  maxOccurs="1" />
```

```
            <xs:element name="addressi" type="ADDRESS" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="addressj" type="ADDRESS" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="addressk" type="ADDRESS" minOccurs="0"
                  maxOccurs="1" />
      </xs:sequence>
</xs:complexType>
<xs:simpleType name="MSG">
      <xs:restriction base="xs:string">
            <xs:maxLength value="21000" />
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="FILINGTIME">
      <xs:restriction base="xs:string">
            <xs:maxLength value="6" />
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="IFPLID">
      <xs:restriction base="xs:string">
            <xs:maxLength value="10" />
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="HHMM">
      <xs:restriction base="xs:integer">
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="FL">
      <xs:restriction base="xs:string">
            <xs:maxLength value="5" />
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="POINT">
      <xs:restriction base="xs:string">
            <xs:maxLength value="5" />
      </xs:restriction>
```

```
</xs:simpleType>
<xs:simpleType name="FPL_2012_VERSION">
      <xs:restriction base="xs:string">
            <xs:enumeration value="PRE_FPL_2012" />
            <xs:enumeration value="POST_FPL_2012" />
            <xs:enumeration value="PRE_AND_POST_FPL_2012" />
            <xs:enumeration value="UNKNOWN_FPL_2012" />
      </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ADJ_UNIT_NAME">
      <xs:restriction base="xs:string">
      </xs:restriction>
</xs:simpleType>
<xs:complexType name="fields">
      <xs:all>
            <xs:element name="MSG" type="MSG" minOccurs="0" maxOccurs="1" />
            <xs:element name="PRIORITYINDICATOR" type="PRIORITYINDICATOR"
                  minOccurs="0" maxOccurs="1" />
            <xs:element name="MSG_HEADER" type="MSG" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="ADDRESSEES" type="ADDRESSEES" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="FILINGTIME" type="FILINGTIME" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="ORIGINATOR" type="ADDRESS" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="ADADDRESSEES" type="ADDRESSEES"
                  minOccurs="0" maxOccurs="1" />
            <xs:element name="MSG_BODY" type="MSG" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="IFPLID" type="IFPLID" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="CTOT" type="HHMM" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="ELDT" type="HHMM" minOccurs="0"
                  maxOccurs="1" />
```

```
            <xs:element name="CFL" type="FL" minOccurs="0" maxOccurs="1" />
            <xs:element name="F3" type="F3" minOccurs="0" maxOccurs="1" />
            <xs:element name="F5" type="F5" minOccurs="0" maxOccurs="1" />
            <xs:element name="F7" type="F7" minOccurs="0" maxOccurs="1" />
            <xs:element name="F8" type="F8" minOccurs="0" maxOccurs="1" />
            <xs:element name="F9" type="F9" minOccurs="0" maxOccurs="1" />
            <xs:element name="F10" type="F10" minOccurs="0" maxOccurs="1" />
            <xs:element name="F13A" type="F13A" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="F13" type="F13" minOccurs="0" maxOccurs="1" />
            <xs:element name="F14" type="F14" minOccurs="0" maxOccurs="1" />
            <xs:element name="F15" type="F15" minOccurs="0" maxOccurs="1" />
            <xs:element name="F16A" type="F16A" minOccurs="0"
                  maxOccurs="1" />
            <xs:element name="F16" type="F16" minOccurs="0" maxOccurs="1" />
            <xs:element name="F17" type="F17" minOccurs="0" maxOccurs="1" />
            <xs:element name="F18" type="F18" minOccurs="0" maxOccurs="1" />
            <xs:element name="F19" type="F19" minOccurs="0" maxOccurs="1" />
            <xs:element name="F22" type="F22" minOccurs="0" maxOccurs="1" />
            <xs:element name="F14A" type="F14A" minOccurs="0" maxOccurs="1" />
      </xs:all>
</xs:complexType>
<xs:complexType name="route">
      <xs:sequence>
            <xs:element name="route-item" minOccurs="0" maxOccurs="unbounded">
                  <xs:complexType>
                        <xs:attribute name="BREAKTEXT" type="xs:string" use="optional" />
                        <xs:attribute name="NAME" type="xs:string" use="required" />
                        <xs:attribute name="TYPE" type="xs:string" use="required" />
                        <xs:attribute name="RULES" type="xs:string" use="optional" />
                        <xs:attribute name="SUBTYPE" type="xs:string" use="required" />
                        <xs:attribute name="START_INDEX" type="xs:string" use="optional" />
                        <xs:attribute name="END_INDEX" type="xs:string" use="optional" />
                        <xs:attribute name="SPEED" type="xs:string" use="optional" />
                        <xs:attribute name="HEIGHT" type="xs:string" use="optional" />
                        <xs:attribute name="SPEED_ICAO" type="xs:string" use="optional" />
```

