Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL DEL TFG: Desenvolupament del mòdul ASXDEC del sistema SSDA (Surveillance Systems Data Analysis)

TITULACIÓ: Grau en Enginyeria d'Aeronavegació

AUTOR: Xavier Marin Serra

DIRECTOR: Albert Prades Gimeno

DATA: 22 d'octubre de 2021

Títol: Desenvolupament del mòdul ASXDEC del sistema SSDA (Surveillance Systems Data Analysis)

Autor: Xavier Marin Serra

Director: Albert Prades Gimeno

Data: 22 d'octubre de 2021

Resum

Aquest treball s'emmarca dins del projecte europeu SSDA (*Surveillance Sensor Data Analysis*) que té com a objectiu l'anàlisi de dades provinents de sensors de vigilància. Quan parlem de vigilància en aquest context, ens referim a la detecció d'aeronaus (o blancs) per tal d'obtenir la seva posició, juntament amb altra informació significativa (velocitat, identificació, etc.), que ens permeti posicionar (ubicar) i monitoritzar el trànsit aeri i aeroportuari en temps real. Les dades de vigilància enviades per aquests sensors estan codificades seguint l'estàndard d'intercanvi d'informació ASTERIX, implementat per EUROCONTROL per primer cop l'any 1997.

Com qualsevol altre sistema, un sensor de vigilància requereix de comprovacions periòdiques que ens ajudin a determinar el seu correcte funcionament i prestacions, dins de les especificacions que el fabricant hagi establert i el que exigeixi la normativa vigent. El programa SSDA neix en resposta als requeriments imposats pel reglament EU1207/1, davant la necessitat d'avaluar de forma continuada diferents indicadors de qualitat d'aquests sistemes, així com d'efectuar altres tipus d'anàlisi i/o utilitzar les dades de vigilància per a múltiples aplicacions: anàlisi d'indicadors, anàlisi de format, representació gràfica de trajectòries radar,...

Presentat el marc del projecte SSDA, l'objectiu d'aquest treball és desenvolupar el mòdul ASXDEC que s'encarregarà de la descodificació i posterior emmagatzematge de les dades de vigilància entrants, en un format de sortida seleccionat (taula en base de dades, CSV...). Aquest mòdul constitueix la punta de llança, que permetrà que subsegüents mòduls (altres aplicacions) puguin utilitzar aquest repositori global de dades, per als requeriments esmentats anteriorment.

Tot i que ja existeixen altres programaris de descodificació de dades de vigilància, el valor afegit d'aquesta aplicació és el seu disseny modular, flexible i escalable a futures categories ASTERIX (tipus de sensors) i/o formats de sortida sol·licitats, com a part dels requeriments establerts amb ENAIRE. Això s'ha aconseguit principalment per la introducció innovadora de fitxers XML que poden ser editats (configuració) per l'usuari i redueixen la necessitat de codi informàtic. Així mateix, l'aplicació permet filtrar les dades d'interès en funció de diferents paràmetres i/o establir les diferents sortides, per mitjà d'un fitxer de configuració.

Title: Development of SSDA System (Surveillance Systems Data Analysis) ASXDEC module

Author: Xavier Marin Serra

Director: Albert Prades Gimeno

Date: October 22th, 2021

Overview

This project takes part in a bigger European initiative called SSDA (Surveillance Sensor Data Analysis) the aim of which is the analysis of data from surveillance sensor. When we talk about surveillance in this context, we mean the detection of aircrafts (or targets) in order to obtain their position, as well as other important information (speed, identification…) that will allow us to have a real-time monitoring of the en-route and airports aircraft traffic. The surveillance data from these sensors is encoded following the information-exchange protocol known as ASTERIX, which was first time implemented by EUROCONTROL at 1997.

As any other system, a sensor is required to have periodic inspections to check that they are working properly and their performances are nominal, within the specifications that the manufacturer has given and the current regulation establishes. SSDA program was born as a response to the requirements established by EU1207/1 regulation, due to the burgeoning need of continuously evaluating different quality indicators of these systems, as well as performing any other types of analysis for multiple applications: indicators analysis, format analysis, graphical representation of radar trajectories…

Introduced the framework of SSDA program, the particular aim of this project is to develop the ASXDEC application module that will decode the input surveillance data and then store it in a selected output format (data base table, CSV…). ASXDEC is the first module that will provide the global repository of this data needed by the subsequent modules (other apps) that will fulfil the requirements explained before.

Although, it currently exists other software for decoding surveillance data, the key points of this module are its modular design, flexibility and scalability to future ASTERIX categories (types of sensors) and/or required output format, as part of the requirements established with ENAIRE. This has been mainly achieved by the innovative introduction of XML files that can be edited (configuration) by the user and reduce the need of extra coding. Likewise, the module allows to filter the data of interest depending on different parameters and/or set the different format outputs, by the use of a configuration file.

ÍNDEX

LLISTA DE FIGURES

LLISTA DE TAULES

INTRODUCCIÓ

L'objectiu d'aquest treball és el desenvolupament del mòdul ASXDEC per a la creació d'un repositori global de dades provinents de sistemes de vigilància que serà utilitzat per aplicacions client d'ENAIRE. Donat que la informació transmesa per aquests sistemes està codificada en el format ASTERIX, l'aplicació s'encarregarà de descodificar la informació continguda en les gravacions (fitxers amb extensió .AST) d'aquests sensors i presentar-la en un format de sortida seleccionat (taula en base de dades, CSV...).

Aquest software ha estat desenvolupat en llenguatge C#, i presenta un disseny modular que consta de 3 submòduls principals. Simplificant molt la seva operativa, el primer submòdul s'encarrega de la lectura dels fitxers d'entrada, el segon de la descodificació de la informació del les gravacions ASTERIX, i el tercer de l'escriptura o emmagatzematge de la informació en el format de sortida.

La funcionalitat dels tres submòduls està regida per la parametrització definida en un fitxer XML d'inicialització i de configuració. La utilització d'aquests fitxers dona flexibilitat a l'usuari per: establir les rutes del tots els fitxers que utilitza l'aplicació, configurar la connexió a la base de dades i les taules, filtrar únicament la informació que interessi descodificar i incloure al repositori (en base a la categoria del missatge, *Data Items* d'interès dins de la categoria o el sensor que transmet la informació), descartar la d'informació errònia d'un sistema de vigilància en una categoria determinada, entre d'altres funcionalitats. Tot això, permet adaptar el repositori de dades a l'entorn operatiu o propòsit del projecte, i a les aplicacions client que faran servir les dades.

D'altra banda, donat que el format ASTERIX contempla múltiples aplicacions o categories (que estableixen un tipus d'informació tramesa concreta en cada cas), l'aplicació introdueix una característica innovadora que permet definir la especificació de cada categoria ASTERIX únicament per mitjà de fitxers XML, en comptes de codi. L'aplicació interpreta aquests fitxers, de manera que la addició de noves categories es limita a priori, a la incorporació de nous fitxers XML d'especificació de la categoria sense requerí de la modificació de codi intern de l'aplicació. Tot això, sempre i quan no s'implementin nous formats d'estructura de *Data Items* per part d'EUROCONTROL (llarg termini). La definició d'aquests fitxers la pot portar a terme un usuari amb coneixements bàsics XML i seguint les directrius que s'inclouen en aquest document. En resum, això aporta una gran escalabilitat de cara a la incorporació de noves categories ASTERIX que es vulguin descodificar.

L'aplicació està orientada a la seva execució en línia de comandes (CLI), en cas que vulgui ser utilitzada de forma puntual per l'usuari, o integrada en una sèrie scripts d'execució automatitzada.

El document s'ha dividit en els següents capítols:

1. **Coneixements previs:** S'han tractat els principis del format d'intercanvi d'informació ASTERIX, des de la codificació binària dels elements d'informació més petits, passant per l'estructura general d'un missatge i arribant a les gravacions ASTERIX (fitxers .AST). Aquestes nocions bàsiques ens permeten entendre com porta a terme la descodificació de la informació de vigilància d'una categoria ASTERIX. També s'inclou un apartat en referència als sistemes o sensors de vigilància, que seran els qui utilitzin el format en les seves comunicacions.

2. **El submòdul de descodificació: ASXDEC_CORE:** A partir d'aquest capítol comencem a parlar del desenvolupament pràctic de l'aplicació. S'explicarà com el submòdul de descodificació basa el seu funcionament en la definició de les especificacions de les categories ASTERIX (que utilitza per a al descodificació) per mitjà de fitxers XML externs a l'aplicació, i les implicacions que ha tingut això en el disseny de l'arquitectura interna del submòdul, la definició d'objectes (programació) i la utilització addicional de llibreries, entre d'altres qüestions.

3. **El submòdul d'entrada: IOSS_INPUT:** En aquest capítol es tractarà la definició del fitxer XML d'inicialització, que permetrà a l'usuari donar les rutes (*paths)* de tots els fitxers que l'aplicació rep com a entrada (*input)*. També és comentarà el format de nom (mandatori) que tindran les gravacions ASTERIX entrants.

4. **El submòdul de sortida: LOSS_OUTPUT:** Es parlarà de la implementació en format taula en base de dades, com a format de sortida seleccionat per a emmagatzemar la informació descodificada. En relació a la creació de les taules (en base de dades), es detalla el procés de creació automàtica de les mateixes, limitació d'emmagatzematge,... Addicionalment, també s'inclou informació del funcionament dels missatges de consola i del seu posterior registre en fitxers .LOG.

5. **Configuració i utilització de l'aplicació:** Aquest últim capítol es tracta l'estructura del fitxer XML de configuració, així com les diferents funcionalitats o característiques que permet definir (en detall). També s'inclou un apartat en relació als requeriments addicionals de software i el procés d'instal·lació de l'aplicació. Finalment, es donen unes nocions bàsiques per tal que l'usuari pugui cridar (executar) l'aplicació des de consola (CLI).

A continuació s'inclouen les paraules i abreviacions clau per a la comprensió del document:

ADS-B	Automatic Dependent Surveillance Broadcast
ASTERIX	**A**ll Purpose **St**ructured **E**UROCONTROL Su**r**veillance **I**nformation E**x**change
CAT	ASTERIX Category
CLI	Command Line Interface
CSV	Comma Separated Values
DSI	Data Source Identifier
EUROCONTROL	European Organisation for the Safety of Air Navigation
FRN	Field Reference Number
FSPEC	Field Specification
FX	Field Extension Indicator
KML	Keyhole Markup Language
LEBL	Codi OACI de l'aeroport de Barcelona
LEN	Length Indicator
LSB	Less Significant Bit
MLAT	Multilateration System
MSB	Most Significant Bit
OSI	Open Systems Interconnection
PSR	Primary Surveillance Radar
REP	Field Repetition Indicator
SAC	System Area Code
SACTA	Sistema Automatizado de Control de Tránsito Aéreo
SIC	System Identification Code
SMR	Surface Movement Radar
SSDA	Surveillance Sensor Data Analysis
SSR	Secondary Surveillance Radar
UAP	User Application Profile
UTC	Coordinated Universal Time
XML	E**x**tensible **M**arkup **L**anguage

CAPÍTOL 1. CONEIXEMENTS PREVIS

1.1 El format ASTERIX

ASTERIX és un format d'intercanvi d'informació per a sistemes de vigilància aèria (denominats sensors) i altres aplicacions del món ATM. Ha estat dissenyat per transmetre la informació (i que pugui ser compresa) minimitzant la càrrega de dades que aquesta ocupa, imprescindible per a la seva utilització en sistemes amb un ample de banda limitat. Els protocols/formats d'intercanvi d'informació moderns segueixen el model del protocol de referència d'interconnexió (OSI), on s'estableixen les diferents capes o nivells del protocol (veure **Fig. 1.1**). La capa de nivell més alt (d'aplicació) és amb la que normalment interactua l'usuari, mentre que la capa més inferior, la física, es constitueix primordialment de bits (uns i zeros).

Fig. 1.1 Model de la pila de protocols OSI

Les especificacions ASTERIX es situen a les capes d'aplicació i presentació, les més superiors. Les capes inferiors, queden fora del abast del format. La informació codificada utilitzant ASTERIX pot ser transmesa utilitzant qualsevol mitjà de comunicació que estigui disponible: una xarxa d'àrea local (LAN), una xarxa de llarg abast (WAN),... Els nivells inferiors del protocol de telecomunicacions hauran d'estar acordats entre els socis que intercanvien la informació.

El format ASTERIX, com a protocol a la capa de presentació, defineix l'estructura de la informació que s'intercanviarà a través del mitjà de comunicació, partint de la codificació de la informació bit a bit fins a

l'organització de la informació en elements més complexes que constitueixen un missatge ASTERIX (bloc d'informació).

En cas que hi hagi informació comuna a tots els sistemes que utilitzen ASTERIX, el format estableix uns requeriments mínims a nivell de la capa d'aplicació, que faciliten l'intercanvi de dades entre aplicacions diferents. La comunicació entre dos sistemes diferents és possible establint un marc de referència comú en relació a la informació transmesa, de la mateixa manera que a la capa de presentació.

1.2 Sistemes de vigilància o sensors

Els sensors seran els qui primordialment utilitzin el format ASTERIX per transmetre la informació. Un sensor és un dispositiu que permet la detecció d'aeronaus, la determinació de la seva posició, i la transmissió d'aquesta informació (no només la posició) a un controlador aeri. Aquest sensors s'utilitzen per detectar blancs (aeronaus) que poden estar als voltants de l'aeròdrom, la seva superfície o en ruta. La primera classificació del sensors es fa en relació a si requereixen la cooperació o no cooperació de l'aeronau a la que pretenen monitoritzar per detectar-la. A continuació es presenten els principals tipus de sensors que s'utilitzen a l'actualitat, en base aquesta classificació [1]:

- Detecció no cooperativa:
  - PSR (*Primary Surveillance Radar)*
  - SMR (*Surface Movement Radar*)

- Detecció cooperativa:
  - SSR (*Secondary Surveillance Radar*)
  - MLAT (*Multilateration Systems*)
  - ADS-C (*Automatic Dependant Surveillance Contract*)
  - ADS-B (*Automatic Dependant Surveillance Broadcast*)

1.3 Categories ASTERIX i els *Data Items*

La informació intercanviada utilitzant el format ASTERIX ha estat classificada en primer lloc, en categories. Aquestes categories estandarditzen el tipus d'informació que pot ser transmesa per a les diferents aplicacions en que s'intercanvia informació, que normalment aniran lligades al tipus de sensor que envia la informació. Cadascuna de les categories, inclou un catàleg d'elements d'informació més petits (estandarditzats), que s'anomenen *Data Items* (veure **Fig. 1.2**). La categoria especifica un context en que s'envia la informació (p. e. informació d'un radar de superfície), i els *Data Items* estableixen una primera divisió dels diferents tipus d'informació que podran ser transmesos en aquest context (p. e. velocitat de l'aeronau, identificació de l'aeronau, coordenades radar del blanc...). És per això, que els *Data Items* que es defineixin per a una

categoria (catàleg) no seran necessàriament els mateixos que es defineixin en una altre.

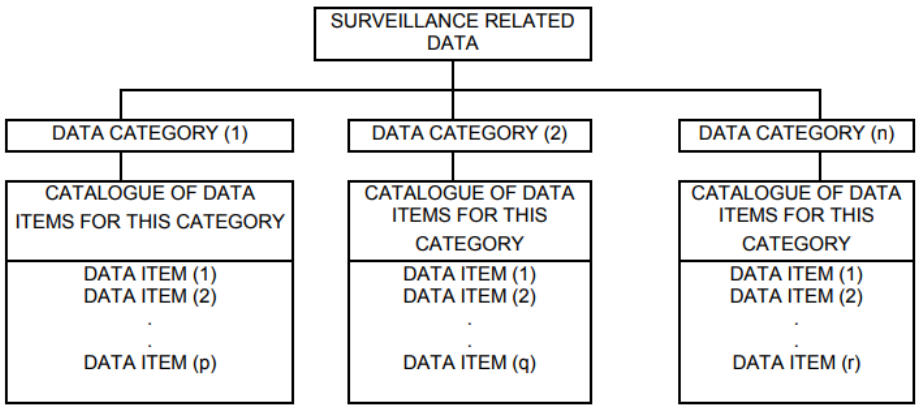

Fig. 1.2 Categories ASTERIX i els seus respectius catàlegs de *Data Items*

Tot i el que s'ha comentat anteriorment, hi hauran sensors que podran transmetre en diferents categories; pot ser que més d'un context s'adeqüi a la informació que transmet un tipus de sensor; sovint perquè hi ha una categoria que agrupa informació de diferents tipus de sensors alhora que s'ha definit una categoria individual per a cadascun d'ells (a mesura que el format ha anat evolucionant). En cas que puguin utilitzar més d'una categoria, els sensors d'ENAIRE seleccionen una o vàries en funció del que esperin les aplicacions de tractament de dades.

A cada categoria se li assigna també una referència de 3 dígits (000-255) que es coneix també com el número de la categoria (CAT010, CAT021...). Així mateix, cada *Data Item* té una referència de 3 dígits (0-255) que l'identifica inequívocament dins d'una categoria.

1.4 Formats estàndard d'estructura dels *Data Items*

Els *Data Items* utilitza els formats d'estructura estandarditzats que afecten a la seva longitud en octets (bytes) que ocupen:

- Longitud fixe o *fixed*: estan formats per un nombre fixe (n) d'octets.
- Explícit o *explicit*: estan formats per un primer octet que indica la longitud total en octets (L) incloent-hi l'octet que ja ocupa l'indicador de longitud .

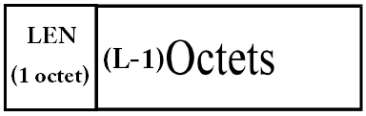

Fig. 1.3 Estructura amb format explícit

- Longitud variable o *variable:* es divideixen en subelements (un subelement primari i a continuació un nombre indefinit d' extensions o subelements secundaris). Cada subelement té una longitud en octets fixe ("k" el primari i "i" els secundaris). Per mitjà de l'últim bit de l'últim octet d'un subelement (FX), s'estableix la continuïtat (1) o no continuïtat (0) al subelement continu.

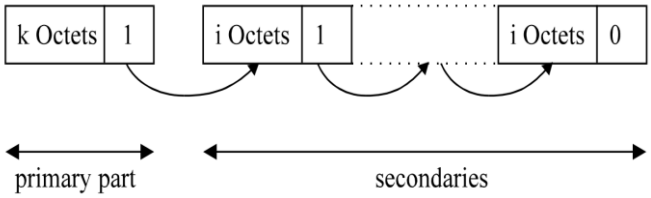

Fig. 1.4 Estructura amb format de longitud variable

- Repetitiu o *repetitive*: consta d'un primer octet que indica el nombre de repeticions (REP) d'un subelement de longitud fixe en octets ("j") que vindran a continuació.

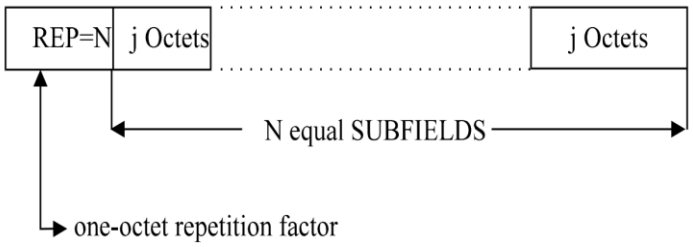

Fig. 1.5 Estructura amb format repetitiu

- Compost *o compound*: una estructura complexa i de longitud variable. (veure pàgina 20 i 22 de la documentació per a més detalls [2]).

1.5 Subelements d'un *Data Item* (*Subfields* i *Fields*)

Dins d'un Data Item trobem la següent divisió en subelements:

- ***Subfields**:* comprèn els subelements que s'utilitzen com a subdivisió addicional dels *Data Items*. Propi dels *Data Items* amb format variable (subelement primari i extensions) o amb format compost. No confondre els *Subfields* amb els que menciona la figura (**Fig. 1.5**).

- ***Fields***: són les unitats d'informació més petites que trobem dins d'un *Data Item*. Cada *Field* conté una porció d'informació molt específica (component X de la velocitat en cartesianes de l'aeronau, "Callsign" de l'aeronau) que ja es materialitza en un valor (-125m/s, VLG325,...). Els *Fields* són els últims elements d'informació; ja no estableixen cap subdivisió de la informació. A la figura (**Fig. 1.6**) es mostra un *Data Item* de longitud fixe que conté dos *Fields* referents a les components cartesianes (radar) X i Y de la posició d'una aeronau respectivament.

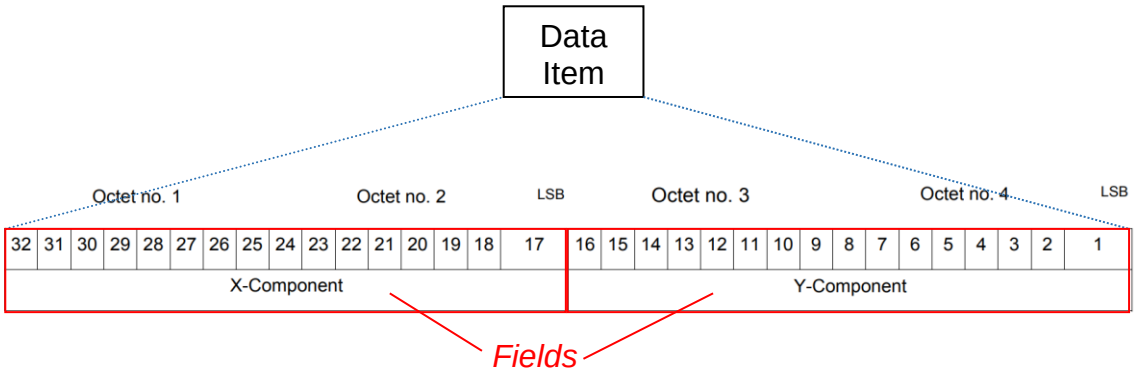

Fig. 1.6 Composició d'un *Data Item* de longitud fixe

Cada *Field* comprèn una seqüència de bits que tindrà el seu respectiu valor binari. Aquest valor binari és el resultat de representar (expressar) el valor real de la magnitud o informació continguda en el *Field* en base dos. El valor real de la informació continguda pot ser decimal (numèric) positiu o negatiu, hexadecimal (numèric i/o caràcters), octal o base 64 (veure **Fig. 1.7**). Els nombre negatius s'expressen en binari utilitzant el complement a dos.

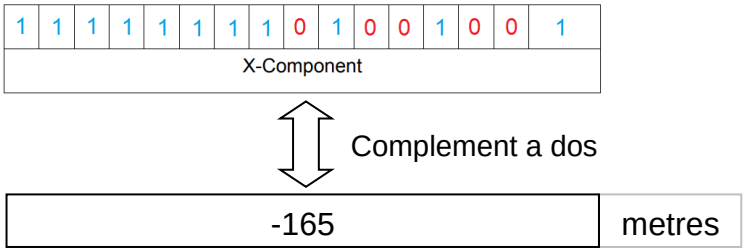

Fig. 1.7 Informació continguda en un *Field*

L'objectiu de la descodificació ASTERIX es centra en obtenir la informació (no binària, p. e. -165) continguda en els *Fields*.

1.6 Estructura general d'un missatge

Un missatge ASTERIX constitueix un bloc d'informació, que agrupa la informació transmesa en el context d'una única categoria. L'estructura d'un missatge és la següent (veure **Fig. 1.8**):

- Un primer camp d'un octet (byte), que indica la categoria (identificació) del missatge.
- Un segon camp de dos octets, que indica la longitud total del missatge en octets (sempre serà múltiple d'un octet), incloent-hi els tres octets que conjuntament ja ocupen aquests dos primers camps.
- En tercer lloc, un *Data Record* (registre) que inclou:
  - un FSPEC, que comprèn un nombre indeterminat d'octets
  - un nombre variable de *Data Items* (del catàleg de la categoria)

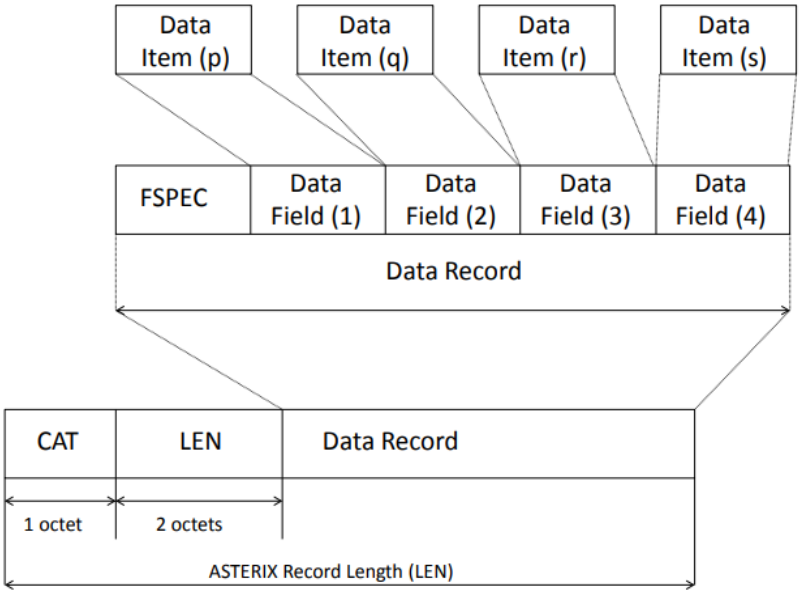

Fig. 1.8 Estructura general d'un missatge ASTERIX a partir de la Edició 2.2

El FSPEC (*Field Specification*), és un com una taula de continguts que indica amb els seus bits la presència (1) o no (0) dels diferents *Data Items* del catàleg en el missatge. Un exemple de la seva utilitat és quan els sensors no envien informació d'una aeronau, sinó del seu *status* o funcionament (periòdicament). Per tant, de tots els *Data Items (*destinats a enviar informació d'un radar de superfície p. e*.)*, només transmetrà en aquest cas, aquells que tenen com a fi expressar el seu *status*. De la mateixa manera, quan enviï informació d'un blanc (aeronau) s'obviaran aquests *Data Items*, i s'utilitzaran uns altres. En cada cas, el FSPEC indicarà la presencia o no dels corresponents *Data Items.*

A la figura (**Fig. 1.9**), es mostra un exemple d'un FSPEC de longitud un octet. Dels 8 bits que formen l'octet, només tenen valor 1 els que es troben a la posició 1, la posició 4, i la posició 7 (d'esquerra a dreta). Per cada posició d'un bit amb valor 1, a la taula UAP (*User Application Profile*) de la categoria (com a exemple la CAT10), s'especifica quin *Data Item* s'inclou en el missatge. El FSPEC d'aquesta figura està dient que al missatge només s'inclouen els *Data Items* amb FRN (*Field Reference Number*) (veure a la taula de la figura), igual a 1, 4 i 7 respectivament. Si el missatge també contingués el *Data Item amb* FRN igual a 18, el FSPEC hauria d'allargar-se més octets. Per mitjà del bit a la vuitena posició (més a la dreta) de cada octet dels FSPEC (denominat FX o ***F****ield E****x****tension Indicator*), s'indica la continuïtat del FSPEC al octet contigu, i així es pot fer successivament els octets que calgui allargar-se. Per indicar la presència del *Data Item* amb FRN 18, caldria que el bit en la quarta posició relativa al tercer octet del FSPEC, tingués valor 1. A continuació dels FSPEC, s'inclouen com a part del missatge els *Data Items* en ordre de FRN ascendent.

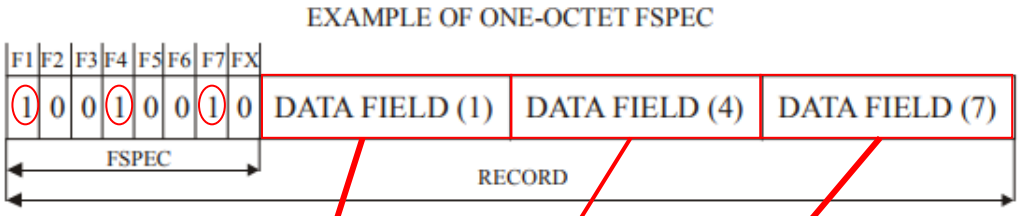

Table 3 - Standard UAP

FRN	Data Item	Information	Length in Octets
1	I010/010	Data Source Identifier	2
2	I010/000	Message Type	1
3	I010/020	Target Report Descriptor	1+
4	I010/140	Time of Day	3
5	I010/041	Position in WGS-84 Co-ordinates	8
6	I010/040	Measured Position in Polar Co-ordinates	4
7	I010/042	Position in Cartesian Co-ordinates	4
FX	-	Field Extension Indicator	-
8	I010/200	Calculated Track Velocity in Polar Co-ordinates	4
9	I010/202	Calculated Track Velocity in Cartesian Coord.	4
10	I010/161	Track Number	2
11	I010/170	Track Status	1+
12	I010/060	Mode-3/A Code in Octal Representation	2
13	I010/220	Target Address	3
14	I010/245	Target Identification	7
FX	-	Field Extension Indicator	-
15	I010/250	Mode S MB Data	1+8n
16	I010/300	Vehicle Fleet Identification	1
17	I010/090	Flight Level in Binary Representation	2
18	I010/091	Measured Height	2
19	I010/270	Target Size & Orientation	1+
20	I010/550	System Status	1
21	I010/310	Pre-programmed Message	1
FX	-	Field Extension Indicator	-
22	I010/500	Standard Deviation of Position	4
23	I010/280	Presence	1+2n
24	I010/131	Amplitude of Primary Plot	1
25	I010/210	Calculated Acceleration	2
26	Spare		
27	SP	Special Purpose Field	1+
28	RE	Reserved Expansion Field	1+
FX	-	Field Extension Indicator	-

FSPEC

Fig. 1.9 El FSPEC i la seva relació amb la taula UAP de la categoria (extreta de la categoria 10)

1.7 L'especificació d'una categoria ASTERIX

La especificació d'una categoria ASTERIX estableix el catàleg de *Data Items* que poden ser transmesos en el context d'una categoria, així com la composició de cadascun dels *Data Items* (incloent-hi el format d'estructura estàndard que utilitzen, els subelements que conté...) i el seu ordre d'inclusió en el missatge per mitjà de la taula UAP. Lògicament, hi haurà una especificació diferent per a cada categoria. L'especificació constitueix una peça fonamental per a la descodificació de missatges de la mateixa (informació que pot transmetre el missatge de la categoria i com s'estructura). Per exemple, la figura (**Fig. 1.6**) i la taula UAP de la figura (**Fig. 1.9**) s'han extret de la documentació referent a l'especificació de la categoria 10 (veure [4]). La documentació ASTERIX detalla tot els principis bàsics del format que s'han comentat en anteriors apartats, així com l'especificació pròpia de cada categoria (veure [2] [3] [4] [5] [6]).

1.8 Una gravació ASTERIX

Una gravació ASTERIX (o amb format ASTERIX), és una seqüència contínua (en ordre cronològic) de missatges ASTERIX (un rere l'altre), que poden haver estat transmesos per un o múltiples sensors. Donat que la codificació del format ASTERIX a més baix nivell és binària, la gravació és primordialment una seqüencia de bytes (o octets), en que cadascun d'aquest bytes serà al mateix temps una seqüencia de 8 bits (uns i zeros). Aquest gravacions s'enregistren com en un fitxer binari (un fitxer de bytes, que contindrà la informació codificada, de la mateixa manera que un fitxer de text conté una seqüència de caràcters que formen paraules i oracions si les sabem "interpretar" correctament) que tindrà la extensió .AST. La nostra aplicació rebrà com entrada gravacions ASTERIX (fitxers .AST) que haurà de descodificar.

1.9 Convencions del format

Algunes de les convencions del format ASTERIX són les següents:

- La informació de temps transmesa s'expressa en temps UTC (*Co-ordinated Universal Time*).

- S'estableix una identificació única per a cada sensor que utilitza ASTERIX per comunicar-se. Aquesta identificació es composa de dos elements numèrics de 3 dígits (del 000 al 255) : el SAC (*System Area Code*) i el SIC (*System Identification Code*). El SAC s'assigna en base a l'àrea geogràfica on es troba el sensor o país. En conseqüència, cada país té assignat un SAC. EL SIC permet identificar el sensor dins de la mateixa àrea geogràfica on es troba. A cada categoria ASTERIX s'inclou un *Data Item* que transmetrà els dos elements (SAC i SIC) que identifiquen el sensor.

CAPÍTOL 2. EL SUBMÒDUL DE DESCODIFICACIÓ: ASXDEC_CORE

El mòdul ASXDEC presenta un disseny modular, i en conseqüència consta de 3 submòduls principals que són els següents:

- El submòdul d'entrada
- El submòdul de descodificació
- El submòdul de sortida

Començaré parlant del submòdul de descodificació ja que conté punts importants que interessa haver tractat prèviament als altres dos submòduls.

El submòdul de descodificació "ASXDEC_core" és l'encarregat de realitzar la descodificació de les gravacions ASTERIX entrants. Aquest submòdul ha estat programat utilitzant la documentació ASTERIX, on trobem tota la informació sobre l'organització, l'estructura general d'un missatge, la codificació dels elements i totes les altres qüestions del format, que hem tractat al primer capítol i ens ajuden a entendre com s'ha de fer la descodificació. Les categories d'interès que actualment l'aplicació permet descodificar són les següents:

- Cat 10: *Monosensor Surface Movement Data*.
  Informació de sensors SMR, MLAT i ADS-B.
- Cat 21: *ADS-B reports*. Informació de sensors ADS-B.

Gran part del esforços en desenvolupar aquest submòdul, s'ha centrat en la creació d'un model genèric de categoria (especificació) que pugui ser definit per mitjà d'un fitxer XML i permeti la posterior incorporació de noves categories de manera més ràpida i eficient, sense la necessitat d'afegir o modificar el codi intern de l'aplicació. Utilitzant l'actual versió de l'aplicació es podrien decidir afegir altres categories de vital importància per a ENAIRE, com poden ser la 19 o la 20, especialment per a la implementació del nou sistema MLAT a l'aeroport de Barcelona, en un temps significativament menor i de forma autònoma per part de l'usuari.

2.1 Desenvolupar un model genèric de l'especificació d'una categoria

La part més dinàmica del format la constitueixen les diverses categories ASTERIX que representen el ventall d'aplicacions en que s'utilitza ASTERIX. Cada categoria inclou una especificació en que se'ns dona l'ordre i elements que poden aparèixer en un missatge ASTERIX de la categoria. Per tal que l'aplicació pugui portar a terme la descodificació és imprescindible que conegui les especificacions de les categories (cal parametritzar-les dins de l'aplicació). La manera de traslladar-les al món de la programació en C# és per mitjà de la definició d'objectes.

Un objecte en el context de la programació en C#, és un recipient abstracte que permet representar o modelar un tipus de cosa, i que capsula dintre seu, les característiques (atributs, subelements...) que aquesta posseeix, així com el seu comportament (accions). De moment obviarem el comportament. A grans trets, definirem les característiques bàsiques del nostre objecte com atributs.

Per posar un exemple tangible del que estic parlant, modelaré un cotxe per mitjà d'un objecte. En primer lloc, formalitzarem el nostre objecte de tipus "Cotxe", i a continuació, definim el conjunt d'atributs d'un cotxe real que volem que el nostre model tingui: el color de la pintura, la marca del vehicle i el nombre de portes. Al tractar-se d'un model, un pot negligir tot allò que no considera pertinent d'allò que vol modelar (en funció de l'aplicació perquè s'utilitzi).

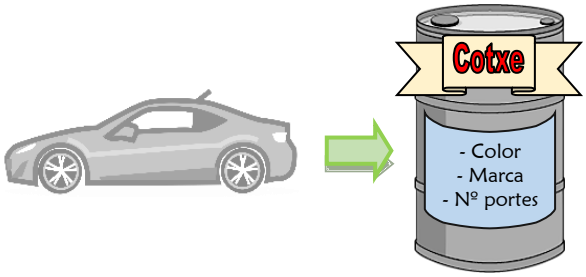

Fig. 2.1 Objecte "Cotxe"

Si el tipus de cosa que estem modelant, no es un tot homogeni, sinó que està constituïda de subelements, haurem de caracteritzar individualment el subelements, per després integrar-los al l'objecte al que pertanyin. Sabent que un cotxe té quatre rodes, modelaré una roda com un objecte de tipus "Roda" amb els següents atributs: color de la llanta i profunditat del dibuix.

