Eurocontrol

The Link2000+ XML Based ATC Information Exchange Format

Document Ref. : TRS1157/TW/4/1/63

Rev. No. : Issue 1.8

Date : 12/09/2014

Author : Tony Whyman

Copyright Notice

CONTENTS **Page**

1 Introduction

1.1 Background

The objective of the LINK 2000+ Programme is to plan and co-ordinate the implementation of operational air/ground data link services for Air Traffic Management (ATM) in the core area of Europe in the timeframe beyond 2000. The programme provides a framework for implementation of Controller Pilot Data Link Communication (CPDLC) in accordance with the internationally standardised operational services defined by the Link 2000+ baseline [1]. The specific services planned for initial implementation are the Data Link Initiation Capability (DLIC), ATC Communications Management (ACM), ATC Clearances (ACL), and ATC Microphone Check (AMC).

SAS commenced Link 2000+ operations as the first pioneer airline in 2004, together with the EUROCONTROL Upper Airspace Control Centre (UAC) Maastricht. Since then a number of further airlines have joined the programme, including FedEx, Air Europa, ATI, Air Berlin, Alitalia. Lufthansa and Hapag Lloyd. Accordingly, a significant increase is expected in the volume of data link traffic in the forthcoming year.

The UAC at Maastricht has developed a Data Link Statistics Analysis Tool (DALISTAT) to produce routine statistical reports on data link usage observed in Link 2000+ operations within Maastricht airspace. The current tool is based on an architecture involving a mixture a Perl scripts, a Microsoft Access database, and a Visual Basic user interface. However, as the Link 2000+ Programme has expanded, the usability of the existing tool has deteriorated it has become apparent the scalability of this architecture will be unlikely to meet the expected increases in data link traffic.

Furthermore, it is desired to be able to perform detailed technical investigations in response to specific observations of data link behaviour, and it is perceived that the current tool provides inadequate flexibility to perform queries to the database that were not foreseen when the tool was designed.

Accordingly, it is intended that an Enhanced Statistical Analysis Environment should be developed, to improve the scalability, usability and flexibility currently offered by the DALISTAT tool. This environment will required a comprehensive database to record the information received from communications logs and provide the basis for reporting and analysis. This new database was originally known as the Central Reporting Office Tool (CROT), and is now called the LINK2000+ Statistics Reporting and Analysis Tool (LISAT).

The LISAT Development Project is responsible for developing the database and it support programs.

1.2 Scope

This document describes the exchange format used for exchanging recorded data between ANSPs and the LINK2000+ Statistics Reporting and Analysis Tool (LISAT).

1.3 References

1. Extensible Markup Language (XML) 1.0 (Fourth Edition), 16 August 2006, (http://www.w3.org/TR/2006/REC-xml-20060816/)

2. Namespaces in XML 1.0 (Second Edition), 16 August 2006 (http://www.w3.org/TR/2006/REC-xml-names-20060816/)

3. Learning XML, Erik T Ray, Published by O'Reilly and Associates, ISBN: 0-596-000464

4. Link 2000+ Baseline – Pioneers Phase - EUROCONTROL V 1.2 March 2006

5. ICAO Doc9705

6. Proposed Database Schema for the Link2000+ Reporting Tool, Eurocontrol, 11/6/07, Ref: LINK/LIT/CRO/DATABASE

7. LISAT:Transport Service Log Import, 12/9/2014, Ref: LISAT/DESIGN/120704

1.4 Changes From Issue 1.0

1. An optional Tail Number has been added to the CM Logon Request

2. An optional Tail Number has been added to the standard attributes.

3. A new communications type attribute (FANS or ATN) has been added to the atc-messages element (default ATN).

1.5 Changes From Issue 1.1

1. Position Reports Section added

2. Correction of Sector Transitions Heading Level to level 2. Note, subsequent sections are re-numbered.

1.6 Changes From Issue 1.2

1. Schema bug fixes resulting from testing of full CPDLC message set.

1.7 Changes from Issue 1.3

1. Correct errors in CPDLC Provider Abort codes

1.8 Changes from Issue 1.4

1. Add support for cm-forward-req and cm-forward-resp

1.9 Changes from Issue 1.5

1. Change to permit an empty report

1.10 Changes from Issue 1.6

1. Add support for PM-CPDLC User Abort reasons.

1.11 Changes from Issue 1.7

1. Major Update to extend the XML Definition to include Transport Protocol Logs.

2 The Development of the LISAT Data Interchange Format

The underlying requirement is for a common interchange format for all ATC related data that could be provided to the LISAT and which could also be used to provided filtered information from the tool to its users. This format should be well standardised and there should be many support tools available for it.

The original data format for the recorded CPDLC messages will be ASN.1 PER. Some messages may be in OLDI format when they are messages exchanged between ATC Centres, and others may be in an internal representation local to the ATC Centre. Some form of data transformation between the source and the LISAT will be necessary. The intended strategy is for the ATSU responsible for the recorded data to transform it into a common standardised format from whatever format they use for local recording and to provide it as such to the LISAT where it is then applied to the database.

The Extensible Markup Language (XML) has been chosen as the basis for the LISAT interchange format.

2.1 The Choice of XML

Although there has been considerable recent interest in XML as a standardised means for exchanging structured data in a text format, it is by no means a recent development.

XML can trace its origins to “hot metal printing” and the mark-up of typed documents with editor’s instructions. When a document or an entire book was prepared for printing, an editor would review the text and mark it up with the required structure into chapters, paragraphs, quotations, emphasised text and so on. It would then be passed to the printer who would then prepare hot metal plates for the actual printing process, by interpreting the editor’s instructions according to the style manual specified for the publication. The style manual would determine the actual font sizes, inter-paragraph gaps and so on. The separation of structure and style was a critical part of this process.

When IBM first introduced computer typesetting, they created a mark-up language (called GML) for text that could be interpreted by the computerised type-setting process. This mark-up language used tags - words delimited by ‘<’ and ‘>’ characters to select and identify areas of text. Additionally, some tags could have attributes, in “name=value” notation to provide additional information. Tags were used to both separate and enclose text. In the latter case tags come in pairs with the closing tag starting with ‘</’ e.g. .... .

This proved a popular approach and was later standardised by ISO as SGML, but the real popularity of SGML based approaches only took off when Tim Berners Leigh developed the Hypertext Mark-up Language (HTML) as a simplified form of SGML. HTML enabled the rapid development of the World Wide Web, and of the mark-up language concept, but was also a retrograde step as it mixes structure and style.

Since then the World Wide Web Consortium (W3C) has put in considerable effort to return HTML to its SGML origins. Separate style sheets have been introduced to facility to separation of style and structure, and a modified version of HTML – XHMTL – has been introduced that returns the structuring of data to its original principles. The development of XML was part of that process and is an SGML compatible set of rules for the specification of structured documents.

XML was developed by W3C with the following goals in mind:

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.

XHMTL is an XML compliant version of HTML, and XML has become the standardised way of exchanging text based structure data.

There are a considerable wealth of tools available for parsing, validating and using XML formatted data, and this consideration alone makes it an ideal choice for the interchange format for the LISAT, notwithstanding all its other advantages.

XML is, in principle, relatively simple, and an XML compliant document is no more than a text document marked up with tags either as single tags in the form `<tag/>` or as enclosing tags contain text and/or other tags, in the form `<tag>....</tag>`. In addition, tags may have attributes comprising a case sensitive white space separated list of name="value" pairs. Note that the value must always be enclosed in single or double quotes.

An XML interpreter will parse a text file for tags and so that '<' and other special characters can be part of the text, the concept of entities is introduced. That is identifiers enclosed in '&' and ';' characters. An entity such as **<** is always interpreted as a '<' character and not the start of a tag. Pre-defined entities exist for many special characters (e.g. copyright symbols) and document specific entities can also be defined.

It is also possible to enclose text in `<!CDATA[...]]>` sections which delimit text that is not parsed by the XML interpreter.

2.2 Using XML for an Interchange Format

It needs to be understood that XML is a series of rules for designing an interchange format rather than the specification of the format itself. It is possible to write informal documents that are XML compliant as well as formal documents that reference a well defined syntax.

For example:

```
<document>
  <header>
    <title>The Link2000+ XML Based ATC Information Exchange Format</title>
    <author>Tony Whyman</author>
    <ref publisher="Eurocontrol" date="17/9/2007">LINK LIT/CRO/XMLFmt</ref>
  </header>
  <body>
    <heading1>Introduction</heading1>
   .....
  </body>
</document>
```

is an example of an XML compliant document. It is compliant because it has well formed tags, with each tag paired with a closing tag (e.g. (`<body>...</body>`) and where tags have attributes, they are in "Name=Value" format with the values enclosed in quotation marks. Between the tags is plain text.

It would be possible to develop a style sheet for this XML document which gave the rules for interpretation and layout of the tags so that it could be printed or displayed in some appropriate way. As it stands, the use of English Language tags makes it readily interpretable by a human user.

However, a tag like `<document>` is not defined by XML, XML only says that it is a valid tag. We may interpret it as the delimiter for a "document" but that is only our intuitive interpretation based on its natural language meaning. It would be just as XML compliant to have used `<smorgasbord>` as the opening and closing tags and as long as the style sheet referred to these tags instead of `<document>` the resulting printed document would be just the same.

In practice, a document that is simply XML compliant is not that useful. For example, if XML was being used by document editors to mark up a document for printing, it would not be that useful if each used a personal set of tags for mark up. The printer would need a different style sheet variant for each editor. For an XML document to be truly useful requires that the XML tags, their attributes, their interpretation and their relationship to each other is well defined. Standard style sheets can then be prepared and standard computer interpretations can be made.

A formal definition of the XML format used for LISAT exchange is essential as it will need to be computer interpreted and formal semantics applied to the information exchanged.

2.3 Defining an XML Document Schema

The XML 1.0 specification **[1]** includes the specification of a Document Type Definition (DTD) Language, which is used for formal definition of the schema a specific XML compliant document type. The DTD is inherited by XML from SGML, however, quoting from Wikipedia:

While DTD support in XML tools is widespread due to its inclusion in the XML 1.0 standard, it is seen as limited for the following reasons:

- *No support for newer features of XML — most importantly, namespaces.*
- *Lack of expressivity. Certain formal aspects of an XML document cannot be captured in a DTD.*

- *Custom non-XML syntax to describe the schema, inherited from SGML. (namely 'Extended Backus Naur Form')*

Three newer XML schema languages that are much more powerful are increasingly favored over DTDs:

- *XML Schema, also referred to as XML Schema Definition (XSD), has achieved Recommendation status within the W3C.*
- *RELAX NG, which is also a part of DSDL, is an ISO international standard.*
- *Document Structure Description (DSD), attempts to combine an expressive schema balanced with ease of use.*

The XSD should probably be the preferred choice for new XML document schemas. However, tool support is only becoming available and, in particular, the *libxml* package that is expected to be used by the LISAT only offers partial support for XSD at present.

The XML DTD will thus be used to define the LISAT interchange format. At some time in the future, when the specification is revised, this may be changed to XSD. However, this will not change the format of the documents and, as tools are available for the automated transformation of DTD specifications to XSD specifications, it is unlikely to be a serious issue.

2.4 The Document Type Definition

```
<!ELEMENT people_list (person*)>
<!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT birthdate (#PCDATA)>
<!ELEMENT gender (#PCDATA)>
<!ELEMENT socialsecuritynumber (#PCDATA)>
```

Above are some examples of a DTD. In this case defining a number of elements (XML tags). Each element is given a name (e.g. people_list), which is then followed by a list of the elements it can contain (in parentheses). In the above, people_list can contain zero, one or more “person” elements, while the person element comprises a sequence of elements: name, birthdate, gender and social security number. Other than the name, all of the other elements are optional (indicated by the ‘?’).

Note that the DTD borrows from regular expression syntax in that the ‘*’ superscript implies zero, one or more occurrences of the preceding symbol or group of symbols, and the ‘+’ superscript implies one or more occurrences of the preceding symbol or group of symbols.

The name, birthdate, gender and social security number elements all contain PCDATA, that is Parsed Character Data. PCDATA is essentially plain text but may contain “entities” which are parsed and interpreted. Entities are items such as **<** which is interpreted as a literal ‘<’ character, rather than the start of a tag.