```
                        <xs:attribute name="HEIGHT_ICAO" type="xs:string" use="optional" />
                        <xs:attribute name="BEARING" type="xs:string" use="optional" />
                        <xs:attribute name="LATITUDE" type="xs:string" use="optional" />
                        <xs:attribute name="LONGITUDE" type="xs:string" use="optional" />
                        <xs:attribute name="SUBTYPE.class" type="xs:string"
                              use="optional" />
                        <xs:attribute name="TYPE.class" type="xs:string" use="optional" />
                  </xs:complexType>
            </xs:element>
      </xs:sequence>
</xs:complexType>
<xs:complexType name="errors">
      <xs:sequence>
            <xs:element name="error" minOccurs="0" maxOccurs="unbounded">
                  <xs:complexType>
                        <xs:simpleContent>
                              <xs:extension base="xs:string">
                                    <xs:attribute name="TOKEN" type="xs:string" use="optional" />
                                    <xs:attribute name="START" type="xs:string" use="optional" />
                                    <xs:attribute name="STOP" type="xs:string" use="optional" />
                                    <xs:attribute name="FIELD_OFFSET" type="xs:string"
                                          use="optional" />
                                    <xs:attribute name="FIELD_ID" type="xs:string" use="optional" />
                              </xs:extension>
                        </xs:simpleContent>
                  </xs:complexType>
            </xs:element>
      </xs:sequence>
</xs:complexType>
<xs:complexType name="info">
      <xs:all>
            <xs:annotation>
                  <xs:documentation>Used to store various flight plan attributes:
                  </xs:documentation>
            </xs:annotation>
            <xs:element name="FF" type="xs:boolean" minOccurs="0"
```

```
      maxOccurs="1">
      <xs:annotation>
            <xs:documentation>the priority indicator is 'FF' then a field FF is
                  present containing the value 'true'</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="DD" type="xs:boolean" minOccurs="0"
      maxOccurs="1">
      <xs:annotation>
            <xs:documentation>the priority indicator is 'DD' then a field DD is
                  present containing the value 'true'</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="GG" type="xs:boolean" minOccurs="0"
      maxOccurs="1">
      <xs:annotation>
            <xs:documentation>the priority indicator is 'GG' then a field GG is
                  present containing the value 'true'</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="SS" type="xs:boolean" minOccurs="0"
      maxOccurs="1">
      <xs:annotation>
            <xs:documentation>the priority indicator is 'SS' then a field SS is
                  present containing the value 'true'</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="__UNKNOWN__" type="xs:boolean"
      minOccurs="0" maxOccurs="1">
      <xs:annotation>
            <xs:documentation>Indicates an unknown priority indicator is
                  present, PRIORITYINDICATOR will be 'XX'</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="ICAO" type="xs:boolean" minOccurs="0"
      maxOccurs="1">
```