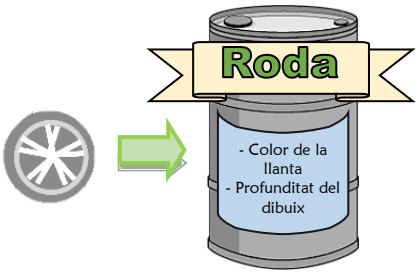

Fig. 2.2 Objecte “Roda”

A continuació integraré 4 objectes de tipus “Roda”, que modelaran l’estat de les quatre rodes del vehicle, en el meu objecte principal “Cotxe”. Quan integrem objectes dintre d’altres objectes s’estableix una estructura jeràrquica que podem representar per mitjà d’un esquema d’arbre, en que els elements d’un nivell inferior pertanyen o queden auto-continguts dins del nivell immediatament superior.

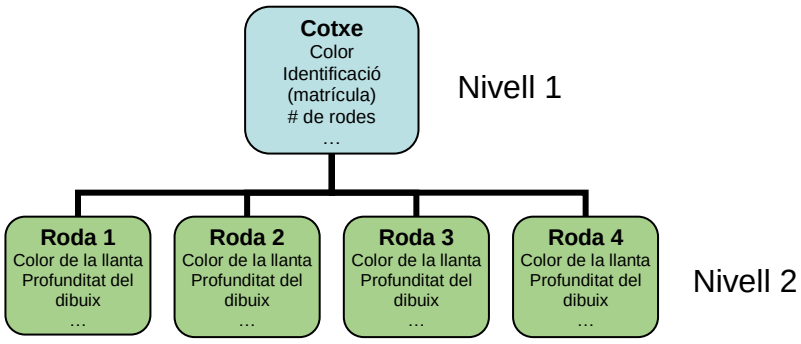

Fig. 2.3 Esquema d’arbre que representa model d’un cotxe (després integrar els objectes)

D’aquesta manera, el meu model d’un cotxe no inclourà només uns atributs genèrics que es refereixen a tot el vehicle (color, matrícula...), sinó que també el conjunt de subelements que el formen amb els seus respectius atributs. Lògicament, un cotxe té mes parts que podria decidir incloure al model.

Addicionalment, un subelement pot estar al mateix temps constituït d’altres subelements. En el cas de la roda, hagués pogut distingir la part del neumàtic i la de la llanta per exemple. A l’esquema d’arbre hauria inclòs un nivell més per sota, en que cada “Roda” tindria dos subelements (ramificacions): “Llanta” i “Neumàtic”, amb els seus atributs propis. Un pot aprofundir en el model tants nivells com cregui necessari (a mesura que inclou subelements de subelements).

Tractarem d’aplicar ara el que acabem de veure al nostre cas. L’especificació d’una categoria pot semblar una cosa abstracta i donada la diversitat de

categories ASTERIX, sembla complicat definir una plantilla tipus que pugui representar qualsevol de les especificacions. No obstant això, totes elles es composen d'elements comuns amb un format i atributs estandarditzats. Seguint l'exemple del cotxe, he caracteritzat els elements i subelements que conformen l'especificació, així com els seus respectius atributs. Els objectes "FieldUnit" i "FieldRange" constitueixen una ramificació (subelements) de l'objecte "Field":

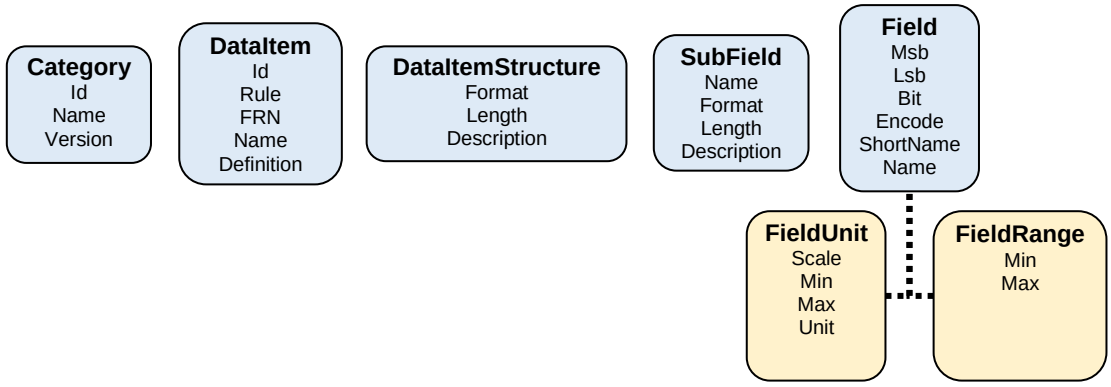

Fig. 2.4 Caracterització dels elements i subelements d'una categoria en objectes.

A la taula següent (**Taula 2.1**) s'afegeix una breu descripció dels elements i subelements (objectes) que hem definit en relació amb allò que modelen de l'especificació de la categoria:

Taula 2.1 Arquitectura del submòdul

Elements i subelements (objectes)	Descripció
Category	Element primari, que representa l'especificació de la categoria i integrarà tots els altres elements i subelements.
DataItem	Model tipus d'un *Data Item.*
DataItemStructure	Model tipus de la estructura d'un *Data Item.* Actua com element integrador del format i la composició (subelements) d'un Data Item
SubField	Model tipus d'un *Subfield,* subdivisions del *Data Item* que apareixen en alguns formats estandarditzats.
Field	Model tipus d'un Field, les unitats més bàsiques d'informació del *Data Item*.
FieldUnit	Modela la unitat numèrica que pugui tenir un *Field* (amb informació numèrica) i el seu rang de valors.
FieldRange	Modela el rang de valors d'un *Field* amb informació numèrica.

Només amb aquests set elements, podré modelar l'especificació de qualsevol categoria a priori.

2.1.1 Jerarquia dels objectes (integració)

Caracteritzats cadascun dels elements i subelements que formen l'especificació caldrà integrar-los per tal de construir el model de l'especificació d'una categoria. Tanmateix, el model serà diferent en cada cas, adaptant-se a l'estructura i elements que l'especificació defineixi. Principalment, dependrà dels diferents formats estàndards (fixe, variable...) que adoptin els *Data Items* que la composen.

Els següents esquemes d'arbre representen el model de forma jeràrquica d'esquerra a dreta (utilitzant colors també). En aquests esquemes es considera que pot haver un nombre variable d'elements dins d'un mateix nivell, i no es faran constar novament els atributs dels mateixos. Fins al nivell 3, el model és comú a totes les categories ASTERIX. No s'inclouran els objectes "FieldUnit" i "FieldRange".

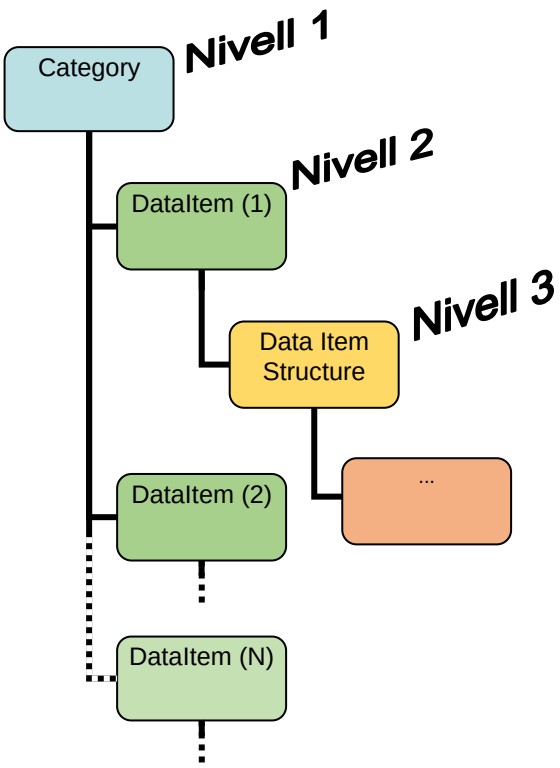

Fig. 2.5 Esquema d'arbre parcial del model d'especificació d'una categoria fins al tercer nivell.

A mesura que continuem avançant (més nivells), el model ha d'adaptar-se a les diferents casuístiques dels formats estandarditzats (fixe, variable....) que trobem en un *Data Item* i un *Subfield* i que defineixen la seva estructura tal i com hem comentat al primer capítol (veure apartat **1.4**). És per això que a partir del nivell 3 he construït tres esquemes d'arbre que pretenen exemplificar els diferents casos possibles d'estructura i integració dels elements.

En el cas d'un *Data Item* amb format fixe o repetitiu, els *Fields* queden directament integrats a l'estructura (com a exemple, veure "Data Item I021/010, Data Source Identification", pàg. 10 del document [5]).

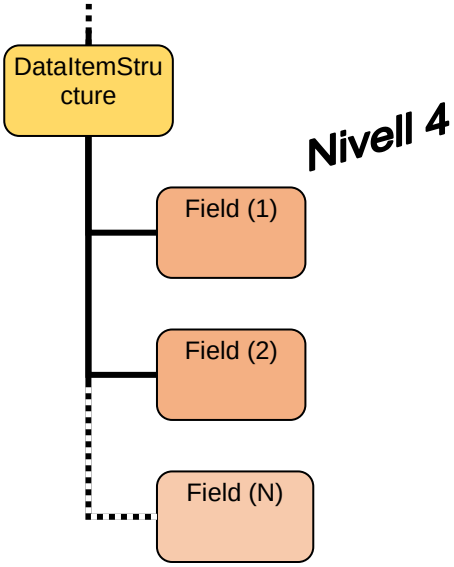

Fig. 2.6 Esquema d'arbre d'un DI amb format fixe o repetitiu.

Si el *Data Item* és variable o explícit, s'han de definir uns *Subfields* (nivell 4), que es corresponen amb les extensions o parts (de longitud fixe) que defineix el format, i que contindran els *Fields* (nivell 5). Si el Data Item és compost, cal definir el conjunt de *Subfields* (nivell 4) que pot contenir, i que en el cas que aquests siguin de longitud fixe o format repetitiu integraran directament els *Fields* (nivell 5), (com exemples, veure "Data Item I021/040, Target Report Descriptor", pàg. 13 del document [5] o veure "Data Item I021/220, Met Information", pàg. 45 del document [5]).

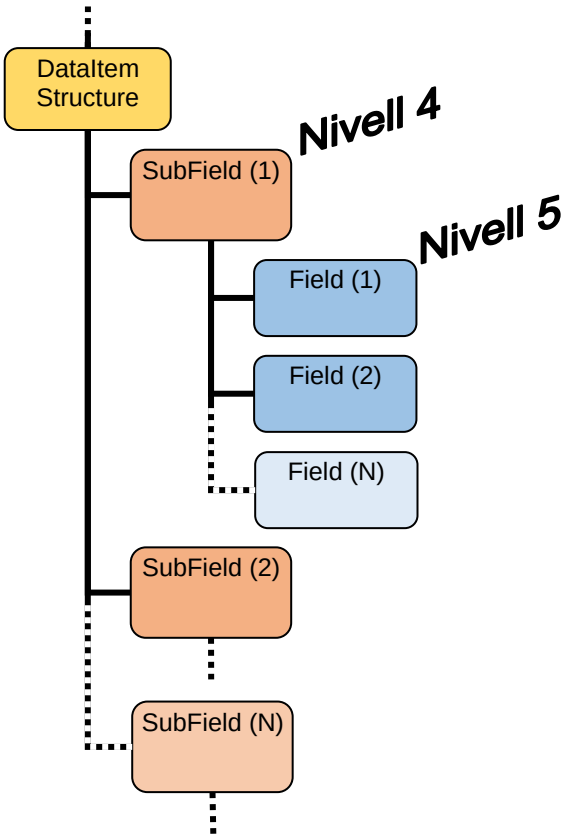

Fig. 2.7 Esquema d'arbre parcial d'un DI amb format variable o explícit, o amb format compost i el respectiu *Subfield* amb format fixe o repetitiu.

L'últim cas es correspon amb un *Data Item* compost, en el que definits el conjunt de *Subfields* (nivell 4) que conté, trobem casos en els que aquests tindran un format variable o explícit (no poden ser compostos) de manera que integraran primerament les extensions o parts com a *Subfields* (nivell 5), amb els seus corresponents *Fields* (nivell 6). No s'ha donat cap cas d'un *Data Item* d'aquest tipus en les categories 10 i 21.

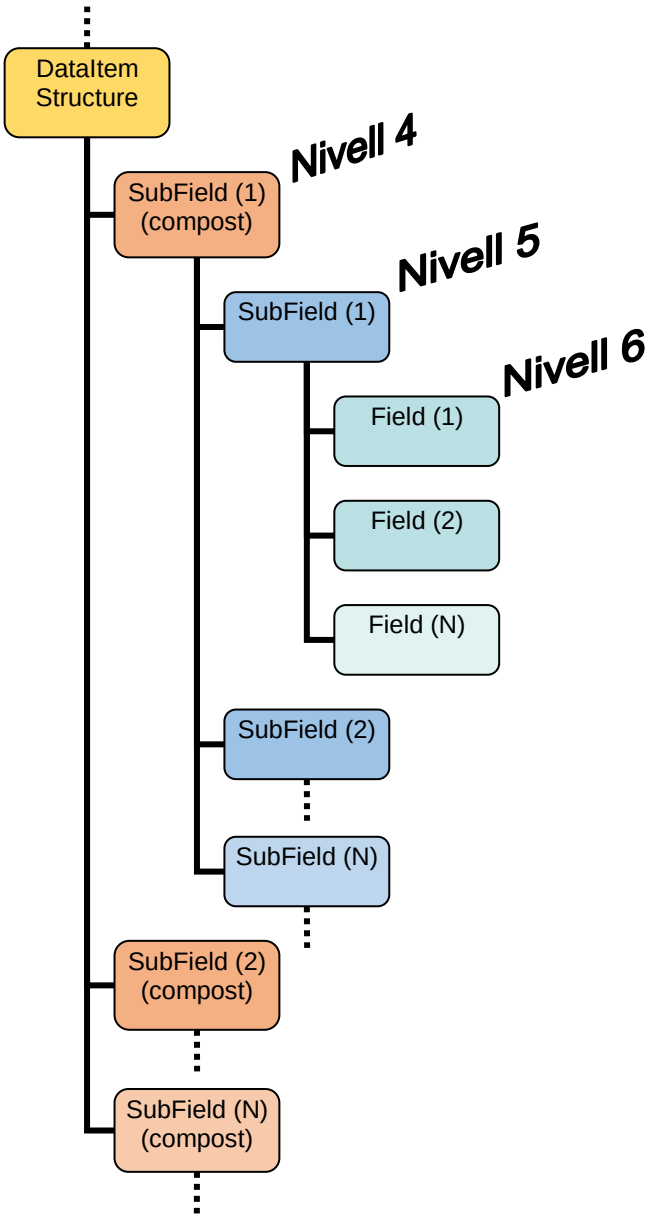

Fig. 2.8 Esquema d'arbre parcial d'un DI amb format variable, o compost i el respectiu *Subfield* (nivell 5) amb format variable o explícit.

2.2 Materialitzar l'especificació de la categoria per mitjà de la seva definició en fitxers XML

L'objectiu del model de l'especificació, és utilitzar-lo per a materialitzar una categoria en particular. Materialitzar el model implica integrar/estructurar els diferents elements i subelements que hem definit individualment, i donar valor als atributs dels mateixos, per al cas particular que volem representar.

Caracteritzat el model genèric de l'especificació, i establerta la jerarquia dels elements (no hi ha problema en que sigui flexible mentre s'hagin considerat totes les casuístiques), és pot materialitzar/definir l'especificació de la categoria per mitja de fitxers XML i la utilització d'una llibreria C# que permet la "deserialització de fitxers XML" (veure [9]).

```
<?xml version="1.0" encoding="UTF-8"?>
<Category id="010" name="Transmission of Monosensor Surface Movement Data" ver="1.1">
  <!--Data Item I010/000, Message Type-->
  <DataItem id="000" rule="mandatory" frn="2">
    <DataItemName>Message Type</DataItemName>
    <DataItemDefinition>This data item allows for a more convenient handling of the m
    <DataItemStructure format="fixed" length="1" desc="One-octet SubField length Data
      <Field msb="8" lsb="1">
        <FieldShortName>MsgTyp</FieldShortName>
        <FieldName>Type of msg.</FieldName>
        <Range min="1" max="4"></Range>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/010, Data Source Identifier-->
  <DataItem id="010" rule="mandatory" frn="1">
    <DataItemName>Data Source Identifier</DataItemName>
    <DataItemDefinition>Identification of the system from which the data are received
    <DataItemStructure format="fixed" length="2" desc="Two-octet SubField length Data
      <Field msb="16" lsb="9">
        <FieldShortName>SAC</FieldShortName>
        <FieldName>System Area Code SubField to zero</FieldName>
      </Field>
      <Field msb="8" lsb="1">
        <FieldShortName>SIC</FieldShortName>
        <FieldName>System Identification Code</FieldName>
      </Field>
```

Fig. 2.9 Definició XML de l'especificació de la CAT10 v1.1

El que l'usuari defineix al fitxer XML s'acaba materialitzant internament a l'aplicació en l'esquema d'objectes que hem vist anteriorment, en aquest cas, per a la CAT 10 v1.1 (veure **Fig. 2.10**). D'aquí que les pautes de nomenclatura i estructura (s'inclouen l'**Annex A**) que es segueixen en la definició (construcció) del fitxer XML, estiguin molt vinculades al model i la jerarquia de l'especificació que s'ha tractat en aquests apartats. La definició del fitxer XML lògicament requerirà seguir les especificacions pròpies del format XML (capçaleres,...).

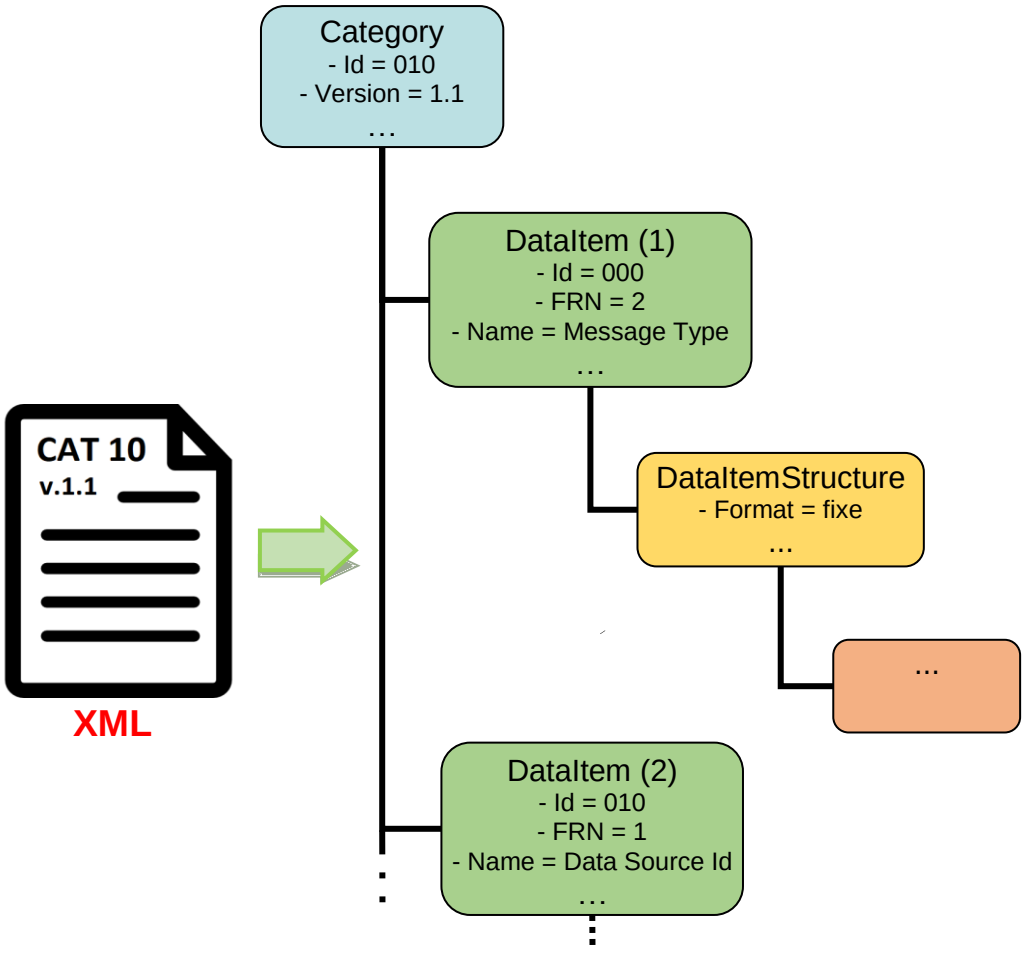

Fig. 2.10 "Deserialització" del fitxer XML de l'especificació en un esquema d'objectes

Tot els passos que s'han seguit des de la creació del model fins a la seva materialització, han anat necessàriament acompanyats del desenvolupament de mètodes (programació) genèrics d'alta complexitat, dels que no s'inclouen més detalls, que seran els que realment s'encarregaran de realitzar la descodificació dels diferents formats d'estructura (fixe, variable...) dels *Data Items* i obtenció dels valors dels *Fields*, utilitzant aquesta parametrització de l'especificació de la categoria (esquema de la **Fig. 2.10**).

2.3 Informació addicional de temps

Addicionalment als *Fields*, s'ha considerat proveir informació del temps de transmissió de cadascun dels missatges descodificats en el següent format tipus data i hora "yyMMdd HH:mm:ss.sss", donat que els *Data Items* que transmeten informació de temps donen únicament la hora UTC en segons transcorreguts des de la mitjanit del dia en curs. Un exemple d'aquest format és "210908 11:48:18.343", que representa el dia 08 de setembre de 2021 a les 11 hores, 48 minuts i 18,343 segons. La incorporació d'aquest format de temps permetrà a les aplicacions client que utilitzin el repositori de dades generat per la nostra aplicació, realitzar avaluacions de períodes de temps concrets i més enllà de les 24 hores d'un mateix dia, gràcies a la incorporació de la data.

En totes les especificacions de les categories trobarem normalment un *Data Item* que expressa el temps de transmissió del missatge (o un altre temps de referència): el *Data Item* "I010/140 Time of Day*"* a la categoria 10, el *Data Item* "I021/077 Time of ASTERIX Report Transmission*"* a la categoria 21,... D'aquí podrem obtenir l'hora de cada missatge (aplicant una conversió). Per tal de conèixer la data, s'utilitza un format de nom en els fitxers ASTERIX que inclou la data de la gravació (veure apartat **3.2**). Combinant la data amb l'hora de cada missatge, l'aplicació podrà donar el temps de transmissió en el format data i hora establert, per a cada missatge.

CAPÍTOL 3. El SUBMÒDUL D'ENTRADA: IOSS_INPUT

La nostra aplicació utilitza nombrosos fitxers d'entrada per al seu funcionament. Com ja sabem, per una banda tenim els fitxers .XML corresponents a les especificacions de les categories per saber com descodificar-les. Addicionalment, també hi ha un fitxer (.XML) de configuració de l'aplicació del que parlarem més endavant. Finalment, també tenim els fitxers ASTERIX (.AST) que seran objecte de la descodificació. Aquest submòdul d'entrada serà l'encarregat de llegir i carregar tota la informació continguda en aquests fitxers, així com també realitzar algunes comprovacions de format, verificació de rutes... sobre els mateixos. Tot i que el submòdul de descodificació s'hagi explicat abans (per una qüestió de comprensió del document), des d'un punt de vista seqüencial en l'execució, aquest submòdul va abans.

3.1 El fitxer d'inicialització: "asxdec_ini.xml"

És imprescindible, que la nostra aplicació conegui les rutes corresponents a tots els fitxers que hem mencionat anteriorment. És per això, que l'aplicació incorpora un fitxer XML d'inicialització en el qual l'usuari pot proporcionar aquestes rutes (veure **Fig. 3.1**). Aquests fitxer ve incorporat per defecte a l'arrel de la carpeta d'instal·lació perquè sigui editat per l'usuari.

```
<ASXDECini>
    <ASXDECconfigFilePath>
        asxdec_config.xml
    </ASXDECconfigFilePath>
    <ASTERIXspecificationFilesPaths>
        asterix_cat021_2_1_RE1_1.xml
        asterix_cat010_1_1.xml
    </ASTERIXspecificationFilesPaths>
    <ASTERIXtodecodeFilePaths>
        201002-lebl-080001.ast
    </ASTERIXtodecodeFilePaths>
</ASXDECini>
```

Fig. 3.1 Fragment del fitxer XML d'inicialització amb les respectives rutes dels fitxers

El segon nivell del fitxer XML consta de tres divisions (capçaleres) en les que incloure les diferents rutes (veure **Taula 3.1**):

Taula 3.1 Descripció de les rutes que escriurem en cadascuna de les divisions (capçaleres) del fitxer d'inicialització

<ASXDECconfigFilePath> ... </ASXDECconfigFilePath>	Escriurem únicament la ruta del fitxer (.XML) de configuració de l'aplicació
<ASTERIXspecificationFilesPaths> ... </ASTERIXspecificationFilesPaths>	Escriurem les rutes del fitxers (.XML) d'especificació de les categories
<ASTERIXtodecodeFilePaths> ... </ASTERIXtodecodeFilePaths>	Escriurem les rutes dels fitxers ASTERIX (.AST) que volem descodificar

És important tenir en compte, que si proveïm únicament el nom dels fitxer (no la seva ruta completa) l'aplicació els buscarà a l'arrel de la carpeta d'instal·lació de la mateixa. En el cas particular dels fitxers corresponents a les especificacions de les categories, si només es dona el nom del fitxer, l'aplicació els buscarà dins la carpeta "specification-folder" que es troba a l'arrel de la carpeta d'instal·lació. En cas que els fitxers no es trobin en algun d'aquests supòsits (altres ubicacions), caldrà donar la seva ruta completa. Per exemple, en el cas que el fitxer de configuració estigués a la carpeta de documents d'usuari escriuríem la seva ruta completa com: "C:\Usuarios*{USUARI}*\Documents\asxdec_config.xml".

Abans de començar a llegir i carregar els fitxers, el submòdul verificarà que les rutes que es donen efectivament existeixen, ja que en cas d'un possible error tipogràfic en alguna de les rutes o equivocació es podria detectar des d'un bon principi de l'execució.

Finalment, serà molt important proporcionar també la ruta del fitxer d'inicialització a l'aplicació. Això és farà en el moment de l'execució o crida de l'aplicació, donant la ruta d'aquest fitxer com argument de consola. A l'apartat *5.3* s'explicarà com fer-ho. Si l'usuari decideix executar més d'una instància de l'aplicació, podrà proporcionar un fitxer d'inicialització diferent en cada cas.

3.2 El format de nom dels fitxers ASTERIX (.AST)

S'ha seguit el conveni per al nom dels fitxers ASTERIX (gravacions) que utilitza ENAIRE en els seus gravadors automàtics (365dx24h). El format de nom és el següent: "yyMMdd-????-hhmmss", on "yyMMdd" i "hhmmss" indiquen la data i hora UTC d'inici de la gravació respectivament. Els caràcters situats entre els dos guions "????" són de lliure elecció (no necessàriament quatre). Un exemple de format de nom és "201002-lebl-080001", corresponent a una gravació del dia 2 d'octubre de 2020 que va començar a les 8 hores i un segon, a l'aeroport de BCN. La data del fitxer s'utilitza per als fins que es descriuen a l'apartat **2.3**.

CAPÍTOL 4. EL SUBMÒDUL DE SORTIDA: LOSS_OUTPUT

L'objectiu primordial de l'aplicació és obtenir la informació descodificada dels fitxers ASTERIX, i en conseqüència, oferir-la a la sortida en un format determinat. Aquesta submòdul de sortida, serà l'encarregat de portar a terme les diferents implementacions que emmagatzemin la descodificació en un format estructurat (base de dades, .CSV...) i accessible per a altres aplicacions que vulguin utilitzar les dades. En el nostre cas, s'ha implementat únicament la sortida en format taula de base de dades, però gràcies al disseny escalable del submòdul, nous formats de sortida poden ser incorporats en un futur, sense necessàriament haver de realitzar modificacions en els altres dos submòduls. També s'inclou com a part d'aquest submòdul, els missatges de consola que l'aplicació proporcionarà a l'usuari en relació a l'estat de l'execució, alertes... i que posteriorment registrarà en fitxers .LOG per a la seva posterior consulta.

4.1 Sortida en format taula en base de dades

El sistema de gestió base de dades que s'ha seleccionat per a la nostra aplicació és "MariaDB" (molt semblant a "MySQL"). La utilització del mateix comportarà requeriments de *software* addicionals (veure apartat **5.2**). Dins d'una base de dades definim taules en les que emmagatzema la informació.

4.1.1 Creació automàtica de les taules (base de dades)

Una taula consta primordialment de files i columnes. En el nostre cas, les files es correspondran amb cadascun dels missatges que descodificarem de les gravacions ASTERIX entrants. Per a cadascun d'aquests missatges, a les columnes trobarem els diferents *Fields* o camps d'informació que podran contenir aquests missatges. Com els *Fields* seran diferents per a cada categoria ASTERIX (diferent especificació) i en conseqüència les columnes de la taula, hi haurà una taula diferent per a cada categoria.

La creació d'una taula en base de dades, requereix primordialment la definició de cadascuna de les columnes incloent-hi el seu nom, tipus de dades (numèric, caràcters...), acceptació de valors nuls..., i el nom de la taula. Per a cadascuna de les categories, el submòdul s'encarregarà de crear la corresponent taula de forma autònoma, sense necessitat que l'usuari hagi de fer-ho manualment. Tanmateix, la base de dades si haurà d'haver estat creada prèviament per l'usuari de forma manual. Per tal de donar un nom a la taula, l'aplicació consultarà el fitxer XML de configuració on l'usuari ha definit prèviament el nom de cadascuna de les taules (veure apartat **5.1.1.1**). En quant a les columnes, l'aplicació consultarà l'especificació de cada categoria per obtenir els *Fields* que aquesta tingui. L'atribut "FieldShortName" definit en cadascun dels "Fields" al

XML de l'especificació (que representen els *Fields)* s'utilitzarà per donar nom a cada columna en combinació amb la identificació (id) del *Data Item* al qual pertanyi cada *Field* en qüestió. El format del nom serà el següent: "di_{ID del *DataItem* al que pertany el *Field*}_{"FieldShortName"}". Per posar un exemple, el nom de la columna corresponent al *Field* "Message Type" de la categoria 010, que s'ha definit al XML (veure **Fig. 4.1**) i que forma part del *Data Item* amb identificació "000" quedaria així: "di_000_MsgTyp".

```
<Field msb="8" lsb="1">
  <FieldShortName>MsgTyp</FieldShortName>
  <FieldName>Type of msg.</FieldName>
  <Range min="1" max="4"></Range>
</Field>
```

Fig. 4.1 Fragment del fitxer XML d'especificació de la categoria 010, on es defineix un "Field" i el seu atribut "FieldShortName"

Pel que respecte al tipus de les dades en cada columna, dependrà principalment del "encode" (tipus de descodificació) que s'hagi definit en cada *Field*, entenent que aquest afecta directament al tipus de dades que s'obtenen a la sortida de la descodificació (veure **Taula 4.1**). Si per exemple descodifiquem el "Target Address" d'una aeronau, obtindrem una adreça en format hexadecimal que haurem d'expressar necessàriament en caràcters (no només números).

Taula 4.1 Relació entre el "encode" definit per al *Field* i el tipus de dades seleccionat per a la columna a la taula (base de dades)

"encode" del *Field*	**tipus de dades a la columna de la taula (BD)**
uint (valors enters)	UINT
int (valors enters)	INT
uint o int (valors amb decimals)[1]	DOUBLE
octal (octal) hex (hexadecimal) sixbitschar ("base64")	VARCHAR(50)[2]

[1] Tindran valors amb decimals aquells *Fields* en que el seu factor d'escala no sigui enter.
[2] En el tipus de dades "VARCHAR" cal definir la longitud màxima en caràcters que pot contenir cada cel·la de la columna entre parèntesis, per defecte 50.

Finalment, es considera que totes les columnes poden tenir valors nuls, ja que no tots els missatges transmeten tota la informació disponible per a una categoria (FSPEC).

4.1.1.1 Definició de les columnes en Data Items que tenen o contenen elements en format repetitiu

En el en cas de *Data Items* o *Subfields* amb format repetitiu, per a un mateix missatge (fila) farà falta emmagatzemar el valor de les successives repeticions d'un mateix *Field* (columna). Per fer-ho, s'ha considerat emmagatzemar els diferents valors com una única seqüència, separats entre si per un punt i coma seguit d'un espai (ambdós són caràcters). Les repeticions quedaran encadenades en el format següent: "{valor_rep1}; {valor_rep2}; ..." Com a conseqüència de la introducció de caràcters, haurem de considerar tota la seqüència com una cadena de text independentment del tipus de "encode" que el *Field* tingui. Per tant, el tipus de dades d'aquestes columnes serà necessàriament "VARCHAR" en tots els casos. Durant la creació de les columnes amb tipus de dades "VARCHAR" és obligatori definir el màxim de caràcters que pot contenir una cel·la de la columna. Aquest màxim dependrà clarament de la longitud de la seqüència i del nombre de repeticions (entre 1 i 255) que pugui incloure (depenent de cada missatge). Per eliminar aquesta incertesa, s'ha adoptat que l'usuari estableixi el nombre màxim de repeticions que podran ser emmagatzemades a la taula en aquests casos. Serà decisió de l'usuari definir un valor més conservador o no. A l'apartat **5.1.1.1** és tractarà com l'usuari haurà definir aquest màxim. En aquells missatges que continguin més repeticions que el llindar establert es descartaran els valors de les repeticions que l'excedeixin (es truncaran).

Fixat ara un nombre màxim de repeticions, l'aplicació podrà calcular la longitud màxima ($L_{màx}$) de la seqüència (veure fórmula **4.1**) que s'utilitzarà per crear les columnes.

$$L_{màx}\ (\text{seqüència}) = REP_{màx} \cdot L\ (\text{d'un valor}) + (REP_{màx} - 1) \cdot c \qquad \textbf{(4.1)}$$

Nota: on c es el nombre de caràcters separadors (dos) i L és la longitud en caràcters d'un únic valor (una repetició).

La longitud en caràcters que ocupa un únic valor d'un *Field* (valor descodificat) pot ser calculada en cada cas. Quan descodifiquem un *Field* sempre partim d'una cadena binària (bits), que majoritàriament haurem d'expressar en una altra base: decimal, octal, hexadecimal... Per tant, la base de partida sempre serà dos (binaria). Coneixent la longitud de la cadena binària (l), podem obtenir el nombre de dígits o caràcters en la base de sortida (y) aplicant la següent expressió (veure fórmula **4.2**). Igualem el rang de valors en ambdues bases i aïllem la longitud en caràcters en la base de sortida (L). Sempre haurem d'arrodonir L a l'enter superior més pròxim. Aquesta longitud (L) es correspon amb la de l'anterior expressió (d'un valor).

$$2^{l} = y^{L} \quad \textbf{(4.2)}$$

Finalment, és important remarcar que aquesta limitació d'emmagatzematge, en quant al nombre de repeticions, afecta estrictament a la sortida en format taula en base de dades, i no necessàriament a cap dels altres formats de sortida que puguin ser implementats.

4.1.1.2 Columnes addicionals

A part de les columnes corresponents als *Fields*, la taula inclourà una primera columna per al temps de transmissió de cada missatge, en relació amb el que s'ha comentat a l'apartat **2.3**.