An example of an XML compliant document using this DTD would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<people_list>
  <person>
    <name>John Doe</name>
    <gender>male</name>
  </person>
  <person>
    <name>Jane Smith</name>
    <birthdate>29/2/1985</birthdate>
    <gender>female</gender>
  </person>
</people_list>
```

Note that in the above, the birthday is obviously fictitious. However, it is still valid XML as there is no constraint on the birthdate in the DTD other than it is character data.

Elements can have attributes and if so, they may be declared as follows:

```
<!ATTLIST          person
                   id           ID                          #REQUIRED
                   status       (dead|alive)  “alive”
>
```

The above defines two attributes for the “person” element. The “id” attribute must be present and is defined as a unique identifier for the person. This identifier must be unique within the document. A second attribute “status”, indicates whether the person is dead or alive and, if omitted, defaults to “alive”.

Note that attributes can also be identified as “#IMPLIED” which is interpreted as optiona.

DTDs can also contain Entity specifications, where entities are essentially macros for either the XML document itself or the DTD (parameter entities).

2.5 Using the LISAT DTD

In practice, ATSUs will be expected to provide data to the LISAT as an XML Document compliant with the LISAT DTD. These documents will first be validated by the LISAT and then interpreted and their data entered into the database.

There is currently no intention to provide a style sheet for the formatted printing of XML LISAT documents. However, one may be produced in the future if a requirement to do so arises.

2.6 XML Limitations

XML is a text format and it is verbose. It does not make for small documents, nor can it readily encode binary data. On the other hand, the text format does make XML documents human readable and editable.

As a text document, XML documents are readily compressible and should be compressed for long term storage or transfer. When compressed, the frequent occurrence of well defined symbols should make for a high compression ratio thus offsetting and disadvantages of the text format. LISAT documents should normally be compressed for storage or transfer.

Binary format elements can be included by reference from the XML document and this how they should be handled.

3 The LISAT Interchange Format

3.1 Objectives

The following is an example of the intended LISAT interchange document:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE l2kx_document PUBLIC -//MWA//DTD LISAT Document 1.6//EN
                        "http://lisat.mwasoftware.co.org/l2kx6.dtd">
<l2kx-document>
  <header>
    <title>A LISAT Example Document</title>
    <date>2007/10/4</date>
    <comments>Anything goes</comment>
  </header>
  <atc-messages reporting-ATSU=”EDYY”>
    <cpdlc-message flight=”SAS0123” icao-24-bit=”0011aa”
                   logged-at=”2007/01/13 12:13:10”>
      ....
    </cpdlc-message>
  </ atc-messages >
  <sector-transitions reporting-ATSU=”EDYY” >
    <transition sector=RUHR_ID”>
      <date>2003/01/06</date>
    ....
    </transition>
  </sector-transitions>
  <position-reports reporting-ATSU="EDYY">
    <position-report flight=”SAS0123” icao-24-bit=”0011aa”
                   logged-at=”2007/01/13 12:15:20”>
    …
    </position-report>
  <position_reports>
</l2kx-document>
```

As indicated by the first line, it is an XML version 1.0 compliant document encoded as 8 bit UTF characters (variable length). In practice, CPDLC does not deviate from the ASCII character set and hence extended characters are not expected.

The XML header is then followed by a reference to the document type definition. A parser uses this to parse and validate the rest of the document. The DTD is identified by its Universal Resource Identifier (URI). The element *l2kx-document* is also identified as the head element of the document. This may be omitted and when not present, the LISAT parser will assume the most recent DTD version.

The l2kx-document then comprises a header followed by zero, one or more *atc-messages* followed by zero, one or more sector transition messages, and zero one or more position reports. There is no requirement for a single document to contain each type of information. However, the format does allow this.

The header is informational only and provides a title, the date created and additional comments.

The *atc-messages* element identifies the reporting ATSU and then comprises one or more CPDLC messages. The CPDLC messages are in no particular order and, in this case, the containing element identifies the flight by both its flight ID and ICAO 24-bit aircraft address. The contents of the message will comprise the message header, message elements and route information.

The format does allow a single file to contain CPDLC and CM messages from more than one reporting ATSU. Each will have its own atc-messages element.

The sector transitions follow a similar pattern, with a list of sector transitions for each reporting ATSU. Similarly, the position reports contains a list of position reports obtained from surveillance data.

As a general point, the DTD for the l2kx-document is straightforward, with the main complexity being due to the large amount of information transferred by CPDLC.

3.2 The L2KX Document Type Definition

```
<!ELEMENT l2kx-document (header?,(atc-messages|sector-transitions|
position-reports|network-trace)*) >
```

The l2kx-document (LINK2000+ eXchange Document) should comprise at least one atc-messages, sector-transitions, position-reports element, or network-trace element, which can then be followed by others, in any order.

3.2.1 The Header

The Document Header is informational only:

```
<!ELEMENT header (title?,date?,comment?) >
<!ELEMENT title (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
```

The title and comments are both free text fields, with the title used as the document title on any reports. The Date element is the date the document was created and is defined below.

3.2.2 Common Elements

The following are common elements for both Sector Transitions and ATC Messages.

```
<!ELEMENT airport (#PCDATA) >
<!ATTLIST airport
                  role         (departure|destination)    #REQUIRED >
```

An airport is identified by its four digit ICAO Airport Code. The role of the airport must also be identified by the role attribute. Use of the role attribute is specified under those elements in which it may be used.

```
<!ELEMENT date (#PCDATA) >
```

A UTC date in yyyy/mm/dd format.

```
<!ELEMENT facilityDesignation (#PCDATA)>
```

The facility designation is constrained to the ICAO assigned facility designation.

```
<!ELEMENT flight-id (#PCDATA)>
```

This element contains an aircraft's Flight Identification (Callsign).

```
<!ELEMENT icao-24-bit (#PCDATA) >
```

This element contains the ICAO 24-bit aircraft address expressed as a six digit hexadecimal number.

```
<!ELEMENT level (leveldata|( leveldata, leveldata))>
<!ATTLIST level
                  atwTolerance (at|atOrAbove|atOrBelow) #IMPLIED
                  type         (current|waypoint) #IMPLIED>
<!ELEMENT leveldata (#PCDATA)>
<!ATTLIST leveldata
                  units        (feet|metres|flightLevel|
                                metricFlightLevel)       #REQUIRED
                  role         (current|waypoint)        #IMPLIED>
```

The level can be either a single level, or a block level, in which case the two bounding levels are provided. Either level may be expressed in feet, or metres or as a normal or metric Flight Level. Use of the role attribute is specified under those elements in which it may be used.

The atwTolerance attribute may only be used when the level is contained in an atwAlongTrackWaypoint element.

The "type" attribute is for use in position reports.

```
<!ENTITY % cm-abort-reasons
                  "timer-expired|undefined-error|invalid-PDU|
                  protocol-error|
                  dialogue-acceptance-not-permitted|
                  dialogue-end-not-accepted|
                  communication-service-error|
                  communication-service-failure|
                  invalid-QOS-parameter|
                  expected-PDU-missing"
```

The entity cm-abort-reasons specifiy the abort reason codes for Contenxt Management.

```
<!ELEMENT time (#PCDATA) >
<!ATTLIST time
                  role         (departure|arrival|expiry|current|
                                next|nextPlusOne|destination|
                                waypoint|remainingFuel|
                                clearanceExpected|allocated|
                               controlled|expected) #IMPLIED
                  tolerance    (at|atOrBefore|atOrAfter) #IMPLIED>
```

This element contains the UTC time in the format hh:mm:ss. It may also have a role attribute when this is needed to identified what the time applies to. Use of the role attribute is specified under those elements in which it may be used.

A tolerance attribute is required when the time is a “controlled time” in a departure clearance.

```
<!ENTITY          % stdatts
                  “flight-id                CDATA        #REQUIRED
                   icao-24-bit  CDATA       #IMPLIED
                   tail-number  CDATA       #IMPLIED
                   logged-at                CDATA        #REQUIRED”>
```

The parameter entity *stdatts* defines the common attributes for all ATC Messages. These are:

- flight-id: The Aircraft’s Flight ID (Callsign).
- icao-24-bit: The Aircraft’s ICAO 24-bit Address, if available. This is expressed as six hexadecimal digits (least significant bit first).
- tail-number: The Aircraft's Tail Number (Registration Code).
- Logged-at: The UTC timestamp that the message was sent or received by the reporting system. This is in the format “yyyy/mm/dd hh:mm:ss.nnn” where:
    - yyyy is the four digit year
    - mm is the two digit month number
    - dd is the two digit day number including leading zeroes.
    - hh is the two digit hour including leading zeroes
    - mm is the two digit minutes including leading zeroes
    - ss is the two digit seconds including leading zeroes
    - nnn is the three milliseconds including leading zeroes (this and the preceding separator character may be omitted)

```
<!ENTITY % direction “direction (up|down) #IMPLIED”>
```

The parameter entity *direction* defines an attribute for use with all ATC messages for which the direction of transfer is not implicit in the message type.

3.3 Sector Transitions

```
<!ELEMENT sector-transitions (transition+) >
<!ATTLIST sector-transitions
                  reporting-ATSU            CDATA        #REQUIRED >
```

The sector-transitions element comprises one or more transition elements and has a single required attribute – the Ground Facility Designation of the reporting ATSU.

```
<!ELEMENT transition (date,flight-id,airframe,
                      (tail-number|icao-24-bit)?,
                      airport,airport,entry,exit) >
<!ATTLIST transition
                  sector                    CDATA        #REQUIRED >
```

The transition element comprises:

- the transition date,
- the flight ID (Callsign),
- the airframe type identifier (e.g. B744),
- optionally, either the tail number or the ICAO 24-bit aircraft identifier,
- the departure and destination airport identifiers, each given by an airport element one in each role,
- and entry and exit elements in that order

The sector name is a required attribute to the transition element.

```
<!ELEMENT airframe (#PCDATA) >
```

This element contains the airframe type identifier (e.g. B744).

```
<!ELEMENT tail-number (#PCDATA) >
```

This element contains the aircraft's Tail Number.

```
<!ELEMENT entry (time,level) >
<!ELEMENT exit (time,level) >
```

The entry and exit elements both comprise a time at which the event occurred and the actual flight level.

3.4 ATC Messages

ATC Messages comprises both CM and CPDLC messages and may later expand to include ADS messages.

```
<!ELEMENT atc-messages ((cm-logon-req|cm-logon-resp|cm-update|
                         cm-contact|cm-contact-resp|cm-abort|
                         cm-forward-req|cm-forward-resp|cm-end-req|
                         cpdlc-start-req|cpdlc-start-resp|
                         cpdlc-message|cpdlc-abort|
                         cpdlc-end-req|cpdlc-end-resp)+)>
<!ATTLIST atc-messages
                  reporting-ATSU            CDATA        #REQUIRED
                  protocol-stack            (atn|fans)   "atn" >
```

An *atc-messages* element comprises one or more of the various CM and CPDLC messages that may be recorded air/ground. The order of these messages is not significant, nor is the set of messages required to be complete. For example, the messages are permitted to contain a *cm-response* without a corresponding *cm-request*.

The *atc-messages* element also contains a single required attribute – the Ground Facility Designation of the reporting ATSU. This is assumed to be the ground end point for all ATC Messages contained by this element.

The *atc-messages* element may also contain an optional element identifying the protocol stack used to transfer the messages. This may be either "atn" or "fans" with "atn" as the default.

3.5 Context Management

3.5.1 CM Messages

```
<!ELEMENT cm-logon-req (flight-id,icao-24-bit?,tail-number?,tsap,
                        AEAddress*,AESupport*,
                        facilityDesignation?,airport?,
                        airport?,eotd?)>
<!ATTLIST cm-logon-req %stdatts; >
```

The CM Logon Request (cm-logon-req) element records the receipt of a CM Logon Request message by the reporting ATSU. It comprises:

- The Aircraft's Flight ID (Callsign)
- The Aircraft's ICAO 24-bit Address (optional)
- The Aircraft's Tail number (optional)
- the TSAP i.e. the aircraft's ATN Address
- The list of supported ground initiated applications, given by zero, one or more AEAddress elements
- The list of supported air initiated applications, given by zero, one or more AESupport elements.
- The Ground Facility Designation of the intended ATSU (optional)
- The departure and destination airports (optional and given by the airport element with role=departure and destination, respectively)
- The Estimated Time of Departure (optional).

Note that the Flight ID and icao-24-bit address contained in this element are those recorded from the content of the CM Logon. The Flight ID and icao 24 bit address in the standard attributes should be identical but are sourced from the dialog service instead.

```
<!ELEMENT cm-logon-resp (AEAddress*,AESupport*)>
<!ATTLIST cm-logon-resp
                  %stdatts;
                  maintain-dialog           (yes|no)     “no”>
```

The CM Logon Response (cm-logon-resp) element records the transmission of a CM Logon Response message by the reporting ATSU. It comprises:

- The list of accepted air initiated applications, given by zero, one or more AEAddress elements
- The list of accepted ground initiated applications, given by zero, one or more AESupport elements.

Note that a CM Logon Response with empty sets of air and ground initiated applications is interpreted as a negative response.

```
<!ELEMENT cm-update (tsap,AEAddress*,AESupport*)>
<!ATTLIST cm-update %stdatts; >
```

A CM Update (cm-update) element records the transmission of a CM Update message by the reporting ATSU, and is syntactically identical to a CM Logon Response. It is used by the ground to advise the aircraft of a change in the available applications.

```
<!ELEMENT cm-contact (facilityDesignation,tsap)>
<!ATTLIST cm-contact %stdatts; >
```

The CM Contact Request (cm-contact) element records the transmission of a CM Contact message by the reporting ATSU. It comprises:

- The Ground Facility Designation of the ATSU that the aircraft is instructed to contact.
- The ATN Address of that ATSU (tsap element).

```
<!ELEMENT cm-contact-resp EMPTY>
<!ATTLIST cm-contact-resp
                  %stdatts;
                  result       (success|notSuccessful)    #REQUIRED >
```

The CM Contact Response (cm-contact-resp) element records the receipt of a CM Contact message by the reporting ATSU.

The result attribute is used to indicate when the contact was successful or not successful.

```
<!ELEMENT cm-abort EMPTY>
<!ATTLIST cm-abort
                  %stdatts;
                  %direction;
                  type         (provider|user)           #REQUIRED
                  reason       (%cm-abort-reasons;) #IMPLIED >
```

The CM Abort (cm-abort) element records the transmission or receipt of a CM Abort message.