```
      <xs:annotation>
            <xs:documentation>If a message is in an ICAO message then this
                  field is 'true', in ADEXP format 'false'</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="ATS_MESSAGE" type="xs:boolean" minOccurs="0"
      maxOccurs="1">
      <xs:annotation>
            <xs:documentation>If a message is an ATS message then this
                  field is 'true', if an OLDI message 'false'</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="ADDRESSEES_VALID" type="xs:boolean"
      minOccurs="0">
      <xs:annotation>
            <xs:documentation>This field is used for internal processing only
                  and can be ignored for external access</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="MILITARY" type="xs:boolean" minOccurs="0"
      maxOccurs="1">
      <xs:annotation>
            <xs:documentation>Set to 'true' if ICAO field 8b is 'M', 'false'
                  when F8b not 'M'</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="HAS_HEADER" type="xs:boolean" minOccurs="0"
      maxOccurs="1">
      <xs:annotation>
            <xs:documentation>Set to 'true' if the message has a header,
                  'false' when no header is present</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="SPLIT_IDX" type="xs:integer" minOccurs="0"
      maxOccurs="1">
      <xs:annotation>
```

```
            <xs:documentation>Specifies the index at which the header ends and
                  the message body begins, points to the '(' or '-'
            </xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="AFIL" type="xs:boolean" minOccurs="0"
      maxOccurs="1">
      <xs:annotation>
            <xs:documentation>Set to 'true' if ICAO field 13a is 'AFIL',
                  'false' otherwise</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="OAT_GAT" type="xs:boolean" minOccurs="0"
      maxOccurs="1">
      <xs:annotation>
            <xs:documentation>Present and set to 'true' if ICAO field 15
                  contains ...OAT ... GAT... changes in F15, indicated by the
                  presence of 'OAT'/'GAT' tokens in field 15</xs:documentation>
            <xs:documentation>The field is missing if such rule changes are not
                  present in field 15.</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="IFPSTART_STOP" type="xs:boolean"
      minOccurs="0" maxOccurs="1">
      <xs:annotation>
            <xs:documentation>Present and set to 'true' if ICAO field 15
                  contains ...IFPSTOP ... IFPSTART... in F15, indicated by the
                  presence of 'IFPSTOP'/'IFPSTART' tokens in field 15
            </xs:documentation>
            <xs:documentation>The field is missing if such rule changes are not
                  present in field 15.</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="F10a_VERSION" type="FPL_2012_VERSION"
      minOccurs="0" maxOccurs="1">
      <xs:annotation>
```

```
            <xs:documentation>Indicates if field 10a is pre or post FPL 2012,
                  whether the status</xs:documentation>
            <xs:documentation>is unknown or if it contains both pre and post
                  FPL 2012 tokens.</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="F10b_VERSION" type="FPL_2012_VERSION"
      minOccurs="0" maxOccurs="1">
      <xs:annotation>
            <xs:documentation>Indicates if field 10b is pre or post FPL 2012,
                  whether the status</xs:documentation>
            <xs:documentation>is unknown or if it contains both pre and post
                  FPL 2012 tokens.</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="F10_VERSION" type="FPL_2012_VERSION"
      minOccurs="0" maxOccurs="1">
      <xs:annotation>
            <xs:documentation>Indicates if field 10 is pre or post FPL 2012,
                  whether the status</xs:documentation>
            <xs:documentation>is unknown or if it contains both pre and post
                  FPL 2012 tokens.</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="F18_VERSION" type="FPL_2012_VERSION"
      minOccurs="0" maxOccurs="1">
      <xs:annotation>
            <xs:documentation>Indicates if field 18 is pre or post FPL 2012,
                  whether the status</xs:documentation>
            <xs:documentation>is unknown or if it contains both pre and post
                  FPL 2012 tokens.</xs:documentation>
      </xs:annotation>
</xs:element>
<xs:element name="TARGET_VERSION" type="FPL_2012_VERSION"
      minOccurs="0" maxOccurs="1">
      <xs:annotation>
```

```
                        <xs:documentation>Indicates the target FPL 2012 version when
                        translating a message from one version to another</xs:documentation>
                  </xs:annotation>
            </xs:element>
            <xs:element name="ADJ_UNIT_NAME" type="ADJ_UNIT_NAME"
                  minOccurs="0" maxOccurs="1">
                  <xs:annotation>
                        <xs:documentation>Contains the unit name that set a message to the parser,
                        arrives a spart of F3 or MSGREF fields</xs:documentation>
                  </xs:annotation>
            </xs:element>
      </xs:all>
</xs:complexType>