4.2 Missatges de consola i generació de fitxers .LOG

Durant l'execució, l'aplicació mostrarà a la consola diferents missatges que informaran a l'usuari que està fent en cada moment l'aplicació i comprovacions que estigui portant a terme (veure **Fig. 4.2**). L'ordre d'aparició dels missatges segueix el procés seqüencial d'execució de l'aplicació:

1. Lectura i comprovació del fitxer d'inicialització (rutes).
2. Lectura i comprovació dels fitxers d'especificació de les categories.
3. Lectura i comprovació del fitxer de configuració
4. Inici de la descodificació de les gravacions

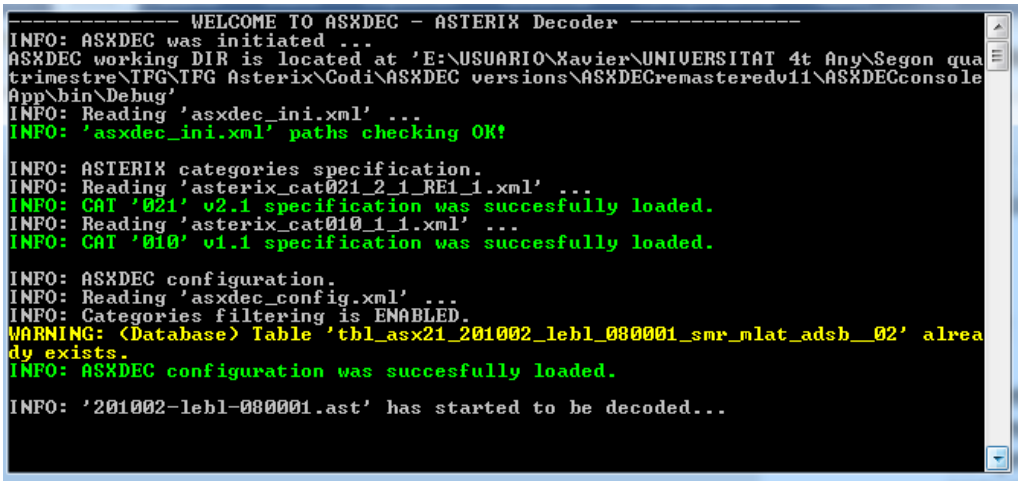

Fig. 4.2 Captura de la consola durant l'execució de l'aplicació

Els missatges són autodescriptius i estan escrits en anglès. Utilitzen un codi de tres colors per indicar la seva tipologia o gravetat :

- Verd (Informació): informa d'una comprovació correcta o que un procés ha finalitzat satisfactòriament.
- Groc (Alertes): alerta a l'usuari de possibles incoherències en la configuració de l'aplicació o potencials errors, però que no afecten a la correcta execució del programa.
- Vermell (Errors): errors crítics que afecten a la continuïtat de l'execució del programa.

D'altra banda, donat que el client de l'aplicació automatitzi la creació del repositori d'informació i no estigui present durant l'execució, a la carpeta "logs" que es troba dins de la carpeta d'instal·lació de l'aplicació, l'aplicació crearà un fitxer de text amb extensió .LOG, on registrarà els missatges de cada execució (un fitxer per execució). Aquests fitxers sempre podran ser consultats per l'usuari a posteriori. Cada fitxer LOG tindrà com a nom el temps d'inici de l'execució, en el següent format de data i hora "yyyyMMdd_HHmmss". Un fitxer LOG amb nom "20210910_191536.log" es correspon amb una execució de l'aplicació del dia 10 de setembre de 2021, a les 19 hores, 15 minuts i 36 segons.

CAPÍTOL 5. CONFIGURACIÓ I UTILITZACIÓ DE L'APLICACIÓ

5.1 El fitxer de configuració

El fitxer de configuració de l'aplicació és un fitxer amb extensió XML, en que es defineixen primordialment els paràmetres corresponents al format de sortida de la informació, concretament de la base de dades i les taules de la mateixa, i per una altra banda, la configuració dels filtres que l'aplicació implementa durant la descodificació. Aquest fitxer de configuració s'inclou a la carpeta d'instal·lació de l'aplicació com "asxdec_config.xml, i és lliure de ser editat per l'usuari perquè l'adapti als seus requeriments, respectant el format del mateix.

5.1.1 Configuració primària

5.1.1.1 Base de dades i les taules

A la configuració primària, l'usuari pot configurar en primer lloc la connexió a la base de dades, així com definir el nom de les taules on s'emmagatzemarà la informació descodificada per a cadascuna de les categories ASTERIX. Haurà de proveir el nom del *host* ("MariaDB" treballa amb servidors, també "localment"), juntament amb l'usuari i contrasenya d'accés de la sessió, així com el nom de la base de dades o *schema*. A la següent figura (**Fig. 5.1**), es mostra la relació entre aquests paràmetres, i el que l'usuari haurà configurat al programa "HeidiSQL" que incorpora "MariaDB", en relació a la sessió d'accés i la base de dades.

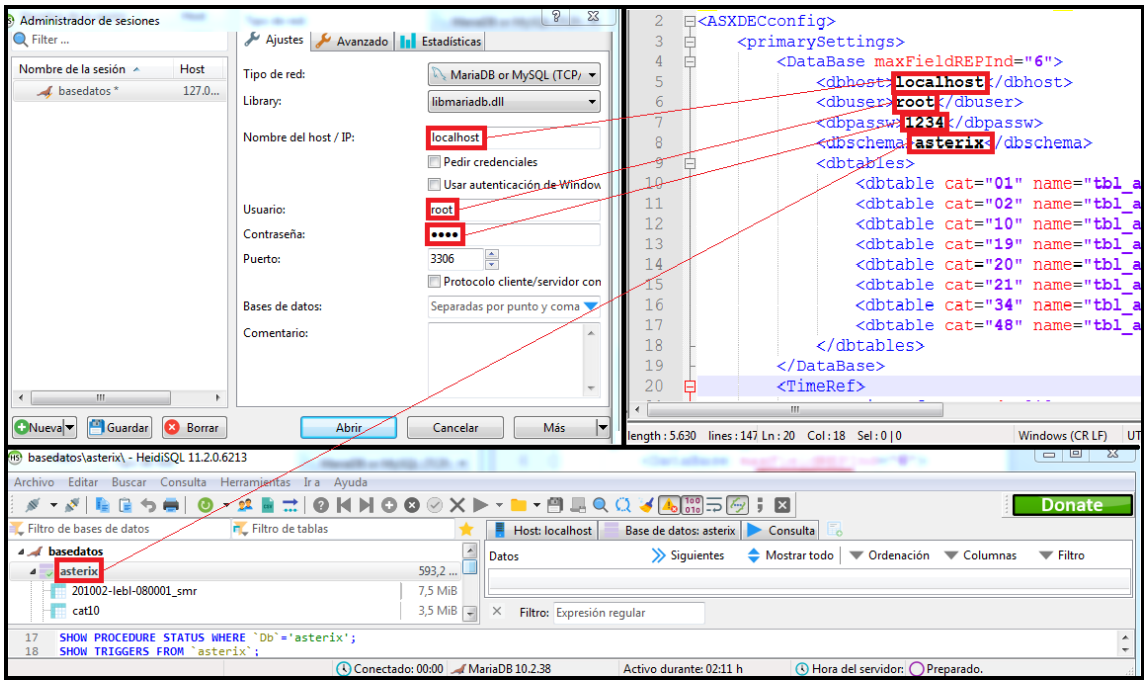

Fig. 5.1 Captura que relaciona els paràmetres de la base de dades (fitxer) i el que s'estableix al programa "HeidiSQL"

Seguidament, l'usuari podrà definir les taules, indicant el nombre de la categoria a la qual pertany ("cat") i el nom de la taula ("name"). És important considerar no utilitzar guions o altres caràcters no permesos (per la sintaxis "MySQL") en el nom. Recordar que serà l'aplicació qui s'encarregui de crear automàticament durant l'execució les taules utilitzant aquests paràmetres i les especificacions de les categories (XML) proporcionades, sense necessitar que ho faci l'usuari manualment. Tanmateix, la base de dades si haurà d'haver estat creada prèviament per l'usuari.

```
<DataBase maxFieldREPInd="6">
    <dbhost>localhost</dbhost>
    <dbuser>root</dbuser>
    <dbpassw>1234</dbpassw>
    <dbschema>asterix</dbschema>
    <dbtables>
        <dbtable cat="01" name="tbl_asx01"> </dbtable>
        <dbtable cat="02" name="tbl_asx02"> </dbtable>
        <dbtable cat="10" name="tbl_asx10_asxd_133602" maxFieldREPInd="1"> </dbtable>
        <dbtable cat="19" name="tbl_asx19" maxFieldREPInd="8"> </dbtable>
        <dbtable cat="20" name="tbl_asx20"> </dbtable>
        <dbtable cat="21" name="tbl_asx21"> </dbtable>
        <dbtable cat="34" name="tbl_asx34"> </dbtable>
        <dbtable cat="48" name="tbl_asx48"> </dbtable>
    </dbtables>
</DataBase>
```

Fig. 5.2 Captura de la part de configuració de la base de dades

Finalment, l'usuari podrà establir el límit de repeticions que seran emmagatzemades en aquells camps que pertanyen a elements amb format repetitiu, i que com ja hem comentat anteriorment (veure apartat **4.1.1.1**), obliguen a emmagatzemar més d'un valor d'informació per a un mateix *Field* o columna en un missatge. Per defecte, es considerarà un valor màxim de repeticions per a totes les taules que serà igual a 6. Aquest valor global per defecte, podrà ser definit/modificat al atribut "maxFieldREPInd" de l'element "DataBase" (requadrat en verd a la **Fig. 5.2**). Si es vol definir un valor màxim individualitzat per a cada taula (categoria), caldrà definir/modificar l'atribut "maxFieldREPInd" directament a la taula ("dbtable"), requadrat en vermell a la **Fig. 5.2**. En les taules en que es defineixi el valor individual aquest lògicament tindrà prioritat sobre del valor per defecte comentat anteriorment.

5.1.1.2 Establir la referència de temps per cada categoria

En aquesta secció del fitxer de configuració, l'usuari haurà d'indicar per a cada categoria quin serà el *Data Item* que l'aplicació haurà de consultar per a obtenir el temps de transmissió dels missatges (o el temps que es consideri com a referència) per als fins que es descriuen a l'apartat 2.3, indicant la identificació de la categoria ("catId") i del *Data Item* ("TRDataItemId") en qüestió (veure **Fig. 5.3**). És imprescindible que l'usuari proveeixi aquesta informació per a les categories que s'hagi donat el fitxer XML amb l'especificació.

```
<TimeRef>
    <CatTimeRefDI catId="010" TRDataItemId="140"></CatTimeRefDI>
    <CatTimeRefDI catId="021" TRDataItemId="077"></CatTimeRefDI>
</TimeRef>
```

Fig. 5.3 Captura de la part de configuració de la referència de temps de cada categoria

5.1.1.3 Activació i desactivació dels filtres

L'ultima secció de la configuració primària, permet l'activació o desactivació per mitjà d'un booleà (*true* o *false*) dels tres filtres que l'aplicació incorpora (veure **Fig. 5.4**). Aquests hauran estat prèviament configurats, i s'explicaran en més detall al següent apartat.

```
<Filters>
    <enableCatFiltering en="false"></enableCatFiltering>
    <enableDIFiltering en="false"></enableDIFiltering>
    <enableSensorFiltering en="false"></enableSensorFiltering>
</Filters>
```

Fig. 5.4 Captura de la part d'activació dels filtres

5.1.2 Filtres

L'aplicació incorpora tres tipus de filtres que ens permeten acotar la informació d'interès, estalviant temps d'execució de l'aplicació i espai en memòria:

- El filtre de categories
- El filtre de *Data Items* (dins de cada categoria)
- El filtre de sensors (transmissors)

5.1.2.1 Filtratge de categories

El primer filtre que l'usuari pot configurar és el filtre de categories ASTERIX. D'aquesta manera només seran descodificats els missatges d'aquelles categories que l'usuari hagi llistat, i estiguin activades (*true*). Així mateix, s'habilita l'opció de considerar també aquelles categories que no estiguin explícitament llistades com a actives, o no actives, per mitjà del booleà "enableNotListedCat" per defecte *false* (o no actives) (requadrat en taronja a la **Fig. 5.5**). Si agafem l'exemple de la figura (**Fig. 5.5**), estaríem considerant només les categories 10, 21 i 48, i descartaríem tots els missatges d'altres categories.

```
<CategoriesFilter enableNotListedCat = "false">
    <Category id="010" en="true"/>
    <Category id="021" en="true"/>
    <Category id="048" en="true"/>
</CategoriesFilter>
```

Fig. 5.5 Captura d'una configuració del filtre de categories amb l'atribut "enableNotListedCat" com a no actiu i les categories llistades com actives

Invertir el valor "enableNotListedCat" a *true* és útil en el cas que ens interessin totes les categories exceptuant alguns casos particulars, que llistarem com a no actives (*false*). En el cas de la **Fig. 5.6**, estaríem descodificant els missatges de totes les categories, exceptuant els de les categories 10, 21 i 48.

```
<CategoriesFilter enableNotListedCat = "true">
    <Category id="010" en="false"/>
    <Category id="021" en="false"/>
    <Category id="048" en="false"/>
</CategoriesFilter>
```

Fig. 5.6 Captura d'una configuració del filtre de categories amb l'atribut "enableNotListedCat" com a actiu i les categories llistades com a no actives

5.1.2.2 Filtratge de Data Items

En aquest segon filtre, l'usuari podrà filtrar els *Data Items* d'interès d'una categoria. El funcionament és molt semblant al de l'anterior filtre, amb l'única diferència que ara parlarem de saltar o no saltar, en comptes de desactivar o activar respectivament. Els *Data Items* llistats o definits dins d'una categoria i configurats per no ser saltats (false) seran descodificats. El booleà "skipNotListedDI" estableix la consideració que tindran aquells *Data Items* que no estiguin explícitament llistats, per defecte *true* (no seran descodificats).

```
<DataItemsFilter>
    <CatDataItems catId="010" skipNotListedDI = "true">
        <DataItem id="000" skip="false"/>
        <DataItem id="010" skip="false"/>
        <DataItem id="020" skip="false"/>
    </CatDataItems>
    <CatDataItems catId="019" skipNotListedDI = "true">
        <DataItem id ="001" skip ="false"/>
```

Fig. 5.7 Captura d'una configuració del filtre de *Data Items* amb "skipNotListedDI" com a *true*

En el cas de la imatge (**Fig. 5.7**), només descodificaríem els *Data Items* 0, 10 i 20 de la categoria 10, i el *Data Item* 1 de la categoria 19. Tots els altres *Data Items* d'aquestes dues categories serien saltats/descartats.

Definir el valor "skipNotListedDI" com a *false* (no els saltem) és útil si ens interessen tots els *Data Items* d'una determinada categoria exceptuant alguns en particular que llistarem amb booleà *true* (els saltem).

```
<DataItemsFilter>
    <CatDataItems catId="010" skipNotListedDI = "false">
        <DataItem id="000" skip="true"/>
        <DataItem id="010" skip="true"/>
        <DataItem id="020" skip="true"/>
    </CatDataItems>
```

Fig. 5.8 Captura d'una configuració del filtre de *Data Items* amb "skipNotListedDI" com a *false*

Si utilitzem la imatge (**Fig. 5.8**) com a exemple, l'aplicació descodificaria tots els *Data Items* de al categoria 10, exceptuant el 0, el 10 i el 20, que els descartaria.

5.1.2.3 Sensors

Abans de tractar el tercer i últim filtre, cal parlar del funcionament de l'aplicació en relació als sensors que l'usuari introdueix al fitxer de configuració. Donat que l'aplicació està enfocada a generar un repositori d'informació provinent de sensors coneguts, dels que sovint es voldrà portar a terme una avaluació posterior (anàlisis), aquesta ha estat desenvolupada per treballar únicament amb els sensors que l'usuari inclogui a la llista. Els missatges provinents d'algun sensor que no estigui a la llista del "SensorsFilter" no seran tinguts en compte (descartats), i és responsabilitat de l'usuari incloure'ls. Aquesta llista es mostra a continuació a la figura (**Fig. 5.9**) on cada sensor queda definit com un element ("<radar> ... </radar>"). La part que es correspon amb la definició de cada sensor és el que s'ha requadrat en taronja a la imatge (**Fig. 5.9)**; obviarem de moment els booleans que no s'inclouen al requadre en *false*.

```
<SensorsFilter defaultDSIid="010">
    <radar id="SMR_GCXO" sac="0" sic="1" en="false">
        <CatSensor id="010" ver="0101" en="true" />
    </radar>
    <radar id="SMR_LEAS" sac="0" sic="5" en="false">
        <CatSensor id="010" ver="0101" en="true" />
    </radar>
    <radar id="MLAT_GCXO" sac="0" sic="101" en="false">
        <CatSensor id="010" ver="0101" en="true" />
        <CatSensor id="021" ver="0201" en="true" />
    </radar>
```

Fig. 5.9 Captura en que s'han definit els sensors d'interès

Per cada sensor l'usuari haurà de donar el seu nom o identificació ("id"), i el seu corresponent SAC i SIC (aquests dos paràmetres s'han explicat a l'apartat **1.9**). Així mateix, serà imprescindible especificar de quines categories ASTERIX podran ser els missatges enviats per cadascun d'aquests sensors. Per a cada categoria, s'haurà donat el seu número ("id"), versió, i si el sensor està autoritzat o no (en cas afirmatiu: *true*) a transmetre missatges de la mateixa. Els missatges provinents d'un sensor definit a la llista, però d'una categoria que no ha estat definida per aquell sensor (com activa o no activa) seran descartats. Si temporalment un sensor no estigués enviant correctament els missatges en una categoria determinada, podríem desactivar els missatges definint com a inactiva (*false*) la categoria en aquell sensor.

Per al primer sensor definit a la imatge, l'aplicació només considerarà els missatges de la categoria 10. Pel tipus de sensor que es tracta, un *Surface Movement Radar* (SMR), té tot el sentit que només s'hagi definit aquesta categoria ja que és l'única que utilitzen aquest tipus de radars. No obstant, hi ha sensors que poden transmetre missatges de diferents categories.

Finalment, el camp “defaultDSIid” (al començament de la llista) s’utilitza per proporcionar la identificació del *Data Item* (“Data Source Identifier”) que contindrà el SAC i el SIC que identifica al sensor que transmet cada missatge, necessari per a totes les comprovacions anteriors. Afortunadament, totes les categories assignen a aquest *Data Item* la identificació “010”.

5.1.2.4 Filtratge de sensors

Tot i que el que s’ha explicat a l’anterior apartat pot ja semblar un filtre en si mateix, és un comportament que no ve definit per l’usuari. El que s’explica a continuació, és el que l’usuari pot activar o desactivar com a filtre de sensors, a l’apartat **5.1.1.3**.

La part de filtratge de sensors permet activar *(true)* o desactivar *(false)* la descodificació dels missatges provinents d’un sensor de la llista, utilitzant el booleà que a la figura (**Fig. 5.10**) s’ha requadrat en verd, i que a l’apartat anterior havíem obviat. Aquest filtratge és útil en cas que l’usuari només vulgui centrar-se en alguns dels sensors d’interès que ja ha inclòs a la llista, deshabilitant els altres sense necessitat d’esborrar-los de la llista. Agafant la figura (**Fig. 5.10**) com a exemple, l’usuari hauria desactivat els dos primers sensors de la llista i l’aplicació estaria considerant únicament els missatges provinents del tercer sensor.

```
<SensorsFilter defaultDSIid="010">
    <radar id="SMR_GCXO" sac="0" sic="1" en="false">
        <CatSensor id="010" ver="0101" en="true" />
    </radar>
    <radar id="SMR_LEAS" sac="0" sic="5" en="false">
        <CatSensor id="010" ver="0101" en="true" />
    </radar>
    <radar id="MLAT_GCXO" sac="0" sic="101" en="true">
        <CatSensor id="010" ver="0101" en="true" />
        <CatSensor id="021" ver="0201" en="true" />
    </radar>
```

Fig. 5.10 Activació o desactivació de sensors (requadrat en verd) per al seu filtratge

La desactivació d’un sensor en el filtre és completa, però en cas que estigui activat (*true*), això no eximirà les comprovacions de l’apartat anterior en relació amb les categories que els sensors tingui o no habilitades.

5.2 Instal·lació i requeriments addicionals de software

El *setup* (instal·lador) de l'aplicació consta d'un fitxer .EXE que haurem d'executar. Fet això, només caldrà seguir els passos que ens marca el *setup* fins que finalitzi la instal·lació. Al llarg de la instal·lació l'usuari podrà escollir la ruta d'instal·lació de l'aplicació. És important no escollir una carpeta protegida. Tampoc és podrà instal·lar a les carpetes tradicionals d'instal·lació d'aplicacions de *Windows,* "ProgramFiles" i "ProgramFilesx86", ja que l'edició dels fitxers XML s'ha comprovat que pot ocasionar problemes. Es recomana instal·lar-ho directament a l'arrel del disc ("C:\").

D'altra banda, tal i com hem comentat al capítol anterior, la sortida en format taula en base de dades comporta el requeriment addicional d'una aplicació de gestió de base de dades, que en el nostre cas serà "MariaDB". Abans d'utilitzar la nostra aplicació caldrà haver instal·lat prèviament una versió aquest programa (es recomana la v10.2). També serà necessari un programa d'accés i visualització/edició de bases de dades com pot ser "HeidiSQL", que en principi ja incorpora el mateix instal·lador de "MariaDB" (veure [10]).

5.3 Crida de l'aplicació des de consola (CLI)

Per tal de cridar la nostra aplicació des de consola, el primer que l'usuari haurà de fer es executar el programa "Símbol del sistema", preferentment com administrador. Podem trobar-lo fàcilment utilitzant la barra de cerca. Un cop oberta la consola, escriurem la ruta de l'executable de la nostra aplicació ("ASXDEC.exe") que es troba a la carpeta d'instal·lació de l'aplicació, per exemple "C:\ASXDEC\ASXDEC.exe". Premem espai i continuem escrivint ara els arguments de consola de l'aplicació. Només haurem de donar un únic argument que serà la ruta del fitxer XML d'inicialització de l'aplicació ("asxdec_ini.xml"). Si aquest fitxer ja es troba a la carpeta d'instal·lació de l'aplicació, podrem escriure directament el seu nom en comptes de la ruta completa tal i com es pot veure a la figura (**Fig. 5.11**). Finalment, premem *Enter* per iniciar l'execució.

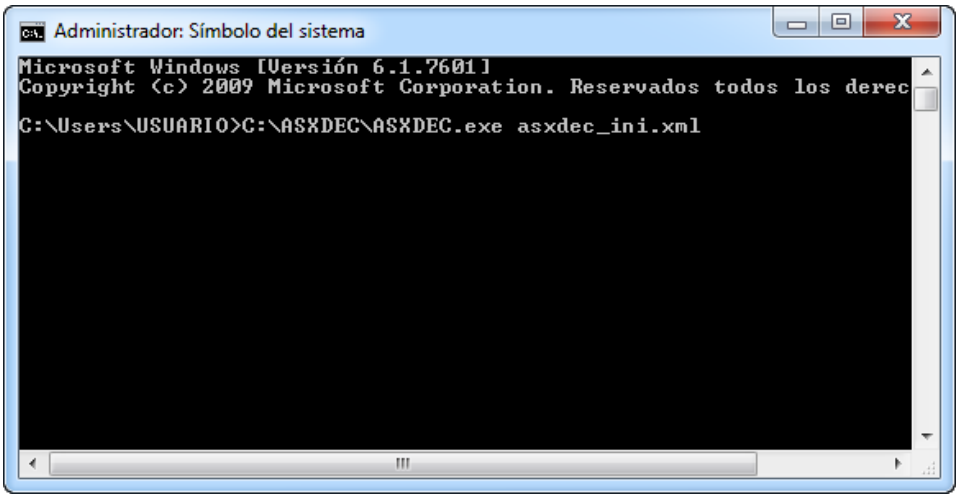

Fig. 5.11 Captura en que es crida a l'aplicació per mitjà de consola.

També es podrà executar l'aplicació des d'un *script* o una altre aplicació (automatitzat), així com d'executar simultàniament múltiples instàncies de la mateixa, que utilitzin diferents fitxers de configuració (p. e. aplica dos filtratges diferents a una mateixa gravació ASTERIX) i d'inicialització.

CONCLUSIONS

L'objectiu d'aquest treball ha estat el desenvolupament d'una aplicació única en el seu tipus, que permet la descodificació d'informació de vigilància de diferents categories ASTERIX (sensors), per al seu emmagatzematge en un repositori global de dades accessible a múltiples aplicacions client que l'utilitzin.

Com a punts innovadors del mòdul ASXDEC s'inclouen:

- La **escalabilitat** que ofereix la implementació de noves categories ASTERIX per mitjà de la introducció de fitxers XML, que elimina la necessitat d'integrar cada especificació de la categoria com a part del codi intern de l'aplicació, al mateix temps que es redueix el temps d'implementació.

- La **flexibilitat** a l'hora de permetre configurar a l'usuari l'aplicació en funció del seu entorn operatiu o propòsit del projecte. Crida simultània de múltiples instàncies de l'aplicació amb diferents configuracions que s'adaptin el repositori als diferents projectes i objectius en que s'utilitza. Potser a l'usuari no l'interessa una taula d'informació descodificada de la CAT21, que inclogui tots els *Data Items* (filtratge) per a una aplicació client en particular, però si que necessita una taula que inclogui tota la informació per un altre projecte.

Així mateix, també permet en relació a les aplicacions client:

- Unificar l'origen de les dades que utilitzen. Eficiència en termes de "data_Storage", ja que estem eliminant la redundància en les dades comunes que utilitzin múltiples aplicacions.

- Eliminar la necessitat que siguin les mateixes aplicacions les que hagin de realitzar la descodificació i el tractament de les dades que utilitzen (**disseny modular**).

Altres característiques a tenir en compte que ofereix l'aplicació són:

- La **robustesa** del sistema, comprovacions del format i contingut dels fitxers XML o detecció d'incoherències en la configuració de l'aplicació que resulten en l'enviament d'alertes, entre d'altres.

- L'**elevada potència de càlcul** que li permet descodificar gravacions amb un alt volum de missatges.

- L'**eficiència** en la programació dels mètodes i algoritmes interns de l'aplicació, amb l'objectiu de reduir el temps d'execució

- La interacció **amigable** i senzilla de l'usuari amb l'aplicació per mitjà de consola.

Finalment, com a línies de futur que donaran continuïtat al que s'ha desenvolupat en aquest treball tenim:

- La implementació de l'aplicació a curt termini a ENAIRE en règim de funcionament 24x7 (automatitzat), per començar a crear el repositori global de dades.

- La incorporació de noves categories ASTERIX per a la seva descodificació, introduint nous fitxers XML:
  - CAT1, CAT 2 i CAT 8: informació radar (PSR, SSR,...) excloent Mode-S.
  - CAT19 i CAT20: informació de sensors MLAT.
  - CAT34 i CAT48: informació radar incloent Mode-S.
  - CAT62: informació del SACTA.

- La implementació de nous formats de sortida com són CSV o per exemple KML (que s'utilitza per a la representació de trajectòries).

- El desenvolupament dels mòduls que venen a continuació del mòdul ASXDEC en el marc del projecte SSDA, i noves aplicacions client que utilitzin les dades (per anàlisis d'indicadors, de format, representació de trajectòries radar...).

BIBLIOGRAFIA

[1] ENAIRE. (2021). *Vigilancia*. https://www.enaire.es/servicios/cns/vigilancia

[2] EUROCONTROL, Specification for Surveillance Data Exchange - Part 1: All Purpose Structured Eurocontrol Surveillance Information Exchange (ASTERIX), 2.4 ed. 2016.

[3] EUROCONTROL, Specification for Surveillance Data Exchange - Part 1: All Purpose Structured Eurocontrol Surveillance Information Exchange (ASTERIX), 1.26 ed. 2000.

[4] EUROCONTROL, Standard Document for Surveillance Data Exchange Part 7: Category 010 Transmission of Monosensor Surface Movement Data, 1.1 ed. 2007.

[5] EUROCONTROL, Standard Document for Surveillance Data Exchange - Part 12: Category 021 ADS-B Messages, 2.1 ed. 2011.

[6] EUROCONTROL, ASTERIX Part 12 Category 021 Appendix A Coding rules for "Reserved Expansion Field", 1.1 ed. 2011.

[7] Microsoft .NET Documentation. (2021). How to deserialize an object using XmlSerializer. https://docs.microsoft.com/en-us/dotnet/standard/serialization/introducing-xml-serialization

[8] Microsoft .NET Documentation. (2021). Attributes That Control XML Serialization. https://docs.microsoft.com/en-us/dotnet/standard/serialization/attributes-that-control-xml-serialization

[9] Patel, K. (2017). Code Project. Convert XML to C# Object. https://www.codeproject.com/Articles/1163664/Convert-XML-to-Csharp-Object

[10] MariaDB.org. (2021). Download MariaDB Server - MariaDB.org https://mariadb.org/download/

[11] W3Schools. (2021). W3Schools. XML Elements vs. Attributes. https://www.w3schools.com/xml/xml_dtd_e

No s’ha inclòs, la nombrosa quantitat d’informació consultada via web relacionada amb qüestions de programació (codi) de l’aplicació.

ANNEXOS

Annex A. Pautes per a la definició de fitxers XML d'especificació de categories

Taula A-1. Informació dels elements i atributs en quant a la seva implementació al XML.

Nomenclatura XML	Element o atribut	Tipus de definició XML[3]	Valor per defecte[4]	Valors fixats[5]
Category	**Element**	Element		
id	Atribut	Atribut	N/A	NO
name	Atribut	Atribut	N/A	NO
ver	Atribut	Atribut	N/A	NO
DataItem	**Element**	Element		
id	Atribut	Atribut	N/A	NO
rule	Atribut	Atribut	N/A	NO
frn	Atribut	Atribut	N/A	NO
DataItemName	Atribut	Element	N/A	NO
DataItemDefinition	Atribut	Element	N/A	NO
DataItemStructure	**Element**	Element		
format	Atribut	Atribut	N/A	"fixed", "variable", "compound", "repetitive", "explícit"
length	Atribut	Atribut	N/A	NO
desc	Atribut	Atribut	N/A	NO
SubField	**Element**	Element		
name	Atribut	Atribut	N/A	NO
format	Atribut	Atribut	N/A	"fixed", "variable", "compound", "repetitive", "explícit"
length	Atribut	Atribut	N/A	NO
desc	Atribut	Atribut	N/A	NO
Field	**Element**	Element		
msb	Atribut	Atribut	N/A	NO
lsb	Atribut	Atribut	N/A	NO
bit	Atribut	Atribut	N/A	NO
encode	Atribut	Atribut	"uint"	"uint", "int", "hex", "octal", "sixbitschar"
FieldShortName	Atribut	Element	N/A	NO
FieldName	Atribut	Element	N/A	NO
FieldUnit	**Element**	Element		

[3] Tipus de definició de l'element o atribut en el format XML (veure [11])
[4] Tot i que l'atribut no es declari al XML, es considerà un valor per defecte
[5] Els atributs que tenen valors fixats, només se'ls hi pot assignar un valor llistat

scale	Atribut	Atribut	1	NO
min	Atribut	Atribut	N/A	NO
max	Atribut	Atribut	N/A	NO
unit	Atribut	Text de l'element	N/A	NO
Range	**Element**	Element		
min	Atribut	Atribut	N/A	NO
max	Atribut	Atribut	NO	NO

Taula A-2. Descripció dels elements i atributs definits al XML, obligatorietat i comentaris addicionals

Nomenclatura XML	Oblig.[6]	Descripció	Comentaris addicionals
Category			
id	SI	Identificació (3 dígits) de la categoria (p. e. 010, 021...)	
name	NO	Nom de la categoria (p.e cat. 010 => "Transmission of Monosensor Surface Movement Data").	
ver	NO	Versió/edició de l'especificació de la categoria. (p.e 1.1, 2.1,...)	
DataItem			
id	SI	Identificació (3 dígits) del *Data Item*. A la documentació apareix amb el nom del *Data Item*. (p. e. "Data Item I021/010", requadrat en vermell)	
rule	NO	Informació sobre la obligatorietat que el *Data Item* estigui present o no en qualsevol missatge (p. e. "This Item shall be present in every ASTERIX record").	
frn	SI	Valor del FRN (*Field Reference Number*) que es dona a la taula "Standard UAP (*User Application Profile*)" i estableix l'ordre d'aparició del *Data Item* al missatge	
DataItemName	SI	Nom del *Data Item* (p.e. "Data Source Identifier")	
DataItemDefinition	NO	Definició que es dona del *Data Item*, (p.e.	

[6] Obligatorietat de declarar l'element o atribut, en cas que el seu valor per defecte (si en té) no sigui l'idoni

		"Identification of the system from which the data are received.")	
DataItemStructure			
format	SI	Format del *Data Item*, d'entre els diferents *Standard Data Field Formats.* (p.e. *fixed)*	
length	SI[7]	Longitud en octets del *Data Item* només si té un format fixe.	Només és obligatori si el *Data Item* té un format fixe.
desc	NO	Descripció que es dona del format del *Data Item* (p.e. "Two-octet fixed length Data Item")	
SubField			
name	NO	Nom del *Subfield* (p.e. "Wind Speed"). El nom només apareix en els *Subfields* d'un *Data Item* amb format compost (*compound*).	
format	SI	Format del *Subfield*, d'entre els diferents *Standard Data Field Formats*. (p.e. *fixed*)	
length	SI[6]	Longitud en octets del *Subfield* només si té un format fixe (p. e. extensions de longitud fixe d'un *Data Item* amb format variable, *Subfields* amb format fixe d'un *Data Item* amb format compost,...).	Només és obligatori si el *Subfield* té un format fixe.
desc	NO	Descripció que es dona del format del *Subfield.*	
Field			
msb	SI[7]	Posició del MSB (*Most Significant Bit*) del *Field* dins del *Data Item*, extensió o *Subfield*.	Es poden declarar o bé el "msb" i el "lsb", o únicament el "bit" (en el cas de *Fields* de longitud un bit), mai els tres a la vegada ja que són redundants.
lsb	SI[7]	Posició del LSB (*Less Significant Bit*) del *Field* dins del *Data Item*, extensió o *Subfield*.	
bit	SI[7]	Posició del únic bit del *Field,* dins del *Data Item*, extensió o *Subfield*. Només en *Fields* amb longitud igual a un bit, on MSB = LSB.	
encode	SI	Codificació que cal aplicar al conjunt bits d'un *Field* per obtenir la informació que contenen (p. e. *uint*)	
FieldName	SI	Nom del *Field*, tal i com	

[7] Consulta "Comentaris addicionals" a la respectiva fila

		es dona a la documentació, preferentment no acrònims (p. e. “System Area Code”)	
FieldShortName	SI	Versió curta del nom del *Field*, sovint per mitjà d’acrònims (p. e. SAC). Es pot donar que coincideixi amb el nom del *Field*.	El seu valor s’utilitzarà per a la creació de les taules en base de dades (veure apartat 4.1.1).
FieldUnit			Aquest element només es declara en *Fields* amb informació numèrica, factor d’escala i unitat de mesura.
scale	SI	Factor d’escala o valor del LSB (*Less Significant Bit*), que cal multiplicar pel valor decimal del conjunt de bits d’un *Field*.	
min	NO[7]	Valor mínim q pot prendre el *Field.* (p.e. -180, -90, 0). A la documentació sovint s’inclou el rang de valors q pot prendre el *Field.*	Si només es defineix un dels dos límits (“min” o “max”), l’aplicació interpretarà l’altre límit seguint els criteris que s’expliquen a continuació de la taula.
max	NO[7]	Valor màxim q pot prendre el *Field.* (p.e. 180, 90, 255). A la documentació sovint s’inclou el rang de valors q pot prendre el *Field.*	
unit	SI	Unitat de mesura del valor del *Field* (p. e. m, m/s, º...).	
Range			Aquest element només es pot declara en *Fields* amb informació numèrica, sempre i quan no s’ha declarat ja un element “FieldUnit”
min	SI[7]	Valor mínim q pot prendre el *Field.* (p.e. -180, -90, 0).	S’ha de definir almenys algun dels dos límits (“min” o “max”). En aquest cas, l’aplicació interpretarà l’altre límit seguint els criteris que s’expliquen a continuació de la taula.
max	SI[7]	Valor màxim q pot prendre el *Field.* (p.e. 180, 90, 255).	