The reason attribute of the cm-abort element gives the reason for the abort, if available. The type attribute indicates whether this is a provider or user abort.

```
<!ELEMENT cm-end-req EMPTY>
<!ATTLIST cm-end-req            %stdatts;>
```

The CM End Request (cm-end-req) element records the transmission of a CM End Request message.

```
<!ELEMENT cm-forward-req (flight-id,icao-24-bit?,tail-number?,tsap,
                          AEAddress*,AESupport*,
                          facilityDesignation?,airport?,
                          airport?,eotd?)>

<!ATTLIST cm-forward-req %stdatts;
                  source-ATSU   CDATA        #REQUIRED >
```

The CM Forward Request element records the receipt of a forwarded of a CM Logon Request.

```
<!ELEMENT cm-forward-resp EMPTY>

<!ATTLIST cm-forward-resp
                  %stdatts;
                  response     (success|incompatible-version|
                                service-not-supported)   #REQUIRED >
```

The CM Forward Response element records the return of a response to a cm-forward-req.

3.5.2 CM Supporting XML Elements

```
<!ELEMENT AEAddress (AEQualifier,AEVersion,tsap)>
<!ELEMENT AEQualifier (#PCDATA)>
<!ELEMENT AEVersion (#PCDATA)>
```

The AE Address identifies a supported ATN Application, its version number and the ATN Address on which it may be found. The AEAddress element contains:

- An AEQualifier element formatted as a decimal number in the range 0 to 255.
- An AEVersion element formatted as a decimal number in the range 1 to 255.
- A tsap containing the application's ATN Address.

```
<!ELEMENT AESupport (AEQualifier,AEVersion)>
```

An AE Support element identifies a supported ATN Application and its version number only.

```
<!ELEMENT tsap (rdp?,ars?,locSysNselTsel)>
<!ELEMENT rdp (#PCDATA)>
<!ELEMENT ars (#PCDATA)>
<!ELEMENT locSysNselTsel (#PCDATA)>
```

A tsap comprises:

- an optional Routing Domain Part (rdp) formatted as 10 hexadecimal characters.
- an optional Administrative Region Selector (ars) formatted as 6 hexadecimal characters.
- The remaining components of the network address plus the transport selector, formatted as 20 or 22 hexadecimal characters.

Note that ATN Addresses are binary numbers formatted as hexadecimal in a “little endian” manner. When the rdp element is present this corresponds to the CM ASN.1 longTSAP syntax. Otherwise, this corresponds to the shortTSAP ASN.1 syntax.

```
<!ELEMENT eotd (date,time)>
```

The Estimated Time of Departure (eotd) element comprises date and time elements, which together define the estimated time of departure.

3.6 CPDLC

3.6.1 CPDLC Messages

```
<!ENTITY % message “header,msg-element+,route-clearance?,
                    route-clearance?” >
```

The parameter entity message defines the elements that make up a CPDLC message. These are:

- the CPDLC header,
- one to five message elements, and
- zero to two route clearances.

```
<!ELEMENT cpdlc-start-req ((%message;)?)>
<!ATTLIST cpdlc-start-req
                  %stdatts;
                  %direction;
                  mode         (cpdlc|dsc)              #REQUIRED>
```

The CPDLC Start Request (cpdlc-start-req) element records the transmission or receipt of a CPDLC Start Request message by the reporting ATSU. It comprises:

- A CPDLC Message (optional)

The cpdlc-start-req element also contains a mode attribute to indicate whether it is for a CPDLC or DSC connection. Note that “dsc” is only valid for aircraft initiated CPDLC Start Requests.

```
<!ELEMENT cpdlc-start-resp ((%message;)?)>
<!ATTLIST cpdlc-start-resp
                  %stdatts;
                  %direction;
                  result       (accepted|rejected)       #REQUIRED>
```

The CPDLC Start Response (cpdlc-start-resp) element records the transmission or receipt of a CPDLC Start Response message by the reporting ATSU. It comprises:

- A CPDLC Message (optional, and only permitted with result=”rejected”).

The cpdlc-start-resp element also contains two required attributes to indicate whether this was an uplink or downlink CPDLC Start Response and, the result of the original request (result=accepted or rejected).

```
<!ELEMENT cpdlc-message (%message;) >
<!ATTLIST cpdlc-message
                  %stdatts;
                  %direction;>
```

The cpdlc-message element comprise a CPDLC message.

```
<!ELEMENT cpdlc-abort EMPTY >
<!ATTLIST cpdlc-abort
                  %stdatts;
                  %direction;
                  type         (provider|user)           #REQUIRED
                  reason       (%cpdlc-user-abort-reasons;|
                                %cpdlc-provider-abort-reasons;) #REQUIRED>
<!ENTITY % cpdlc-user-abort-reasons
                  “undefined|
                  no-message-identification-numbers-available|
                  duplicate-message-identification-numbers|
                  no-longer-next-data-authority|
                  current-data-authority-abort|
                  commanded-termination|
                  invalid-response|
                  time-out-of-synchronisation|
                  unknown-integrity-check|
                  validation-failure|
                  unable-to-decode-message|
                  invalid-pdu|
                  invalid-CPDLC-message” >

<!ENTITY % cpdlc-provider-abort-reasons
                  "timer-expired|undefined-error|invalid-PDU|
                  protocol-error|
                  communication-service-error|
                  communication-service-failure|
                  invalid-QOS-parameter|
                  expected-PDU-missing" >
```

The cpdlc-abort element records that a CPDLC connection has been aborted.

The reason attribute of the cpdlc -abort element gives the reason for the abort. The type attribute indicates whether this is a provider or user abort, and the direction attribute indicates, for a User Abort, whether it was aircraft (direction=down) or ground (direction=up) initiated.

```
<!ELEMENT cpdlc-end-req ((%message;)?)>
<!ATTLIST cpdlc-cpdlc-end-req
                  %stdatts;
                  %direction;>
```

The CPDLC End Request (cpdlc-end-req) element records the transmission or receipt or a CPDLC End Request message by the recording ATSU. It comprises:

- A cpdlc-message, if present.

```
<!ELEMENT cpdlc-end-resp ((%message;)?)>
<!ATTLIST cpdlc-cpdlc-end-resp
                  %stdatts;
                  %direction;
                  result       (accepted|rejected)       #REQUIRED>
```

The CPDLC End Response (cpdlc-end-resp) element records the transmission or receipt or a CPDLC End Response message by the recording ATSU. It comprises:

- A cpdlc-message, if present.

3.6.2 CPDLC Support XML Elements

```
<!ELEMENT header (msg-id,msg-ref?,date,time,logical-ack-req?) >
<!ELEMENT msg-id (#PCDATA)>
<!ELEMENT msg-ref (#PCDATA)>
<!ELEMENT logical-ack-req EMPTY>
```

The message header comprises:

- the message identifier (an integer, range 0 to 63),
- an optional message reference (if this is a response message),
- the message timestamp (date and time elements)
- an indication of a logical ack request (element present), no ack requested (element not present).

```
<!ELEMENT msg-element (%params;) >
<!ATTLIST msg-element
                  id           CDATA        #REQUIRED
                  error        (unrecognizedMsgReferenceNumber|
                                logicalAcknowledgmentNotAccepted|
                                insufficientResources|
                                invalidMessageElementCombination|
                                invalidMessageElement) #IMPLIED >
```

The msg-element comprises an optional parameter list and is required to have a single attribute – the message element number (id). This is a number in the range specified by ICAO. UM159 and DM62 may also have an error attribute giving error information.

The parameter list is defined by a parameter entity which is defined below.

3.6.3 CPDLC Message Element Parameters

```
<!ENTITY  % params         "(level|time|position|speed|distance|
                           direction|degrees|
                           departureClearance|routeClearanceIndex|
                           procedure|holdClearance|
                           unitName|code|speedType|frequency|
                           altimeter|atisCode|positionReport|
                           facilityDesignation|traffic|freeText|
                           verticalRate|runwayRVR| clearance|
                           remainingFuelPersonsOnBoard|
                           versionNumber|airport)*">
```

The parameter entity "params" defines the possible parameters for a CPDLC message element. The syntax allows for one or more of each of the above parameters to be provided in any order and each parameter may appear more than once. Semantically, the number of parameters for each CPDLC message element is fully defined by the ICAO SARPs and their order is significant for interpretation. However, this is not checked by the XML DTD.

For a given message element the required set of parameters must be present in the order specified in the SARPs, or the parameters omitted altogether (deprecated and supported only for compatibility with Maastricht old log format data).

```
<!ELEMENT altimeter (#PCDATA)>
<!ATTLIST altimeter
                  units        (inches|hectopascals)     #REQUIRED >
```

The altimeter setting may be given in inches of mercury or Hectopascals:

- Inches Mercury: Range (22.00 .. 32.00), resolution = 0.01
- Hectopascal: Range (750.0..1250.0), resolution = 0.1

```
<!ELEMENT atisCode (#PCDATA)>
```

The one digit ATIS Code.

```
<!ELEMENT clearance EMPTY>
<!ATTLIST clearance
                  type         (noneSpecified|approach|departure|
                                further|start-up|pushback|taxi|
                                take-off|landing|oceanic|
                                en-route|downstream)     #REQUIRED>
```

The clearance element has a single required attribute, the type of the clearance.

```
<!ELEMENT code (#PCDATA)>
```

Constrained to four octal digits (the Mode A Code).

```
<!ELEMENT degrees (#PCDATA)>
<!ATTLIST degrees
                  units        (magnetic|true|latitude|
                                longitude)              #REQUIRED
                  role         (bearing|heading|trackAngle)
                                                        #IMPLIED>
```

A bearing measured in degrees, measured as:

- magnetic: Range (1..360), resolution = 1
- true: Range (1..360), resolution = 1
- latitude: Range (0..90), resolution = 0.001
- longitude: Range (0..180), resolution = 0.001

A degrees element is usually used as a bearing, but in a position report can also be used as a heading or track angle, as indicated by the role attribute. A degrees element may also be

```
<!ELEMENT direction EMPTY >
<!ATTLIST direction
                  value        (left|right|eitherSide|north|south|
                                east|west|northEast|northWest|
                                southEast|southWest)     #REQUIRED >
```

The direction element has a single attribute “value” which gives the required direction.

```
<!ELEMENT distance (#PCDATA) >
<!ATTLIST distance
                  units        (nautical-miles|kilometres)
                                                        #REQUIRED
                  atwTolerance (plus|minus)  #IMPLIED
                  specified    (yes|no)      “no” >
```

The distance is specified in either nautical miles or kilometres and may be expressed as either a whole number or with a decimal fraction:

- If specified=”no”:
    - Kilometres, Range (0..2000), resolution = 0.25,
    - Nautical Miles: Range (0..999.9), resolution = 0.1.
- If specified-“yes”
    - Kilometres, Range (1..500), resolution = 1,
    - Nautical Miles: Range (Range (1..250), resolution = 1.

Note the specified=”yes” corresponds to use in the CPDLC ASN.1 type distanceSpecifiedDirection.

The ATW Tolerance attribute is only permitted when the distance is contained within an atwAlongTrackWaypoint element.

```
<!ELEMENT facilityName (#PCDATA)>
```

The facility name is the plain text name of the facility.

```
<!ELEMENT fix (latLong?)>
<!ATTLIST fix
                  name          CDATA                       #REQUIRED>
```

The fix has a single attribute which is the actual fix name and may have an optional element – the lat-long co-ordinates of the fix.

```
<!ELEMENT frequency (#PCDATA)>
<!ATTLIST frequency
                  band          (hf|vhf|uhf|sat)            #REQUIRED>
```

Depending on the specified band, the frequency is interpreted as:

- hf: Kilohertz, Range (2850..28000), resolution = 1
- vhf: Megahertz, Range (118.000..136.990), resolution = 0.005
- uhf: Megahertz, Range (225.000..399.975), resolution = 0.025
- sat: corresponds to a 12 digit telephone number

```
<!ELEMENT freetext (#PCDATA)>
```

Up to 256 characters of free text.

```
<!ELEMENT humidity (#PCDATA)>
```

This element contains the reported humidity (unit = Percent humidity, Range (0..100), resolution = 1).

```
<!ELEMENT icing EMPTY>
<!ATTLIST icing
                  value         (trace|light|moderate|severe)    #REQUIRED>
```

The value attribute of this element gives the reported icing level.

```
<!ELEMENT latLong (latitude?,longitude?)>
<!ELEMENT latitude (degrees,minutes?,seconds?)>
<!ATTLIST latitude
                  direction     (north|south)“north” >
<!ELEMENT longitude (degrees,minutes?,seconds?)>
<!ATTLIST longitude
                  direction     (east|west)                 “east”>
```

The latLong element defines a latitude and longitude. Each is expressed in degrees, minutes and seconds and a direction, north/south or east/west respectively. The latitude or longitude may be expressed as either:

- Degrees including a decimal fraction of a degree
- Whole degrees and minutes, including a decimal fraction of a minute
- Whole degrees, minutes and seconds.

The type attribute of the degrees element must be set to “latitude” or “longitude” as appropriate.

```
<!ELEMENT minutes (#PCDATA)>
```

Minutes of a degree expressed as an integer or as a fixed point number with up to two decimal places (Range (0..59.99), resolution = 0.01).

```
<!ELEMENT navaid (latLong?)>
<!ATTLIST navaid
                  name         CDATA                      #REQUIRED>
```

The navaid has a single attribute which is the actual navaid name and may have an optional element – the lat-long co-ordinates of the fix.

```
<!ELEMENT position (((fix|navaid),(degrees,distance)?)|
                              airport|latLong) >
<!ATTLIST position
                  direction    (to|from)                  #IMPLIED
                  role         (current|next|nextPlusOne|waypoint)
                                                          #IMPLIED>
```

The position may be expressed as either a fix or navaid, with an optional degrees and distance specification, or as an airport identifier or a Lat-Long.