<xs:complexType name="Primary_Basic_Fields" mixed="true">
      <xs:annotation>
            <xs:documentation>This tag is only output if a message has been flagged with</xs:documentation>
            <xs:documentation>'adexpOutput=true' to indicate that all ADEXP fields will</xs:documentation>
            <xs:documentation>be saved to the XML output file.</xs:documentation>
            <xs:documentation>This tag will contain all ADEXP primary basic fields</xs:documentation>
      </xs:annotation>
      <xs:sequence>
            <xs:any processContents='lax' />
      </xs:sequence>
</xs:complexType>

<xs:complexType name="Primary_List_Fields" mixed="true">
      <xs:annotation>
            <xs:documentation>This tag is only output if a message has been flagged with</xs:documentation>
            <xs:documentation>'adexpOutput=true' to indicate that all ADEXP fields will</xs:documentation>
            <xs:documentation>be saved to the XML output file.</xs:documentation>
            <xs:documentation>This tag will contain all ADEXP primary list fields</xs:documentation>
      </xs:annotation>
      <xs:sequence>
            <xs:any processContents='lax' />
      </xs:sequence>
```

```
</xs:complexType>

<xs:complexType name="Primary_Structured_Fields" mixed="true">
      <xs:annotation>
            <xs:documentation>This tag is only output if a message has been flagged with</xs:documentation>
            <xs:documentation>'adexpOutput=true' to indicate that all ADEXP fields will</xs:documentation>
            <xs:documentation>be saved to the XML output file.</xs:documentation>
            <xs:documentation>This tag will contain all ADEXP primary structured fields</xs:documentation>
      </xs:annotation>
      <xs:sequence>
            <xs:any processContents='lax' />
      </xs:sequence>
</xs:complexType>

<xs:complexType name="Undefined_for_this_Message" mixed="true">
      <xs:annotation>
            <xs:documentation>This tag is only output if a message has been flagged with</xs:documentation>
            <xs:documentation>'adexpOutput=true' to indicate that all ADEXP fields will</xs:documentation>
            <xs:documentation>be saved to the XML output file.</xs:documentation>
            <xs:documentation>This tag will contain all ADEXP primary fields (all types),</xs:documentation>
            <xs:documentation>that are defined in the XML DB but undefined for the </xs:documentation>
            <xs:documentation>message title for 'this' output file.</xs:documentation>
      </xs:annotation>
      <xs:sequence>
            <xs:any processContents='lax' />
      </xs:sequence>
</xs:complexType>

<xs:complexType name="Unknown_Content" mixed="true">
      <xs:annotation>
            <xs:documentation>This tag is only output if a message has been flagged with</xs:documentation>
            <xs:documentation>'adexpOutput=true' to indicate that all ADEXP fields will</xs:documentation>
            <xs:documentation>be saved to the XML output file.</xs:documentation>
            <xs:documentation>This tag will contain all ADEXP primary fields (all types),</xs:documentation>
            <xs:documentation>that NOT defined in the XML DB and included for the </xs:documentation>
            <xs:documentation>message title for 'this' output file.</xs:documentation>
```

```
                  <xs:documentation></xs:documentation>
            </xs:annotation>
            <xs:sequence>
                  <xs:any processContents='lax' />
            </xs:sequence>
      </xs:complexType>

      <xs:complexType name="FDR">
            <xs:sequence>
                  <xs:element name="fields" type="fields" />
                  <xs:element name="errors" type="errors" />
                  <xs:element name="route" type="route" />
                  <xs:element name="info" type="info" />
                  <xs:element name="Primary_Basic_Fields" minOccurs="0" maxOccurs="1" />
                  <xs:element name="Primary_List_Fields" minOccurs="0" maxOccurs="1" />
                  <xs:element name="Primary_Structured_Fields" minOccurs="0" maxOccurs="1" />
                  <xs:element name="Undefined_for_this_Message" minOccurs="0" maxOccurs="1" />
                  <xs:element name="Unknown_Content" minOccurs="0" maxOccurs="1" />
            </xs:sequence>
      </xs:complexType>
      <xs:element name="fdr" type="FDR" />
</xs:schema>
```