Criteri a l'hora de declarar els atributs "min" i "max"
Els atributs "min" i "max" pertanyen als elements "FieldUnit" i "Range":

- Si definim únicament l'atribut "min" (valor mínim): l'aplicació interpretarà que es tracta d'un rang de valors simètric, $-R/2 \leq x \leq R/2$. S'assignarà al valor de "max" la negació del valor de "min".
- Si definim únicament l'atribut "max" (valor màxim): l'aplicació interpretarà un rang de valors del tipus, $0 \leq x \leq R$. El valor assignat a "min" serà 0.
- Si definim ambdós atributs, tindrem lògicament un rang personalitzat entre els dos valors dels atributs, $valor_{"min"} \leq x \leq valor_{"max"}$.

Taula A-3. Restriccions a l'hora de definir els elements llistats a l'esquerra com a subelements dels elements llistats a dalt al fitxer XML

	Category	**DataItem**	**DataItemStructure**	**SubField**	**Field**
DataItem	Subelement				
DataItemStructure		Subelement			
SubField			Subelement	Subelement	
Field			Subelement	Subelement	
FieldUnit					Subelement
Range					Subelement

Annex B. Codi del software ASXDEC

Subprojecte ASXDECcore

AsterixDecoder.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.IO;
using loss_output;

namespace ASXDECcore
{
   public class CFileDecoder
   {
      #region Object Fields
      private Dictionary<ushort, CCategory> _categories;
      private List<ushort> _notGivenSpecCategories;
      private ASXDECconfig _config;
      private loss_output.CDataBase _db;
      private CAsterixFileTime _time;
      #endregion

      #region Constructor
      public CFileDecoder(string filepath1)
      {
         try
         {
            // Inicializar algunas variables
            _notGivenSpecCategories = new List<ushort>();
            _categories = new Dictionary<ushort, CCategory>();
            _time = new CAsterixFileTime();

            // Leo fichero ini de la app
            CFilePath.CheckFilePathExists(filepath1);
            string filename1 = CFilePath.GetFileName(filepath1).Item1; //
MOVED v.9
            ASXDECApp.InfoMsg(String.Format("Reading '{0}' ...", filename1)); //
int ccl = String.Format("INFO: Reading '{0}' ...", filename1).Length;
            string xmlInput;
            xmlInput = File.ReadAllText(filepath1);
            ASXDECini asxdecIni =
Serializer.Deserialize<ASXDECini>(xmlInput);
            string filepath2 = asxdecIni.ConfigurationFilePath;

            string[] specFilesPaths = asxdecIni.GetSpecificationFilesPaths; // v.9
```

```
            if (specFilesPaths.Length == 0)
                ASXDECApp.ExitApp(String.Format("NO ASTERIX Category
Specification file was given in '{0}'", filename1));

            string[] asterixFilePaths = asxdecIni.GetAsterixFilesPaths;
            if (asterixFilePaths.Length == 0)
                ASXDECApp.ExitApp(String.Format("NO ASTERIX files to decode
were given in '{0}'.", filename1));

            string[] asterixFilenames =
CFilePath.GetFileNames(asterixFilePaths); // v.9

            _time.CheckFileNamesFormat(asterixFilenames); // v.7

            ASXDECApp.SuccessfulMsg(String.Format("'{0}' paths checking
OK!\n", filename1));

            // Leo las especificaciones de cada categoria
            ASXDECApp.InfoMsg("ASTERIX categories specification.");
            for (int i = 0; i < specFilesPaths.Length; i++)
            {
                string filepath3 = specFilesPaths[i];
                xmlInput = File.ReadAllText(filepath3);
                ASXDECApp.InfoMsg(String.Format("Reading '{0}' ...",
CFilePath.GetFileName(filepath3).Item1));
                CCategory category =
Serializer.Deserialize<CCategory>(xmlInput);
                ushort cat = category.Num;
                category.sortDataItemsByFRN();
                _categories.Add(cat, category);
                string msg = String.Format("CAT '{0}' v{1} specification was
succesfully loaded.", category.Id, category.Version);
                if (i == specFilesPaths.Length - 1)
                {
                    msg = msg + "\n";
                }
                ASXDECApp.SuccessfulMsg(msg);
            }

            // Leo asterix_config.xml
            string filename2 = CFilePath.GetFileName(filepath2).Item1; //
MOVED v.9
            ASXDECApp.InfoMsg("ASXDEC configuration.");
            ASXDECApp.InfoMsg(String.Format("Reading '{0}' ...", filename2));
            xmlInput = File.ReadAllText(filepath2);
            this._config = Serializer.Deserialize<ASXDECconfig>(xmlInput);

            // Primary Settings
            CDataBaseParams databaseParams =
_config.Evaluation.DataBaseParams;
```

```
            _db = new CDataBase(databaseParams.DbHost,
databaseParams.DbSchema, databaseParams.DbUser,
databaseParams.DbPass);

            CTimeRef timeRef = _config.Evaluation.TimeRef;

            CFiltersSelection filterSelection = _config.Evaluation.FiltersSelection;

            // Filtering
            CFiltering filtering = _config.Filtering;

            CCategoriesFilter catFilter = null;
            if (filterSelection.EnableCatFiltering.Enable)
            {
                ASXDECApp.InfoMsg("Categories filtering is ENABLED.");

filtering.PreDecodingFilter.CategoriesFilter.CheckCategoriesHaveSpecification(
_categories);
                catFilter = filtering.PreDecodingFilter.CategoriesFilter;
            }

            CSensorsFilter sensorFilter =
filtering.PreDecodingFilter.SensorsFilter; // v.8
            if (filterSelection.EnableSensorFiltering.Enable)
            {
                ASXDECApp.InfoMsg("Sensor filtering is ENABLED.");
                sensorFilter.CreateDictForFasterAccess();
            }

            CDataItemsFilter dataItemsFilter = null;
            if (filterSelection.EnableDIFiltering.Enable)
            {
                ASXDECApp.InfoMsg("DataItems filtering is ENABLED.");
                dataItemsFilter = filtering.PreDecodingFilter.DataItemsFilter;
                if (dataItemsFilter.CheckDIHaveSpecification(_categories) == 1)
                {
                    ASXDECApp.ThrowWarning("DataItem filtering was DISABLED
because is empty.");
                    filterSelection.EnableDIFiltering.Enable = false;
                }
            }

            var keys = _categories.Keys.ToList();
            CCatAvailableSensors availableSensors;
            foreach (var key in keys)
            {
                CCategory cat = _categories[key];
                CDBTableParams dbTableParams =
databaseParams.GetDbTable(cat.Num);
                availableSensors = new CCatAvailableSensors();
```

```
            if (filterSelection.EnableCatFiltering.Enable)
            {
                if (!catFilter.CheckIfCategoryIsEnabled(cat))
                {
                    _categories.Remove(key);
                    continue;
                }
                if (dbTableParams == null)
                    ASXDECApp.ExitApp(String.Format("CAT '{0}' has been
ENABLED in the categories filter but NO DataBase table has been provided to
store its decodification.", cat.Id));
            }

            else if (dbTableParams == null)
            {
                ASXDECApp.ThrowWarning(String.Format("NO DataBase table
has been provided for CAT '{0}'. Therefore, its ASTERIX records will not be
decoded and stored.'", cat.Id));
                _categories.Remove(key);
                continue;
            }

            if (sensorFilter.ApplySensorFilter(cat, availableSensors) == 1)
            {
                ASXDECApp.ExitApp(String.Format("Could not found DataItem
'{0}' in CAT '{1}' specification necessary for identifying Data Source Sensor",
sensorFilter.DefaultDSIid, cat.Id));
                _categories.Remove(key);
                continue;
            }

            if (filterSelection.EnableDIFiltering.Enable)
            {
                CCategoryDIFilter catDIFilter =
dataItemsFilter.GetCategoryDIFilter(cat);
                if (catDIFilter != null)
                {
                    if (catDIFilter.IsEnabled && catDIFilter.ApplyDIFilter(cat) ==
1)
                    {
                        _categories.Remove(key);
                        continue;
                    }
                }
            }

            if (sensorFilter.GetSensorsGivenCat(cat,
filterSelection.EnableSensorFiltering.Enable, availableSensors) == 1)
            {
```

```
                    if (filterSelection.EnableCatFiltering.Enable)
                        ASXDECApp.ExitApp(String.Format("CAT '{0}' is ENABLED in
the categories filter but NO sensors are ENABLE to use it.", cat.Id));
                    else
                        _categories.Remove(key); continue;
                }

                cat.DecFieldsTable.AsterixFileTime = _time; // v.7
                cat.AddDatabaseTable(dbTableParams.CreateDBTable(_db));
                timeRef.ApplyTimeRef(cat); //v.7
                cat.OutputAndDecodingConfig();
            }

            if (_categories.Count == 0)
            {
                ASXDECApp.ExitApp("NO message from ANY category would be
decoded after applying the ASXDEC configuration.");
            }
            ASXDECApp.SuccessfulMsg("ASXDEC configuration was succesfully
loaded.\n");

            if (false) // v.9 only 'if' ADDED
            {
                ASXDECApp.AskQuestion("Start to decode?");
                Console.WriteLine("Press Enter to proceed...");
                Console.ReadLine();
            }

            for (int i = 0; i < asterixFilePaths.Length; i++) // v.7
            {
                _time.SetFileTimeFromFileName = asterixFilenames[i];
                _decodeAsterixFile(asterixFilePaths[i]);
            }
        }
        catch (FileNotFoundException ex)
        {
            ASXDECApp.ExitApp(ex.Message);
        }
        catch (Exception ex)
        {
            ASXDECApp.ExitApp(ex.Message);
        }
    }
    #endregion

    #region Methods
    private void _decodeAsterixFile(string filepath) // (ruta del fichero asterix a
decodificar)
    {
```

```
        ASXDECApp.InfoMsg(String.Format("'{0}' has started to be decoded...",
CFilePath.GetFileName(filepath).Item1));
        ushort cat = 0;
        try
        {
            /* Previously checked...
            if (!File.Exists(filepath))
            {
                throw new FileNotFoundException($"{filepath} file was not found");
            }
            */
            byte[] fileBytes = File.ReadAllBytes(filepath);
            ushort dBLength;
            int B_offset = 0;
            while (B_offset < fileBytes.Length)
            {
                cat = fileBytes[B_offset];
                dBLength = _getDataBlockLength(fileBytes, B_offset + 1);

                if (_categories.ContainsKey(cat)) // GRAN MEJORA DE
RENDIMIENTO EN COMPARACION CON EL USO DE LA EXCEPCION
KeyNotFoundException!!!
                {
                    byte[] blockBytes = new ArraySegment<byte>(fileBytes,
B_offset, dBLength).ToArray();
                    _categories[cat].decodeASXRecords(blockBytes);
                }
                else
                {
                    if (!_notGivenSpecCategories.Contains(cat)) //
(NotGivenSpecCategories.Exists(x => x == cat)). En caso que el diccionario
'Categories' no contenga la categoria del paquete para poder decodificarlo.
                        _notGivenSpecCategories.Add(cat);
                }
                B_offset += dBLength;
            }

            foreach (CCategory category in _categories.Values) // En el caso
que queden paquetes por insertar en la base de datos porque no se ha llegado
al threshold establecido.
            {
                category.DecFieldsTable.FinalInsertion();
            }

            // End of the decodification (some warnings)...
            CFiltersSelection filterSelection = _config.Evaluation.FiltersSelection;
            CFiltering filtering = _config.Filtering;
            CCategoriesFilter catFilter = null;

            if (filterSelection.EnableCatFiltering.Enable)
```

```
            {
                catFilter = filtering.PreDecodingFilter.CategoriesFilter;
            }
            Console.WriteLine();
            _notGivenSpecCategories.Sort(); // v.9
//_notGivenSpecCategories.OrderBy(x => x).ToList();
            foreach (ushort catNum in _notGivenSpecCategories)
            {
                if (filterSelection.EnableCatFiltering.Enable)
                {
                    if (catFilter.CheckIfCategoryIsEnabled(catNum))
                    {
                        ASXDECApp.ThrowWarning(String.Format("CAT '{0}' Asterix
records were found but could NOT be decoded because its specification was
NOT given or previous reasons.", catNum.ToString("000")));
                    }
                    continue;
                }
                ASXDECApp.InfoMsg(String.Format("CAT '{0}' Asterix records
were found but could NOT be decoded because its specification was NOT given
or previous reasons.", catNum.ToString("000")));
            }

            foreach (CCategory category in _categories.Values)
            {
                CCatAvailableSensors availableSensors =
category.AvailableSensors;

                if (availableSensors.GetNumNotFoundSensors() != 0)
                {
                    ASXDECApp.ThrowWarning(String.Format("The CAT '{0}'
messages from these sensors (SAC, SIC) were not decoded: " +
availableSensors.GetNotFoundSensorsString() +".", category.Id));
                }

            }
            ASXDECApp.SuccessfulMsg("Asterix recording has been succesfully
decoded!\n"); // Podria afegir amb # warning o # errors. La variable warning
podria anar dins de CDecFields
        }
        catch (FileNotFoundException ex)
        {
            ASXDECApp.ExitApp(ex.Message);
        }
    }

    private ushort _getDataBlockLength(byte[] fileBytes, int B_offset)
    {
        if (BitConverter.IsLittleEndian)
```

```
            return BitConverter.ToUInt16(new byte[] { fileBytes[B_offset + 1],
fileBytes[B_offset] }, 0);
        else
        {
            return BitConverter.ToUInt16(fileBytes, B_offset);
        }
    }
    #endregion
  }
}
```

CCategory.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml;
using System.Xml.Serialization;
using loss_output;

namespace ASXDECcore
{
    [XmlType(TypeName = "Category")]
    public class CCategory
    {
        #region Object Fields
        [XmlIgnore]
        private string _id;
        private ushort _num;
        [XmlIgnore]
        private string _name;
        [XmlIgnore]
        private string _version;
        [XmlIgnore]
        private List<CDataItem> _dataItems;
        [XmlIgnore]
        private List<Tuple<string, string>> _fieldNames;
        [XmlIgnore]
        private CDecFields _decFieldsTable;
        private CCatAvailableSensors _availableSensors;
        #endregion

        #region Properties
        [XmlAttribute(AttributeName = "id")]
        public string Id { get { return this._id; }
            set
            {
                this._id = value;
                this._num = Convert.ToUInt16(value);
            }
        }
        [XmlIgnore]
        public ushort Num { get { return this._num; } }
        [XmlAttribute(AttributeName = "name")]
        public string Name { get { return this._name; } set { this._name = value;
} }
        [XmlAttribute(AttributeName = "ver")]
        public string Version { get { return this._version; } set { this._version =
value; } }
```

```
        [XmlElement(ElementName = "DataItem")]
        public List<CDataItem> DataItems { get { return this._dataItems; } set {
this._dataItems = value; } }
        [XmlIgnore]
        public List<Tuple<string, string>> FieldsNames { get { return
this._fieldNames; } set { this._fieldNames = value; } }
        [XmlIgnore]
        public CDecFields DecFieldsTable { get { return this._decFieldsTable; }
set { this._decFieldsTable = value; } }
        [XmlIgnore]
        public CCatAvailableSensors AvailableSensors { get { return
this._availableSensors; } set { this._availableSensors = value; } }
        #endregion

        #region Constructor
        public CCategory()
        {
            this.FieldsNames = new List<Tuple<string, string>>();
            this.DecFieldsTable = new CDecFields();
        }

        #endregion

        #region Methods
        public void sortDataItemsByFRN() // Ordenar los DataItems en base a su
FRN
        {
            DataItems = DataItems.OrderBy(c => c.FRN).ToList();
        }

        public CDataItem FindDataItemById(string id)
        {
            return _dataItems.Find(x => x.Id == id);
        }

        public void decodeASXRecords(byte[] blockBytes)
        {
            int B_offset = 0;
            B_offset += 3; // Salto la categoria y la longitud del Record
            int sum = 0;
            while (B_offset < blockBytes.Length && sum == 0) // In case we are
following a DataBlock Structure
            {
                byte FX = 1;
                string FSPECbits = "";
                while (FX == 1)
                {
                    FX = CheckByteFX(blockBytes, B_offset);
                    FSPECbits += Convert.ToString(blockBytes[B_offset],
2).PadLeft(8, '0').Remove(7, 1); // Removing the last FX bit.
```

```
                B_offset += 1;
            }

            Func<char, int> char2int = (@char =>
Convert.ToInt32(@char.ToString(), 2));
            //Func < char, int> char2int = (@char => ((ushort)@char-48));
            List<int> FSPEC = FSPECbits.Select(char2int).ToList();
            int n = 0;
            int j = 0;
            while (j < FSPEC.Count && n < DataItems.Count) // No entiendo "n <
N"
            {
                CDataItem dataitem = DataItems[n];
                if (j + 1 == dataitem.FRN)
                {
                    if (FSPEC[j] == 1)
                    {
                        B_offset =
dataitem.DecodeDataItem(dataitem.DecodingMode, blockBytes, B_offset,
_decFieldsTable, _availableSensors);
                        if (B_offset == -2)
                        {
                            _decFieldsTable.InitializeVector();
                            goto noAddRecord;
                        }
                    }
                    else if (dataitem.DecodingMode == 0)
                        _decFieldsTable.SkipFields(dataitem.NumFields);
                    n++;
                }
                j++;
            }
            // DataItems.Last().FRN  == DataItems.Count Check!
            //int SpecOnes = FSPEC.Take(DataItems.Last().FRN).Sum();
            //int allOnes = FSPEC.Sum();
            //int noSpecOnes = allOnes - SpecOnes;
            while (j < FSPEC.Count) // If I'm missing Data Items that should be
decoded (FSPEC == 1) but I do not have their specification (e. g. SP or REF), I
will consider that the bytes that are remaining are not part of a new record in a
DataBlock
            {
                sum += FSPEC[j];
                j++;
            }
            _decFieldsTable.NewRecord();
        }
        noAddRecord:
        ;
    }
```

```
        public byte CheckByteFX(byte[] blockBytes, int B_offset) // Lo utilizo en el
FSPEC
        {
            return (byte)(blockBytes[B_offset] % 2);
        }

        public void AddDatabaseTable(CDBTable dataBaseTable)
        {
            _decFieldsTable.DBTable = dataBaseTable;
        }

        public void OutputAndDecodingConfig()
        {
            foreach (CDataItem di in DataItems)
            {
                if (di.DecodingMode != 1) // Avoid skip mode.
                {
                    CParentsInfo pInfo = new CParentsInfo(this._id, di.Name, di.Id);
                    di.LoopFields(pInfo, DecFieldsTable);
                }
            }
            DecFieldsTable.InitializeVector();
            DecFieldsTable.DBTable.CreateTable(DecFieldsTable.FieldsNames);
        }
        #endregion
    }

    public class CDecFields
    {
        #region Object Fields
        private ushort _categoryId; // Quizas no es necesario
        private uint _numDecPackages = 0;
        private List<string> _fieldsNames; // Database columns
        private ushort _numFields = 0;
        private List<Type> _fieldsDataType; // Quizas no es necesario
        private List<string[]> _decFieldsValues;
        private string[] _currentPackageDecFields;
        private ushort _defaultFieldCounter = 0;
        private ushort _fieldCounter = 0;
        private CDBTable _dbTable;
        private int _insertionCounter = 0;
        private const uint _insertionThreshold = 50; //100
        private CAsterixFileTime _asterixFileTime; // v.7
        #endregion

        #region Properties
        public ushort CategoryId { get { return this._categoryId; } set {
this._categoryId = value; } }
        public uint NumDecPackages { get { return this._numDecPackages; } set
{ this._numDecPackages = value; } }
```

```
        public List<string> FieldsNames { get { return this._fieldsNames; } set {
this._fieldsNames = value; } }
        public List<Type> FieldsDataType { get { return this._fieldsDataType; }
set { this._fieldsDataType = value; } }
        public List<string[]> DecFieldsValues { get { return
this._decFieldsValues; } set { this._decFieldsValues = value; } }
        public CDBTable DBTable { get { return this._dbTable; } set {
this._dbTable = value; } }
        public CAsterixFileTime AsterixFileTime { get { return
this._asterixFileTime; } set { this._asterixFileTime = value; } } // v.7

        #endregion

        #region Constructor
        public CDecFields()
        {
            this._fieldsNames = new List<string>();
            this._decFieldsValues = new List<string[]>();
        }
        #endregion

        #region Methods
        public void AddFieldName(string fieldName)
        {
            _fieldsNames.Add(fieldName);
            _numFields++;
        }


        public string GetFieldFromCurrentPos(sbyte offset) // v.7
        {
            return _currentPackageDecFields[_fieldCounter + offset];
        }                                       //

        public void IncreaseOneDefaultFieldCounter()
        {
            _defaultFieldCounter++;
            _fieldCounter = _defaultFieldCounter;
        }

        public void SetCurrentPackageTime(string utcTimeSeconds, ushort pos =
0) // v.7
        {
            TimeSpan currentPckgTime =
TimeSpan.FromSeconds(Convert.ToDouble(utcTimeSeconds,
System.Globalization.CultureInfo.InvariantCulture));
            if (_numDecPackages == 0)
            {
                AsterixFileTime.LastPckgTime = currentPckgTime;
            }
```

```
            this._currentPackageDecFields[pos] =
_asterixFileTime.Add(currentPckgTime, "yyMMdd HH:mm:ss.fff");
        }                                                    //

        public void SetField(string fieldValue, ushort pos = 0)
        {
            this._currentPackageDecFields[pos] = fieldValue;
        }

        public void AddFieldName(string fieldName, Type t)
        {
            AddFieldName(fieldName);
            _fieldsDataType.Add(t);
            _numFields++;
        }

        public void AddDecFieldValue(string decFieldValue)
        {
            try
            {
                if (_currentPackageDecFields[_fieldCounter] == null) // If it is the first
time I save data in a field slot
                    _currentPackageDecFields[_fieldCounter] = decFieldValue;
                else
                    _currentPackageDecFields[_fieldCounter] += "; " + decFieldValue;
// I case I am adding new data to a field of a repetitive type DI.
                _fieldCounter++;
            }
            catch (OverflowException ex) // In case that index (_fieldCounter) is
beyond vector size.
            {
                ASXDECApp.ThrowError(ex.Message);
            }
        }

        public void AddDecFieldValue(string decFieldValue, ushort
numSkippedFields)
        {
            _fieldCounter += numSkippedFields;
            try
            {
                _currentPackageDecFields[_fieldCounter] = decFieldValue;
                _fieldCounter += 1;
            }
            catch (OverflowException ex) // In case that index (_fieldCounter) is
beyond vector size.
            {
                ASXDECApp.ThrowError(ex.Message);
            }
```

```
        }

        public void InitializeVector() { this._currentPackageDecFields = new
string[_numFields]; }

        public void NewRecord()
        {
            _decFieldsValues.Add(_currentPackageDecFields);
            _fieldCounter = _defaultFieldCounter;
            _currentPackageDecFields = new string[_numFields];
            _numDecPackages++;
            _insertionCounter++;
            if (_insertionCounter == _insertionThreshold)
            {
                if (_dbTable != null) // In case a database table was not given for the
category
                {
                    _dbTable.InsertValues(_decFieldsValues);
                }
                _decFieldsValues.Clear();
                _insertionCounter = 0;
            }
        }

        public void FinalInsertion() { DBTable.InsertValues(_decFieldsValues); }

        public void JumpBackFields(ushort numJumpedBackFields)
        {
            _fieldCounter -= numJumpedBackFields;
        }

        public void SkipFields(ushort numSkippedFields)
        {
            _fieldCounter += numSkippedFields;
        }
        #endregion
    }

    public class CAsterixFileTime // v.7
    {
        #region Object Fields
        private DateTime _fileTime;
        private TimeSpan _lastPckgTime;
        private const uint _timeThreshold = 84000;
        private const string _outputFormat = "yyMMdd HH:mm:ss.fff";
        #endregion

        #region Properties
```

```
        public string SetFileTime { private get { return _fileTime.ToString(); } set
{ this._fileTime = DateTime.ParseExact(value, "yyMMdd",
System.Globalization.CultureInfo.InvariantCulture); } }
        public string SetFileTimeFromFileName { private get { return
_fileTime.ToString(); } set { SetFileTime = value.Split('-')[0]; } }
        public DateTime GetFileTime { get { return this._fileTime; } private set {
} }
        public TimeSpan LastPckgTime { get { return this._lastPckgTime; } set {
this._lastPckgTime = value; } }
        #endregion

        #region Methods

        public void CheckFileNamesFormat(string[] filenames)
        {
            foreach (string filename in filenames)
            {

                string[] parts = filename.Split('-');
                DateTime time;
                int num;
                if (!(parts.Length == 3 && DateTime.TryParseExact(parts[0],
"yyMMdd", System.Globalization.CultureInfo.InvariantCulture,
System.Globalization.DateTimeStyles.None, out time) &&
int.TryParse(parts[2].Remove(parts[2].Length - 4, 4), out num)))
                {
                    ASXDECApp.ExitApp(String.Format("'{0}' filename format is not
accepted. Remember to write it like this 'yyMMdd-xxxx-hhmmss'", filename));
                }
            }
        }

        public DateTime Add(TimeSpan s)
        {
            if ((_lastPckgTime - s).TotalSeconds > _timeThreshold)
            {
                _fileTime.AddDays(1);
            }
            return _fileTime.Add(s);
        }

        public string Add(TimeSpan s, string format = _outputFormat)
        {
            return Add(s).ToString(format);
        }
        #endregion      //
    }

    public class CCatAvailableSensors // v.8
    {
```

```
#region Object Fields
private Dictionary<Tuple<byte, byte>, bool> _sensors;
private Dictionary<Tuple<byte, byte>, bool> _notAllowedSensors;
#endregion;

#region Properties
#endregion

#region Constructor
public CCatAvailableSensors()
{
    this._sensors = new Dictionary<Tuple<byte, byte>, bool>();
    this._notAllowedSensors = new Dictionary<Tuple<byte, byte>, bool>();
}

public CCatAvailableSensors(Dictionary<Tuple<byte, byte>, bool> dict)
{
    this._sensors = dict;
}

public bool CheckIfSensorIsEnabled(byte sac, byte sic)
{
    Tuple<byte, byte> sensor = new Tuple<byte, byte>(sac, sic);
    if (_sensors.ContainsKey(sensor))
        return true;
    else
    {
        if (!_notAllowedSensors.ContainsKey(sensor))
            _notAllowedSensors.Add(sensor, false);
        return false;
    }
}

public List<Tuple<byte, byte>> GetNotFoundSensors()
{
    return _notAllowedSensors.Keys.ToList();
}

public string GetNotFoundSensorsString()
{
    string str = "";
    ushort i = 0;
    foreach (Tuple<byte, byte> sensor in _notAllowedSensors.Keys)
    {
        if (i > 0) { str += ","; }
        str += String.Format("({0},{1})", sensor.Item1, sensor.Item2);
        i++;
    }
    return str;
```

```
        }

        public int GetNumNotFoundSensors()
        {
            return _notAllowedSensors.Count;
        }
        #endregion

        #region Methods
        public void AddSensor(byte sac, byte sic, bool enable)
        {
            _sensors.Add(new Tuple<byte, byte>(sac, sic), enable);
        }

        public int GetNumAvailableSensors()
        {
            return this._sensors.Count;
        }
        #endregion
    }
}
```

CDataItem.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml;
using System.Xml.Linq;
using System.Xml.Serialization;


namespace ASXDECcore
{
    [XmlType(TypeName = "DataItem", IncludeInSchema = true)]
    public class CDataItem
    {
        #region Object Fields
        [XmlIgnore]
        private string _id; // | MANDATORY
        //private ushort _num;
        [XmlIgnore]
        private string _rule; // | OPTIONAL
        [XmlIgnore]
        private ushort _frn; // | MANDATORY
        [XmlIgnore]
        private string _name; // | MANDATORY
        [XmlIgnore]
        private string _definition; // | OPTIONAL
        [XmlIgnore]
        private CDataItemStructure _DIstructure; // | MANDATORY
        [XmlIgnore]
        private ushort _num_Fields;
        private byte _decodingMode = 0; // Default mode is NOT skipping DI and
so, decode it
        #endregion

        #region Properties
        [XmlAttribute(AttributeName = "id")]
        public string Id
        {
            get { return this._id; }
            set
            {
                ushort num;
                if (ushort.TryParse(value, out num))
                    this._id = num.ToString().PadLeft(3, '0');
                else
                    this._id = value;
                //this._num = Convert.ToUInt16(value);
```

```
            }
        }
        [XmlAttribute(AttributeName = "rule")]
        public string Rule { get { return this._rule; } set { this._rule = value; } }
        [XmlAttribute(AttributeName = "frn")]
        public ushort FRN { get { return this._frn; } set { this._frn = value; } }
        [XmlElement(ElementName = "DataItemName")]
        public string Name { get { return this._name; } set { this._name = value;
} }
        [XmlElement(ElementName = "DataItemDefinition")]
        public string Definition { get { return this._definition; } set {
this._definition = value; } }
        [XmlElement(ElementName = "DataItemStructure")]
        public CDataItemStructure DIStructure { get { return this._DIstructure; }
set { this._DIstructure = value; } }
        [XmlIgnore]
        public ushort NumFields { get { return this._num_Fields; } set {
this._num_Fields = value; } }
        [XmlIgnore]
        public byte DecodingMode { get { return this._decodingMode; } set {
this._decodingMode = value; } }
        #endregion

        #region Methods
        public int DecodeDataItem(byte mode, byte[] blockBytes, int B_offset,
CDecFields decFieldsTable, CCatAvailableSensors availablesensors)
        {
            return _DIstructure.DecodeDIformat(mode, blockBytes, B_offset,
decFieldsTable, availablesensors);
        }