When the direction attribute is present, this element corresponds to the ASN.1 *toFromPosition* type.

The position role is used to distinguish different types of positions in a Position Report.

```
<!ELEMENT positionReport (position+,time+,level+,
                            temperature?,winds?,turbulence?,
                            icing?,speed*,verticalRate?,degrees*,
                            distance?,humidity?) >
```

A Position Report comprises:

- The current Position
- The time at the current position (time element, role=current)
- The current level (level element, role=current), and

- Optionally:
    - The position at the next fix (position element, role=next)
    - The time at the next fix (time element, role=next)
    - The position at the next fix plus one (position element, role=nextPlusOne)
    - The time at the next fix plus one (time element, role=nextPlusOne)
    - The time at the destination (time element, role=destination)
    - The Remaining Fuel On Board (time element, role=remainingFuel)
    - The Temperature
    - The Winds
    - Turbulence
    - Icing
    - Speed (speed element, type ≠ ground or groundMetric)
    - Ground Speed (speed element, type=ground or groundMetric)
    - The vertical rate of change (includes a direction attribute)
    - The track angle (degrees element, role=trackAngle)
    - The heading (degrees element, role=heading)
    - the distance
    - The Humidity
    - Reported Waypoint Position (position element, role=waypoint)
    - Report Waypoint Time (time element, role=waypoint)
    - Reported Waypoint Level (level element, role=waypoint)

```
<!ELEMENT procedure            (procedure-transition?) >
<!ATTLIST procedure
                  name         CDATA                          #REQUIRED
                  type         (arrival|approach|departure) #REQUIRED>
<!ELEMENT procedure-transition (#PCDATA)>
```

The procedure element has a name attribute and a “type” attribute that identifies the type of the procedure. It may also contain a procedure transition name.

```
<!ELEMENT remainingFuelPersonsOnBoard (time,persons) >
<!ELEMENT persons (#PCDATA)>
```

The remaining fuel is given by the time (role=remainingFuel), while the remaining persons, is the value of the persons element and given as an integer.

```
<!ELEMENT routeClearanceIndex EMPTY>
<!ATTLIST routeClearanceIndex
                  index        (1|2)                      #REQUIRED>
```

The index attribute of this element is an index into the list of route clearances provided with this CPDLC message.

```
<!ELEMENT runwayRVR (runway,rvr) >
<!ELEMENT runway (#PCDATA) >
<!ATTLIST runway
                  config       (left|right|center|none) #REQUIRED
                  role         (departure|arrival)        #IMPLIED>
```

The runway element contains the runway direction given as an integer in the range 1 to 36, with the runway configuration given by the config attribute. The optional attribute role may be used where it is necessary to distinguish whether the departure or destination runway is being referenced.

```
<!ELEMENT rvr (#PCDATA)>
<!ATTLIST rvr
                  units        (feet|metres)#REQUIRED>
```

The Runway Visible Range (RVR) is given either in feet or metres:

- feet: Range (0..6100), resolution = 1
- metres: Meters (0..1500), resolution = 1

```
<!ELEMENT seconds (#PCDATA)>
```

An integer in the range 0 to 60.

```
<!ELEMENT speed (#PCDATA)>
<!ATTLIST speed
                  units        (indicated|indicatedMetric|true|
                                trueMetric|ground|groundMetric|
                                mach|windSpeed|windSpeedMetric)
                                                          #REQUIRED
                  use          (low|high)                 #IMPLIED >
```

The speed element contains the value of the current speed, which must be interpreted according to the units attribute:

- indicated: unit = Knots, Range (0..400), resolution = 1
- indicatedMetric: unit = Kilometers/Hour, Range (0..800), resolution = 1
- true: unit = Knots, Range (0..2000), resolution = 1

- trueMetric: unit = Kilometers/Hour, Range (0..4000), resolution = 1
- ground: unit = Knots, Range (-50..2000), resolution = 1
- groundMetric: unit = Kilometers/Hour, Range (-100..4000), resolution = 1
- mach: unit = Mach Range (0.5 to 4.0), resolution = 0.001
- windSpeed: unit = Knot, Range (0..255), resolution = 1
- windSpeedMetric: unit = Kilometer/Hour, Range (0..511), resolution = 1

A speed "use" needs to be provided when the speed use used in a HoldAtWaypoint element. Otherwise, it is ignored.

```
<!ELEMENT speedType EMPTY>
<!ATTLIST speedType
                  type         (noneSpecified|indicated|true|ground|
                                mach|approach|cruise|minimum|maximum)
                                                          #REQUIRED >
```

The speed type is given by its type attribute.

```
<!ELEMENT temperature (#PCDATA)>
```

The temperature element contains the temperature
(unit = Degree Celsius, Range (-100..100), resolution = 1).

```
<!ELEMENT traffic EMPTY>
<!ATTLIST traffic
                  type         (noneSpecified|oppositeDirection|
                                sameDirection|converging|
                                crossing|diverging)      #REQUIRED>
```

The traffic element has a single required attribute giving the traffic type.

```
<!ELEMENT turbulence EMPTY>
<!ATTLIST turbulence
                  value        (light|moderate|severe)    #REQUIRED>
```

This value attribute of this element gives the indicated turbulence.

```
<!ELEMENT unitName (facilityDesignation,facilityName?)>
<!ATTLIST unitName
                  function     (center|approach|tower|final|
                                groundControl|clearanceDelivery|
                                departure|control|radio) #REQUIRED>
```

The unit name comprises the ICAO facility designation and an optional facility name. It has a single attribute, which identifies the facility function.

```
<!ELEMENT versionNumber (#PCDATA)>
```

The version number is an integer in the range 0 to 15.

```
<!ELEMENT verticalRate (#PCDATA)>
<!ATTLIST verticalRate
                  units        (feetPerMinute|metresPerMinute)
                                                    #REQUIRED
                  direction    (up|down)            #IMPLIED>
```

The vertical rate is given either as:

- feet/minute: Range (0..30000), resolution = 10
- Meters/Minute: Range (0..10000), resolution = 10

A direction may also be required. For example, when used in a position report.

```
<!ELEMENT winds (WindDirection,speed)>
```

The wind is given by a direction and the speed of the wind. The speed element must have a type of windSpeed or WindSpeedMetric.

```
<!ELEMENT WindDirection (#PCDATA)>
```

This element contains the wind direction
(unit = Degree, Range (1..360), resolution = 1).

3.6.4 Clearances

```
<!ELEMENT atsRouteDesignator (#PCDATA)>
```

The ATS Route Designator is a character string from two to seven characters.

```
<!ELEMENT aTWAlongTrackWaypoint (position,distance,speed?,
                                    level?,level?) >
```

An ATW Along Track Waypoint comprises a position and distance and, optionally, a speed and one or two levels. Both the distance and level elements must include an atwTolerance attribute.

```
<!ELEMENT degreeIncrement (#PCDATA)>
```

This elements contains a degree increment for reporting points
(unit = Degree, Range (1..20), resolution = 1).

```
<!ELEMENT departureClearance (flight-id,position,flightInformation?,
                              code?,(unitName,frequency)?,time?,
                              airport?,airport?,departureTime?,
                              runway?,
                              revisionNumber?,atisCode?) >
```

A departure clearance comprises:

- The Flight ID (Callsign)
- the clearance limit *position*,
- flight information (optional),
- the Mode A *code* (optional),
- the Unit Name and Frequency for the Departure (optional)
- the clearance expiry *time* (optional),
- the departure airport (optional),
- the destination airport (optional),
- the departure time (optional),
- the departure runway (optional)
- the revision number (optional), and
- the ATIS Code (optional).

Note that the role attribute of the airport element is required to distinguish the airport elements. Use of the runway role element is recommended but is not essential in this context.

```
<!ELEMENT departureTime (time?,time?,time?,departureMinInterval?)>
```

The departure time can be expressed as some or all of:

- The allocated departure time (time element, role=departure)
- The controlled time (time element with a tolerance attribute)
- The time a departure clearance is expected (time element, role=clearanceExpected)
- The minimum departure interval.

```
<!ELEMENT departureMinInterval (#PCDATA)>
```

unit = Minute, Range (0.1..15.0), resolution = 0.1

```
<!ELEMENT flightInformation (routeInformation|levelsOfFlight|
                             (routeInformation,levelsOfFlight)) >
```

Flight Information comprises either route information, levels of flight, or both route information and levels of flight.

```
<!ELEMENT holdAtWaypoint (position,speed?,level?,speed?,direction?,
                          degrees?,time?,leg?)>
```

The Hold At Waypoint is described by:

- Its position, and optionally:
    - The low speed (speed element, use=low)
    - The Level (level element including an atwTolerance attribute)
    - The high speed (speed element, use=high)
    - The direction
    - The bearing (degrees element)
    - The eFC Time
    - The indicated leg

```
<!ELEMENT holdClearance (position,level,degrees,direction,leg?)>
```

The Hold Clearance comprises a position, level, a bearing, a direction and optionally *leg* information.

```
<!ELEMENT interceptCourseFrom ((%publishedIdentifier;|latLong|
                                (placeBearing,(placeBearing|
                                 distance))),degrees)>
```

A intercept course is defined by either:

- A published identifier
- A Latitude Longitude co-ordinate,
- a “place bearing” and another “place bearing” or a distance.

It also contains a degrees element used as a bearing.

```
<!ELEMENT leg (#PCDATA)>
<!ATTLIST leg
                  units        (distance-nm|distance-km|time)
                                                    #REQUIRED>
```

A leg is expressed as a distance in nautical miles, a distinance in kilometres or as a time:

- Nautical Mile, Range (0..50), resolution = 1
- Kilometre, Range (1..128), resolution = 1
- Time: Minutes, Range (0..10), resolution = 1

```
<!ELEMENT levelsOfFlight (level|procedure|(level,procedure))>
```

The levels of flight information is either level information, a procedure or a level followed by a procedure.

```
<!ENTITY % publishedIdentifier “(fix|navaid)” >
```

A published identifier is either a fix or a navaid.

```
<!ELEMENT placeBearing (%publishedIdentifier;,degrees)>
```

A place bearing is bearing from a fix or navaid.

```
<!ELEMENT route-clearance       ((airport|runway|procedure)*,
                                 routeInformation*,
                                 aTWAlongTrackWaypoint*,
                                 reportingPoints?,
                                 interceptCourseFrom*,
                                 holdAtWaypoint*,
                                 waypointSpeedLevel*,
                                 rTARequiredTimeArrival*) >
<!ATTLIST route-clearance
                  index                      (1|2)        “1”>
```

The route clearance may comprise:

- The Departure Airport
- The Destination Airport
- The Departure Runway
- The Departure Procedure
- The Arrival Runway
- The Approach Procedure
- The Arrivals Procedure
- Up to 128 Route Information items
- Up to 8 ATW Along Track Waypoints
- Reporting Points

- Up to 4 Intercept Course From points
- Up to 32 waypoint speed levels
- Up to 32 Require Time of Arrival points

Note that the airport, runway and procedure elements must all include their role attributes in order to distinguish the context in which they are used.

The route-clearance has a single attribute, its index number which may be either 1 or 2.

```
<!ELEMENT reportingPoints ((latitude|longitude),
                           degreeIncrement)>
```

The reportingPoints element comprises a latitude reporting point or a longitude reporting point, plus a degree increment. The latitude or longitude is restricted to containing a latitude degrees element only, or a longitude degrees element only.

```
<!ELEMENT revisionNumber (#PCDATA)>
```

The revision number is an integer in the range 1 to 16.

```
<!ELEMENT routeInformation      (%publishedIdentifier;|latLong|
                                 (placeBearing,
                                  (placeBearing|distance))|
                                 atsRouteDesignator)>
```

Route Information is either a published identifier, a lat-long co-ordinate, a place bearing followed by another place bearing or a distance, or an ATS Route Designator.

```
<!ELEMENT rTARequiredTimeArrival (position,time,rtaTolerance?)>
```

The rTARequiredTimeArrival element comprises a position and arrival time, and an optional tolereance.

```
<!ELEMENT rtaTolerance (#PCDATA)>
```

The rtaTolerance element contains the tolerance for the required time of arrival (unit= Minute, Range (0.1..15.0), resolution = 0.1).

```
<!ELEMENT waypointSpeedLevel (position,speed?,level?,level?)>
```

The waypoint speed levels comprise:

- A position, and optionally:
    - A speed,

- One or two levels (atwTolerance attribute required)

3.7 Position Reports

The reports given in the Position Reports section are typically obtained from surveillance data. CPDLC position reports are described in 3.6.

```
<!ELEMENT position-reports (position-report+) >
<!ATTLIST position-reports
                  Reporting-ATSU            CDATA        #REQUIRED >
```

The postion-reports element comprises one or more position reports and has a single required attribute – the Ground Facility Designation of the reporting ATSU.

```
<!ELEMENT position-report (level,latLong,x-velocity?,y-velocity?) >
<!ATTLIST position-report %stdatts; >
```

The position-report element identifies the flight and the time of the position report through the standard attribute set and its Level, Lat Long and optional X and Y velocity through the element contents.

```
<!ENTITY x-velocity (#PCDATA) >
<!ATTLIST x-velocity
                  units        (indicated|indicatedMetric|true|
                                trueMetric|ground|groundMetric|
                                mach|windSpeed|windSpeedMetric)
                                                        #IMPLIED >
```

The x-velocity gives the component of the aircraft's speed in the X (west to east) direction.

```
<!ENTITY y-velocity (#PCDATA) >
<!ATTLIST y-velocity
                  units        (indicated|indicatedMetric|true|
                                trueMetric|ground|groundMetric|
                                mach|windSpeed|windSpeedMetric)
                                                        #IMPLIED >
```

The y-velocity gives the component of the aircraft's speed in the Y (south to north) direction

3.8 Transport Logs

3.8.1 Network Trace

```
<!ELEMENT network-trace (clnp-packet*)>

<!ATTLIST network-trace
                  reporting-ATSU            CDATA        #REQUIRED>
```

The network trace element identifies the reporting ATSU and is a container for a set of CLNP packets.

3.8.2 CLNP

3.8.2.1 The CLNP Packet Element

```
<!ENTITY  %       priority
                  "(0|1|2|3|4|5|6|7|8|9|10|11|12|13|14|15)">

<!ELEMENT clnp-packet (destination-nsap,source-nsap,data-unit-identifer?,
lifetime,
checksum,globally-unique-qos?,atn-security?,npdu-data?,
(cr-tpdu|cc-tpdu|dr-tpdu|dc-tpdu|dt-tpdu|ed-tpdu|ak-tpdu|ea-tpdu|er-tpdu)*)
>
<!ATTLIST clnp-packet
                  type
          (data|error-report|echo-request|echo-response)   #REQUIRED
                  logged-at                 CDATA          #REQUIRED
                  bad-checksum              (yes|no)       "no"
                  congestion-experienced    (yes|no)       "no"
                  discard-reason            CDATA          #IMPLIED
                  priority                  %priority;     "0"
                  direction                 (uplink|downlink) #REQUIRED>
```

The clnp-packet is not intended to be a proper representation of a CLNP packet. Instead, its objective is to record the information needed by LISAT.

It comprises a destination and source NSAP Address, the PDU lifetime, Checksum, Globally Unique Qos, if present, the ATN Security Label if present, and the user data payload.

The element also has attributes to indicate whether or not congestion was experienced (i.e. the CE bit is set), the packet priority, the time it was logged and direction of transfer. It may also include a discard reason code, if the DL-FEP discarded the packet. A bad checksum is also flagged if the checksum cannot be validated.

3.8.2.2 Destination and Source NSAP Addresses

```
<!ELEMENT destination-nsap (#PCDATA) >
<!ELEMENT source-nsap (#PCDATA) >
```

The Destination and Source NSAP Addresses are encoded as hexadecimal character strings.

3.8.2.3 Data Unit Identifier

```
<!ELEMENT data-unit-idenitifer (#PCDATA) >
```

The Data Unit Identifier is given as four hex digits.

3.8.2.4 PDU Lifetime

```
<!ELEMENT lifetime (#PCDATA) >
```

The PDU Lifetime is given as a decimal number in units of 500 milliseconds.

3.8.2.5 Checksum

```
<!ELEMENT checksum (#PCDATA) >
```

The checksum is given as four hex digits.

3.8.2.6 Globally Unique Qos

```
<!ELEMENT globally-unique-qos (sequencing-vs-transit-delay?,
    transit-delay-vs-cost?, residual-error-prob-vs-transit-delay?,
    residual-error-prob-vs_cost?) >
```

The globally unique qos element may contain up to four elements, when present, the sub-element indicates that the corresponding flag has been set. These elements are always empty.

```
<!ELEMENT sequencing-vs-transit-delay EMPTY>
<!ELEMENT transit-delay-vs-cost EMPTY>
<!ELEMENT residual-error-prob-vs-transit-delay EMPTY>
<!ELEMENT residual-error-prob-vs_cost EMPTY>
```

3.8.2.7 The ATN Security Label

```
<!ELEMENT atn-security (atsc-tag|aoc-tag) >
```

The ATN Security Label can be either an ATSC Security tag or an AOC Security Tag.

```
<!ELEMENT atsc-tag EMPTY>
<!ATTLIST atsc-tag
          class    (U|A|B|C|D|E|F|G|H)       "U" >
```

The ATSC Tag identified the traffic class preference expressed by the originator. This is given explicitly by the *class* attribute. When not present, or when given the value "U", the semantic is "No Traffic Type Policy Preference". This element has no sub-elements.

```
<!ELEMENT aoc-tag EMPTY>
<!ATTLIST aoc-tag
          via      (gatelink|vdl|satcom|hf|mode_s|gatelink-then-vdl|
                   gatelink-then-vdl-then-satcom|
                   gatelink-then-vdl-then-hf-then-satcom)  #IMPLIED>
```

The AOC tag expresses the originator's air/ground routing preference. This can specify the only air/ground network permitted, or an ordered list in preference order. Note that only certain specific orderings are allowed by the SARPs. When no attribute is present then "No Traffic Type Policy Preference" is given. This element has no sub-elements.

3.8.2.8 NPDU User Data

```
<!ELEMENT npdu-data (#PCDATA) >
```

The undecoded data portion of an NPDU, if present, is given as a string of hexadecimal digits in transmission order.

3.8.3 TP4

The CLNP user data in a data NPDU is decoded into one or more tpdu. The resulting TPDUs may also be included in the clnp-packet element in the order in which they are decoded.

3.8.3.1 CR TPDU

```
<!ELEMENT cr-tpdu (calling-tsap,called-tsap,tp4-version?, src-ref,
                   tpdu-size?,ack-time,inactivity-time,
                   initial-credit, priority,checksum?, atn-checksum?,
                   user-data?)>

<!ATTLIST cr-tpdu
                   bad-checksum              (yes|no)     "no"
                   allow-expedited           (yes|no)     "no"
                   extended-format           (yes|no)     "no">
```

The CR TPDU is used to transfer a Transport Connect Request. The elements it contains are the connect request parameters.

3.8.3.2 CC TDU

```
<!ELEMENT cc-tpdu (calling-tsap,called-tsap,tp4-version?,dst-ref,src-ref,
                   tpdu-size?, ack-time,inactivity-time,initial-credit,
                   priority?, checksum?, atn-checksum?,user-data?)>

<!ATTLIST cc-tpdu
                   bad-checksum              (yes|no)     "no"
                   allow-expedited           (yes|no)     "no"
                   extended-format           (yes|no)     "no">
```

The CC TPDU is used to transfer a transport connect confirm. The elements it contains are the connect request parameters.

3.8.3.3 DR TPDU

```
<!ELEMENT dr-tpdu (dst-ref,src-ref, checksum?,
                   atn-checksum?,user-data?) >

<!ATTLIST dr-tpdu
                   bad-checksum              (yes|no)     "no"
                   dr-reason (user-disconnect|remote-te-congestion|
          connection-negotiation-failed|duplicate-src-reference|
          Mismatched-references|protocol-error|reference-overflow|
          cr-refused|hdr-or-param-length-invalid|reason-not-specified|
          congestion-at-tsap|session-entity-not-attached|
          address-unknown) #REQUIRED>
```

The DR TPDU is used to transfer a transport disconnect request.

3.8.3.4 DC TPDU

```
<!ELEMENT dc-tpdu (dst-ref, src-ref, checksum?,
                   atn-checksum?) >

<!ATTLIST dc-tpdu
                   bad-checksum              (yes|no)     "no">
```

The DC TPDU is used to transfer a transport disconnect confirm.

3.8.3.5 DT TPDU

```
<!ELEMENT dt-tpdu (dst-ref,tpdu-nr, checksum?,
                   atn-checksum?, user-data) >

<!ATTLIST dt-tpdu
                   bad-checksum              (yes|no)     "no"
          eot      (yes|no)                  “yes” >
```

The DT TPDU is used to transfer a user data fragment. Each fragment has a sequence number unique within the current transmission window. If the EOT attribute is set, then this signals the end of the TSDU.

3.8.3.6 ED TPDU

```
<!ELEMENT ed-tpdu (dst-ref,tpdu-nr, checksum?,
                   atn-checksum?, user-data) >

<!ATTLIST ed-tpdu
                   bad-checksum              (yes|no)     "no">
```

The ED TPDU is used to transfer an expedited user data fragment outside of the normal flow control window. The user data is limited to 16 octets.

3.8.3.7 AK TPDU

```
<!ELEMENT ak-tpdu (dst-ref,yr-tpdu-nr,credit,ak-subsequence-nr?,
                   flow-control-confirmation?,selective-ak*, checksum?,
                   atn-checksum?) >

<!ATTLIST ak-tpdu
                   bad-checksum              (yes|no)     "no">
```

The AK TPDU is used to acknowledge the successful receipt of one or more DT TPDUs and/or to update the transmission credit.

3.8.3.8 EA TPDU

```
<!ELEMENT ea-tpdu (dst-ref,yr-tpdu-nr, checksum?,
                   atn-checksum?) >

<!ATTLIST ea-tpdu
                   bad-checksum              (yes|no)     "no">
```

The EA TPDU is used to acknowledge successful receipt of an expedited TPDU.

3.8.3.9 ER TPDU

```
<!ELEMENT er-tpdu (dst-ref, invalid-tpdu?, checksum?,
                   atn-checksum?) >

<!ATTLIST er-tpdu
                   bad-checksum              (yes|no)     "no"
                   reject-cause (not-specified| invalid-parameter-code|
       invalid-tpdu-type| invalid-parameter-value) #REQUIRED>
```

The ER TPDU is used to report rejection of an invalid. TPDU

3.8.4 TPDU Elements

3.8.4.1 Checksum

```
<!ELEMENT checksum (#PCDATA) >
```

The TP4 standard checksum is encoded as a two byte bit pattern expressed as four hexadecimal digits. The byte order is the same as encoded into the TPDU.

3.8.4.2 ATN Checksum

```
<!ELEMENT atn-checksum (#PCDATA) >
```

The ATN Extended checksum is encoded as a four byte bit pattern expressed as eight hexadecimal digits. The byte order is the same as encoded into the TPDU.

3.8.4.3 TSAP Identifiers

```
<!ELEMENT calling-tsap (#PCDATA) >
<!ELEMENT called-tsap (#PCDATA) >
```

These elements contain addressed information used to convey the sender and destination TSAP Identifiers. Each identifier is given as a hexadecimal byte stream in the same order as encoded in the TPDU header.

3.8.4.4 Destination and Source References

```
<!ELEMENT dst-ref (#PCDATA) >
<!ELEMENT src-ref (#PCDATA) >
```

Each of these elements contains the 2 byte destination and source references expressed as four hexadecimal digits. The bytes are given in the order encoded into the TPDU header.

3.8.4.5 TPDU Size

```
<!ELEMENT tpdu-size (#PCDATA) >
```

The TPDU Size element is used to convey the maximum TPDU size accepted by the sender to the CR or CC TPDU. This is expressed as a decimal number.

3.8.4.6 TP4 Version

```
<!ELEMENT tp4-version (#PCDATA) >
```

The TP4 version number contains the protocol version number expressed as a decimal number. If omitted, then the default is one.

3.8.4.7 ACK Time

```
<!ELEMENT ack-time (#PCDATA) >
```

The ACK Time element is used to convey the acknowledgement time (delay) used by the sender. It is expressed as a decimal number in milliseconds.

3.8.4.8 Inactivity Time

```
<!ELEMENT inactivity-time (#PCDATA) >
```

The inactivity time is used to convey the inactivity time used by the sender. It is expressed as a decimal number in milliseconds.

3.8.4.9 Initial Credit

```
<!ELEMENT initial-credit (#PCDATA) >
```

The initial credit is used in a CR/CC TPDU to convey the initial transmission credit offered by the sender. It is expressed as a decimal number.

3.8.4.10 Priority

```
<!ELEMENT priority (#PCDATA) >
```

The priority, when present, is expressed as a decimal number.

3.8.4.11 User Data

```
<!ELEMENT user-data (#PCDATA) >
```

The user data is expressed as a hexadecimal byte stream encoded in the same order as in the TPDU.

3.8.4.12 TPDU Number

```
<!ELEMENT tpdu-nr (#PCDATA) >
```

This element contains the TPDU sequence number expressed as a decimal number.

3.8.4.13 Your TPDU Number

```
<!ELEMENT yr-tpdu-nr (#PCDATA) >
```

This element contains the returned sequence number given in an AK or EA TPDU, expressed as a decimal number.

3.8.4.14 Credit

```
<!ELEMENT credit (#PCDATA) >
```

This element contains the transmission credit offered by the sender in an AK TPDU, expressed as a decimal number.

3.8.4.15 AK Subsequence Number

```
<!ELEMENT ak-subsequence-nr (#PCDATA) >
```

This element contains the sender's ack subsequence number given in an AK TPDU, expressed as a decimal number.

3.8.4.16 Flow Control Confirmation

```
<!ELEMENT flow-control-confirmation (lower-window-edge?,
                  yr-subsequence-nr?,yr-credit) >
```

3.8.4.17 Lower Window Edge

```
<!ELEMENT lower-window-edge (#PCDATA) >
```

This element contains the lower-window edge, expressed as a decimal number.

3.8.4.18 Your Subsequence Number

```
<!ELEMENT yr-subsequence-nr (#PCDATA) >
```

This element returns the last received ack subsequence number given in an AK TPDU, expressed as a decimal number.