        public ushort LoopFields(CParentsInfo pInfo, CDecFields decFieldsTable)
        {
            ushort n = _DIstructure.LoopFields(pInfo, decFieldsTable);
            this.NumFields = n;
            return n;
        }
        #endregion
    }
}
```

CDataItemStructureAndSubField.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Serialization;
using loss_output;

namespace ASXDECcore
{
    /// <summary>
    /// Used for DataItemStructure and SubField classes
    /// Contains the implementation of 'fixed', 'repetitive' and 'variable' DI format types
    /// </summary>
    public class BaseClass
    {
        #region Object Fields
        [XmlIgnore]
        protected short _length; // In octets (bytes). Used in DI with 'fixed', 'repetitive' (excluding the rep. factor (N) octet) and 'variable' (to indicate the length (usually one octet) of each fixed part/extent) formats to indicate their length.
        [XmlIgnore]
        protected string _dataItemFormat; // Indicates the DI format. It can be fixed, variable, compound, repetitive or explicit (SP).
        [XmlIgnore]
        protected List<CField> _fields;
        [XmlIgnore]
        protected List<CSubField> _subFields;
        [XmlIgnore]
        protected ushort _numFields;
        // protected CSensorsFilter _sensorFilter; // v.7 DELETED
        #endregion

        #region Properties
        [XmlAttribute(AttributeName = "length")]
        public short Length { get { return _length; } set { _length = value; } }
        [XmlAttribute(AttributeName = "format")]
        public string DataItemFormat
        {
            get
            {
                return this._dataItemFormat;
            }
            set
            {
```

```
            #region IsFormatWithinOptions // Quizas esto no es necesario si
utilizo el fichero .dtd
            bool check = Enum.IsDefined(typeof(DIFormats), value);
            try
            {
               if (!check)
               {
                  throw new ArgumentOutOfRangeException();
               }
            }
            catch (ArgumentOutOfRangeException)
            {
               string diFormats = "";
               foreach (object item in Enum.GetValues(typeof(DIFormats)))
               {
                  diFormats += item + ", ";
               }
               ASXDECApp.ThrowError(String.Format("The format '{0}' for the
DataItemStructure or Subfield does not correspond to any of the allowed format
types (" + diFormats + ")."", value));
            }
            #endregion
            this._dataItemFormat = value;
         }
      }
      [XmlElement("Field")]
      public List<CField> Fields { get { return this._fields; } set { this._fields =
value; } }
      [XmlElement("SubField")]
      public List<CSubField> SubFields { get { return this._subFields; } set {
this._subFields = value; } }
      [XmlIgnore]
      public ushort NumFields { get { return this._numFields; } set {
this._numFields = value; } }
      #endregion

      #region Methods
      protected int _decodeTime(byte[] blockBytes, int B_offset, CDecFields
decFieldsTable) // v.7
      {
         B_offset = _decodeFixedFormatYes(blockBytes, B_offset,
decFieldsTable);
         byte fieldPos = 1;
         string time_seconds =
decFieldsTable.GetFieldFromCurrentPos((sbyte)(-_numFields - 1 + fieldPos));
         decFieldsTable.SetCurrentPackageTime(time_seconds);
         return B_offset;
      }                                                              //
```

```
        protected int _decodeDSI(byte[] blockBytes, int B_offset, CDecFields
decFieldsTable, CCatAvailableSensors availableSensors)
        {
            if (availableSensors.CheckIfSensorIsEnabled(blockBytes[B_offset],
blockBytes[B_offset + 1])) // v.8
                return _decodeFixedFormatYes(blockBytes, B_offset,
decFieldsTable);
            else
                return -2;
        }

        protected int _decodeFixedFormatYes(byte[] blockBytes, int B_offset,
CDecFields decFieldsTable)
        {
            int i = 1;
            int j = 0;
            int N = Fields.Count;
            int total_bits_length = Length * 8;
            string bits = Convert.ToString(blockBytes[B_offset], 2).PadLeft(8, '0'); //
ATENCION: He añadido .PadLeft(8, '0')
            CField field;
            while (j < N)
            {
                field = Fields[j];
                if (field.Lsb > (Length - i) * 8)
                {
                    int bits_length = (field.Msb - field.Lsb) + 1;
                    string input = bits.Substring(total_bits_length - field.Msb,
bits_length);
                    field.decodeField(input, decFieldsTable);
                    j += 1;
                }
                else if (i <= Length) // ATENCION: He cambiado "<" por "<="
                {
                    i += 1;
                    B_offset += 1;
                    bits += Convert.ToString(blockBytes[B_offset], 2).PadLeft(8, '0');
                }
            }
            return B_offset + 1;
        }

        protected int _decodeFixedFormat(byte mode, byte[] blockBytes, int
B_offset, CDecFields decFieldsTable, CCatAvailableSensors availableSensors)
        {
            switch (mode)
            {
                case 3: // Time Mode // v.7
                    return _decodeTime(blockBytes, B_offset, decFieldsTable); // v.7
                case 2: // DSI Mode
```

```
                return _decodeDSI(blockBytes, B_offset, decFieldsTable,
availableSensors);
            case 1: // Skipping Mode
                return B_offset + Length;
            default: // Decoding Mode
                //(int a, int b) = decodeFixedDataItemYes(blockBytes, B_offset);
                return _decodeFixedFormatYes(blockBytes, B_offset,
decFieldsTable);
        }
    }

    protected int _decodeRepetitiveFormat(byte mode, byte[] blockBytes, int
B_offset, CDecFields decFieldsTable)
    {
        //byte maxNumReps = 4;
        byte N_rep = blockBytes[B_offset]; // CHANGED v.9
        //byte N = Math.Min(N_rep, maxNumReps);
        B_offset += 1;
        switch (mode)
        {
            case 1: // Skipping Mode
                return B_offset + N_rep * _length;
            default: // Decoding Mode
                {
                    int n = 0;
                    while (n < N_rep)
                    {
                        if (n > 0 && (mode == 0)) {
decFieldsTable.JumpBackFields(_numFields); } // Because I will have to add
new repeated data to fields already decoded
                        B_offset = _decodeFixedFormatYes(blockBytes, B_offset,
decFieldsTable);
                        n++;
                    }
                    return B_offset;
                }
        }
    }
    /// <summary>
    /// To be developed.
    /// </summary>
    /// <param name="mode"></param>
    /// <param name="blockBytes"></param>
    /// <param name="B_offset"></param>
    /// <param name="decFieldsTable"></param>
    /// <returns></returns>
    protected int _decodeExplicitFormat(byte mode, byte[] blockBytes, int
B_offset, CDecFields decFieldsTable)
    {
```

```
        int lengthInd = blockBytes[B_offset]; // The length indicator gives the
explicit length including itselfs (one-octet)
        switch (mode)
        {
            case 1: // Skipping Mode
                return B_offset + lengthInd;
            default: // Decoding Mode
                {
                    B_offset += 1;
                    int i = 0;
                    string bits = "";
                    while (i < lengthInd)
                    {
                        bits += Convert.ToString(blockBytes[B_offset], 2).PadLeft(8,
'0');
                        B_offset++;
                        i++;
                    }
                    CField f = new CField();
                    f.decodeField(bits, decFieldsTable);
                    return B_offset;
                }
        }
    }

    protected int _decodeVariableFormat(byte mode, byte[] blockBytes, int
B_offset, CDecFields decFieldsTable)
    {
        int n = 0;
        int N = SubFields.Count;
        int FX = 1;
        while (n < N && FX == 1)
        {
            CSubField Subfield = SubFields[n];
            FX = Subfield.CheckFX(blockBytes, B_offset);
            B_offset = Subfield._decodeFixedFormat(mode, blockBytes, B_offset,
decFieldsTable, null); // Aunque el modo sea '1', tengo que decodificar el DI
para saber la longitud en octetos del DI en función de las extensiones.
            n += 1;
        }
        while (n < N && (mode == 0)) // In case that not all the subfields
appears on the variable DI, we have to consider the fields of those which are
not appearing as nulls;
        {
            decFieldsTable.SkipFields(SubFields[n].NumFields);
            n++;
        }
        return B_offset;
    }
```

```
        public ushort LoopFields(CParentsInfo pInfo, CDecFields decFieldsTable)
        {
            ushort n = 0;
            if (Fields.Count == 0) // In case of 'variable' or 'compound' DI format
            {
                foreach (CSubField subF in SubFields)
                {
                    n += subF.LoopFields(pInfo, decFieldsTable);
                }
            }
            else // In case of 'fixed' or 'repetitive' DI format
            {
                if (decFieldsTable != null)
                {
                    foreach (CField f in Fields)
                    {
                        f.PInfo = pInfo;
                        if (_dataItemFormat == DIFormats.repetitive.ToString())
                        {

decFieldsTable.AddFieldName($"REP_di_{pInfo.DataItemId}_{f.ShortName}");
// CHANGED v.9
                        }
                        else
                        {

decFieldsTable.AddFieldName($"di_{pInfo.DataItemId}_{f.ShortName}");
                        }

decFieldsTable.DBTable.AddColumnDataType(f.GetFieldEncode(),
_dataItemFormat, f.GetBitsLength());
                    }
                }
                else
                {
                    foreach (CField f in Fields)
                    {
                        f.PInfo = pInfo;
                    }
                }

                n = (ushort)Fields.Count;
            }
            _numFields = n;
            return n;
        }
        #endregion
    }

    /// <summary>
```

```
    ///
    /// Adds the implementation of 'compound' DI format type
    /// </summary>
    [XmlType(TypeName = "DataItemStructure")]
    public class CDataItemStructure : BaseClass
    {
        #region Object Fields
        [XmlIgnore]
        private string _description; // | OPTIONAL
        // private short _length; // In case of 'fixed', 'repetitive' DI format
        // private string _dataItemFormat; It can be fixed, variable, compound,
repetitive or explicit (SP).
        // private List<CField> _fields; // In case of 'fixed' or 'repetitive' DI format
        // private List<CSubField> _subFields; // In case of 'variable' or 'compound'
DI format
        #endregion

        #region Properties
        [XmlAttribute(AttributeName = "desc")]
        public string Description { get { return this._description; } set {
this._description = value; } }
        #endregion

        #region Constructor // Empty
        public CDataItemStructure()
        {

        }
        #endregion

        #region Methods
        public int DecodeDIformat(byte mode, byte[] blockBytes, int B_offset,
CDecFields decFieldsTable, CCatAvailableSensors availableSensors)
        {
            switch (_dataItemFormat)
            {
                case "fixed":
                    return _decodeFixedFormat(mode, blockBytes, B_offset,
decFieldsTable, availableSensors);
                case "repetitive":
                    return _decodeRepetitiveFormat(mode, blockBytes, B_offset,
decFieldsTable);
                case "variable":
                    return _decodeVariableFormat(mode, blockBytes, B_offset,
decFieldsTable);
                case "compound":
                    return _decodeCompoundDataItem(mode, blockBytes, B_offset,
decFieldsTable);
                default:
```

```
                Console.WriteLine("Unknown DI format type '{0}'",
_dataItemFormat);
                Console.ReadLine();
                return -1;
            }
        }

        public byte CheckByteFX(byte[] blockBytes, int B_offset) // Used in
compound type to check if FSPEC continues
        {
            return (byte)(blockBytes[B_offset] % 2);
        }

        private int _decodeCompoundDataItem(byte mode, byte[] blockBytes, int
B_offset, CDecFields decFieldsTable)
        {
            byte FX = 1;
            string FSPECbits = "";
            while (FX == 1)
            {
                FX = CheckByteFX(blockBytes, B_offset);
                FSPECbits += Convert.ToString(blockBytes[B_offset],
2).PadLeft(8,'0').Remove(7, 1); // Removing the last FX bit.
                B_offset += 1;
            }

            Func<char, int> char2int = (@char =>
Convert.ToInt32(@char.ToString(), 2));
            //Func < char, int> char2int = (@char => ((ushort)@char-48));
            List<int> FSPEC = FSPECbits.Select(char2int).ToList();
            int n = 0;
            while (n < FSPEC.Count && n < SubFields.Count) // No entiendo "n <
N"
            {
                CSubField subfield = SubFields[n];
                if (FSPEC[n] == 1)
                    B_offset = subfield.DecodeSubField(mode, blockBytes, B_offset,
decFieldsTable);
                else if (mode == 0)
                    decFieldsTable.SkipFields(subfield.NumFields);
                n++;
            }
            while (n < SubFields.Count && (mode == 0)) // In case that not all the
subfields appears on compound DI "FSPEC", we have to consider the fields of
those which are not appearing as nulls;
            {
                decFieldsTable.SkipFields(SubFields[n].NumFields);
                n++;
            }
            return B_offset;
```

```
        }
        #endregion
    }
    [XmlType(TypeName = "SubField")]
    public class CSubField : BaseClass
    {
        #region Object Fields
        [XmlIgnore]
        private string _name; // | OPTIONAL
        // private short _length; // In case of 'fixed', 'repetitive' DI format
        // private string _dataItemFormat; It can be fixed, variable, (NOT
compound), repetitive or explicit (SP).
        // private List<CField> _fields; // In case of 'fixed' or 'repetitive' DI format
        // private List<CSubField> _subFields; // In case of 'variable' (NOT
'compound') DI format
        #endregion

        #region Properties
        [XmlAttribute(AttributeName = "name")]
        public string Name { get { return this._name; } set { this._name = value;
} }
        #endregion

        #region Methods
        public int DecodeSubField(byte mode, byte[] blockBytes, int B_offset,
CDecFields decFieldsTable) // Falta añadir el formato 'explicit'
        {
            switch (_dataItemFormat)
            {
                case "fixed":
                    return _decodeFixedFormat(mode, blockBytes, B_offset,
decFieldsTable, null);
                case "repetitive":
                    return _decodeRepetitiveFormat(mode, blockBytes, B_offset,
decFieldsTable);
                case "variable":
                    return _decodeVariableFormat(mode, blockBytes, B_offset,
decFieldsTable);
                default:
                    Console.WriteLine("Unknown DI format type '{0}'",
_dataItemFormat);
                    Console.ReadLine();
                    return -1;
            }
        }

        public int CheckFX(byte[] blockBytes, int B_offset) // Used in the subfields
of a 'variable' format DI
        {
            byte last_byte = blockBytes[B_offset + Length - 1];
```

```
            return (last_byte % 2);
        }
        #endregion
    }
}
```

CField.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Serialization;
using loss_output;

namespace ASXDECcore
{
    /// <summary>
    /// Class to decode a single field
    /// </summary>
    [XmlType(TypeName = "Field")] // The name for the object in the XML file
    public class CField
    {
        #region Object Fields
        [XmlIgnore]
        private short _msb; // Most significant bit | MANDATORY
        [XmlIgnore]
        private short _lsb; // Less significant bit | MANDATORY
        [XmlIgnore]
        private string _encode; // Encode for decoding the field. The options are uint (base10 positive), int(base10 complement two), hex (base16), octal (base8) and sixbitschar (similar to a base64). | IF NOT GIVEN, DEFAULT 'uint'
        [XmlIgnore]
        private string _name; // The complete name of the field | MANDATORY
        [XmlIgnore]
        private string _shortName; // A shorter name for the field | MANDATORY
        [XmlIgnore]
        private CUnit _unit; // Object that contains the unit, the scale and the limits (range of allowed values) for the field. | ONLY APPLICABLE FOR numeric encode types
        [XmlIgnore]
        private CRange _range; // Used in case a field does not have units (CUnit) but needs to have limits (range of allowed values) | OPTIONAL, (NOT USED IF 'CUnit _unit' IS DEFINED)
        [XmlIgnore]
        private string _description; // A description of the field | OPTIONAL
        [XmlIgnore]
        private CParentsInfo _pInfo;
        #endregion

        #region Properties
        [XmlAttribute(AttributeName = "msb")]
        public short Msb { get { return this._msb; } set { this._msb = value; } }
        [XmlAttribute(AttributeName = "lsb")]
        public short Lsb { get { return this._lsb; } set { this._lsb = value; } }
```

```
        [XmlAttribute(AttributeName ="bit")] // v.9
        public short Bit { get { return this._msb; } set { this._msb = value;
this._lsb = value; } }
        [XmlAttribute(AttributeName = "encode")]
        public string Encode
        {
            get
            {
                return this._encode;
            }
            set
            {
                #region IsEncodeWithinOptions // Quizas esto no es necesario si
utilizo el fichero .dtd
                bool check = Enum.IsDefined(typeof(EncodingOptions), value);
                try
                {
                    if (!check)
                    {
                        throw new ArgumentOutOfRangeException();
                    }
                }
                catch (ArgumentOutOfRangeException)
                {
                    string outputFormats = "";
                    foreach (object item in
Enum.GetValues(typeof(EncodingOptions)))
                    {
                        outputFormats += item + ", ";
                    }
                    ASXDECApp.ThrowError(String.Format("The encode '{0}' for field
'{1}' does not correspond to any of the allowed encoding options (" +
outputFormats + ")., value, this._name)); // El nombre del field aparecera en
blanco porque aun no he asignado su valor. La solucion es definir el 'encode'
como un elemento, y no como un atributo.
                }
                #endregion
                this._encode = value;
            }
        }
        [XmlElement(ElementName = "FieldName")]
        public string Name { get { return this._name; } set { this._name = value;
} }
        [XmlElement(ElementName = "FieldShortName")]
        public string ShortName { get { return this._shortName; } set {
this._shortName = value; } }
        [XmlElement(ElementName = "FieldUnit")]
        public CUnit Unit { get { return this._unit; } set { this._unit = value; } }
        [XmlElement(ElementName = "Range")]
```

```
        public CRange Range { get { return this._range; } set { this._range =
value; } }
        [XmlElement(ElementName = "desc")]
        public string Description { get { return this._description; } set {
this._description = value; } }
        [XmlIgnore]
        public CParentsInfo PInfo { get { return this._pInfo; } set { this._pInfo =
value; } }
        #endregion

        #region Constructor
        public CField()
        {
            this._encode = EncodingOptions.@uint.ToString(); // Default encoding
        }
        #endregion

        #region Methods
        public string decodeField(string input, CDecFields decFieldsTable)
        {
            string output = null;
            double num;
            try
            {
                switch (_encode)
                {
                    case "uint": // Positive number
                        if (_unit == null) // Always integer (no scale)
                        {
                            num = Convert.ToUInt32(input, 2);
                            if (_range != null)
                                _range.IsWithinMargins(num);
                            output = num.ToString();
                        }
                        else // Could be a double (decimals)
                        {
                            num = _unit.ApplyScale(Convert.ToUInt32(input, 2));
                            //_unit.IsWithinMargins(num); // v.7 isn't necessary, is already
done in ApplyScale()
                            output =
num.ToString(System.Globalization.CultureInfo.InvariantCulture); //
System.Globalization.CultureInfo.InvariantCulture (format) is used to keep dot
as the decimal separator for the conversion to String.

                        }
                        break;
                    case "int": // Negative number
                        if (_unit == null) // Always integer (no scale)
                        {
                            if (input.StartsWith("0"))
```

```
                        num = Convert.ToInt32(input, 2);
                    else // Two's complement
                        num = Convert.ToInt32(input.PadLeft(32, '1'), 2);
                    if (_range != null)
                        _range.IsWithinMargins(num);
                    output = num.ToString();
                }
                else // Could be a double (decimals)
                {
                    if (input.StartsWith("0"))
                        num = _unit.ApplyScale(Convert.ToInt32(input, 2));
                    else // Two's complement
                        num = _unit.ApplyScale(Convert.ToInt32(input.PadLeft(32,
'1'), 2));
                    //_unit.IsWithinMargins(num); v.7 isn't necessary, is already
done in ApplyScale(
                    output =
num.ToString(System.Globalization.CultureInfo.InvariantCulture);
                }
                break;
            case "hex":
                // output = Convert.ToString(Convert.ToUInt32(input, 2),16); //
v.4
                output = string.Format("{0:X}", Convert.ToUInt64(input, 2)); // v.5
Need a larger integer to store the values from input.
                break;
            case "octal":
                output = Convert.ToString(Convert.ToUInt32(input, 2), 8);
                break;
            case "sixbitschar":
                output = _decodeIcaoabc(input);
                break;
            default: // Already checked in the property accesor (set) of
'_outputFormat' field.
                ASXDECApp.ThrowError(String.Format("The encode '{0}' for
field '{1}' does not correspond to any of the allowed encoding options.",
_encode, this._name));
                output = "error";
                break;
        }
        decFieldsTable.AddDecFieldValue(output);
        return output;
    }
    catch (ArgumentOutOfRangeException ex)
    {
        ASXDECApp.ThrowWarning(String.Format(PInfo.ParentsInfo +
"\n'{0}' is outside margins. " + ex.Message, _name));
        decFieldsTable.AddDecFieldValue(output); // v.5 It wasn't storing the
values of the fields with this exception. Consequently, the fieldCounter didn't
advance +1.
```

```
            return output;
            //'{0}' is outside margins. " + ex.Message,this._name);
        }
        catch (Exception ex)
        {
            ASXDECApp.ThrowError(String.Format(PInfo.ParentsInfo + "\nWhen trying to apply encode '{0}' to '{1}'. " + ex.Message, _encode, _name));
            decFieldsTable.AddDecFieldValue("decError"); // v.5 It wasn't storing the values of the fields with this exception (at least as 'Error'). Consequently, the fieldCounter didn't advance +1;
            return "decError";
        }

    }

    private string _decodeIcaoabc(string input)
    {
        UsefulFunctions.ConvertIcao6bits2char func = new UsefulFunctions.ConvertIcao6bits2char();
        int i = 0;
        string charBits;
        string decodedChars = "";
        while (i < input.Length)
        {
            charBits = input.Substring(i, 6);
            decodedChars += func.Icao6bits2char(charBits);
            i += 6;
        }
        return decodedChars;
    }

    public Tuple<string,bool> GetFieldEncode()
    {
        if (_unit != null) // For uint and int
        {
            return new Tuple<string, bool>(_encode, _unit.IsScaleDouble);
        }
        else
        {
            return new Tuple<string, bool>(_encode, false);
        }
    }

    public ushort GetBitsLength()
    {
        return (ushort)(this._msb - this._lsb + 1);
    }
    #endregion
}
```

```
    [XmlType(TypeName = "Range")]
    public class CRange // Temporary public
    {
        #region Object Fields
        [XmlIgnore]
        private double? _minValue; // Minimum numerical value the field could
have | REQUIRED AT LEAST ONE OF THE TWO FIELDS
        [XmlIgnore]
        private double? _maxValue; // Maximum numerical value the field could
have | REQUIRED AT LEAST ONE OF THE TWO FIELDS
        [XmlIgnore]
        private bool _isLimited = false; // It's value will be true if at least one of the
two previous Object Fields is defined.
        #endregion

        #region Properties
        [XmlAttribute(AttributeName = "min")]
        public string MinValue
        {
            get
            {
                return Convert.ToString(this._minValue,
System.Globalization.CultureInfo.InvariantCulture);
            }
            set
            {
                this._minValue = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
                _isLimited = true;
            }
        }
        [XmlAttribute(AttributeName = "max")]
        public string MaxValue
        {
            get
            {
                return Convert.ToString(this._maxValue,
System.Globalization.CultureInfo.InvariantCulture);
            }
            set
            {
                this._maxValue = Convert.ToDouble(value,
System.Globalization.CultureInfo.InvariantCulture);
                _isLimited = true;
            }
        }
        #endregion

        #region Methods
        public bool IsWithinMargins(double num)
```

```
        {
            bool isWithingMargins;
            if (_isLimited)
            {
                if (_minValue == null) // In case I only provide the upper limit of the
range => [0, maxValue]
                    isWithingMargins = ((num >= 0) && (num <= _maxValue));
                else if (_maxValue == null) // In case I only provide the lower limit
(considering is negative) of the range => [minValue(<0),-minValue(>0)]
                    isWithingMargins = ((num >= _minValue) && (num <= -
_minValue));
                else // In case I provide both limits of the range => [minValue,
maxValue]
                    isWithingMargins = ((num >= _minValue) && (num <=
_maxValue));
                if (!isWithingMargins)
                {
                    throw new ArgumentOutOfRangeException($"Field value '{num}'
should be within {_minValue} and {_maxValue}.", new Exception());
                    //throw new
ArgumentOutOfRangeException(fieldName,num,$"Argument should be within
{_minValue} and {_maxValue}");
                }
                return isWithingMargins;
            }
            else
            {
                return true;
            }
        }
        #endregion
    }

    [XmlType(TypeName = "FieldUnit")]
    public class CUnit : CRange // Temporary public
    {
        #region Object Fields
        [XmlIgnore]
        private double _scale = 1; // The decimal (base10) value of the bits is
multiplyied by this factor | IF NOT PRESENT, DEFAULT 1
        [XmlIgnore]
        private bool _isScaleDouble; // Necessary for chosing database data type.
Checks if '_scale' has decimals.
        [XmlIgnore]
        private string _unit; // Defines the unit of the field (m, cm,...) | OPTIONAL
(but should be)
        #endregion

        #region Properties
        [XmlAttribute(AttributeName = "scale")]
```

```
        public double Scale
        {
            get
            {
                return this._scale;
            }
            set
            {
                _isScaleDouble = IsDouble(value);
                this._scale = value;
            }
        }

        [XmlIgnore]
        public bool IsScaleDouble { get { return this._isScaleDouble; } private
set { this._isScaleDouble = value; } }
        [XmlText(DataType = "string")]
        public string Unit { get { return this._unit; } set { this._unit = value; } }
        #endregion

        #region Constructor
        public CUnit() // Default values for Fields
        {
            this._scale = 1;
        }
        #endregion

        #region Methods
        static private bool IsDouble(double num)
        {
            return (num != Convert.ToInt32(num));
        }

        public double ApplyScale(uint num)
        {
            double res = num * Scale;
            IsWithinMargins(res);
            return res;
        }

        public double ApplyScale(int num)
        {
            double res = num * Scale;
            IsWithinMargins(res);
            return res;
        }
        #endregion
    }

    public class CParentsInfo
```

```
    {
        #region Object Fields
        private string _categoryId; // | MANDATORY
        private string _dataItemName; // | MANDATORY
        private string _dataItemId; // | MANDATORY
        private string _parentsInfo => $"Data Item I{_categoryId}/{_dataItemId},
{_dataItemName}";
        #endregion

        #region Properties
        public string DataItemId { get { return this._dataItemId; } }
        public string ParentsInfo { get { return this._parentsInfo; } }
        #endregion

        #region Constructor
        public CParentsInfo(string categoryId, string dataItemName, string
dataItemId)
        {
            this._categoryId = categoryId;
            this._dataItemName = dataItemName;
            this._dataItemId = dataItemId;
        }
        #endregion
    }
}
```

Serializer.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Xml;
using System.Xml.Serialization;

namespace ASXDECcore
{
   public static class Serializer
   {
      /// <summary>
      /// populate a class with xml data
      /// </summary>
      /// <typeparam name="T">Object Type</typeparam>
      /// <param name="input">xml data</param>
      /// <returns>Object Type</returns>
      static public T Deserialize<T>(string input) where T : class
      {
         System.Xml.Serialization.XmlSerializer ser = new
System.Xml.Serialization.XmlSerializer(typeof(T));
         StringReader x = new StringReader(input);
         using (StringReader sr = new StringReader(input))
         {
            return (T)ser.Deserialize(sr);
         }
      }
      /// <summary>
      /// convert object to xml string
      /// </summary>
      /// <typeparam name="T"></typeparam>
      /// <param name="ObjectToSerialize"></param>
      /// <returns></returns>
      static public string Serialize<T>(T ObjectToSerialize)
      {
         XmlSerializer xmlSerializer = new
XmlSerializer(ObjectToSerialize.GetType());

         using (StringWriter textWriter = new StringWriter())
         {
            xmlSerializer.Serialize(textWriter, ObjectToSerialize);
            return textWriter.ToString();
         }
      }
   }
}
```

UsefulFunctions.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ASXDECcore
{
   static public class UsefulFunctions
   {
      public class ConvertIcao6bits2char
      {
         private string[,] Matrix_IA_5;

         public ConvertIcao6bits2char()
         {
            Matrix_IA_5 = new string[,] { { "", "P", " ", "0" }, { "A", "Q", "", "1"
},{"B","R","","2"},
         {"C","S","","3"},{"D","T","","4"},
{"E","U","","5"},{"F","V","","6"},{"G","W","","7"},{"H","X","","8"},
         {"I","Y","","9"},{"J","Z","",""}
,{"K","","","",""},{"L","","","",""},{"M","","","",""},{"N","","","",""},{"O","","","",""}};
         }

         public string Icao6bits2char(string sixCharBits)
         {
            int decBits4_1 = Convert.ToInt32(sixCharBits.Substring(2, 4), 2);
            int decBits5_6 = Convert.ToInt32(sixCharBits.Substring(0, 2), 2);
            return Matrix_IA_5[decBits4_1, decBits5_6];
         }
      }
   }
}
```

Subprojecte ioss_input

CAsxdec_ini.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Serialization;
using System.IO;
using loss_output;
using ASXDECcore;

namespace ioss_input
{
    [XmlType(TypeName = "ASXDECini")]
    public class ASXDECini
    {
        #region Object Fields
        private string _configurationFilePath;
        //private string _specificationFilesPath; // DELETED v.9
        //private Func<char, bool> func = x => !(x.Equals('\n') || x.Equals('\t') ||
x.Equals(' '));
        private string[] _specificationFilesPaths;
        //private string _asterixFilesPath; // DELETED v.9
        private string[] _asterixFilesPaths;
        private Func<char, bool> func01 = x => !x.Equals('\t');
        private Func<string, bool> func02 = x => !(String.IsNullOrWhiteSpace(x) ||
x.Equals(""));
        #endregion

        #region Properties
        [XmlElement(ElementName = "ASXDECconfigFilePath")]
        public string ConfigurationFilePath
        {
            get { return this._configurationFilePath; }
            set
            {
                Func<char, bool> func = x => !(x.Equals('\n') || x.Equals('\t') ||
x.Equals(' '));
                this._configurationFilePath = String.Join("", value.Where(func));
                CFilePath.CheckFilePathExists(_configurationFilePath);
            }
        }
        [XmlElement(ElementName = "ASTERIXspecificationFilesPaths")]
        public string SetSpecificationFilesPaths
        {
```

```
            //get { return this._specificationFilesPath; } // CHANGED v.9
            get { return String.Join(",", this._specificationFilesPaths); } // v.9
            set
            {
                //this._specificationFilesPath = value; // DELETED v.9
                this._specificationFilesPaths =
CFilePath.ModifyPathIfNecessary(value.Trim('\n', '\t', ' ').Split('\n').Select(x =>
x.Trim('\t', ' ')).ToArray(), @".\specification-folder\"); // MOD v.9
                CFilePath.CheckFilePathsExist(this._specificationFilesPaths);
            }
        }
        [XmlElement(ElementName = "ASTERIXtodecodeFilePaths")]
        public string SetAsterixFilesPaths
        {
            //get { return this._asterixFilesPath; } // CHANGED v.9
            get { return String.Join(",", this._asterixFilesPaths); } // v.9
            set
            {
                //this._asterixFilesPath = value; // DELETED v.9
                this._asterixFilesPaths = value.Trim('\n', '\t', ' ').Split('\n').Select(x =>
x.Trim('\t', ' ')).ToArray(); // MOD v.9
                CFilePath.CheckFilePathsExist(_asterixFilesPaths);
            }
        }
        [XmlIgnore]
        public string[] GetSpecificationFilesPaths { get { return
this._specificationFilesPaths; } private set { } }
        [XmlIgnore]
        public string[] GetAsterixFilesPaths { get { return this._asterixFilesPaths;
} private set { } }
        #endregion
    }

    public static class CFilePath // v.9
    {
        #region Properties
        public static string FileName1 = "asxdec_ini.xml";
        public static string FileName2 = "asxdec_config.xml";
        #endregion

        #region Methods
        public static string ModifyPathIfNecessary(string filepath, string
relativePath) // MOVED v.9
        {
            Tuple<string, byte> filename = GetFileName(filepath);
            if (filename.Item2 == 1)
                return relativePath + filename.Item1;
            else
                return filepath;
        }
```

```
        public static string[] ModifyPathIfNecessary(string[] filePaths, string
relativePath) // v.9
        {
            return filePaths.Select(x => ModifyPathIfNecessary(x,
relativePath)).ToArray();
        }

        public static Tuple<string, byte> GetFileName(string filepath) // MOVED
v.9
        {
            filepath = filepath.Replace('/', (char)92);
            string[] filepathParts = filepath.Split((char)92);
            return new Tuple<string, byte>(filepathParts.Last(),
(byte)filepathParts.Length);
        }

        public static string[] GetFileNames(string[] filePaths) // v.9
        {
            return filePaths.Select(x =>
CFilePath.GetFileName(x).Item1).ToArray();
        }

        public static void CheckFilePathExists(string filePath) // v.9
        {
            try
            {
                if (!File.Exists(filePath))
                {
                    throw new FileNotFoundException($"'{filePath}' file was not
found");
                }
            }
            catch (FileNotFoundException ex)
            {
                ASXDECApp.ExitApp(ex.Message);
            }
        }

        public static void CheckFilePathsExist(string[] filePaths)
        {
            foreach (string filePath in filePaths)
            {
                CheckFilePathExists(filePath); // v.9
            }
        }
        #endregion
    }
}
```

CAsxdec_config.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Xml.Serialization;
using loss_output;
using ASXDECcore;

namespace ioss_input
{
    [XmlType(TypeName = "ASXDECconfig")]
    public class ASXDECconfig
    {
        #region Object Fields
        private CEvaluation _evaluation;
        private CFiltering _filtering;
        #endregion

        #region Properties
        [XmlElement(ElementName = "primarySettings")]
        public CEvaluation Evaluation { get { return this._evaluation; } set {
this._evaluation = value; } }
        [XmlElement(ElementName = "Filtering")]
        public CFiltering Filtering { get { return this._filtering; } set { this._filtering
= value; } }
        #endregion

    }

    #region primarySettings
    public class CEvaluation
    {
        #region Object Fields
        private CDataBaseParams _dataBaseParams;
        private CTimeRef _timeRef; // v.7
        private CFiltersSelection _filtersSelection;
        #endregion

        #region Properties
        [XmlElement(ElementName = "DataBase")]
        public CDataBaseParams DataBaseParams { get { return
this._dataBaseParams; } set { this._dataBaseParams = value; } }
        [XmlElement(ElementName ="TimeRef")] // v.7
        public CTimeRef TimeRef { get { return this._timeRef; } set {
this._timeRef = value; } } // v.7
        [XmlElement(ElementName = "Filters")]
```

```
        public CFiltersSelection FiltersSelection { get { return
this._filtersSelection; } set { this._filtersSelection = value; } }
        #endregion
    }

    #region DataBase
    //[XmlType(TypeName = "DataBase")]
    public class CDataBaseParams
    {
        #region Object Fields
        private static byte _maxREP = 6;
        private string _dbHost;
        private string _dbSchema;
        private string _dbUser;
        private string _dbPassw;
        private List<CDBTableParams> _dbTables;
        #endregion

        #region Properties
        [XmlElement(ElementName = "dbhost")]
        public string DbHost { get { return this._dbHost; } set { this._dbHost =
value; } }
        [XmlElement(ElementName = "dbschema")]
        public string DbSchema { get { return this._dbSchema; } set {
this._dbSchema = value; } }
        [XmlElement(ElementName = "dbuser")]
        public string DbUser { get { return this._dbUser; } set { this._dbUser =
value; } }
        [XmlElement(ElementName = "dbpassw")]
        public string DbPass { get { return this._dbPassw; } set { this._dbPassw
= value; } }
        [XmlArray(ElementName = "dbtables", IsNullable = true)]
        public List<CDBTableParams> DbTables { get { return this._dbTables; }
set { this._dbTables = value; } }
        [XmlAttribute(AttributeName = "maxFieldREPInd")]
        public byte MaxREPXML { get { return _maxREP; } set { _maxREP =
value; } } // v.9
        [XmlIgnore]
        public static byte MaxREP { get { return _maxREP; } set { _maxREP =
value; } }

        #endregion

        #region Methods
        public CDBTableParams GetDbTable(ushort catId)
        {
            return DbTables.Find(x => x.CategoryId == catId);
        }
        #endregion
    }
```

```
    [XmlType(TypeName = "dbtable")]
    public class CDBTableParams
    {
        #region Object Fields
        private byte _maxREP = CDataBaseParams.MaxREP; // v.9
        private ushort _categoryId;
        private string _tableName;
        #endregion

        #region Properties
        [XmlAttribute(AttributeName = "cat")]
        public ushort CategoryId { get { return this._categoryId; } set {
this._categoryId = value; } }
        [XmlAttribute(AttributeName = "name")]
        public string TableName { get { return this._tableName; } set {
this._tableName = value; } }
        [XmlAttribute(AttributeName = "maxFieldREPInd")]
        public byte MaxREP { get { return this._maxREP; } set { this._maxREP
= value; } } // v.9
        #endregion

        #region Methods
        public CDBTable CreateDBTable(CDataBase db)
        {
            return new CDBTable(db, _tableName, _maxREP);
        }
        #endregion
    }
    #endregion

    #region TimeReference // v.7
    public class CTimeRef
    {
        #region Object Fields
        private List<CatTimeRef> _categories;
        #endregion

        #region Properties
        [XmlElement(ElementName = "CatTimeRefDI")]
        public List<CatTimeRef> Categories { get { return this._categories; } set
{ this._categories = value; } }
        #endregion

        #region Methods
        public void ApplyTimeRef (CCategory cat)
        {
            CatTimeRef catTR = _categories.Find(x => cat.Num == x.CatNum);
            if (catTR != null)
            {
                catTR.ApplyTimeRef(cat);
```

```
            }
            else
            {
                ASXDECApp.ExitApp(String.Format("Missing Time Reference for CAT '{0}'."/* Consequently, there will not be a column for the 'time' in this category."*/, cat.Id));
            }
        }
        #endregion
    }

    public class CatTimeRef
    {
        #region Object Fields
        private string _catId;
        private ushort _catNum;
        private string _timeRefDataItemId;
        private uint _daySeconds = 86400;
        #endregion

        #region Properties
        [XmlAttribute(AttributeName = "catId")]
        public string CatId { get { return this._catId; }
            set
            {
                this._catId = value.PadLeft(3, '0');
                this._catNum = Convert.ToUInt16(value);
            }
        }
        [XmlIgnore]
        public ushort CatNum { get { return this._catNum; } private set { } }
        [XmlAttribute(AttributeName = "TRDataItemId")]
        public string TimeRefDataItemId { get { return this._timeRefDataItemId; }
            set
            {
                ushort num;
                if (ushort.TryParse(value, out num))
                    _timeRefDataItemId = num.ToString().PadLeft(3, '0');
                else
                    _timeRefDataItemId = value;
            }

        }
        #endregion
        #region Methods
        public void ApplyTimeRef(CCategory cat)
        {
            CDataItem dataItem = cat.FindDataItemById(_timeRefDataItemId);
            if (dataItem != null)
            {
```

```
                dataItem.DecodingMode = 3;
                cat.DecFieldsTable.AddFieldName("time");
                cat.DecFieldsTable.DBTable.AddColumnDataType(); // Time column
                cat.DecFieldsTable.IncreaseOneDefaultFieldCounter();
            }
            else
            {
                ASXDECApp.ExitApp(String.Format("CAT '{0}' DataItem '{1}' could
not be found in specification for Time Reference." /*Consequently, there will not
be a column for the 'time' in this category."*/,cat.Id,_timeRefDataItemId));
            }
        }
        #endregion
    }

    #endregion // v.7

    #region Filters
    public class CFiltersSelection
    {
        #region Object Fields
        private CEnable _enableDIFiltering;
        private CEnable _enableCatFiltering;
        private CEnable _enableSensorFiltering;
        #endregion

        #region Properties
        [XmlElement(ElementName = "enableDIFiltering")]
        public CEnable EnableDIFiltering { get { return this._enableDIFiltering; }
set { this._enableDIFiltering = value; } }
        [XmlElement(ElementName = "enableCatFiltering")]
        public CEnable EnableCatFiltering { get { return this._enableCatFiltering;
} set { this._enableCatFiltering = value; } }
        [XmlElement(ElementName = "enableSensorFiltering")]
        public CEnable EnableSensorFiltering { get { return
this._enableSensorFiltering; } set { this._enableSensorFiltering = value; } }
        #endregion

    }

    public class CEnable
    {
        #region Object Fields
        private bool _enable;
        #endregion

        #region Properties
        [XmlAttribute(AttributeName = "en")]
        public bool Enable { get { return this._enable; } set { this._enable =
value; } }
```

```
        #endregion
    }
    #endregion

    #endregion

    #region Filtering
    public class CFiltering
    {
        #region Object Fields
        private CPreDecodingFilter _preDecodingFilter;
        #endregion

        #region Properties
        [XmlElement(ElementName = "InputFilterSelection")]
        public CPreDecodingFilter PreDecodingFilter { get { return
this._preDecodingFilter; } set { this._preDecodingFilter = value; }
            #endregion

        }
    }

    //[XmlType(TypeName = "InputFilterSelection")]
    public class CPreDecodingFilter
    {
        #region Object Fields
        private CDataItemsFilter _dataItemsFilter;
        private CCategoriesFilter _categoriesFilter;
        private CSensorsFilter _sensorsFilter;
        #endregion