3.8.4.19 Your Credit

```
<!ELEMENT yr-credit (#PCDATA) >
```

This element returns the last received transmission credit given, expressed as a decimal number.

3.8.4.20 Selective Acknowledgement

```
<!ELEMENT selective-ak (lower-window-edge,upper-window-edge) >
```

3.8.4.21 Upper Window Edge

```
<!ELEMENT upper-window-edge (#PCDATA) >
```

This element contains the upper-window edge, expressed as a decimal number.

3.8.4.22 Invalid TPDU

```
<!ELEMENT invalid-tpdu (#PCDATA) >
```

When present, provides the invalid TPDU as a hexadecimal character string.

4 Example Document

The following is an example ATC Messages XML document compliant with the DTD in section 3. Comments are embedded in the document in order to aid interpretation.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE l2kx-document PUBLIC -//MWA//DTD LISAT Document 1.0//EN
                      "http://www.l2kbaseline.org/l2kx.dtd">
<l2kx-document>
  <header>
    <title>Example UACC Maastricht Log</title>
    <date>2007/10/04</date
  </header>
  <atc-messages reporting-ATSU=”EDYY”>

       <!-- The recording starts with a CM Logon for AAL123. This is
        recorded as taking place on March 29, 2002 at 10:01:30 GMT. The CM
        Logon records the ATN Address of the aircraft and that it supports
        ground initiated CPDLC. The relative ATN Address of the application
        is also provided. The Flight is further identified by the Departure
        Airport (Heathrow) and the destination (Berlin Templehof) and an
        estimated departure time of 09:30. -->

     <cm-logon-req flight-id=”AAL123” icao-24-bit=”A4A08E”
                    logged-at=”2002/03/29 10:01:30”>
       <flight-id>AAL123</flight-id>
       <tsap>
         <rdp>0011223344</rdp>
          <ars>556677</ars>
          <locSysNselTsel>88990011223377889900</locSysNselTsel>
       </tsap>
       <AEAddress>
         <AEQualifier>2</AEQualifier>
         <AEVersion>1</AEVersion>
         <tsap>
            <locSysNselTsel>8899AABBCCDDEEFF0011</locSysNselTsel>
         </tsap>
       </AEAddress>
       <facilityDesignation>EDYY</facilityDesignation>
       <airport role=”departure”>EGLL</airport>
       <airport role=”destination”>EDII</airport>
       <eotd>
         <date>2002/03/29</date>
         <time>09:30:00</time>
       </eotd>
     </cm-logon-req>

       <!-- The CM Logon Response indicates that CPDLC has been accepted by
        Maastricht -->

     <cm-logon-resp flight-id=”AAL123” icao-24-bit=”A4A08E”
                    logged-at=”2002/03/29 10:01:31”>
       <AESupport>
          <AEQualifier>2</AEQualifier>
         <AEVersion>1</AEVersion>
       </AESupport>
     </cm-logon-resp>


<!-- The next record in the log is an uplink CM Start Request for a CPDLC
        connection with AAL123.. This is accepted, and the response
        received eight seconds later. -->
```

```
 <cpdlc-start-req flight-id="AAL123" icao-24-bit="A4A08E"
                  logged-at="2002/03/29 10:40:22"
                  direction="up" mode="cpdlc" />
 <cpdlc-start-resp flight-id="AAL123" icao-24-bit="A4A08E"
                  logged-at="2002/03/29 10:40:30"
                  direction="down" result="accepted"/>

    <!--A CPDLC message is now recorded. This is um19 MAINTAIN LEVEL
     300. A LACK is requested. -->

<cpdlc-message flight-id="AAL123" direction="up"
                  icao-24-bit="A4A08E"
                  logged-at="2002/03/29 10:59:34">
  <header>
    <msg-id>1</msg-id>
    <date>2002/03/29</date>
    <time>10:59:33</time>
    <logical-ack-req />
  </header>
  <msg-element id="19">
    <level>
       <leveldata units="flightLevel">300</leveldata>
    </level>
 </msg-element>
</cpdlc-message>

    <!-- The LACK is received at 10:59:48 and correlates with the
     original instruction through the msg-ref element -->

<cpdlc-message flight-id="AAL123" direction="down"
                  icao-24-bit="A4A08E"
                  logged-at="2002/03/29 10:59:48">
  <header>
    <msg-id>1</msg-id>
    <msg-ref>1</msg-ref>
    <date>2002/03/29</date>
    <time>10:59:42</time>
  </header>
  <msg-element id="100" />
</cpdlc-message>

    <!-- Finally the WILCO is received -->

<cpdlc-message flight-id="AAL123" direction="down"
                  icao-24-bit="A4A08E"
                  logged-at="2002/03/29 10:59:49">
  <header>
    <msg-id>2</msg-id>
    <msg-ref>1</msg-ref>
    <date>2002/03/29</date>
    <time>10:59:45</time>
  </header>
  <msg-element id="0" />
</cpdlc-message>

    <!-- The next uplink instruction is a CONTACT um117 message that
     instructs the pilot to change voice frequency. A LACK is again
     returned followed by the WILCO -->

<cpdlc-message flight-id="AAL123" direction="up"
                  icao-24-bit="A4A08E"
                  logged-at="2002/03/29 11:12:00">
  <header>
    <msg-id>2</msg-id>
```

```
      <date>2002/03/29</date>
      <time>11:11:59</time>
      <logical-ack-req />
    </header>
    <msg-element id=”117”>
      <unitName function=”control”>
        <facilityDesignation>EDYY</facilityDesignation>
      </unitName>
      <frequency band=”vhf”>120.000</frequency >
  </msg-element>
</cpdlc-message>

<cpdlc-message flight-id=”AAL123” direction=”down”
               icao-24-bit=”A4A08E”
               logged-at=”2002/03/29 11:12:22”>
  <header>
    <msg-id>3</msg-id>
    <msg-ref>2</msg-ref>
    <date>2002/03/29</date>
    <time>11:12:16</time>
  </header>
  <msg-element id=”100” />
</cpdlc-message>
<cpdlc-message flight-id=”AAL123” direction=”down”
               icao-24-bit=”A4A08E”
               logged-at=”2002/03/29 11:12:30”>
  <header>
    <msg-id>4</msg-id>
    <msg-ref>2</msg-ref>
    <date>2002/03/29</date>
    <time>11:12:28</time>
  </header>
  <msg-element id=”0” />
</cpdlc-message>

   <!--In preparation for transfer of communications the NDA message is
    uplinked, and a LACK received. -->

<cpdlc-message flight-id=”AAL123” direction=”up”
               icao-24-bit=”A4A08E”
               logged-at=”2002/03/29 11:14:00”>
  <header>
    <msg-id>3</msg-id>
    <date>2002/03/29</date>
    <time>11:14:00</time>
    <logical-ack-req />
  </header>
  <msg-element id=”160”>
    <facilityDesignation>EDGG</facilityDesignation>
  </msg-element>
</cpdlc-message>

<cpdlc-message flight-id=”AAL123” direction=”down”
               icao-24-bit=”A4A08E”
               logged-at=”2002/03/29 11:14:18”>
  <header>
    <msg-id>5</msg-id>
    <msg-ref>3</msg-ref>
    <date>2002/03/29</date>
    <time>11:14:12</time>
  </header>
  <msg-element id=”100” />
</cpdlc-message>
```

```
      <!-- Transfer of communications is now effected by closing the CPDLC
        connection with a CONTACT message -->

   <cpdlc-end-req flight-id="AAL123" direction="up"
                  icao-24-bit="A4A08E"
                  logged-at="2002/03/29 11:17:30">
     <header>
       <msg-id>4</msg-id>
       <date>2002/03/29</date>
       <time>11:17:29</time>
     </header>
     <msg-element id="117">
       <unitName function="control">
         <facilityDesignation>EDGG</facilityDesignation>
       </unitName>
       <frequency band="vhf">117.000</frequency >
    </msg-element>
   </cpdlc-end-req>

   <!-- The end of the CPDLC dialog is confirmed with a WILCO -->

   <cpdlc-end-resp flight-id="AAL123" direction="down"
                  icao-24-bit="A4A08E"
                  logged-at="2002/03/29 11:18:22"
                  result="accepted">
     <header>
       <msg-id>6</msg-id>
       <msg-ref>4</msg-ref>
       <date>2002/03/29</date>
       <time>11:18:18</time>
     </header>
     <msg-element id="0" />
   </cpdlc-end-resp>

</atc-messages>

 <sector-transitions reporting-ATSU="EDYY" >

 <!-- This element contains the section transitions reported by EDYY (UAC
       Maastricht). A single transition is reported for the sector RUHR-ID
       and this is for Flight ID "LH401". The transition occurred on Jan
       6, 2003 and the aircraft type is reported as a Boeing 747-400 with
       a tail number of VH-OJD. The sector was entered as 11:12:14 and
       FL300, the aircraft exited the sector at 11:18:14 at FL320. -->

   <transition sector="RUHR_ID">
     <date>2003/01/06</date>
     <flight-id>DLH401</flight-id>
     <airframe>B744</airframe>
     <tail-number>VH-OJD</tail-number>
     <airport role="departure">KIAD</airport>
     <airport role="destination">EDDF</airport>
     <entry>
       <time>11:12:14</time>
       <level>
         <leveldata units="flightLevel">300</leveldata>
       </level>
     </entry>
     <exit>
       <time>11:18:14</time>
       <level>
         <leveldata units="flightLevel">320</leveldata>
       </level>
```

```
     </exit>
    </transition>
  </sector-transitions>
</l2kx-document>
```

Annex A Message Element Parameters

This annex is an integral part of this specification and identifies the message element number supported and the parameter elements required for each message element.

CPDLC Message Element Number and Name	Link2000+ Status	Parameter Elements and required attributes
UM0 UNABLE	M	
UM1 STANDBY	M	
UM3 ROGER	O	
UM4 AFFIRM	O	
UM5 NEGATIVE	O	
UM19 MAINTAIN [level]	M	level
UM20 CLIMB TO [level]	M	level
UM23 DESCEND TO [level]	M	level
UM26 CLIMB TO REACH [level] BY [time]	O	level,time
UM27 CLIMB TO REACH [level] BY [position]	O	level,position
UM28 DESCEND TO REACH [level] BY [time]	O	level,time
UM29 DESCEND TO REACH [level] BY [position]	O	level,position
UM46 CROSS [position] AT [level]	O	position,level
UM47 CROSS [position] AT OR ABOVE [level]	O	position,level
UM48 CROSS [position] AT OR BELOW [level]	O	position,level
UM51 CROSS [position] AT [time]	O	position,time
UM52 CROSS [position] AT OR BEFORE [time]	O	position,time
UM53 CROSS [position] AT OR AFTER [time]	O	position,time
UM54 CROSS [position] BETWEEN [time] AND [time]	O	position,time,time Note the order of the time elements is significant.
UM55 CROSS [position] AT [speed]	O	position,speed
UM61 CROSS [position] AT AND MAINTAIN [level] AT [speed]	O	position,level,speed
UM64 OFFSET [specifiedDistance] [direction] OF ROUTE	O	distance,direction Note that distance must have the specified="yes" attribute
UM72 RESUME OWN NAVIGATION	O	
UM74 PROCEED DIRECT TO [position]	M	position
UM79 CLEARED TO [position] VIA [routeClearance]	O	position,routeClearanceIndex

CPDLC Message Element Number and Name	Link2000+ Status	Parameter Elements and required attributes
UM80 CLEARED [route clearance]	O	routeClearanceIndex
UM82 CLEARED TO DEVIATE UP TO [specifiedDistance] [direction] OF ROUTE	O	distance,direction Note that distance must have the specified=”yes” attribute
UM92 HOLD AT [position] AS PUBLISHED MAINTAIN [level]	O	position,level
UM94 TURN [direction] HEADING [degrees]	O	direction,degrees Note: degress must have a role=”heading” attribute
UM96 CONTINUE PRESENT HEADING	O	
UM106 MAINTAIN [speed]	O	speed
UM107 MAINTAIN PRESENT SPEED	O	
UM108 MAINTAIN [speed] OR GREATER	O	speed
UM109 MAINTAIN [speed] OR LESS	O	speed
UM116 RESUME NORMAL SPEED	O	
UM117 CONTACT [unitname] [frequency]	M	unitName,frequency
UM120 MONITOR [unitname] [frequency]	O	unitName,frequency
UM123 SQUAWK [code]	O	code
UM133 REPORT PRESENT LEVEL	O	
UM148 WHEN CAN YOU ACCEPT [level]	O	level
UM157 CHECK STUCK MICROPHONE [frequency]	M	frequency
UM159 ERROR [errorInformation]	M	errorInformation
UM160 NEXT DATA AUTHORITY [facility]	M	facilityDesignation Note: CPDLC allows this parameter to be omitted.
UM162 SERVICE UNAVAILABLE	M	
UM165 THEN	O	
UM171 CLIMB AT [verticalRate] MINIMUM	O	verticalRate
UM172 CLIMB AT [verticalRate] MAXIMUM	O	verticalRate
UM173 DESCEND AT [verticalRate] MINIMUM	O	verticalRate
UM174 DESCEND AT [verticalRate] MAXIMUM	O	verticalRate
UM179 SQUAWK IDENT	O	
UM183 [freetext]	M	freetext
UM190 FLY HEADING [degrees]	M	degrees Note: degress must have a role=”heading” attribute

CPDLC Message Element Number and Name	Link2000+ Status	Parameter Elements and required attributes
UM196 [freetext]	O	freetext
UM203 [freetext]	O	freetext
UM205 [freetext]	O	freetext
UM211 REQUEST FORWARDED	O	
UM213 [facilitydesignation] ALTIMETER [altimeter]	O	facilityDesignation,altimeter Note: CPDLC allows the facilityDesignation to be omitted
UM215 TURN [direction] [degrees]	O	direction,degrees Note: degress must have a role="heading" attribute
UM222 NO SPEED RESTRICTION	O	
UM227 LOGICAL ACKNOWLEDGMENT	M	
UM228 REPORT ETA [position]	O	position
UM231 STATE PREFERRED LEVEL	O	
UM232 STATE-TOP-OF-DESCENT	O	

CPDLC Message Element Number and Name	Link2000+ Status	Parameter Elements and required attributes
DM0 WILCO	M	
DM1 UNABLE	M	
DM2 STANDBY	M	
DM3 ROGER	M	
DM4 AFFIRM	M	
DM5 NEGATIVE	M	
DM6 REQUEST [level]	M	level
DM9 REQUEST CLIMB TO [level]	O	level
DM10 REQUEST DECENT TO [level]	O	level
DM18 REQUEST [speed]	M	speed
DM22 REQUEST DIRECT TO [position]	M	position
DM27 REQUEST WEATHER DEVIATION UP TO [specifiedDistance] [direction] OF ROUTE	O	distance,direction Note that distance must have the specified=”yes” attribute
DM32 PRESENT LEVEL [level]	M	level
DM62 ERROR [errorInformation]	M	errorInformation
DM63 NOT CURRENT DATA AUTHORITY	M	
DM65 DUE TO WEATHER	M	
DM66 DUE TO AIRCRAFT PERFORMANCE	M	
DM81 WE CAN ACCEPT [level] AT [time]	M	level, time
DM82 WE CANNOT ACCEPT [level]	M	level
DM89 MONITORING [unitname] [frequency]	M	unitName, frequency
DM98 [freetext]	M	freetext
DM99 CURRENT DATA AUTHORITY	M	
DM100 LOGICAL ACKNOWLEDGMENT	M	
DM106 PREFERRED LEVEL [level]	M	level
DM107 NOT AUTHORIZED NEXT DATA AUTHORITY	M	
DM109 TOP OF DESCENT [time]	M	time

Annex B Formal DTD Definition

The formal DTD follows. This is the contents of the DTD file and is laid out in a syntactically correct order:

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- This is the Document Type Definition for the LISAT Interchange format
        version 1.8 dated 4/7/13. This DTD should be referenced from a
compliant
          XML document using:

         <!DOCTYPE l2kx-document PUBLIC "-//MWA//DTD CROT Document 1.8//EN"
         "http://lisat.mwasoftware.co.uk/l2kx1-8.dtd" >
 -->

<!ENTITY  % params
                              "(level|time|position|speed|distance|

direction|degrees|departureClearance|routeClearanceIndex|
                              procedure|holdClearance|
unitName|code|speedType|frequency|
                              altimeter|atisCode|positionReport|
facilityDesignation|traffic|freetext|
                              verticalRate|runwayRVR|
clearance|remainingFuelPersonsOnBoard |
                              versionNumber|airport)*">

<!ENTITY          % stdatts
                  "flight-id                CDATA        #REQUIRED
                   icao-24-bit  CDATA        #IMPLIED
                   tail-number  CDATA        #IMPLIED
                   logged-at                 CDATA        #REQUIRED">

<!ENTITY % direction "direction (up|down) #IMPLIED">

<!ENTITY % cm-abort-reasons
                  "timer-expired|undefined-error|invalid-PDU|
                  protocol-error|
                  dialogue-acceptance-not-permitted|
                  dialogue-end-not-accepted|
                  communication-service-error|
                  communication-service-failure|
                  invalid-QOS-parameter|
                  expected-PDU-missing" >

<!ENTITY % cpdlc-provider-abort-reasons
                  "timer-expired|undefined-error|invalid-PDU|
                  protocol-error|
                  communication-service-error|
                  communication-service-failure|
                  invalid-QOS-parameter|
                  expected-PDU-missing" >

<!ENTITY % cpdlc-user-abort-reasons
                  "undefined|
                  no-message-identification-numbers-available|
                  duplicate-message-identification-numbers|
                  no-longer-next-data-authority|
                  current-data-authority-abort|
                  commanded-termination|
                  invalid-response|
                  time-out-of-synchronisation|
                  unknown-integrity-check|
                  validation-failure|
                  unable-to-decode-message|
                  invalid-pdu|
                  invalid-CPDLC-message" >
```

```
<!ENTITY % message "header,msg-element+,route-clearance?,
                    route-clearance?" >

<!ENTITY % publishedIdentifier "(fix|navaid)" >



<!ELEMENT l2kx-document (docheader?,(atc-messages|sector-
transitions|position-reports|network-trace)*) >

<!ELEMENT docheader (title?,date?,comment?) >
<!ELEMENT title (#PCDATA)>
<!ELEMENT comment (#PCDATA)>

<!ELEMENT airport (#PCDATA) >

<!ATTLIST airport
                  role         (departure|destination)    #IMPLIED >

<!ELEMENT date (#PCDATA) >

<!ELEMENT facilityDesignation (#PCDATA)>

<!ELEMENT flight-id (#PCDATA)>

<!ELEMENT icao-24-bit (#PCDATA) >

<!ELEMENT level (leveldata|( leveldata, leveldata))>

<!ATTLIST level
                  atwTolerance (at|atOrAbove|atOrBelow) #IMPLIED
                               type         (current|waypoint) #IMPLIED>

<!ELEMENT leveldata (#PCDATA)>

<!ATTLIST leveldata
                  units        (feet|metres|flightLevel|
                                metricFlightLevel)       #REQUIRED
                  role         (current|waypoint)        #IMPLIED>

<!ELEMENT time (#PCDATA) >

<!ATTLIST time
                  role         (departure|arrival|expiry|current|
                                next|nextPlusOne|destination|
                                waypoint|remainingFuel|

clearanceExpected|allocated|controlled|expected) #IMPLIED
                  tolerance    (at|atOrBefore|atOrAfter) #IMPLIED>

<!ELEMENT sector-transitions (transition+) >

<!ATTLIST sector-transitions
                  reporting-ATSU            CDATA        #REQUIRED >

<!ELEMENT transition (date,flight-id,airframe,
                       (tail-number|icao-24-bit)?,
                       airport,airport,entry,exit) >

<!ATTLIST transition
                  sector                    CDATA        #REQUIRED >

<!ELEMENT position-reports (position-report+) >
<!ATTLIST position-reports
```

```
                  reporting-ATSU            CDATA        #REQUIRED >

<!ELEMENT position-report (level,latLong,x-velocity?,y-velocity?) >
<!ATTLIST position-report %stdatts; >


<!ELEMENT airframe (#PCDATA) >

<!ELEMENT tail-number (#PCDATA) >

<!ELEMENT entry (time,level) >

<!ELEMENT exit (time,level) >

<!ELEMENT atc-messages ((cm-logon-req|cm-logon-resp|cm-update|
                         cm-contact|cm-contact-resp|cm-abort|
                         cm-forward-req|cm-forward-resp|cm-end-req|
                         cpdlc-start-req|cpdlc-start-resp|
                         cpdlc-message|cpdlc-abort|
                         cpdlc-end-req|cpdlc-end-resp)+)>

<!ATTLIST atc-messages
                  reporting-ATSU            CDATA        #REQUIRED
                  protocol-stack            (atn|fans)   "atn" >

<!ELEMENT cm-logon-req (flight-id,icao-24-bit?,tail-number?,tsap,
                         AEAddress*,AESupport*,
                         facilityDesignation?,airport?,
                         airport?,eotd?)>

<!ATTLIST cm-logon-req %stdatts; >

<!ELEMENT cm-logon-resp (AEAddress*,AESupport*)>

<!ATTLIST cm-logon-resp %stdatts;
                                           maintain-dialog       (yes|no)
    "no">

<!ELEMENT cm-update (tsap,AEAddress*,AESupport*)>

<!ATTLIST cm-update %stdatts; >

<!ELEMENT cm-contact (facilityDesignation,tsap)>

<!ATTLIST cm-contact %stdatts; >

<!ELEMENT cm-contact-resp EMPTY>

<!ATTLIST cm-contact-resp
                  %stdatts;
                  result       (success|notSuccessful)   #REQUIRED >

<!ELEMENT cm-abort EMPTY>

<!ATTLIST cm-abort
                  %stdatts;
                  %direction;
                  type         (provider|user)           #REQUIRED
                  reason       (%cm-abort-reasons;) #IMPLIED >

<!ELEMENT cm-end-req EMPTY>

<!ATTLIST cm-end-req           %stdatts;
                               %direction;>
```

```
<!ELEMENT cm-end-resp EMPTY>

<!ATTLIST cm-end-resp          %stdatts;
                               %direction;>

<!ELEMENT cm-forward-req (flight-id,icao-24-bit?,tail-number?,tsap,
                         AEAddress*,AESupport*,
                         facilityDesignation?,airport?,
                         airport?,eotd?)>

<!ATTLIST cm-forward-req %stdatts;
                  source-ATSU   CDATA        #REQUIRED >

<!ELEMENT cm-forward-resp EMPTY>

<!ATTLIST cm-forward-resp
                  %stdatts;
                  response     (success|incompatible-version|
                                 service-not-supported)   #REQUIRED >


<!ELEMENT AEAddress (AEQualifier,AEVersion,tsap)>

<!ELEMENT AEQualifier (#PCDATA)>

<!ELEMENT AEVersion (#PCDATA)>

<!ELEMENT AESupport (AEQualifier,AEVersion)>

<!ELEMENT tsap (rdp?,ars?,locSysNselTsel)>

<!ELEMENT rdp (#PCDATA)>

<!ELEMENT ars (#PCDATA)>

<!ELEMENT locSysNselTsel (#PCDATA)>

<!ELEMENT eotd (date,time)>

<!ELEMENT cpdlc-start-req ((%message;)?)>

<!ATTLIST cpdlc-start-req
                  %stdatts;
                  %direction;
                  mode         (cpdlc|dsc)               #REQUIRED>

<!ELEMENT cpdlc-start-resp ((%message;)?)>

<!ATTLIST cpdlc-start-resp
                  %stdatts;
                  %direction;
                  result       (accepted|rejected)       #REQUIRED>


<!ELEMENT cpdlc-message (%message;) >

<!ATTLIST cpdlc-message
                  %stdatts;
                  %direction;>

<!ELEMENT cpdlc-abort EMPTY >

<!ATTLIST cpdlc-abort
```

```
                  %stdatts;
                  %direction;
                  type         (provider|user)           #REQUIRED
                  reason       (%cpdlc-user-abort-reasons;|
                                %cpdlc-provider-abort-reasons;) #REQUIRED>



<!ELEMENT cpdlc-end-req ((%message;)?)>

<!ATTLIST cpdlc-end-req
                  %stdatts;
                  %direction;>

<!ELEMENT cpdlc-end-resp ((%message;)?)>

<!ATTLIST cpdlc-end-resp
                  %stdatts;
                  %direction;
                  result       (accepted|rejected)       #REQUIRED>

<!ELEMENT header (msg-id,msg-ref?,date,time,logical-ack-req?) >

<!ELEMENT msg-id (#PCDATA)>

<!ELEMENT msg-ref (#PCDATA)>

<!ELEMENT logical-ack-req EMPTY>

<!ELEMENT msg-element (%params;) >

<!ATTLIST msg-element
                  id           CDATA        #REQUIRED
                  params       (yes|no)     "yes"
                  error        (unrecognizedMsgReferenceNumber|
                                logicalAcknowledgmentNotAccepted|
                                insufficientResources|
                                invalidMessageElementCombination|