        #region Properties
        [XmlElement(ElementName = "DataItemsFilter")]
        public CDataItemsFilter DataItemsFilter { get { return
this._dataItemsFilter; } set { this._dataItemsFilter = value; } }
        [XmlElement(ElementName = "CategoriesFilter")]
        public CCategoriesFilter CategoriesFilter { get { return
this._categoriesFilter; } set { this._categoriesFilter = value; } }
        [XmlElement(ElementName = "SensorsFilter")]
        public CSensorsFilter SensorsFilter { get { return this._sensorsFilter; }
set { this._sensorsFilter = value; } }
        #endregion
    }

    #region SensorFiltering
    public class CSensorsFilter
    {
        #region Object Fields
        private List<CEnableSensor> _sensors;
        private Dictionary<Tuple<byte, byte>, bool> _isSensorEnableDict;
```

```
        private bool _enableNotListedSensors = false; // Default value false
        private string _defaultDSIid = "010"; // Default Id por Data Source Identifier
        #endregion

        #region Properties
        [XmlElement(ElementName = "radar")]
        public List<CEnableSensor> Sensors { get { return this._sensors; } set {
this._sensors = value; } }
        [XmlAttribute(AttributeName = "enableNotListedRadars")]
        public bool EnableNotListedSensors { get { return
this._enableNotListedSensors; } set { this._enableNotListedSensors = value; }
}
        [XmlAttribute(AttributeName = "defaultDSIid")]
        public string DefaultDSIid { get { return this._defaultDSIid; }
            set
            {
                ushort num;
                if (ushort.TryParse(value, out num))
                    _defaultDSIid = num.ToString().PadLeft(3, '0');
                else
                    _defaultDSIid = value;
            }
        }
        #endregion

        #region Methods
        public void CreateDictForFasterAccess()
        {
            CEnableSensor sen = null;
            try
            {
                _isSensorEnableDict = new Dictionary<Tuple<byte, byte>, bool>();
                ushort i = 0;
                while (i < _sensors.Count)
                {
                    sen = _sensors[i];
                    _isSensorEnableDict.Add(new Tuple<byte, byte>(sen.Sac,
sen.Sic), sen.Enable);
                    i++;
                }
            }
            catch (ArgumentException)
            {
                ASXDECApp.ExitApp(string.Format("There are two or more radars
with the same SAC '{0}' and SIC '{1}'", sen.Sac, sen.Sic));
            }
        }

        public byte GetSensorsGivenCat(CCategory cat, bool sensorsFiltEnabled,
CCatAvailableSensors availableSensors) // v.8
```

```
        {
            foreach (CEnableSensor sensor in _sensors)
            {
                if ((!sensorsFiltEnabled || sensor.Enable) &&
sensor.FindCatSensor(cat))
                {
                    availableSensors.AddSensor(sensor.Sac, sensor.Sic, true);
                }
            }
            if (availableSensors.GetNumAvailableSensors() == 0)
            {
                if (sensorsFiltEnabled)
                    ASXDECApp.ThrowWarning(String.Format("NO ENABLED sensor
is able to send CAT '{0}' messages. Therefore, messages from this category will
not be decoded and stored.", cat.Id));
                else
                    ASXDECApp.ThrowWarning(String.Format("NO sensor is able to
send CAT '{0}' messages. Therefore, messages from this category will not be
decoded and stored.", cat.Id));
                return 1;
            }
            return 0;
        }                                                                    //

        public byte ApplySensorFilter(CCategory category, CCatAvailableSensors
availableSensors /*added v.8*/)
        {
            CDataItem di_DSI = category.DataItems.Find(x => x.Id ==
_defaultDSIid);
            if (di_DSI != null)
            {
                di_DSI.DecodingMode = 2;
                // di_DSI.DIStructure.SensorFilter = this; // v.7 DELETED
                category.AvailableSensors = availableSensors; // v.8
                return 0;
            }
            return 1;
        }
        #endregion
    }

    public class CEnableSensor
    {
        #region Object Properties
        private string _id;
        private byte _sac;
        private byte _sic;
        private bool _enable;
        private List<CEnableCategorySensor> _sensorCategories;
        #endregion
```

```
        #region Properties
        [XmlAttribute(AttributeName = "id")]
        public string Id { get { return this._id; } set { this._id = value; } }
        [XmlAttribute(AttributeName = "sac")]
        public byte Sac { get { return this._sac; } set { this._sac = value; } }
        [XmlAttribute(AttributeName = "sic")]
        public byte Sic { get { return this._sic; } set { this._sic = value; } }
        [XmlAttribute(AttributeName = "en")]
        public bool Enable { get { return this._enable; } set { this._enable =
value; } }
        [XmlElement(ElementName = "CatSensor")]
        public List<CEnableCategorySensor> SensorCategories { get { return
this._sensorCategories; } set { this._sensorCategories = value; } }
        #endregion

        #region Methods
        public bool FindCatSensor(CCategory cat)
        {
            CEnableCategorySensor enCatSen = _sensorCategories.Find(x =>
x.CatId == cat.Id);
            if (enCatSen != null)
            {
                return enCatSen.IsEnabled;
            }
            else
            {
                return false;
            }
        }

        #endregion
    }

    public class CEnableCategorySensor: CEnableCategory
    {
        #region Object Fields
        private string _version;
        #endregion

        #region Properties
        [XmlAttribute(AttributeName = "ver")]
        public string Version {
            get
            {
                return Convert.ToByte(this._version.Substring(0, 2)) + "." +
Convert.ToByte(this._version.Substring(2, 2));
            }
            set
            {
```

```
            this._version = value;
        }
    }

    #endregion
}
#endregion

#region CategoriesFiltering
public class CCategoriesFilter
{
    #region Object Fields
    private List<CEnableCategory> _categories;
    private bool _enableNotListedCat;
    #endregion

    #region Properties
    [XmlElement(ElementName = "Category")]
    public List<CEnableCategory> Categories { get { return this._categories;
} set { this._categories = value; } }
    [XmlAttribute(AttributeName = "enableNotListedCat")]
    public bool EnableNotListedCat { get { return this._enableNotListedCat; }
set { this._enableNotListedCat = value; } }
    #endregion

    #region Methods
    public bool CheckIfCategoryIsEnabled(CCategory cat)
    {
        CEnableCategory cEnCat = _categories.Find(x => x.CatNum ==
cat.Num);
        if (cEnCat != null)
            return cEnCat.IsEnabled;
        else
            return _enableNotListedCat;
    }

    public bool CheckIfCategoryIsEnabled(ushort catNum)
    {
        CEnableCategory cEnCat = _categories.Find(x => x.CatNum ==
catNum);
        if (cEnCat != null)
            return cEnCat.IsEnabled;
        else
            return _enableNotListedCat;
    }

    public ushort CheckCategoriesHaveSpecification(Dictionary<ushort,
CCategory> categoriesSpecs)
    {
        foreach (CEnableCategory catEn in _categories)
```

```
            {
                if (catEn.IsEnabled &&
!categoriesSpecs.ContainsKey(catEn.CatNum))
                {
                    ASXDECApp.ExitApp(String.Format("CAT '{0}' is ENABLED in the
categories filter but its specification has NOT been provided.", catEn.CatId)); //
v.8 previously Warning, now ExitApp Error
                }
            }
            return 0;
        }
        #endregion
    }

    public class CEnableCategory
    {
        #region Object Fields
        protected string _catId;
        protected ushort _catNum;
        protected bool _isEnabled;
        #endregion

        #region Properties
        [XmlAttribute(AttributeName = "id")]
        public string CatId { get { return this._catId; }
            set
            {
                this._catId = value.PadLeft(3, '0');
                this._catNum = Convert.ToUInt16(value);
            }
        }
        [XmlAttribute(AttributeName = "en")]
        public bool IsEnabled { get { return this._isEnabled; } set {
this._isEnabled = value; } }
        [XmlIgnore]
        public ushort CatNum { get { return this._catNum; } private set {
this._catNum = value; } }
        #endregion
    }
    #endregion

    #region DataItemsFiltering
    //[XmlType(TypeName = "DataItemsFilter")]
    public class CDataItemsFilter
    {
        #region Object Fields
        private List<CCategoryDIFilter> _categories;
        #endregion

        #region Properties
```

```
        [XmlElement(ElementName = "CatDataItems")]
        public List<CCategoryDIFilter> Categories { get { return this._categories;
} set { this._categories = value; } }
        #endregion

        #region Methods
        public CCategoryDIFilter GetCategoryDIFilter(CCategory cat)
        {
            return _categories.Find(x => x.CatId == cat.Num);
        }

        public byte CheckDIHaveSpecification (Dictionary<ushort,CCategory>
categoriesSpec)
        {
            ushort n = 0;
            ushort p = 0;
            foreach (CCategoryDIFilter catDIFilter in _categories)
            {
                if (categoriesSpec.ContainsKey(catDIFilter.CatId))
                {

catDIFilter.CheckDIHaveSpecification(categoriesSpec[catDIFilter.CatId]);
                    if (!catDIFilter.IsEnabled)
                        n++;
                    p++;
                }
            }
            if (n == p)
                return 1;
            return 0;
        }

        #endregion
    }

    [XmlType(TypeName = "CatDataItems")]
    public class CCategoryDIFilter
    {
        #region Object Fields
        private ushort _catID;
        private List<CSkipDataItem> _dataItems;
        private bool _skipNotListedDI = true;
        private bool _isEnabled = true;
        #endregion

        #region Properties
        [XmlAttribute(AttributeName = "catId")]
        public ushort CatId { get { return this._catID; } set { this._catID = value; }
}
        [XmlAttribute(AttributeName = "skipNotListedDI")]
```

```
        public bool SkipNotListedDI { get { return this._skipNotListedDI; } set {
this._skipNotListedDI = value; } }
        [XmlElement(ElementName = "DataItem")]
        public List<CSkipDataItem> DataItems { get { return this._dataItems; }
set { this._dataItems = value; } }
        [XmlIgnore]
        public bool IsEnabled { get { return this._isEnabled; } private set { } }
        #endregion

        #region Methods

        public void CheckDIHaveSpecification(CCategory cat)
        {
            if (_dataItems.Count == 0)
            {
                _isEnabled = false;
                //return 0;
            }
            else
            {
                // ushort n = 0;
                for (int i = 0; i < _dataItems.Count; i++)
                {
                    CSkipDataItem skipDI = _dataItems[i];
                    bool exists = cat.DataItems.Exists(x => x.Id == skipDI.DataItemID);
                    if (!exists)
                    {
                        Console.WriteLine("WARNING: DataItem '{0}' from CAT '{1}' is in
the DataItems filter but is not in the category specification.", skipDI.DataItemID,
this._catID);
                        _dataItems.Remove(skipDI);
                        i--;
                    }
                }
            }

        }

        public bool CheckIfDataItemIsSkipped(CDataItem dataItem)
        {
            CSkipDataItem skipDI = _dataItems.Find(x => x.DataItemID ==
dataItem.Id);
            if (skipDI != null)
                return skipDI.IsSkipped;
            else
                return _skipNotListedDI;
        }

        public byte ApplyDIFilter(CCategory cat)
        {
```

```
            ushort i = 0;
            foreach (CDataItem di in cat.DataItems)
            {
                if (CheckIfDataItemIsSkipped(di))
                {
                    if (di.DecodingMode == 0)
                        di.DecodingMode = 1;
                    i++;
                }
            }
            if (i == cat.DataItems.Count)
            {
                Console.WriteLine("WARNING: DataItems filter skips all DataItems
from CAT '{0}' specification. Therefore, its ASTERIX records will not be decoded
and stored. Consider using the category filter instead.", _catID);
                return 1;
            }
            return 0;
        }
        #endregion
    }

    //[XmlType(TypeName = "DataItem")]
    public class CSkipDataItem
    {
        #region Object Fields
        private string _dataItemID;
        //private ushort _dataItemNum;
        private bool _isSkipped;
        #endregion

        #region Properties
        [XmlAttribute(AttributeName = "id")]
        public string DataItemID
        {
            get { return this._dataItemID; }
            set
            {
                ushort num;
                if (ushort.TryParse(value, out num))
                    this._dataItemID = num.ToString().PadLeft(3, '0');
                else
                    this._dataItemID = value;
                //this._dataItemNum = Convert.ToUInt16(value);
            }
        }
        //[XmlIgnore]
        //public ushort DataItemNum { get { return this._dataItemNum; } private set
{ } }
        [XmlAttribute(AttributeName = "skip")]
```

```
        public bool IsSkipped { get { return this._isSkipped; } set {
this._isSkipped = value; } }
        #endregion
    }
    #endregion

    #endregion
}
```

Subprojecte loss_output

CDataBase.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.Common;
//using MySqlConnector;
using MySql.Data.MySqlClient; // MySql.Data NuGet Package before v6.79, now v6.9.12
using System.Xml;
using System.Xml.Serialization;

namespace loss_output
{
   public class CDataBase
   {
      #region Object Fields
      private MySqlConnection cnx;
      private string _cnxStr;
      #endregion

      #region Constructor
      public CDataBase(string dataSource, string dbName, string userID, string password)
      {
         _cnxStr = "DataSource=" + dataSource + ";Database=" + dbName + ";User ID=" + userID + ";Password='" + password + "';";
         this.cnx = new MySqlConnection(_cnxStr);
         this.Open();
      }
      #endregion

      #region Methods
      public void Open()
      {
         try
         {
            if (!cnx.State.Equals(ConnectionState.Open))
               this.cnx.Open();
         }
         catch (MySqlException)
         {
            ASXDECApp.ExitApp("Cannot open data base.");
```

```
        }
    }

    public void Close()
    {
        this.cnx.Close();
    }

    public void CreateTable(string tableName, string columnsStr)
    {
        Open();
        string createTableStr = "CREATE TABLE " + tableName + " (" +
columnsStr + ")";
        MySqlCommand createTableCmd = new
MySqlCommand(createTableStr, this.cnx);
        try
        {
            if ((int)createTableCmd.ExecuteNonQuery() == 0)
            {
                ASXDECApp.SuccessfulMsg("(Database) Table '" + tableName + "'
was succesfully created.");
            }

        }
        catch (MySqlException exception)
        {
            if ((MySqlErrorCode)exception.Number ==
MySqlErrorCode.TableExists)
            {
                ASXDECApp.ThrowWarning("(Database) Table '" + tableName + "'
already exists.");
            }
            else
            {
                ASXDECApp.ExitApp("When trying to create table '" + tableName
+ "'\n" + exception.Message);
            }
        }
        Close();
    }


    public async Task<int> ExecuteInsertCommandAsync(MySqlCommand
insertCmd)
    {
        return insertCmd.ExecuteNonQuery();
    }
    /// <summary>
```

```
        /// Podria haber insertado cada valor con su correspondiente columna "(column1, column2,...) VALUES (column1value, column2value,...)" por lo que no haria falta añadir nulls en las columnas vacias.
        /// No obstante, en el caso de querer hacer una inserción multifila como es nuestro caso, como no todas las filas (paquetes Asterix) tienen siempre
        /// las mismas columnas (campos) esto no es posible.
        /// </summary>
        /// <param name="tableName"></param>
        /// <param name="values"></param>
        public void InsertIntoTable(string tableName, string valuesStr)
        {
            Open();
            string query = "INSERT INTO " + tableName + " VALUES " + valuesStr;
            MySqlCommand insertCmd = new MySqlCommand(query, this.cnx);
            try
            {
                int res = insertCmd.ExecuteNonQuery();
            }
            catch (MySqlException ex)
            {
                ASXDECApp.ThrowError(String.Format("(Database) Cannot insert data into '{0}' table\n" + ex.Message, tableName)); Console.ReadKey();
            }
            Close();
        }
        #endregion
    }

    public class CDBTable
    {
        #region Object Fields
        private List<string> _dBColumnsDataTypes;
        private string _tableName;
        private List<bool> _isRepField; // v.9
        private byte _maxREP;
        private loss_output.CDataBase _dataBase;
        #endregion

        #region Properties
        public string TableName { get { return this._tableName; } set { this._tableName = value; } }
        public CDataBase DataBase { get { return this._dataBase; } set { this._dataBase = value; } }
        [XmlAttribute(AttributeName = "maxREP")]
        public byte MaxREP { get { return this._maxREP; } set { this._maxREP = value; } } // v.9
        #endregion

        #region Constructors
        public CDBTable(CDataBase dB, string tableName, byte maxREP)
```

```
        {
            List<object> p = new List<object>();
            this._dBColumnsDataTypes = new List<string>();
            this._dataBase = dB;
            this._tableName = tableName;
            this._maxREP = maxREP;
            this._isRepField = new List<bool>();
        }
        #endregion

        #region Methods
        public void AddColumnDataType(string dataType = "varchar(50) NULL")
        {
            _dBColumnsDataTypes.Add(dataType);
            _isRepField.Add(false);
        }

        public void AddColumnDataType(Tuple<string, bool> dataType, string
dataItemFormat, ushort bits_length) // v.9
        {
            if (DIFormats.repetitive == (DIFormats)Enum.Parse(typeof(DIFormats),
dataItemFormat)) // v.9
            {
                //EncodingOptions p =
(EncodingOptions)Enum.Parse(typeof(EncodingOptions) as Type,
dataType.Item1);
                int numChars =
ComputeChars((byte)((EncodingOptions)Enum.Parse(typeof(EncodingOptions)
, dataType.Item1)), bits_length);
                if (dataType.Item2) // Further investigation needed
                    numChars += 15 + 1; // 15 decimal digits double precision plus
decimal dot
                numChars = numChars * _maxREP + 2 * (_maxREP - 1);
                _dBColumnsDataTypes.Add($"varchar({numChars}) NULL");
                _isRepField.Add(true); // CHANGED v.9
            }
            else
            {
                AddColumnDataTypeNoRep(dataType, dataItemFormat);
            }
            //EncodingOptions p =
Enum.GetNames(EncodingOptions).ToList().Find(x => x == dataItemFormat);
        }

        private byte ComputeChars(byte @base, ushort bits_length)
        {
            return (byte)Math.Ceiling(Math.Log(Math.Pow(2, bits_length), @base));
        }
```

```
        private void AddColumnDataTypeNoRep(Tuple<string, bool> dataType,
string dataItemFormat) // CHANGED v.9
        {
            if (dataType.Item2)
            {
                _dBColumnsDataTypes.Add("double NULL");
            }
            else
            {
                switch (dataType.Item1)
                {
                    case ("uint"):
                        {
                            _dBColumnsDataTypes.Add("int UNSIGNED NULL");
                            break;
                        }
                    case ("int"):
                        {
                            _dBColumnsDataTypes.Add("int SIGNED NULL");
                            break;
                        }
                    case ("hex"):
                    case ("octal"):
                    case ("sixbitschar"):
                        {
                            _dBColumnsDataTypes.Add("varchar(50) NULL");
                            break;
                        }
                }
            }
            _isRepField.Add(false);
        }

        public void CreateTable(List<string> dBColumnsNames)
        {
            if (dBColumnsNames.Count != 0)
            {
                if (_dBColumnsDataTypes.Count == dBColumnsNames.Count)
                {
                    string columnsStr = "";
                    ushort i = 0;
                    while (i < _dBColumnsDataTypes.Count)
                    {
                        if (i > 0) { columnsStr += ", "; }
                        columnsStr += dBColumnsNames[i] + " " +
_dBColumnsDataTypes[i];
                        i++;
                    }
                    _dataBase.CreateTable(_tableName, columnsStr);
                }
```

```
            else
            {
                ASXDECApp.ExitApp("(DataBaseTable => CreateTable) The
number of columns is not the same as the number of column data types");
            }
        }
        else
        {
            ASXDECApp.ThrowWarning(String.Format("No attempt to create '{0}'
will be performed because there are no columns.", _tableName));
        }

    }

    public void InsertValues(List<string[]> decFieldValues) // CHANGED v.9
    {
        if (decFieldValues.Count != 0) // v.8 only the if // CRITICAL ERROR
solved
        {
            string valuesStr = "";
            int i = 0;
            int L = decFieldValues.Count;
            while (i < L)
            {
                if (i > 0)
                {
                    valuesStr += "),";
                }
                ushort j = 0;
                string[] onePackageDecFields = decFieldValues[i];
                while (j < onePackageDecFields.Length)
                {
                    string value = onePackageDecFields[j];
                    if (string.IsNullOrWhiteSpace(value))
                    {
                        value = "NULL";
                    }
                    else if (_dBColumnsDataTypes[j].StartsWith("var")) // Cuando el
valor que se inserta es una varchar hay que ponerlo entre comillas => 'VALUE'
                    {
                        #region // v.9
                        if (_isRepField[j])
                        {
                            string[] trozos = value.Split(';');
                            byte rep = (byte)trozos.Length;
                            if (rep > _maxREP)
                            {
                                value = String.Join(";", trozos.Take(_maxREP));
                            }
                        }
```

```
                #endregion
                value = """ + value + """;
            }
            if (j == 0) { valuesStr += $"({value}"; }
            else
            {
                valuesStr += $",{value}";
            }
            j++;
        }
        i++;
    }
    valuesStr += ")";
    _dataBase.InsertIntoTable(_tableName, valuesStr);
  }
}

public static void ObtainValue(List<string[]> decFieldValues)
{
  string valuesInsertSrt = "";
  int i = 0;
  int L = decFieldValues.Count;
  while (i < L)
  {
    if (i > 0)
    {
      valuesInsertSrt += "),";
    }
    ushort j = 0;
    string[] onePackageDecFields = decFieldValues[i];
    while (j < onePackageDecFields.Length)
    {
      if (j == 0) { valuesInsertSrt += $"({onePackageDecFields[j]}"; }
      else
      {
        valuesInsertSrt += $",{onePackageDecFields[j]}";
      }
      j++;
    }
    i++;
  }
  valuesInsertSrt += ")";
}

public static void ObtainValue(List<string[]> decFieldValues, int
startIndex, int length)
{
  string valuesInsertSrt = "";
  int i = startIndex;
  int L = startIndex + length;
```

```
        while (i < L)
        {
            if (i > startIndex)
            {
                valuesInsertSrt += "),";
            }
            ushort j = 0;
            string[] onePackageDecFields = decFieldValues[i];
            while (j < onePackageDecFields.Length)
            {
                if (j == 0) { valuesInsertSrt += $"({onePackageDecFields[j]}"; }
                else
                {
                    valuesInsertSrt += $",{onePackageDecFields[j]}";
                }
                j++;
            }
            i++;
        }
        valuesInsertSrt += ")";
    }
    #endregion Methods
}


#region Further implementation
public class DataBaseType
{
    public string ColumnFormatTypes { get; set; }
}

[XmlType(TypeName = "DBDataTypes")]
public class DBDataTypes
{
    [XmlArray(ElementName = "integer")]
    public List<DBIntType> IntegerTypes { get; set; }
    [XmlArray(ElementName = "real")]
    public List<DBRealType> RealTypes { get; set; }
    [XmlArray(ElementName = "string")]
    public List<DBStringType> StringTypes { get; set; }
}

[XmlType(TypeName = "type")]
public class DBDataType
{
    [XmlText(DataType = "string")]
    public string Name { get; set; }
}

public class DBIntType : DBDataTypes
```

```
    {
        [XmlAttribute(AttributeName = "bits")]
        public ushort Bits { get; set; }
    }

    public class DBRealType : DBDataTypes
    {
        [XmlAttribute(AttributeName = "smallest")]
        public decimal Smallest { get; set; }
        [XmlAttribute(AttributeName = "largest")]
        public decimal Biggest { get; set; }
    }

    public class DBStringType : DBDataTypes
    {
        [XmlAttribute(AttributeName = "chars")]
        public uint Chars { get; set; }
    }
    #endregion
}
```

ASXDECApp.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace loss_output
{
   public class ASXDECApp
   {
      #region Object Static Fields
      private static StreamWriter sw;
      private static string _filepath;
      public static string FilePath { get { return _filepath; } set { _filepath =
value; } }
      public static ConsoleColor WarningColor = ConsoleColor.Yellow;
      public static ConsoleColor ErrorColor = ConsoleColor.Red;
      public static ConsoleColor QuestionColor = ConsoleColor.DarkCyan;
      public static ConsoleColor SuccesfulColor = ConsoleColor.Green;
      public static bool ReleaseMode = false;
      #endregion

      #region Static Methods
      public static byte CreateFoldersIfNotExist(string relPath)
      {
         string logsPath = Path.Combine(relPath, "logs");
         if (!Directory.Exists(logsPath))
         {
            SetLogFilePath(logsPath);
            Directory.CreateDirectory(logsPath);
            Console.WriteLine(String.Format("Creating directory '{0}' ...",
logsPath));
         }
         else
            SetLogFilePath(logsPath);

         string specsFilePath = Path.Combine(relPath, "specification-folder");
         if (!Directory.Exists(specsFilePath))
         {
            Console.WriteLine(String.Format("Creating directory '{0}' ...",
specsFilePath));
            Console.WriteLine("Here you need to add the ASTERIX categories
specification XML files.");
            Directory.CreateDirectory(specsFilePath);
            return 1;
         }
         return 0;
```

```
    }

    public static void SetLogFilePath(string relPath)
    {
        string fileName = DateTime.Now.ToString("yyyyMMdd_HHmmss") +
".log";
        _filepath = Path.Combine(relPath, fileName);
    }

    public static void Open()
    {
        sw = File.AppendText(_filepath);
    }

    public static void Close()
    {
        sw?.Close();
    }

    public static void Write(string message)
    {
        Open();
        sw?.WriteLine(DateTime.Now.ToString("HH:mm:ss - ") + message);
        Close();
    }

    public static void ExitApp(string errorMessage)
    {
        ThrowError(errorMessage);
        Console.WriteLine("\a"); // v.5 Audible beep added.
        Write("EXITING ASXDEC...");
        Environment.Exit(-1);
    }

    public static void AskQuestion(string questionMessage)
    {
        Console.ForegroundColor = QuestionColor;
        Console.WriteLine("QUESTION: " + questionMessage);
        Console.ResetColor();
    }

    public static void ThrowWarning(string warningMessage)
    {
        Write("WARNING: " + warningMessage);
        Console.ForegroundColor = WarningColor;
        //sw.Close();
        Console.WriteLine("WARNING: " + warningMessage);
        Console.ResetColor();
    }
```

```
    public static void ThrowError(string errorMessage)
    {
        Write("ERROR: " + errorMessage);
        Console.ForegroundColor = ErrorColor;
        Console.WriteLine("ERROR: " + errorMessage);
        Console.ResetColor();
    }

    public static void SuccessfulMsg (string successfulMessage)
    {
        Write("INFO: " + successfulMessage);
        Console.ForegroundColor = SuccesfulColor;
        Console.WriteLine("INFO: " + successfulMessage);
        Console.ResetColor();
    }

    public static void InfoMsg(string infoMessage)
    {
        Write("INFO: " + infoMessage);
        Console.WriteLine("INFO: " + infoMessage);
    }

    public static void InfoMsg(string infoMessage, bool extraLineBefore)
    {
        string msg;
        if (extraLineBefore)
            msg = "\nINFO: " + infoMessage;
        else
            msg = "INFO: " + infoMessage + "\n";
        Write(msg);
        Console.WriteLine(msg);
    }
    #endregion
}

/// <summary>
/// Available field encoding options
/// </summary>
///
public enum EncodingOptions // MOVED v.9
{
    @uint = 10,
    @int = 10,
    hex = 16,
    octal = 8,
    sixbitschar = 64,
}

/// <summary>
///
```

```
    /// </summary>
    public enum DIFormats // MOVED v.9
    {
        @fixed,
        variable,
        compound,
        repetitive,
        @explicit,
    }
}
```

Aplicació de consola (main)

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using ASXDECcore;
using loss_output;
using System.IO;
using System.Runtime;

namespace ConsoleApp
{
   class Program
   {
      static void Main(string[] args) // OPTION 1 Args: "asxdec_config.xml" and "asterixfile.ast" // "asterix.ini" at App basepath for ASTERIX categories specs paths.
      {                              // OPTION 2 Args: "asterix_ini.xml" and "asterixfile.ast"
//  "asterix_ini.xml" contains ASTERIX categories specs paths and "asxdec_config.xml" path
         string appPath = Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location);
         Environment.CurrentDirectory = appPath;
         if (ASXDECApp.CreateFoldersIfNotExist(appPath) == 1)
         {
            ASXDECApp.InfoMsg("Application needs to be RESTARTED.");
            Console.WriteLine("Press Enter to exit ...");
            Console.ReadLine();
            Environment.Exit(1);
         }

         Console.WriteLine("--------------" + " WELCOME TO ASXDEC - ASTERIX Decoder " + "--------------", (char)169);
         ASXDECApp.InfoMsg("ASXDEC was initiated ...");
         Console.WriteLine(String.Format("ASXDEC working DIR is located at '{0}'", appPath));

         if (args.Length == 0)
            args = new string[] { "asxdec_ini.xml" };
            //ASXDECApp.ExitApp(String.Format("NO command-line arguments were provided. You need to provide the path for '{0}'", CFilePath.FileName1));

         byte maxAppArgs = 1;
         if (args.Length > maxAppArgs)
```

```
        {
            ASXDECApp.ThrowWarning(String.Format("Extra command-line
arguments ({0}) will be ignored.", args.Length - maxAppArgs));
        }

        CFileDecoder fd = new CFileDecoder(args[0]); // v.9
        ASXDECApp.InfoMsg("Execution completed!");
        //Console.ReadLine();
        Environment.Exit(0);
    }
  }
}
```

Annex C. Fitxers XML (inicialització, configuració i especificacions de categories)

asxdec_ini.xml (exemple)

```
<?xml version="1.0" encoding="UTF-8"?>
<ASXDECini>
   <ASXDECconfigFilePath>
      asxdec_config.xml
   </ASXDECconfigFilePath>
   <ASTERIXspecificationFilesPaths>
      asterix_cat021_2_1_RE1_1.xml
      asterix_cat010_1_1.xml
   </ASTERIXspecificationFilesPaths>
   <ASTERIXtodecodeFilePaths>
      210917-leb2-100001.ast
   </ASTERIXtodecodeFilePaths>
</ASXDECini>
```

asxdec_config.xml (exemple)

```
<?xml version="1.0" encoding="UTF-8"?>
<ASXDECconfig>
   <primarySettings>
      <DataBase maxFieldREPInd="6">
         <dbuser>root</dbuser>
         <dbhost>localhost</dbhost>
         <dbpassw>1234</dbpassw>
         <dbschema>asterix</dbschema>
         <dbtables>
            <dbtable cat="01" name="tbl_asx01"> </dbtable>
            <dbtable cat="02" name="tbl_asx02"> </dbtable>
            <dbtable cat="10"
name="tbl_asx10_201002_lebl_080001_smr_mlat_adsb__02"
maxFieldREPInd="1"> </dbtable>
            <dbtable cat="19" name="tbl_asx19" maxFieldREPInd="8">
</dbtable>
            <dbtable cat="20" name="tbl_asx20"> </dbtable>
            <dbtable cat="21"
name="tbl_asx21_201002_lebl_080001_smr_mlat_adsb__02"> </dbtable>
            <dbtable cat="34" name="tbl_asx34"> </dbtable>
            <dbtable cat="48" name="tbl_asx48"> </dbtable>
         </dbtables>
      </DataBase>
      <TimeRef>
         <CatTimeRefDI catId="010" TRDataItemId="140"></CatTimeRefDI>
         <CatTimeRefDI catId="021" TRDataItemId="077"></CatTimeRefDI>
      </TimeRef>
      <Filters>
         <enableCatFiltering en="false"></enableCatFiltering>
         <enableDIFiltering en="false"></enableDIFiltering>
         <enableSensorFiltering en="false"></enableSensorFiltering>
      </Filters>
   </primarySettings>

   <Filtering>
      <InputFilterSelection>
         <CategoriesFilter enableNotListedCat = "false">
            <Category id="010" en="false"/>
            <Category id="021" en="true"/>
            <Category id="048" en="false"/>
         </CategoriesFilter>
         <DataItemsFilter>
            <CatDataItems catId="010" skipNotListedDI = "false">
               <!--<DataItem id="000" skip="false"/>-->
               <DataItem id="001" skip="false"/>
               <DataItem id="010" skip="true"/>
               <DataItem id="020" skip="true"/>
               <DataItem id="040" skip="true"/>
```

```
            <DataItem id="041" skip="true"/>
            <DataItem id="042" skip="true"/>
            <DataItem id="060" skip="true"/>
            <DataItem id="090" skip="true"/>
            <DataItem id="091" skip="true"/>
            <DataItem id="131" skip="true"/>
            <DataItem id="140" skip="true"/>
            <DataItem id="161" skip="true"/>
            <DataItem id="170" skip="true"/>
            <DataItem id="200" skip="true"/>
            <DataItem id="202" skip="true"/>
            <DataItem id="210" skip="true"/>
            <DataItem id="220" skip="true"/>
            <DataItem id="245" skip="true"/>
            <DataItem id="250" skip="true"/>
            <DataItem id="270" skip="true"/>
            <DataItem id="280" skip="true"/>
            <DataItem id="300" skip="true"/>
            <DataItem id="310" skip="true"/>
            <DataItem id="500" skip="true"/>
            <DataItem id="550" skip="true"/>
            <DataItem id="SP" skip="true"/>
        </CatDataItems>
        <CatDataItems catId="019" skipNotListedDI = "true">
            <DataItem id ="001" skip ="true"/>
        </CatDataItems>
        <CatDataItems catId="020">

        </CatDataItems>
        <CatDataItems catId="021">

        </CatDataItems>
    </DataItemsFilter>
    <SensorsFilter defaultDSIid="010">
        <radar id="SMR_GCXO" sac="0" sic="1" en="false">
            <CatSensor id="010" ver="0101" en="true" />
        </radar>
        <radar id="SMR_LEPA" sac="0" sic="2" en="false">
            <CatSensor id="010" ver="0101" en="true" />
        </radar>
        <radar id="SMR_LEMD-S" sac="0" sic="3" en="false">
            <CatSensor id="010" ver="0101" en="true" />
        </radar>
        <radar id="SMR_LEMD-N" sac="0" sic="4" en="true">
            <CatSensor id="010" ver="0101" en="true" />
        </radar>
        <radar id="SMR_LEAS" sac="0" sic="5" en="false">
            <CatSensor id="010" ver="0101" en="true" />
        </radar>
        <radar id="SMR_LEST" sac="0" sic="6" en="false">
```

```
    <CatSensor id="010" ver="0101" en="true" />
</radar>
<radar id="SMR_LEBL" sac="0" sic="7" en="false">
    <CatSensor id="010" ver="0101" en="true" />
</radar>
<radar id="SMR_LEBB" sac="0" sic="8" en="false">
    <CatSensor id="010" ver="0101" en="true" />
</radar>
<radar id="MLAT_GCXO" sac="0" sic="101" en="false">
    <CatSensor id="010" ver="0101" en="true" />
    <CatSensor id="021" ver="0201" en="true" />
</radar>
<radar id="MLAT_LEPA" sac="0" sic="102" en="false">
    <CatSensor id="010" ver="0101" en="true" />
</radar>
<radar id="MLAT_LEMD" sac="0" sic="104" en="true">
    <CatSensor id="010" ver="0101" en="true" />
    <CatSensor id="019" ver="0103" en="false" />
    <CatSensor id="020" ver="0107" en="false" />
    <CatSensor id="021" ver="0201" en="false" />
</radar>
<radar id="MLAT_LEAS" sac="0" sic="105" en="false">
    <CatSensor id="010" ver="0101" en="true" />
    <CatSensor id="019" ver="0103" en="false" />
    <CatSensor id="020" ver="0107" en="false" />
    <CatSensor id="021" ver="0201" en="false" />
</radar>
<radar id="MLAT_LEBL" sac="0" sic="107" en="false">
    <CatSensor id="010" ver="0101" en="true" />
    <CatSensor id="019" ver="0103" en="true" />
    <CatSensor id="020" ver="0107" en="true" />
    <CatSensor id="021" ver="0201" en="true" />
</radar>

<radar id="SSRS_BEG" sac="20" sic="132" en="true">
    <CatSensor id="034" ver="0101" en="true" />
    <CatSensor id="048" ver="0103" en="true" />
    <CatSensor id="021" ver="0201" en="false" />
</radar>
<radar id="SSRS_BCN" sac="21" sic="132" en="true">
    <CatSensor id="034" ver="0101" en="true" />
    <CatSensor id="048" ver="0103" en="true" />
    <CatSensor id="021" ver="0201" en="true" />
</radar>
<radar id="SSRS_BCN" sac="22" sic="132" en="false">
    <CatSensor id="034" ver="0101" en="true" />
    <CatSensor id="048" ver="0103" en="true" />
    <CatSensor id="021" ver="0201" en="false" />
</radar>
<radar id="ADSB_unknown00" sac="20" sic="219" en="false">
```

```
                <CatSensor id="021" ver="0201" en="true" />
            </radar>
        </SensorsFilter>
    </InputFilterSelection>
</Filtering>
</ASXDECconfig>
```