                                invalidMessageElement) #IMPLIED >



<!ELEMENT altimeter (#PCDATA)>

<!ATTLIST altimeter
                  units        (inches|hectopascals)     #REQUIRED >

<!ELEMENT atisCode (#PCDATA)>

<!ELEMENT clearance EMPTY>

<!ATTLIST clearance
                  type         (noneSpecified|approach|departure|
                                further|start-up|pushback|taxi|
                                take-off|landing|oceanic|
                                en-route|downstream)     #REQUIRED>

<!ELEMENT code (#PCDATA)>

<!ELEMENT degrees (#PCDATA)>

<!ATTLIST degrees
                  units        (magnetic|true|latitude|
                                longitude)               #REQUIRED
```

```
                  role         (bearing|heading|trackAngle)
                                                        #IMPLIED>

<!ELEMENT direction EMPTY >

<!ATTLIST direction
                  value        (left|right|eitherSide|north|south|
                                east|west|northEast|northWest|
                                southEast|southWest)     #REQUIRED >

<!ELEMENT distance (#PCDATA) >

<!ATTLIST distance
                  units        (nautical-miles|kilometres)
                                                        #REQUIRED
                  atwTolerance (plus|minus)  #IMPLIED
                  specified    (yes|no)     "no" >

<!ELEMENT facilityName (#PCDATA)>

<!ELEMENT fix (latLong?)>

<!ATTLIST fix
                  name         CDATA                    #REQUIRED>

<!ELEMENT frequency (#PCDATA)>

<!ATTLIST frequency
                  band         (hf|vhf|uhf|sat)         #REQUIRED>

<!ELEMENT freetext (#PCDATA)>

<!ELEMENT humidity (#PCDATA)>

<!ELEMENT icing EMPTY>

<!ATTLIST icing
                  value        (trace|light|moderate|severe)    #REQUIRED>

<!ELEMENT latLong (latitude?,longitude?)>

<!ELEMENT latitude (degrees,minutes?,seconds?)>

<!ATTLIST latitude
                  direction    (north|south)"north" >

<!ELEMENT longitude (degrees,minutes?,seconds?)>

<!ATTLIST longitude
                  direction    (east|west)              "east">

<!ELEMENT minutes (#PCDATA)>

<!ELEMENT navaid (latLong?)>
<!ATTLIST navaid
                  name         CDATA                    #REQUIRED>

<!ELEMENT position (((fix|navaid),(degrees,distance)?)|
                              airport|latLong) >

<!ATTLIST position
                  direction    (to|from)                #IMPLIED
                  role         (current|next|nextPlusOne|waypoint)
                                                        #IMPLIED>
```

```
<!ELEMENT positionReport (position+,time+,level+,
                          temperature?,winds?,turbulence?,
                          icing?,speed*,verticalRate?,degrees*,
                          distance?,humidity?) >

<!ELEMENT procedure             (procedure-transition?) >

<!ATTLIST procedure
                  name          CDATA                           #REQUIRED
                  type          (arrival|approach|departure) #REQUIRED>

<!ELEMENT procedure-transition (#PCDATA)>

<!ELEMENT remainingFuelPersonsOnBoard (time,persons) >

<!ELEMENT persons (#PCDATA)>

<!ELEMENT revisionNumber (#PCDATA)>

<!ELEMENT routeClearanceIndex EMPTY>

<!ATTLIST routeClearanceIndex
                  index         (1|2)                       #REQUIRED>

<!ELEMENT runwayRVR (runway,rvr) >

<!ELEMENT runway (#PCDATA) >

<!ATTLIST runway
                  config        (left|right|center|none) #REQUIRED
                  role          (departure|arrival)         #IMPLIED>

<!ELEMENT rvr (#PCDATA)>

<!ATTLIST rvr
                  units         (feet|metres)#REQUIRED>

<!ELEMENT seconds (#PCDATA)>

<!ELEMENT speed (#PCDATA)>

<!ATTLIST speed
                  units         (indicated|indicatedMetric|true|
                                 trueMetric|ground|groundMetric|
                                 mach|windSpeed|windSpeedMetric)
                                                              #REQUIRED
                  use           (low|high)                  #IMPLIED>

<!ELEMENT speedType EMPTY>

<!ATTLIST speedType
                  type          (noneSpecified|indicated|true|ground|
                                 mach|approach|cruise|minimum|maximum)
                                                              #REQUIRED>

<!ELEMENT temperature (#PCDATA)>

<!ELEMENT traffic EMPTY>

<!ATTLIST traffic
                  type          (noneSpecified|oppositeDirection|
                                 sameDirection|converging|
                                 crossing|diverging)       #REQUIRED>
```

```
<!ELEMENT turbulence EMPTY>

<!ATTLIST turbulence
                  value        (light|moderate|severe)    #REQUIRED>

<!ELEMENT unitName (facilityDesignation,facilityName?)>

<!ATTLIST unitName
                  function     (center|approach|tower|final|
                                groundControl|clearanceDelivery|
                                departure|control|radio) #REQUIRED>

<!ELEMENT versionNumber (#PCDATA)>

<!ELEMENT verticalRate (#PCDATA)>

<!ATTLIST verticalRate
                  units        (feetPerMinute|metresPerMinute)
                                                          #REQUIRED
                  direction    (up|down)                 #IMPLIED>

<!ELEMENT winds (WindDirection,speed)>

<!ELEMENT WindDirection (#PCDATA)>

<!ELEMENT atsRouteDesignator (#PCDATA)>

<!ELEMENT aTWAlongTrackWaypoint (position,distance,speed?,
                                     level?,level?) >

<!ELEMENT degreeIncrement (#PCDATA)>

<!ELEMENT departureClearance (flight-id,position,flightInformation?,
                                code?,(unitName,frequency)?,time?,
                                airport?,airport?,departureTime?,
                                runway?,
                                revisionNumber?,atisCode?) >

<!ELEMENT departureTime (time?,time?,time?,departureMinInterval?)>

<!ELEMENT departureMinInterval (#PCDATA)>

<!ELEMENT flightInformation (routeInformation|levelsOfFlight|
                               (routeInformation,levelsOfFlight)) >

<!ELEMENT holdAtWaypoint (position,speed?,level?,speed?,direction?,
                            degrees?,time?,leg?)>

<!ELEMENT holdClearance (position,level,degrees,direction,leg?)>

<!ELEMENT interceptCourseFrom ((%publishedIdentifier;|latLong|
                                   (placeBearing,(placeBearing|
                                    distance))),degrees)>

<!ELEMENT leg (#PCDATA)>

<!ATTLIST leg
                  units        (distance-nm|distance-km|time)
                                                          #REQUIRED>

<!ELEMENT levelsOfFlight (level|procedure|(level,procedure))>

<!ELEMENT placeBearing (%publishedIdentifier;,degrees)>
```

```
<!ELEMENT route-clearance       ((airport|runway|procedure)*,
                                 routeInformation*,
                                 aTWAlongTrackWaypoint*,
                                 reportingPoints?,
                                 interceptCourseFrom*,
                                 holdAtWaypoint*,
                                 waypointSpeedLevel*,
                                 rTARequiredTimeArrival*) >

<!ATTLIST route-clearance
                  index                     (1|2)        "1">

<!ELEMENT reportingPoints ((latitude|longitude),
                           degreeIncrement)>

<!ELEMENT routeInformation      (%publishedIdentifier;|latLong|
                                 (placeBearing,
                                  (placeBearing|distance))|
                                 atsRouteDesignator)>

<!ELEMENT rTARequiredTimeArrival (position,time,rtaTolerance?)>

<!ELEMENT rtaTolerance (#PCDATA)>

<!ELEMENT waypointSpeedLevel (position,speed?,level?,level?)>

<!ELEMENT x-velocity (#PCDATA) >
<!ATTLIST x-velocity
                  units        (indicated|indicatedMetric|true|
                                trueMetric|ground|groundMetric|
                                mach|windSpeed|windSpeedMetric)
                                                        #IMPLIED >

<!ELEMENT y-velocity (#PCDATA) >
<!ATTLIST y-velocity
                  units        (indicated|indicatedMetric|true|
                                trueMetric|ground|groundMetric|
                                mach|windSpeed|windSpeedMetric)
                                                        #IMPLIED >

<!ELEMENT network-trace (clnp-packet*)>

<!ATTLIST network-trace
                  reporting-ATSU            CDATA        #REQUIRED>

<!ENTITY  %       priority
                  "(0|1|2|3|4|5|6|7|8|9|10|11|12|13|14|15)">

<!ELEMENT clnp-packet (destination-nsap,source-nsap,data-unit-
identifier?,lifetime,
    checksum,globally-unique-qos?,atn-security?,npdu-data?,
    (cr-tpdu|cc-tpdu|dr-tpdu|dc-tpdu|dt-tpdu|ed-tpdu|ak-tpdu|ea-tpdu|er-
tpdu)*) >
<!ATTLIST clnp-packet
                  type
          (data|error-report|echo-request|echo-response)   #REQUIRED
                  logged-at                 CDATA        #REQUIRED
                  bad-checksum              (yes|no)     "no"
                  congestion-experienced    (yes|no)     "no"
                  discard-reason                         CDATA
    #IMPLIED
                  priority                  %priority;   "0"
                  direction                 (uplink|downlink) #REQUIRED>
```

```
<!ELEMENT destination-nsap (#PCDATA) >
<!ELEMENT source-nsap (#PCDATA) >

<!ELEMENT data-unit-identifier (#PCDATA) >

<!ELEMENT lifetime (#PCDATA) >

<!ELEMENT checksum (#PCDATA) >

<!ELEMENT globally-unique-qos (sequencing-vs-transit-delay?,
    transit-delay-vs-cost?, residual-error-prob-vs-transit-delay?,
    residual-error-prob-vs_cost?) >

<!ELEMENT sequencing-vs-transit-delay EMPTY>
<!ELEMENT transit-delay-vs-cost EMPTY>
<!ELEMENT residual-error-prob-vs-transit-delay EMPTY>
<!ELEMENT residual-error-prob-vs_cost EMPTY>

<!ELEMENT atn-security (atsc-tag|aoc-tag) >

<!ELEMENT atsc-tag EMPTY >
<!ATTLIST atsc-tag
          class    (U|A|B|C|D|E|F|G|H)       "U" >

<!ELEMENT aoc-tag EMPTY>
<!ATTLIST aoc-tag
          via      (gatelink|vdl|satcom|hf|mode_s|gatelink-then-vdl|
                   gatelink-then-vdl-then-satcom|
                   gatelink-then-vdl-then-hf-then-satcom)  #IMPLIED>

<!ELEMENT npdu-data (#PCDATA) >

<!ELEMENT cr-tpdu (calling-tsap,called-tsap,tp4-version?, src-ref,
                   tpdu-size?,ack-time,inactivity-time,
                   initial-credit, priority,checksum?, atn-checksum?,
                   user-data?)>

<!ATTLIST cr-tpdu
                   bad-checksum              (yes|no)     "no"
                   allow-expedited           (yes|no)     "no"
                   extended-format           (yes|no)     "no">


<!ELEMENT cc-tpdu (calling-tsap,called-tsap,tp4-version?,dst-ref,src-ref,
                   tpdu-size?, ack-time,inactivity-time,initial-credit,
                   priority?, checksum?, atn-checksum?,user-data?)>

<!ATTLIST cc-tpdu
                   bad-checksum              (yes|no)     "no"
                   allow-expedited           (yes|no)     "no"
                   extended-format           (yes|no)     "no">


<!ELEMENT dr-tpdu (dst-ref,src-ref, checksum?,
                   atn-checksum?,user-data?) >

<!ATTLIST dr-tpdu
                   bad-checksum              (yes|no)     "no"
                   dr-reason (user-disconnect|remote-te-congestion|
          connection-negotiation-failed|duplicate-src-reference|
          Mismatched-references|protocol-error|reference-overflow|
          cr-refused|hdr-or-param-length-invalid|reason-not-specified|
```

```
          congestion-at-tsap|session-entity-not-attached|
          address-unknown) #REQUIRED>

<!ELEMENT dc-tpdu (dst-ref, src-ref, checksum?,
                   atn-checksum?) >

<!ATTLIST dc-tpdu
                   bad-checksum              (yes|no)     "no">


<!ELEMENT dt-tpdu (dst-ref,tpdu-nr, checksum?,
                   atn-checksum?, user-data) >

<!ATTLIST dt-tpdu
                   bad-checksum              (yes|no)     "no"
          eot      (yes|no)                  "yes" >



<!ELEMENT ed-tpdu (dst-ref,tpdu-nr, checksum?,
                   atn-checksum?, user-data) >

<!ATTLIST ed-tpdu
                   bad-checksum              (yes|no)     "no">


<!ELEMENT ak-tpdu (dst-ref,yr-tpdu-nr,credit,ak-subsequence-nr?,
                   flow-control-confirmation?,selective-ak*, checksum?,
                   atn-checksum?) >

<!ATTLIST ak-tpdu
                   bad-checksum              (yes|no)     "no">


<!ELEMENT ea-tpdu (dst-ref,yr-tpdu-nr, checksum?,
                   atn-checksum?) >

<!ATTLIST ea-tpdu
                   bad-checksum              (yes|no)     "no">

<!ELEMENT er-tpdu (dst-ref,invalid-tpdu?, checksum?,
                   atn-checksum?) >

<!ATTLIST er-tpdu
                   bad-checksum              (yes|no)     "no"
                   reject-cause (not-specified| invalid-parameter-code|
       invalid-tpdu-type| invalid-parameter-value) #REQUIRED>


<!ELEMENT atn-checksum (#PCDATA) >

<!ELEMENT calling-tsap (#PCDATA) >
<!ELEMENT called-tsap (#PCDATA) >

<!ELEMENT dst-ref (#PCDATA) >
<!ELEMENT src-ref (#PCDATA) >

<!ELEMENT tpdu-size (#PCDATA) >

<!ELEMENT tp4-version (#PCDATA) >

<!ELEMENT ack-time (#PCDATA) >

<!ELEMENT inactivity-time (#PCDATA) >
```

```
<!ELEMENT initial-credit (#PCDATA) >

<!ELEMENT priority (#PCDATA) >

<!ELEMENT user-data (#PCDATA) >

<!ELEMENT tpdu-nr (#PCDATA) >

<!ELEMENT yr-tpdu-nr (#PCDATA) >

<!ELEMENT credit (#PCDATA) >

<!ELEMENT ak-subsequence-nr (#PCDATA) >

<!ELEMENT flow-control-confirmation (lower-window-edge?,
                  yr-subsequence-nr?,yr-credit) >

<!ELEMENT lower-window-edge (#PCDATA) >

<!ELEMENT yr-subsequence-nr (#PCDATA) >

<!ELEMENT yr-credit (#PCDATA) >

<!ELEMENT selective-ak (lower-window-edge,upper-window-edge) >

<!ELEMENT upper-window-edge (#PCDATA) >

<!ELEMENT invalid-tpdu (#PCDATA) >
```