asterix_cat010_1_1.xml (Especificació Categoria 10 v1.1)

```
<?xml version="1.0" encoding="UTF-8"?>
<Category id="010" name="Transmission of Monosensor Surface Movement
Data" ver="1.1">
  <!--Data Item I010/000, Message Type-->
  <DataItem id="000" rule="mandatory" frn="2">
    <DataItemName>Message Type</DataItemName>
    <DataItemDefinition>This data item allows for a more convenient handling of
the messages at the receiver side by further defining the type of
transaction.</DataItemDefinition>
    <DataItemStructure format="fixed" length="1" desc="One-octet SubField
length Data Item.">
      <Field msb="8" lsb="1">
        <FieldShortName>MsgTyp</FieldShortName>
        <FieldName>Type of msg.</FieldName>
        <Range min="1" max="4"></Range>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/010, Data Source Identifier-->
  <DataItem id="010" rule="mandatory" frn="1">
    <DataItemName>Data Source Identifier</DataItemName>
    <DataItemDefinition>Identification of the system from which the data are
received.</DataItemDefinition>
    <DataItemStructure format="fixed" length="2" desc="Two-octet SubField
length Data Item.">
      <Field msb="16" lsb="9">
        <FieldShortName>SAC</FieldShortName>
        <FieldName>System Area Code SubField to zero</FieldName>
      </Field>
      <Field msb="8" lsb="1">
        <FieldShortName>SIC</FieldShortName>
        <FieldName>System Identification Code</FieldName>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/020, Target Report Descriptor-->
  <DataItem id="020" rule="mandatory" frn="3">
    <DataItemName>Target Report Descriptor</DataItemName>
    <DataItemDefinition>Type and characteristics of the data as transmitted by a
system.</DataItemDefinition>
    <DataItemStructure format="variable" desc="Variable length Data Item
comprising a first part of one-octet, followed by one-octet extents as
necessary.">
      <SubField name="First Part" format="fixed" length="1">
        <Field msb="8" lsb="6">
          <FieldShortName>TYP</FieldShortName>
          <FieldName>TYP</FieldName>
          <FieldValue val="0">SSR multilateration</FieldValue>
```

```
        <FieldValue val="1">Mode-S multilateration</FieldValue>
        <FieldValue val="2">ADS-B</FieldValue>
        <FieldValue val="3">PSR</FieldValue>
        <FieldValue val="4">Magnetic Loop System</FieldValue>
        <FieldValue val="5">HF multilateration</FieldValue>
        <FieldValue val="6">Not defined</FieldValue>
        <FieldValue val="7">Other types</FieldValue>
      </Field>
      <Field msb="5" lsb="5">
        <FieldShortName>DCR</FieldShortName>
        <FieldName>Differential Correction</FieldName>
        <FieldValue val="0">No differential correction (ADS-B)</FieldValue>
        <FieldValue val="1">Differential correction (ADS-B)</FieldValue>
      </Field>
      <Field msb="4" lsb="4">
        <FieldShortName>CHN</FieldShortName>
        <FieldValue val="0">Chain 1</FieldValue>
        <FieldValue val="1">Chain 2</FieldValue>
      </Field>
      <Field msb="3" lsb="3">
        <FieldShortName>GBS</FieldShortName>
        <FieldName>Ground Bit Setting</FieldName>
        <FieldValue val="0">Ground Bit not set</FieldValue>
        <FieldValue val="1">Ground Bit set</FieldValue>
      </Field>
      <Field msb="2" lsb="2">
        <FieldShortName>CRT</FieldShortName>
        <FieldValue val="0">No Corrupted reply in multilateration</FieldValue>
        <FieldValue val="1">Corrupted reply in multilateration</FieldValue>
      </Field>
    </SubField>
    <SubField name="First Extent" format="fixed" length="1">
      <Field msb="8" lsb="8">
        <FieldShortName>SIM</FieldShortName>
        <FieldValue val="0">Actual target report</FieldValue>
        <FieldValue val="1">Simulated target report</FieldValue>
      </Field>
      <Field msb="7" lsb="7">
        <FieldShortName>TST</FieldShortName>
        <FieldValue val="0">Default</FieldValue>
        <FieldValue val="1">Test Target</FieldValue>
      </Field>
      <Field msb="6" lsb="6">
        <FieldShortName>RAB</FieldShortName>
        <FieldValue val="0">Report from target transponder</FieldValue>
        <FieldValue val="1">Report from field monitor (SubField
transponder)</FieldValue>
      </Field>
      <Field msb="5" lsb="4">
        <FieldShortName>LOP</FieldShortName>
```

```
        <FieldName>LOP</FieldName>
        <FieldValue val="0">Undetermined</FieldValue>
        <FieldValue val="1">Loop start</FieldValue>
        <FieldValue val="2">Loop stop</FieldValue>
        <Range min="0" max="2"></Range>
       </Field>
       <Field msb="3" lsb="2">
        <FieldShortName>TOT</FieldShortName>
        <FieldName>Type of target</FieldName>
        <FieldValue val="0">Undetermined</FieldValue>
        <FieldValue val="1">Aircraft</FieldValue>
        <FieldValue val="2">Ground vehicle</FieldValue>
        <FieldValue val="3">Helicopter</FieldValue>
       </Field>
      </SubField>
      <SubField name="Second Extent" format="fixed" length="1">
       <Field msb="8" lsb="8">
        <FieldShortName>SPI</FieldShortName>
        <FieldValue val="0">Absence of SPI</FieldValue>
        <FieldValue val="1">Special Position Identification</FieldValue>
       </Field>
      </SubField>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/040, Measured Position in Polar Co-ordinates-->
  <DataItem id="040" rule="optional" frn="6">
    <DataItemName>Measured Position in Polar Co-ordinates.</DataItemName>
    <DataItemDefinition>Measured Position of a target in local polar co-
ordinates.</DataItemDefinition>
    <DataItemStructure format="fixed" length="4" desc="Four-Octet fixed length
data item.">
      <Field msb="32" lsb="17">
       <FieldShortName>RHO</FieldShortName>
       <FieldName>RHO</FieldName>
       <FieldUnit scale="1" max="65536">m</FieldUnit>
      </Field>
      <Field msb="16" lsb="1">
       <FieldShortName>Theta</FieldShortName>
       <FieldName>Theta</FieldName>
       <FieldUnit scale="0.0054931640625">deg</FieldUnit>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/041, Position in WGS-84 Co-ordinates-->
  <DataItem id="041" rule="mandatory" frn="5">
    <DataItemName>Position in WGS-84 Coordinates</DataItemName>
    <DataItemDefinition>Position of a target in WGS-84
Coordinates.</DataItemDefinition>
    <DataItemStructure format="fixed" length="8" desc="Eight-octet fixed length
Data Item">
```

```
      <Field msb="64" lsb="33" encode="int">
        <FieldShortName>Lat</FieldShortName>
        <FieldName>Latitude in WGS-84 in two's complement.</FieldName>
        <FieldUnit scale="0.000000083819031715393066406 25" min="-90"
max="90">deg</FieldUnit>
      </Field>
      <Field msb="32" lsb="1" encode="int">
        <FieldShortName>Lon</FieldShortName>
        <FieldName>Longitude in WGS-84 in two's complement.</FieldName>
        <FieldUnit scale="0.000000083819031715393066406 25" min="-180"
max="180">deg</FieldUnit>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/042, Position in Cartesian Co-ordinates-->
  <DataItem id="042" rule="optional" frn="7">
    <DataItemName>Position in Cartesian Coordinates</DataItemName>
    <DataItemDefinition>Position of target in Cartesian Coordinates, in two's
complement form.</DataItemDefinition>
    <DataItemStructure format="fixed" length="4" desc="Four-octet fixed length
Data Item.">
      <Field msb="32" lsb="17" encode="int">
        <FieldShortName>X</FieldShortName>
        <FieldName>X</FieldName>
        <FieldUnit scale="1" min="-32768" max="32768">m</FieldUnit>
      </Field>
      <Field msb="16" lsb="1" encode="int">
        <FieldShortName>Y</FieldShortName>
        <FieldName>Y</FieldName>
        <FieldUnit scale="1" min="-32768" max="32768">m</FieldUnit>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/060, Mode-3/A Code in Octal Representation-->
  <DataItem id="060" rule="optional" frn="12">
    <DataItemName>Mode-3/A Code in Octal Representation</DataItemName>
    <DataItemDefinition>Mode-3/A code converted into octal
representation.</DataItemDefinition>
    <DataItemStructure format="fixed" length="2" desc="Two-octet fixed length
Data Item.">
      <Field msb="16" lsb="16">
        <FieldShortName>V</FieldShortName>
        <FieldName>V</FieldName>
        <FieldValue val="0">Code validated</FieldValue>
        <FieldValue val="1">Code not validated</FieldValue>
      </Field>
      <Field msb="15" lsb="15">
        <FieldShortName>G</FieldShortName>
        <FieldName>G</FieldName>
        <FieldValue val="0">Default</FieldValue>
```

```
      <FieldValue val="1">Garbled code</FieldValue>
    </Field>
    <Field msb="14" lsb="14">
      <FieldShortName>L</FieldShortName>
      <FieldName>L</FieldName>
      <FieldValue val="0">Mode-2 code derived from the reply of the
transponder</FieldValue>
      <FieldValue val="1">Smoothed Mode-2 code as provided by a local
tracker n</FieldValue>
    </Field>
    <Field msb="12" lsb="1" encode="octal">
      <FieldShortName>Mod3A</FieldShortName>
      <FieldName>Mode-3/A reply in octal representation</FieldName>
    </Field>
  </DataItemStructure>
 </DataItem>
 <!--Data Item I010/090, Flight Level in Binary Representation-->
 <DataItem id="090" rule="optional" frn="17">
  <DataItemName>Flight Level in Binary Representation</DataItemName>
  <DataItemDefinition>Flight Level (Mode S Altitude) converted into binary
two's complement representation.</DataItemDefinition>
  <DataItemStructure format="fixed" length="2" desc="Two-octet fixed length
Data Item.">
    <Field msb="16" lsb="16">
      <FieldShortName>V</FieldShortName>
      <FieldValue val="0">Code validated</FieldValue>
      <FieldValue val="1">Code not validated</FieldValue>
    </Field>
    <Field msb="15" lsb="15">
      <FieldShortName>G</FieldShortName>
      <FieldValue val="0">Default</FieldValue>
      <FieldValue val="1">Garbled code</FieldValue>
    </Field>
    <Field msb="14" lsb="1" encode="int">
      <FieldShortName>FL</FieldShortName>
      <FieldName>Flight Level</FieldName>
      <FieldUnit scale="0.25">FL</FieldUnit>
    </Field>
  </DataItemStructure>
 </DataItem>
 <!--Data Item I010/091, Measured Height-->
 <DataItem id="091" rule="optional" frn="18">
  <DataItemName>Measured Height (Local Cartesian
Coordinates)</DataItemName>
  <DataItemDefinition>Height above local 2D co-ordinate system in reference
to the MLT System Reference Point as defined in item I019/610, in two’s
complement form, based on a direct measurement not related to barometric
pressure.</DataItemDefinition>
  <DataItemStructure format="fixed" length="2" desc="Two-octet fixed length
Data Item.">
```

```
      <Field msb="16" lsb="1" encode="int">
        <FieldShortName>MHeight</FieldShortName>
        <FieldName>Measured Height</FieldName>
        <FieldUnit scale="6.25" min="-204800" max="204800">ft</FieldUnit>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/131, Amplitude of Primary Plot-->
  <DataItem id="131" rule="optional" frn="24">
    <DataItemName>Signal Amplitude</DataItemName>
    <DataItemDefinition>Relative strength of received
signal.</DataItemDefinition>
    <DataItemStructure format="fixed" length="1" desc="One-Octet fixed length
data item.">
      <Field msb="8" lsb="1">
        <!--encode="uint"-->
        <FieldShortName>PAM</FieldShortName>
        <FieldName>Primary Plot Amplitude</FieldName>
        <!-- Revisar -->
        <FieldUnit scale="1" min="0" max="255">N/U</FieldUnit>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/140, Time of Day-->
  <DataItem id="140" rule="mandatory" frn="4">
    <DataItemName>Time of Day</DataItemName>
    <DataItemDefinition>Absolute time stamping expressed as
UTC.</DataItemDefinition>
    <DataItemStructure format="fixed" length="3" desc="Three-octet fixed length
Data Item.">
      <Field msb="24" lsb="1">
        <!--encode="uint"-->
        <FieldShortName>ToD</FieldShortName>
        <FieldName>Time of Day</FieldName>
        <FieldUnit scale="0.0078125" max="86400">s</FieldUnit>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/161, Track Number-->
  <DataItem id="161" rule="optional" frn="10">
    <DataItemName>Track Number</DataItemName>
    <DataItemDefinition>An integer value representing a unique reference to a
track record within a particular track file.</DataItemDefinition>
    <DataItemStructure format="fixed" length="2" desc="Two-octet fixed length
Data Item.">
      <Field msb="12" lsb="1">
        <!--encode="uint"-->
        <FieldShortName>TrkNb</FieldShortName>
        <FieldName>Track number</FieldName>
        <Range max="4095"></Range>
```

```
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/170, Track Status -->
  <DataItem id="170" rule="optional" frn="11">
    <DataItemName>Track Status</DataItemName>
    <DataItemDefinition>Status of track.</DataItemDefinition>
    <DataItemStructure format="variable" desc="Variable length Data Item
comprising a first part of one-octet, followed by one-octet extents as
necessary.">
      <SubField format="fixed" length="1">
        <Field msb="8" lsb="8">
          <FieldShortName>CNF</FieldShortName>
          <FieldValue val="0">Confirmed track</FieldValue>
          <FieldValue val="1">Track in initiation phase</FieldValue>
        </Field>
        <Field msb="7" lsb="7">
          <FieldShortName>TRE</FieldShortName>
          <FieldValue val="0">Default</FieldValue>
          <FieldValue val="1">Last report for a track</FieldValue>
        </Field>
        <Field msb="6" lsb="5">
          <FieldShortName>CST</FieldShortName>
          <FieldValue val="0">No extrapolation</FieldValue>
          <FieldValue val="1">Predictable extrapolation due to sensor refresh
period</FieldValue>
          <FieldValue val="2">Predictable extrapolation in masked
area</FieldValue>
          <FieldValue val="3">Extrapoltion due to unpredictable absence of
detection</FieldValue>
        </Field>
        <Field msb="4" lsb="4">
          <FieldShortName>MAH</FieldShortName>
          <FieldValue val="0">Default</FieldValue>
          <FieldValue val="1">Horizontal manoeuvre</FieldValue>
        </Field>
        <Field msb="3" lsb="3">
          <FieldShortName>TCC</FieldShortName>
          <FieldValue val="0">Tracking performed in 'Sensore Plane', i.e. neither
slant range correction nor projection was applied.</FieldValue>
          <FieldValue val="1">Slant range correction and a suitable projection
technique are used to track in a 2D. referance plane, tangential to the earth
model at the sensor Site co-ordinates. </FieldValue>
        </Field>
        <Field msb="2" lsb="2">
          <FieldShortName>STH</FieldShortName>
          <FieldValue val="0">Measured position</FieldValue>
          <FieldValue val="1">Smoothed position</FieldValue>
        </Field>
      </SubField>
```

```
      <SubField format="fixed" length="1">
       <Field msb="8" lsb="7">
        <FieldShortName>TOM</FieldShortName>
        <FieldValue val="0">Unknown type of movement</FieldValue>
        <FieldValue val="1">Taking-off</FieldValue>
        <FieldValue val="2">Landing</FieldValue>
        <FieldValue val="3">Other type of movements</FieldValue>
       </Field>
       <Field msb="6" lsb="4">
        <FieldShortName>DOU</FieldShortName>
        <FieldValue val="0">No doubt</FieldValue>
        <FieldValue val="1">Doubtful correlation (Undetermined
reason)</FieldValue>
        <FieldValue val="2">Doubtful correlation in clutter</FieldValue>
        <FieldValue val="3">Loss of accuracy</FieldValue>
        <FieldValue val="4">Loss of accuracy in clutter</FieldValue>
        <FieldValue val="5">Unstable track</FieldValue>
        <FieldValue val="6">Previously coasted</FieldValue>
        <Range max="6"></Range>
       </Field>
       <Field msb="3" lsb="2">
        <FieldShortName>MRS</FieldShortName>
        <FieldValue val="0">Nerge or split indication undetermined</FieldValue>
        <FieldValue val="1">Track merged by association to plot</FieldValue>
        <FieldValue val="2">Track merged by no-association to
plot</FieldValue>
        <FieldValue val="3">Split track</FieldValue>
       </Field>
      </SubField>
      <SubField format="fixed" length="1">
       <Field msb="8" lsb="8">
        <FieldShortName>GHO</FieldShortName>
        <FieldValue val="0">Default</FieldValue>
        <FieldValue val="1">Ghost track</FieldValue>
       </Field>
      </SubField>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/200, Calculated Track Velocity in Polar Co-ordinates-->
  <DataItem id="200" rule="optional" frn="8">
    <DataItemName>Calculated Track Velocity in Polar Co-
ordinates</DataItemName>
    <DataItemDefinition>Calculated track velocity expressed in polar co-
ordinates.</DataItemDefinition>
    <DataItemStructure format="fixed" length="4" desc="Four-Octet fixed length
data item.">
      <Field msb="32" lsb="17">
       <FieldShortName>GS</FieldShortName>
       <FieldName>Ground Speed</FieldName>
       <FieldUnit scale="0.00006103515625" max="2">NM/s</FieldUnit>
```

```
      </Field>
      <Field msb="16" lsb="1">
        <FieldShortName>TA</FieldShortName>
        <FieldName>Track Angle clockwise reference to True North</FieldName>
        <FieldUnit scale="0.0054931640625">deg</FieldUnit>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/202, Calculated Track Velocity in Cartesian Co-ordinates-->
  <DataItem id="202" rule="optional" frn="9">
    <DataItemName>Calculated Track Velocity in Cartesian
Coordinates</DataItemName>
    <DataItemDefinition>Calculated track velocity ex
    ed in Cartesian Coordinates, in two's complement
representation.</DataItemDefinition>
    <DataItemStructure format="fixed" length="4" desc="Four-octet fixed length
Data Item.">
      <Field msb="32" lsb="17" encode="int">
        <FieldShortName>Vx</FieldShortName>
        <FieldName>Vx</FieldName>
        <FieldUnit scale="0.25" min="-8192" max="8192">m/s</FieldUnit>
      </Field>
      <Field msb="16" lsb="1" encode="int">
        <FieldShortName>Vy</FieldShortName>
        <FieldName>Vy</FieldName>
        <FieldUnit scale="0.25" min="-8192" max="8192">m/s</FieldUnit>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/210, Calculated Acceleration-->
  <DataItem id="210" rule="optional" frn="25">
    <DataItemName>Calculated Acceleration</DataItemName>
    <DataItemDefinition>Calculated Acceleration of the target, in two's
complement form.</DataItemDefinition>
    <DataItemStructure format="fixed" length="2" desc="Two-Octet fixed length
data item.">
      <Field msb="16" lsb="9" encode="int">
        <FieldShortName>Ax</FieldShortName>
        <FieldName>Ax</FieldName>
        <FieldUnit scale="0.25" min="-31" max="31">m/s²</FieldUnit>
      </Field>
      <Field msb="8" lsb="1" encode="int">
        <FieldShortName>Ay</FieldShortName>
        <FieldName>Ay</FieldName>
        <FieldUnit scale="0.25" min="-31" max="31">m/s²</FieldUnit>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/220, Target Address-->
  <DataItem id="220" rule="optional" frn="13">
```

```
    <DataItemName>Target Address</DataItemName>
    <DataItemDefinition>Target address (ICAO 24-bit address) assigned
uniquely to each Target.</DataItemDefinition>
    <DataItemStructure format="fixed" length="3" desc="Three-octet fixed length
Data Item.">
      <Field msb="24" lsb="1" encode="hex">
        <FieldShortName>TAddr</FieldShortName>
        <FieldName>24-bits Target Address</FieldName>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/245, Target Identification-->
  <DataItem id="245" rule="optional" frn="14">
    <DataItemName>Target Identification</DataItemName>
    <DataItemDefinition>Target (aircraft or vehicle) identification in 8
characters.</DataItemDefinition>
    <DataItemStructure format="fixed" length="7" desc="Seven-octet fixed length
Data Item.">
      <Field msb="56" lsb="55">
        <FieldShortName>STI</FieldShortName>
        <FieldName>STI</FieldName>
        <FieldValue val="0">Callsign or registration downlinked from
transponder</FieldValue>
        <FieldValue val="1">Callsign not downlinked from
transponder</FieldValue>
        <FieldValue val="2">Registration not downlinked from
transponder</FieldValue>
        <Range max="2"></Range>
      </Field>
      <Field msb="48" lsb="1" encode="sixbitschar">
        <FieldShortName>TId</FieldShortName>
        <FieldName>Characters 1-8 (coded on 6 bits each) defining target
identification</FieldName>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/250, Mode S MB Data-->
  <DataItem id="250" rule="optional" frn="15">
    <DataItemName>Mode S MB Data</DataItemName>
    <DataItemDefinition>Mode S Comm B data as extracted from the aircraft
transponder.</DataItemDefinition>
    <DataItemStructure format="repetitive" length="8" desc="Repetitive Data
Item starting with a one-octet Field Repetition Indicator (REP) followed by at
least one BDS report comprising one seven octet BDS register and one octet
BDS code.">
      <Field msb="64" lsb="9" encode="hex">
        <FieldShortName>MBData</FieldShortName>
        <FieldName>Mode S Comm B message data</FieldName>
      </Field>
      <Field msb="8" lsb="5">
```

```
      <FieldShortName>BDS1</FieldShortName>
      <FieldName>Comm B Data Buffer Store 1 Address</FieldName>
    </Field>
    <Field msb="4" lsb="1">
      <FieldShortName>BDS2</FieldShortName>
      <FieldName>Comm B Data Buffer Store 2 Address</FieldName>
    </Field>
  </DataItemStructure>
 </DataItem>
 <!--Data Item I010/270, Target Size & Orientation-->
 <DataItem id="270" rule="optional" frn="19">
  <DataItemName>Target Size and Orientation</DataItemName>
  <DataItemDefinition>Target size defined as length and with of the detected
target, and orientation.</DataItemDefinition>
  <DataItemStructure format="variable" desc="Variable Data Item, comprising
a first part of one octet, followed by one-octed as necessary.">
    <SubField name="First Part" format="fixed" length="1">
      <Field msb="8" lsb="2">
        <FieldShortName>Length</FieldShortName>
        <FieldName>Length</FieldName>
        <FieldUnit scale="1">m</FieldUnit>
      </Field>
    </SubField>
    <SubField name="First Extent" format="fixed" length="1">
      <Field msb="8" lsb="2">
        <FieldShortName>Ori</FieldShortName>
        <FieldName>Orientation</FieldName>
        <FieldUnit scale="2.81">deg.</FieldUnit>
      </Field>
    </SubField>
    <SubField  name="Second Extent" format="fixed" length="1">
      <Field msb="8" lsb="2">
        <FieldShortName>Width</FieldShortName>
        <FieldName>Width</FieldName>
        <FieldUnit scale="1">m</FieldUnit>
      </Field>
    </SubField>
  </DataItemStructure>
 </DataItem>
 <!--Data Item I010/280, Presence-->
 <DataItem id="280" rule="optional" frn="23">
  <DataItemName>Presence</DataItemName>
  <DataItemDefinition>Positions of all elementary presences constituting a
plot.</DataItemDefinition>
  <DataItemStructure format="repetitive" length="2" desc="Repetitive data
length item">
    <Field msb="16" lsb="9" encode="int">
      <FieldShortName>DRHO</FieldShortName>
      <FieldName>DRHO</FieldName>
      <FieldUnit scale="1" min="-127" max="127">m</FieldUnit>
```

```
      </Field>
      <Field msb="8" lsb="1" encode="int">
        <FieldShortName>DTHETA</FieldShortName>
        <FieldName>DTHETA</FieldName>
        <FieldUnit scale="0.15" min="-19.05" max="19.05">deg</FieldUnit>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/300, Vehicle Fleet Identification-->
  <DataItem id="300" rule="optional" frn="16">
    <DataItemName>Vehicle Fleet Identification</DataItemName>
    <DataItemDefinition>Vehicle fleet identification number.</DataItemDefinition>
    <DataItemStructure format="fixed" length="1" desc="One octet fixed length
Data Item.">
      <Field msb="8" lsb="1">
        <FieldShortName>VFI</FieldShortName>
        <FieldName>Vehicle Fleet Identification</FieldName>
        <FieldValue val="0">Unknown</FieldValue>
        <FieldValue val="1">ATC equipment maintenance</FieldValue>
        <FieldValue val="2">Airport maintenance</FieldValue>
        <FieldValue val="3">Fire</FieldValue>
        <FieldValue val="4">Bird scarer</FieldValue>
        <FieldValue val="5">Snow plough</FieldValue>
        <FieldValue val="6">Runway sweeper</FieldValue>
        <FieldValue val="7">Emergency</FieldValue>
        <FieldValue val="8">Police</FieldValue>
        <FieldValue val="9">Bus</FieldValue>
        <FieldValue val="10">Tug (push/tow)</FieldValue>
        <FieldValue val="11">Grass cutter</FieldValue>
        <FieldValue val="12">Fuel</FieldValue>
        <FieldValue val="13">Baggage</FieldValue>
        <FieldValue val="14">Catering</FieldValue>
        <FieldValue val="15">Aircraft maintenance</FieldValue>
        <FieldValue val="16">Flyco (follow me)</FieldValue>
        <Range max="16"></Range>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/310, Pre-programmed Message-->
  <DataItem id="310" rule="optional" frn="21">
    <DataItemName>Pre-programmed Message</DataItemName>
    <DataItemDefinition>Number related to a pre-programmed message that can
be transmitted by a vehicle.</DataItemDefinition>
    <DataItemStructure format="fixed" length="1" desc="One octet fixed length
Data Item.">
      <Field msb="8" lsb="8">
        <FieldShortName>TRB</FieldShortName>
        <FieldValue val="0">Default</FieldValue>
        <FieldValue val="1">In Trouble</FieldValue>
      </Field>
```

```
      <Field msb="7" lsb="1">
        <FieldShortName>MSG</FieldShortName>
        <FieldValue val="1">Towing aircraft</FieldValue>
        <FieldValue val="2">"Follow me" operation</FieldValue>
        <FieldValue val="3">Runway check</FieldValue>
        <FieldValue val="4">Emergency operation (fire, medical...)</FieldValue>
        <FieldValue val="5">Work in progress (maintenance, birds scarer,
sweepers...)</FieldValue>
        <Range min="1" max="5"></Range>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/500, Standard Deviation of Position-->
  <DataItem id="500" rule="optional" frn="22">
    <DataItemName>Standard Deviation of POsition</DataItemName>
    <DataItemDefinition>Standard Deviation of Position</DataItemDefinition>
    <DataItemStructure format="fixed" length="4" desc="Four octed fixed length
Data Item.">
      <Field msb="32" lsb="25">
        <FieldShortName>Qx</FieldShortName>
        <FieldName>Standard deviation X component</FieldName>
        <FieldUnit scale="0.25"/>
      </Field>
      <Field msb="24" lsb="17">
        <FieldShortName>Qy</FieldShortName>
        <FieldName>Standard deviation Y component</FieldName>
        <FieldUnit scale="0.25"/>
      </Field>
      <Field msb="16" lsb="1" encode="int">
        <FieldShortName>Qxy</FieldShortName>
        <FieldName>Q(X,Y)</FieldName>
        <FieldUnit scale="0.25"/>
      </Field>
    </DataItemStructure>
  </DataItem>
  <!--Data Item I010/550, System Status-->
  <DataItem id="550" rule="optional" frn="20">
    <DataItemName>System Status</DataItemName>
    <DataItemDefinition>Iformation concerning the configuration and status of a
System.</DataItemDefinition>
    <DataItemStructure format="fixed" length="1" desc="One-octet fixed length
Data Item.">
      <Field msb="8" lsb="7">
        <FieldShortName>NOGO</FieldShortName>
        <FieldName>Operational Release Status of System</FieldName>
        <FieldValue val="0">Operational</FieldValue>
        <FieldValue val="1">Degraded</FieldValue>
        <FieldValue val="2">NOGO</FieldValue>
        <Range max="2"></Range>
      </Field>
```

```
      <Field msb="6" lsb="6">
        <FieldShortName>OVL</FieldShortName>
        <FieldName>Overload indicator</FieldName>
        <FieldValue val="0">NO Overload</FieldValue>
        <FieldValue val="1">Overload</FieldValue>
      </Field>
      <Field msb="5" lsb="5">
        <FieldShortName>TSV</FieldShortName>
        <FieldName>Time Source Validity</FieldName>
        <FieldValue val="0">valid</FieldValue>
        <FieldValue val="1">invalid</FieldValue>
      </Field>
      <Field msb="4" lsb="4">
        <FieldShortName>DIV</FieldShortName>
        <FieldName>Diversity indicator</FieldName>
        <FieldValue val="0">Normal Operation</FieldValue>
        <FieldValue val="1">Diversity degraded</FieldValue>
      </Field>
      <Field msb="3" lsb="3">
        <FieldShortName>TTF</FieldShortName>
        <FieldName>Test Target indicator</FieldName>
        <FieldValue val="0">Test Target Operative</FieldValue>
        <FieldValue val="1">Test Target Failure</FieldValue>
      </Field>
    </DataItemStructure>
  </DataItem>
</Category>
```

asterix_cat021_2_1.xml (Especificació categoría 21 v2.1)

```
<?xml version="1.0" encoding="UTF-8"?>
<Category id="021" name="Surveillance Data Exchange - Part 12 ADS-B
Reports" ver="2.1">
  <!--Data Item I021/008, Aircraft Operational Status-->
    <DataItem id="008" rule="optional" frn="36">
      <DataItemName>Aircraft Operational Status</DataItemName>
      <DataItemDefinition>Identification of the operational services available in
the aircraft while airborne.</DataItemDefinition>
      <DataItemStructure format="fixed" length ="1" desc="One-octet fixed
length Data Item.">
        <Field msb="8" lsb="8">
          <FieldShortName>RA</FieldShortName>
          <FieldName>TCAS Resolution Advisory active</FieldName>
          <FieldValue val="0">TCAS II or ACAS RA not active</FieldValue>
          <FieldValue val="1">TCAS RA active</FieldValue>
        </Field>
        <Field msb="7" lsb="6">
          <FieldShortName>TC</FieldShortName>
          <FieldName>Target Change Report Capability</FieldName>
          <FieldValue val="0">no capability for Trajectory Change
Reports</FieldValue>
          <FieldValue val="1">support for TC+0 reports only</FieldValue>
          <FieldValue val="2">support for multiple TC reports</FieldValue>
          <FieldValue val="3">reserved</FieldValue>
        </Field>
        <Field msb="5" lsb="5">
          <FieldShortName>TS</FieldShortName>
          <FieldName>Target State Report Capability</FieldName>
          <FieldValue val="0">no capability to support Target State
Reports</FieldValue>
          <FieldValue val="1">capable of supporting target State
Reports</FieldValue>
        </Field>
        <Field msb="4" lsb="4">
          <FieldShortName>ARV</FieldShortName>
          <FieldName>Air-Referenced Velocity Report Capability</FieldName>
          <FieldValue val="0">no capability to generate ARV-
reports</FieldValue>
          <FieldValue val="1">capable of generate ARV-reports</FieldValue>
        </Field>
        <Field msb="3" lsb="3">
          <FieldShortName>CDTI_A</FieldShortName>
          <FieldName>Cockpit Display of Traffic Information
airborne</FieldName>
          <FieldValue val="0">CDTI not operational</FieldValue>
          <FieldValue val="1">CDTI operational</FieldValue>
        </Field>
        <Field msb="2" lsb="2">
```

```
            <FieldShortName>not_TCAS</FieldShortName>
            <FieldName>TCAS System Status</FieldName>
            <FieldValue val="0">TCAS operational or unknown</FieldValue>
            <FieldValue val="1">TCAS not installed or not
operational</FieldValue>
         </Field>
         <Field msb="1" lsb="1">
            <FieldShortName>SA</FieldShortName>
            <FieldName>Single Antenna</FieldName>
            <FieldValue val="0">Antenna Diversity</FieldValue>
            <FieldValue val="1">Single Antenna Only</FieldValue>
         </Field>
      </DataItemStructure>
   </DataItem>
  <!--Data Item I021/010, Data Source Identification-->
   <DataItem id="010" rule="mandatory" frn="1">
      <DataItemName>Data Source Identification</DataItemName>
      <DataItemDefinition>Identification of the ADS-B station providing
information.</DataItemDefinition>
      <DataItemStructure format="fixed" length="2" desc="Two-octet fixed length
Data Item.">
         <Field msb="16" lsb="9">
            <FieldShortName>SAC</FieldShortName>
            <FieldName>System Area Code</FieldName>
         </Field>
         <Field msb="8" lsb="1">
            <FieldShortName>SIC</FieldShortName>
            <FieldName>System Identification Code</FieldName>
         </Field>
      </DataItemStructure>
   </DataItem>
  <!--Data Item I021/015, Service Identification-->
   <DataItem id="015" rule="optional" frn="4">
      <DataItemName>Service Identification</DataItemName>
      <DataItemDefinition>Identification of the service provided to one or more
users.</DataItemDefinition>
      <DataItemStructure format="fixed" length="1" desc="One-Octet fixed
length data item.">
         <Field msb="8" lsb="1">
            <FieldShortName>ServiceId</FieldShortName>
            <FieldName>Service Identification</FieldName>
         </Field>
      </DataItemStructure>
   </DataItem>
  <!--Data Item I021/016, Service Management-->
   <DataItem id="016" rule="optional" frn="35">
      <DataItemName>Service Management</DataItemName>
      <DataItemDefinition>Identification of services offered by a ground station
(identified by a SIC code).</DataItemDefinition>
```

```
      <DataItemStructure format="fixed" length="1" desc="One-octet fixed length
Data Item.">
         <Field msb="8" lsb="1">
            <FieldShortName>RP</FieldShortName>
            <FieldName>Report Period</FieldName>
            <FieldUnit scale="0.5" min="0" max="127.5">s</FieldUnit>
         </Field>
      </DataItemStructure>
      <DataItemNote>1. This item contains the same information as item
I023/101 in ASTERIX category 023 [Ref. 3]. Since not all service users receive
category 023 data, this information has to be conveyed in category 021 as well.
2. If this item is due to be sent according to the encoding rule above, it shall be
sent with the next target report</DataItemNote>
   </DataItem>
  <!--Data Item I021/020, Emitter Category-->
   <DataItem id="020" rule="optional" frn="30">
      <DataItemName>Emitter Category</DataItemName>
      <DataItemDefinition>Characteristics of the originating ADS-B
unit.</DataItemDefinition>
      <DataItemStructure format="fixed" length="1" desc="One-Octet fixed
length data item.">
         <Field msb="8" lsb="1">
            <FieldShortName>ECAT</FieldShortName>
            <FieldName>Emitter Category</FieldName>
            <FieldValue val="0">No ADS-B Emitter Category
Information</FieldValue>
            <FieldValue val="1">light aircraft &lt;= 15500 lbs</FieldValue>
            <FieldValue val="2">15500 lbs &lt; small aircraft &lt; 75000
lbs</FieldValue>
            <FieldValue val="3">75000 lbs &lt; medium a/c &lt; 300000
lbs</FieldValue>
            <FieldValue val="4">High Vortex Large</FieldValue>
            <FieldValue val="5">300000 lbs &lt;= heavy aircraft</FieldValue>
            <FieldValue val="6">highly manoeuvrable (5g acceleration capability)
and high speed (&gt;400 knots cruise)</FieldValue>
            <FieldValue val="7">reserved</FieldValue>
            <FieldValue val="8">reserved</FieldValue>
            <FieldValue val="9">reserved</FieldValue>
            <FieldValue val="10">rotocraft</FieldValue>
            <FieldValue val="11">glider / sailplane</FieldValue>
            <FieldValue val="12">lighter-than-air</FieldValue>
            <FieldValue val="13">unmanned aerial vehicle</FieldValue>
            <FieldValue val="14">space / transatmospheric vehicle</FieldValue>
            <FieldValue val="15">ultralight / handglider / paraglider</FieldValue>
            <FieldValue val="16">parachutist / skydiver</FieldValue>
            <FieldValue val="17">reserved</FieldValue>
            <FieldValue val="18">reserved</FieldValue>
            <FieldValue val="19">reserved</FieldValue>
            <FieldValue val="20">surface emergency vehicle</FieldValue>
            <FieldValue val="21">surface service vehicle</FieldValue>
```

```
            <FieldValue val="22">fixed ground or tethered
obstruction</FieldValue>
            <FieldValue val="23">cluster obstacle</FieldValue>
            <FieldValue val="24">line obstacle</FieldValue>
          </Field>
        </DataItemStructure>
    </DataItem>
  <!--Data Item I021/040, Target Report Descriptor-->
    <DataItem id="040" rule="mandatory" frn="2">
        <DataItemName>Target Report Descriptor</DataItemName>
        <DataItemDefinition>Type and characteristics of the data as transmitted by
a system.</DataItemDefinition>
        <DataItemStructure format="variable" desc="Variable Length Data Item,
comprising a primary subfield of one octet, followed by one-octet extensions as
necessary.">
          <SubField name="Primary Subfield" format="fixed" length="1">
            <Field msb="8" lsb="6">
              <FieldShortName>ATP</FieldShortName>
              <FieldName>Address Type</FieldName>
              <FieldValue val="0">24-Bit ICAO address</FieldValue>
              <FieldValue val="1">Duplicate address</FieldValue>
              <FieldValue val="2">Surface vehicle address</FieldValue>
              <FieldValue val="3">Anonymous address</FieldValue>
              <FieldValue val="4">Reserved for future use</FieldValue>
              <FieldValue val="5">Reserved for future use</FieldValue>
              <FieldValue val="6">Reserved for future use</FieldValue>
              <FieldValue val="7">Reserved for future use</FieldValue>
            </Field>
            <Field msb="5" lsb="4">
              <FieldShortName>ARC</FieldShortName>
              <FieldName>Altitude Reporting Capability</FieldName>
              <FieldValue val="0">25 ft</FieldValue>
              <FieldValue val="1">100 ft</FieldValue>
              <FieldValue val="2">Unknown</FieldValue>
              <FieldValue val="3">Invalid</FieldValue>
            </Field>
            <Field msb="3" lsb="3">
              <FieldShortName>RC</FieldShortName>
              <FieldName>Range Check</FieldName>
              <FieldValue val="0">Default</FieldValue>
              <FieldValue val="1">Range Check passed, CPR Validation
pending</FieldValue>
            </Field>
            <Field bit="2">
              <FieldShortName>RAB</FieldShortName>
              <FieldName>Report Type</FieldName>
              <FieldValue val="0">Report from target transponder</FieldValue>
              <FieldValue val="1">Report from field monitor (fixed
transponder)</FieldValue>
            </Field>
```

```
        </SubField>
        <SubField name="First Extension" format="fixed" length="1">
          <Field bit="8">
            <FieldShortName>DCR</FieldShortName>
            <FieldName>Differential Correction</FieldName>
            <FieldValue val="0">No differential correction (ADS-
B)</FieldValue>
            <FieldValue val="1">Differential correction (ADS-B)</FieldValue>
          </Field>
          <Field bit="7">
            <FieldShortName>GBS</FieldShortName>
            <FieldName>Ground Bit Setting</FieldName>
            <FieldValue val="0">Ground Bit not set</FieldValue>
            <FieldValue val="1">Ground Bit set</FieldValue>
          </Field>
          <Field bit="6">
            <FieldShortName>SIM</FieldShortName>
            <FieldName>Simulated Target</FieldName>
            <FieldValue val="0">Actual target report</FieldValue>
            <FieldValue val="1">Simulated target report</FieldValue>
          </Field>
          <Field bit="5">
            <FieldShortName>TST</FieldShortName>
            <FieldName>Test Target</FieldName>
            <FieldValue val="0">Default</FieldValue>
            <FieldValue val="1">Test Target</FieldValue>
          </Field>
          <Field bit="4">
            <FieldShortName>SAA</FieldShortName>
            <FieldName>Selected Altitude Available</FieldName>
            <FieldValue val="0">Equipment capable to provide Selected
Altitude</FieldValue>
            <FieldValue val="1">Equipment not capable to provide Selected
Altitude</FieldValue>
          </Field>
          <Field msb="3" lsb="2">
            <FieldShortName>CL</FieldShortName>
            <FieldName>Confidence Level</FieldName>
            <FieldValue val="0">Report valid</FieldValue>
            <FieldValue val="1">Report suspect</FieldValue>
            <FieldValue val="2">No information</FieldValue>
            <FieldValue val="3">Reserved for future use</FieldValue>
          </Field>
        </SubField>
        <SubField name="Second Extension: Error Conditions" format="fixed"
length="1">
          <Field bit="6">
            <FieldShortName>IPC</FieldShortName>
            <FieldName>Independent Position Check</FieldName>
            <FieldValue val="0">default</FieldValue>
```

```
            <FieldValue val="1">failed</FieldValue>
          </Field>
          <Field bit="5">
            <FieldShortName>NOGO</FieldShortName>
            <FieldName>No-go Bit Status</FieldName>
            <FieldValue val="0">NOGO-bit not set</FieldValue>
            <FieldValue val="1">NOGO-bit set</FieldValue>
          </Field>
          <Field bit="4">
            <FieldShortName>CPR</FieldShortName>
            <FieldName>Compact Position Reporting</FieldName>
            <FieldValue val="0">CPR Validation correct</FieldValue>
            <FieldValue val="1">CPR Validation failed</FieldValue>
          </Field>
          <Field bit="3">
            <FieldShortName>LDPJ</FieldShortName>
            <FieldName>Local Decoding Position Jump</FieldName>
            <FieldValue val="0">LDPJ not detected</FieldValue>
            <FieldValue val="1">LDPJ detected</FieldValue>
          </Field>
          <Field bit="2">
            <FieldShortName>RCF</FieldShortName>
            <FieldName>Range Check</FieldName>
            <FieldValue val="0">default</FieldValue>
            <FieldValue val="1">Range Check failed</FieldValue>
          </Field>
        </SubField>
      </DataItemStructure>
      <DataItemNote>1. Bit 3 indicates that the position reported by the target is
within a credible range from the ground station. The range check is followed by
the CPR validation to ensure that global and local position decoding both
indicate valid position information. Bit 3=1 indicates that the range check was
done, but the CPR validation is not yet completed. Once CPR validation is
completed, Bit 3 will be reset to 0. 2. The second extension signals the reasons
for which the report has been indicated as suspect (indication Confidence Level
(CL) in the first extension). Bit 2 indicates that the Range Check failed, i.e. the
target is reported outside the credible range for the Ground Station. For
operational users such a target will be suppressed. In services used for
monitoring the Ground Station, the target will be transmitted with bit 2 indicating
the fault condition. 3. Bit 5 represents the setting of the GO/NOGO-bit as
defined in item I023/100 of category 023 [Ref. 3]. 4. Bits 8/6 are reserved for
future use.</DataItemNote>
    </DataItem>
  <!--Data Item I021/070, Mode-3/A Code in Octal Representation-->
    <DataItem id="070" rule="optional" frn="19">
      <DataItemName>Mode 3/A Code in Octal
Representation</DataItemName>
      <DataItemDefinition>Mode-3/A code converted into octal
representation.</DataItemDefinition>
```

```
      <DataItemStructure format="fixed" length="2" desc="Two-octet fixed length
Data Item.">
        <Field msb="12" lsb="1" encode="octal">
          <FieldShortName>Mode3A</FieldShortName>
          <FieldName>Mode-3/A reply in octal</FieldName>
        </Field>
      </DataItemStructure>
   </DataItem>
<!--Data Item I021/071, Time of Applicability for Position-->
   <DataItem id="071" rule="optional" frn="5">
      <DataItemName>Time of Applicability for Position</DataItemName>
      <DataItemDefinition>Time of applicability of the reported position, in the
form of elapsed time since last midnight, expressed as
UTC.</DataItemDefinition>
      <DataItemStructure format="fixed" length="3" desc="Three-Octet fixed
length data item.">
        <Field msb="24" lsb="1">
          <FieldShortName>time_applicability_position</FieldShortName>
          <FieldName>Time of Applicability of Position</FieldName>
          <FieldUnit scale="0.0078125" max="86400">s</FieldUnit>
        </Field>
      </DataItemStructure>
      <DataItemNote>1. The time of applicability value is reset to zero at every
midnight. 2. The time of applicability indicates the exact time at which the
position transmitted in the target report is valid.</DataItemNote>
   </DataItem>
<!--Data Item I021/072, Time of Applicability for Velocity-->
   <DataItem id="072" rule="optional" frn="8">
      <DataItemName>Time of Applicability for Velocity</DataItemName>
      <DataItemDefinition>Time of applicability (measurement) of the reported
velocity, in the form of elapsed time since last midnight, expressed as
UTC.</DataItemDefinition>
      <DataItemStructure format="fixed" length="3" desc="Three-Octet fixed
length data item.">
        <Field msb="24" lsb="1">
          <FieldShortName>time_applicability_velocity</FieldShortName>
          <FieldName>Time of Applicability of Velocity</FieldName>
          <FieldUnit scale="0.0078125" max="86400">s</FieldUnit>
        </Field>
      </DataItemStructure>
      <DataItemNote>1. The time of applicability value is reset to zero at every
midnight. 2. The time of applicability indicates the exact time at which the
velocity information transmitted in the target report is valid. 3. This item will not
be available in some ADS-B technologies.</DataItemNote>
   </DataItem>
<!--Data Item I021/073, Time of Message Reception for Position-->
   <DataItem id="073" rule="optional" frn="12">
      <DataItemName>Time of Message Reception for
Position</DataItemName>
```

```
      <DataItemDefinition>Time of reception of the latest position squitter in the
Ground Station, in the form of elapsed time since last midnight, expressed as
UTC.</DataItemDefinition>
      <DataItemStructure format="fixed" length="3" desc="Three-Octet fixed
length data item.">
         <Field msb="24" lsb="1">
            <FieldShortName>time_reception_position</FieldShortName>
            <FieldName>Time of Message Reception of Position</FieldName>
            <FieldUnit scale="0.0078125" max="86400">s</FieldUnit>
         </Field>
      </DataItemStructure>
      <DataItemNote>The time of message reception value is reset to zero at
every midnight.</DataItemNote>
   </DataItem>
<!--Data Item I021/074, Time of Message Reception for Position-High Precision-
->
   <DataItem id="074" rule="optional" frn="13">
      <DataItemName>Time of Message Reception of Position-High
Precision</DataItemName>
      <DataItemDefinition>Time at which the latest ADS-B position information
was received by the ground station, expressed as fraction of the second of the
UTC Time.</DataItemDefinition>
      <DataItemStructure format="fixed" length="4" desc="Four-Octet fixed
length data item.">
         <Field msb="32" lsb="31">
            <FieldShortName>FSI</FieldShortName>
            <FieldName>Full Second Indication</FieldName>
            <FieldValue val="0">TOMRp whole seconds = (I021/073) Whole
seconds</FieldValue>
            <FieldValue val="1">TOMRp whole seconds = (I021/073) Whole
seconds + 1</FieldValue>
            <FieldValue val="2">TOMRp whole seconds = (I021/073) Whole
seconds - 1</FieldValue>
            <FieldValue val="3">Reserved</FieldValue>
         </Field>
         <Field msb="30" lsb="1">

<FieldShortName>time_reception_position_highprecision</FieldShortName>
            <FieldName>Fractional part of the time of message reception for
position in the ground station.</FieldName>
            <FieldUnit scale="0.931322574615478515625">ns</FieldUnit>
         </Field>
      </DataItemStructure>
   </DataItem>
<!--Data Item I021/075, Time of Message Reception for Velocity-->
   <DataItem id="075" rule="optional" frn="14">
      <DataItemName>Time of Message Reception for
Velocity</DataItemName>
```

```
      <DataItemDefinition>Time of reception of the latest velocity squitter in the
Ground Station, in the form of elapsed time since last midnight, expressed as
UTC.</DataItemDefinition>
      <DataItemStructure format="fixed" length="3" desc="Three-Octet fixed
length data item.">
         <Field msb="24" lsb="1">
            <FieldShortName>time_reception_velocity</FieldShortName>
            <FieldName>Time of Message Reception of Velocity</FieldName>
            <FieldUnit scale="0.0078125" max="86400">s</FieldUnit>
         </Field>
      </DataItemStructure>
      <DataItemNote>The time of message reception value is reset to zero at
every midnight.</DataItemNote>
   </DataItem>
<!--Data Item I021/076, Time of Message Reception for Velocity-High Precision-
->
   <DataItem id="076" rule="optional" frn="15">
      <DataItemName>Time of Message Reception of Velocity-High
Precision</DataItemName>
      <DataItemDefinition>Time at which the latest ADS-B velocity information
was received by the ground station, expressed as fraction of the second of the
UTC Time.</DataItemDefinition>
      <DataItemStructure format="fixed" length="4" desc="Four-Octet fixed
length data item.">
         <Field msb="32" lsb="31">
            <FieldShortName>FSI</FieldShortName>
            <FieldName>Full Second Indication</FieldName>
            <FieldValue val="0">TOMRv whole seconds = (I021/073) Whole
seconds</FieldValue>
            <FieldValue val="1">TOMRv whole seconds = (I021/073) Whole
seconds + 1</FieldValue>
            <FieldValue val="2">TOMRv whole seconds = (I021/073) Whole
seconds - 1</FieldValue>
            <FieldValue val="3">Reserved</FieldValue>
         </Field>
         <Field msb="30" lsb="1">

<FieldShortName>time_reception_velocity_highprecision</FieldShortName>
            <FieldName>Fractional part of the time of message reception for
velocity in the ground station.</FieldName>
            <FieldUnit scale="0.9313225746154785">ns</FieldUnit>
         </Field>
      </DataItemStructure>
   </DataItem>
<!--Data Item I021/077, Time of ASTERIX Report Transmission-->
   <DataItem id="077" rule="optional" frn="28">
      <DataItemName>Time of ASTERIX Report
Transmission</DataItemName>
```

```
      <DataItemDefinition>Time of the transmission of the ASTERIX category
021 report in the form of elapsed time since last midnight, expressed as
UTC.</DataItemDefinition>
      <DataItemStructure format="fixed" length="3" desc="Three-Octet fixed
length data item.">
         <Field msb="24" lsb="1">
            <FieldShortName>time_report_transmission</FieldShortName>
            <FieldName>Time of ASTERIX Report Transmission</FieldName>
            <FieldUnit scale="0.0078125" max="86400">s</FieldUnit>
         </Field>
      </DataItemStructure>
      <DataItemNote>The time of ASTERIX report transmission value is reset to
zero at every midnight.</DataItemNote>
   </DataItem>
<!--Data Item I021/080, Target Address-->
   <DataItem id="080" rule="mandatory" frn="11">
      <DataItemName>Target Address</DataItemName>
      <DataItemDefinition>Target address (emitter identifier) asint uniquely to
each target.</DataItemDefinition>
      <DataItemStructure format="fixed" length="3" desc="Three-octet fixed
length Data Item.">
         <Field msb="24" lsb="1" encode="hex">
            <FieldShortName>TAddr</FieldShortName>
            <FieldName>Target Address</FieldName>
         </Field>
      </DataItemStructure>
   </DataItem>
<!--Data Item I021/090, Quality Indicators-->
   <DataItem id="090" rule="mandatory" frn="17">
      <DataItemName>Quality Indicators</DataItemName>
      <DataItemDefinition>ADS-B quality indicators transmitted by a/c according
to MOPS version.</DataItemDefinition>
      <DataItemStructure format="variable" desc="Variable Length Data Item,
comprising a primary subfield of one-octet, followed by one-octet extents as
necessary.">
         <SubField name="Primary Subfield" format="fixed" length="1">
            <Field msb="8" lsb="6">
               <FieldShortName>NUCr_or_NACv</FieldShortName>
               <FieldName>Navigation Uncertainty Category for velocity or
Navigation Accuracy Category for Velocity</FieldName>
            </Field>
            <Field msb="5" lsb="2">
               <FieldShortName>NUCp_or_NIC</FieldShortName>
               <FieldName>Navigation Uncertainty Category for Position NUCp or
Navigation Integrity Category NIC.</FieldName>
            </Field>
         </SubField>
         <SubField name="First extension" format="fixed" length="1">
            <Field bit="8">
               <FieldShortName>NICbaro</FieldShortName>
```

```
                <FieldName>Navigation Integrity Category for Barometric
Altitude</FieldName>
            </Field>
            <Field msb="7" lsb="6">
               <FieldShortName>SIL</FieldShortName>
               <FieldName>Surveillance Integrity Level</FieldName>
            </Field>
            <Field msb="5" lsb="2">
               <FieldShortName>NACp</FieldShortName>
               <FieldName>Navigation Accuracy Category for
Position</FieldName>
            </Field>
         </SubField>
         <SubField name="Second extension" format="fixed" length="1">
            <Field bit="6">
               <FieldShortName>SIL_supplement</FieldShortName>
               <FieldName>Surveillance Integrity Level Supplement</FieldName>
               <FieldValue val="0">measured per flight-hour</FieldValue>
               <FieldValue val="1">measured per sample</FieldValue>
            </Field>
            <Field msb="5" lsb="4">
               <FieldShortName>SDA</FieldShortName>
               <FieldName>Horizontal Position System Design Assurance Level
(as defined in version 2)</FieldName>
            </Field>
            <Field msb="3" lsb="2">
               <FieldShortName>GVA</FieldShortName>
               <FieldName>Geometric Altitude Accuracy</FieldName>
            </Field>
         </SubField>
         <SubField  name="Third extension" format="fixed" length="1">
            <Field msb="8" lsb="5">
               <FieldShortName>PIC</FieldShortName>
               <FieldName>Position Integrity Category</FieldName>
            </Field>
         </SubField>
      </DataItemStructure>
      <DataItemNote>"Version 2" refers to the MOPS version as defined in data
item I021/210, bits 6/4</DataItemNote>
   </DataItem>
<!--Data Item I021/110, Trajectory Intent-->
   <DataItem id="110" rule="optional" frn="34">
      <DataItemName>Trajectory Intent</DataItemName>
      <DataItemDefinition>Reports indicating the 4D intended trajectory of the
aircraft.</DataItemDefinition>
      <DataItemStructure format="compound" desc="Compound Data Item,
comprising a primary subfield of one octet, followed by the indicated
subfields.">
         <SubField format="variable">
            <SubField format="fixed" length="1">
```

```
                <Field bit="8">
                  <FieldShortName>NAV</FieldShortName>
                  <FieldName>NAV</FieldName>
                  <FieldValue val="0">Trajectory Intent Data is available for this
aircraft</FieldValue>
                  <FieldValue val="1">Trajectory Intent Data is not available for
this aircraft</FieldValue>
                </Field>
                <Field bit="7">
                  <FieldShortName>NVB</FieldShortName>
                  <FieldName>NVB</FieldName>
                  <FieldValue val="0">Trajectory Intent Data is valid</FieldValue>
                  <FieldValue val="1">Trajectory Intent Data is not
valid</FieldValue>
                </Field>
              </SubField>
            </SubField>
            <SubField format="repetitive" length="15">
              <Field bit="120">
                <FieldShortName>TCA</FieldShortName>
                <FieldValue val="0">TCP number available</FieldValue>
                <FieldValue val="1">TCP number not available</FieldValue>
              </Field>
              <Field bit="119">
                <FieldShortName>NC</FieldShortName>
                <FieldValue val="0">TCP compliance</FieldValue>
                <FieldValue val="1">TCP non-compliance</FieldValue>
              </Field>
              <Field msb="118" lsb="113">
                <FieldShortName>TcpN</FieldShortName>
                <FieldName>Trajectory Change Point Number</FieldName>
              </Field>
              <Field msb="112" lsb="97" encode="int">
                <FieldShortName>Alt</FieldShortName>
                <FieldName>Altitude</FieldName>
                <FieldUnit scale="10" min="-1500" max="150000">ft</FieldUnit>
              </Field>
              <Field msb="96" lsb="73" encode="int">
                <FieldShortName>Lat</FieldShortName>
                <FieldName>Latitude</FieldName>
                <FieldUnit scale="0.000021457672119140625" min="-90"
max="90">deg</FieldUnit>
              </Field>
              <Field msb="72" lsb="49" encode="int">
                <FieldShortName>Lon</FieldShortName>
                <FieldName>Longitude</FieldName>
                <FieldUnit scale="0.000021457672119140625" min="-180"
max="180">deg</FieldUnit>
              </Field>
              <Field msb="48" lsb="45">
```

```
            <FieldShortName>PType</FieldShortName>
            <FieldName>Point Type</FieldName>
            <FieldValue val="0">Unknown</FieldValue>
            <FieldValue val="1">Fly by waypoint (LT)</FieldValue>
            <FieldValue val="2">Fly over waypoint (LT)</FieldValue>
            <FieldValue val="3">Hold pattern (LT)</FieldValue>
            <FieldValue val="4">Procedure hold (LT)</FieldValue>
            <FieldValue val="5">Procedure turn (LT)</FieldValue>
            <FieldValue val="6">RF leg (LT)</FieldValue>
            <FieldValue val="7">Top of climb (VT)</FieldValue>
            <FieldValue val="8">Top of descent (VT)</FieldValue>
            <FieldValue val="9">Start of level (VT)</FieldValue>
            <FieldValue val="10">Cross-over altitude (VT)</FieldValue>
            <FieldValue val="11">Transition altitude (VT)</FieldValue>
          </Field>
          <Field msb="44" lsb="43">
            <FieldShortName>TD</FieldShortName>
            <FieldValue val="0">N/A</FieldValue>
            <FieldValue val="1">Turn right</FieldValue>
            <FieldValue val="2">Turn left</FieldValue>
            <FieldValue val="3">No turn</FieldValue>
          </Field>
          <Field bit="42">
            <FieldShortName>TRA</FieldShortName>
            <FieldName>Turn Radius Availability</FieldName>
            <FieldValue val="0">TTR not available</FieldValue>
            <FieldValue val="1">TTR available</FieldValue>
          </Field>
          <Field bit="41">
            <FieldShortName>TOA</FieldShortName>
            <FieldValue val="0">TOV available</FieldValue>
            <FieldValue val="1">TOV not available</FieldValue>
          </Field>
          <Field msb="40" lsb="17">
            <FieldShortName>TOV</FieldShortName>
            <FieldName>Time Over Point</FieldName>
            <FieldUnit scale="1">s</FieldUnit>
          </Field>
          <Field msb="16" lsb="1">
            <FieldShortName>TTR</FieldShortName>
            <FieldName>TCP Turn radius</FieldName>
            <FieldUnit scale="0.01" max="655.35">Nm</FieldUnit>
          </Field>
        </SubField>
      </DataItemStructure>
      <DataItemNote>
```

1. NC is set to one when the aircraft will not fly the path described by the TCP data.
2. TCP numbers start from zero.
3. LT = Lateral Type.

4. VT = Vertical Type.
5. TOV gives the estimated time before reaching the point. It is defined as the absolute time from midnight.
6. TOV is meaningful only if TOA is set to 1.

```
        </DataItemNote>
    </DataItem>
<!--Data Item I021/130, Position in WGS-84 Co-ordinates-->
    <DataItem id="130" rule="optional" frn="6">
        <DataItemName>Position in WGS-84 Co-ordinates</DataItemName>
        <DataItemDefinition>Position in WGS-84 Co-
ordinates.</DataItemDefinition>
        <DataItemStructure format="fixed" length="6" desc="Six-octet fixed length
Data Item.">
            <Field msb="48" lsb="25" encode="int">
                <FieldShortName>Lat</FieldShortName>
                <FieldName>Latitude in WGS.84 in two's complement. Range -90
&lt;= Lat &lt;= 90 deg.</FieldName>
                <FieldUnit scale="0.000021457672119140625" min="-90"
max="90">deg</FieldUnit>
            </Field>
            <Field msb="24" lsb="1" encode="int">
                <FieldShortName>Lon</FieldShortName>
                <FieldName>Longitude in WGS.84 in two's complement. Range -180
&lt;= longitude &lt; 180 deg.</FieldName>
                <FieldUnit scale="0.000021457672119140625" min="-180"
max="180">deg</FieldUnit>
            </Field>
        </DataItemStructure>
    </DataItem>
<!--Data Item I021/131, High-Resolution Position in WGS-84 Co-ordinates-->
    <DataItem id="131" rule="optional" frn="7">
        <DataItemName>High-Resolution Position in WGS-84 Co-
ordinates</DataItemName>
        <DataItemDefinition>Position in WGS-84 Co-ordinates in high
resolution.</DataItemDefinition>
        <DataItemStructure format="fixed" length="8" desc="Eight-octet fixed
length Data Item.">
            <Field msb="64" lsb="33" encode="int">
                <FieldShortName>Lat</FieldShortName>
                <FieldName>Latitude In WGS.84 in two's complement. Range -90
&lt;= Lat &lt;= 90 deg.</FieldName>
                <FieldUnit scale="0.000000167638063430786l3" min="-90"
max="90">deg</FieldUnit>
            </Field>
            <Field msb="32" lsb="1" encode="int">
                <FieldShortName>Lon</FieldShortName>
                <FieldName>Longitude In WGS.84 in two's complement. Range -180
&lt;= longitude &lt; 180 deg.</FieldName>
                <FieldUnit scale="0.000000167638063430786l3" min="-180"
max="180">deg</FieldUnit>
```

```
            </Field>
        </DataItemStructure>
    </DataItem>
<!--Data Item I021/132, Message Amplitude-->
    <DataItem id="132" rule="optional" frn="38">
        <DataItemName>Message Amplitude</DataItemName>
        <DataItemDefinition>Amplitude, in dBm, of ADS-B messages received by
the ground station, coded in two's complement.</DataItemDefinition>
        <DataItemStructure format="fixed" length="1" desc="One-Octet fixed
length data item.">
            <Field msb="8" lsb="1" encode="int">
                <FieldShortName>MAM</FieldShortName>
                <FieldName>Message Amplitude</FieldName>
                <FieldUnit scale="1">dBm</FieldUnit>
            </Field>
        </DataItemStructure>
    </DataItem>
<!--Data Item I021/140, Geometric Height-->
    <DataItem id="140" rule="optional" frn="16">
        <DataItemName>Geometric Height</DataItemName>
        <DataItemDefinition>Minimum height from a plane tangent to the earth's
ellipsoid, defined by WGS-84, in two's complement form.</DataItemDefinition>
        <DataItemStructure format="fixed" length="2" desc="Two-Octet fixed
length data item.">
            <Field msb="16" lsb="1" encode="int">
                <FieldShortName>geometric_height</FieldShortName>
                <FieldName>Geometric Height</FieldName>
                <FieldUnit scale="6.25" min="-1500" max="150000">ft</FieldUnit>
            </Field>
        </DataItemStructure>
        <DataItemNote>
1. LSB is required to be less than 10 ft by ICAO.
2. A value of ‘0111111111111111’ indicates that the aircraft transmits a "greater
than" indication.
        </DataItemNote>
    </DataItem>
<!--Data Item I021/145, Flight Level-->
    <DataItem id="145" rule="optional" frn="21">
        <DataItemName>Flight Level</DataItemName>
        <DataItemDefinition>Flight Level from barometric measurements, not QNH
corrected, in two's complement form.</DataItemDefinition>
        <DataItemStructure format="fixed" length="2" desc="Two-Octet fixed
length data item.">
            <Field msb="16" lsb="1" encode="int">
                <FieldShortName>FL</FieldShortName>
                <FieldName>Flight Level</FieldName>
                <FieldUnit scale="0.25" min="-15" max="1500">FL</FieldUnit>
            </Field>
        </DataItemStructure>
    </DataItem>
```

```
<!--Data Item I021/146, Selected Altitude-->
   <DataItem id="146" rule="optional" frn="32">
      <DataItemName>Intermediate State Selected Altitude</DataItemName>
      <DataItemDefinition>The short-term vertical intent as described by either
the FMS selected altitude, the Altitude Control Panel Selected Altitude, or the
current aircraft altitude according to the aircraft's mode of
flight.</DataItemDefinition>
      <DataItemStructure format="fixed" length="2" desc="Two-Octet fixed
length data item.">
         <Field bit="16">
            <FieldShortName>SAS</FieldShortName>
            <FieldName>Source Availability</FieldName>
            <FieldValue val="0">No source information provided</FieldValue>
            <FieldValue val="1">Source Information provided</FieldValue>
         </Field>
         <Field msb="15" lsb="14">
            <FieldShortName>Source</FieldShortName>
            <FieldName>Source</FieldName>
            <FieldValue val="0">Unknown</FieldValue>
            <FieldValue val="1">Aircraft Altitude (Holding Altitude)</FieldValue>
            <FieldValue val="2">FCU/MCP Selected Altitude</FieldValue>
            <FieldValue val="3">FMS Selected Altitude</FieldValue>
         </Field>
         <Field msb="13" lsb="1" encode="int">
            <FieldShortName>Alt</FieldShortName>
            <FieldName>Altitude</FieldName>
            <FieldUnit scale="25" min="-1300" max="100000">ft</FieldUnit>
         </Field>
      </DataItemStructure>
      <DataItemNote>
1. The Selected Altitude provided in this field is not necessarily the "Target
Altitude" as defined by ICAO.
2. The value of "Source" (bits 15/14) indicating "unknown" or "Aircraft Altitude"
is kept for backward compatibility as these indications are not provided by
"version 2" systems as defined by data item I021/210, bits 6/4.
3. Vertical mode indications supporting the determination of the nature of the
Selected Altitude are provided in the Reserved Expansion Field in the subfield
NAV.
      </DataItemNote>
   </DataItem>
<!--Data Item I021/148, Final State Selected Altitude-->
   <DataItem id="148" rule="optional" frn="33">
      <DataItemName>Final State Selected Altitude</DataItemName>
      <DataItemDefinition>The vertical intent value that corresponds with the
ATC cleared altitude, as derived from the Altitude Control Panel
(FCU/MCP).</DataItemDefinition>
      <DataItemStructure format="fixed" length="2" desc="Two-Octet fixed
length data item.">
         <Field bit="16">
            <FieldShortName>MV</FieldShortName>
```

```
            <FieldName>Manage Vertical Mode</FieldName>
            <FieldValue val="0">Not active</FieldValue>
            <FieldValue val="1">Active</FieldValue>
         </Field>
         <Field bit="15">
            <FieldShortName>AH</FieldShortName>
            <FieldName>Altitude Hold Mode</FieldName>
            <FieldValue val="0">Not active</FieldValue>
            <FieldValue val="1">Active</FieldValue>
         </Field>
         <Field bit="14">
            <FieldShortName>AM</FieldShortName>
            <FieldName>Approach Mode</FieldName>
            <FieldValue val="0">Not active</FieldValue>
            <FieldValue val="1">Active</FieldValue>
         </Field>
         <Field msb="13" lsb="1" encode="int">
               <FieldShortName>Alt</FieldShortName>
               <FieldName>Altitude</FieldName>
               <FieldUnit scale="25" min="-1300" max="100000">ft</FieldUnit>
            </Field>
      </DataItemStructure>
      <DataItemNote>
This item is kept for backward compatibility but shall not be used for "version 2" ADS-B systems (as defined by data item I021/210, bits 6/4) for which item 146 will be used to forward the MCP/FCU or the FMS selected altitude information. For "version 2" ADS-B systems, the vertical mode indications will be provided through the Reserved Expansion Field in the subfield NAV.
      </DataItemNote>
   </DataItem>
<!--Data Item I021/150, Air Speed-->
   <DataItem id="150" rule="optional" frn="9">
      <DataItemName>Air Speed</DataItemName>
      <DataItemDefinition>Calculated Air Speed (Element of Air Vector).</DataItemDefinition>
      <DataItemStructure format="fixed" length="2" desc="Two-Octet fixed length data item.">
         <Field bit="16">
            <FieldShortName>IM</FieldShortName>
            <FieldName>Air Speed type</FieldName>
            <FieldValue val="0">Air Speed = IAS</FieldValue>
            <FieldValue val="1">Air Speed = Mach</FieldValue>
         </Field>
         <Field msb="15" lsb="1">
            <FieldShortName>ASpeed</FieldShortName>
            <FieldName>Air Speed (IAS or Mach)</FieldName>
         </Field>
      </DataItemStructure>
   </DataItem>
<!--Data Item I021/151, True Airspeed-->
```

```
   <DataItem id="151" rule="optional" frn="10">
      <DataItemName>True Airspeed</DataItemName>
      <DataItemDefinition>True Air Speed.</DataItemDefinition>
      <DataItemStructure format="fixed" length="2" desc="Two-Octet fixed
length data item.">
         <Field bit="16">
            <FieldShortName>RE</FieldShortName>
            <FieldName>Range Exceeded Indicator</FieldName>
            <FieldValue val="0">Value in defined range</FieldValue>
            <FieldValue val="1">Value exceeds defined range</FieldValue>
         </Field>
         <Field msb="15" lsb="1">
            <FieldShortName>TAS</FieldShortName>
            <FieldName>True Air Speed</FieldName>
            <FieldUnit scale="1">knot</FieldUnit>
         </Field>
      </DataItemStructure>
      <DataItemNote>The RE-Bit, if set, indicates that the value to be
transmitted is beyond the range defined for this specific data item and the
applied technology. In this case the True Air Speed contains the maximum
value that can be downloaded from the aircraft avionics and the RE- bit
indicates that the actual value is greater than the value contained in the
field.</DataItemNote>
   </DataItem>
<!--Data Item I021/152, Magnetic Heading-->
   <DataItem id="152" rule="optional" frn="22">
      <DataItemName>Magnetic Heading</DataItemName>
      <DataItemDefinition>Magnetic Heading (Element of Air
Vector).</DataItemDefinition>
      <DataItemStructure format="fixed" length="2" desc="Two-Octet fixed
length data item.">
         <Field msb="16" lsb="1">
            <FieldShortName>MHdg</FieldShortName>
            <FieldName>Magnetic Heading</FieldName>
            <FieldUnit scale="0.0054931640625">deg</FieldUnit>
         </Field>
      </DataItemStructure>
      <DataItemNote>
True North Heading is defined in the Reserved Expansion Field in the subfield
TNH.
      </DataItemNote>
   </DataItem>
<!--Data Item I021/155, Barometric Vertical Rate-->
   <DataItem id="155" rule="optional" frn="24">
      <DataItemName>Barometric Vertical Rate</DataItemName>
      <DataItemDefinition>Barometric Vertical Rate, in two's complement
form.</DataItemDefinition>
      <DataItemStructure format="fixed" length="2" desc="Two-Octet fixed
length data item.">
         <Field bit="16">
```

```
            <FieldShortName>RE</FieldShortName>
            <FieldName>Range Exceeded Indicator</FieldName>
            <FieldValue val="0">Value in defined range</FieldValue>
            <FieldValue val="1">Value exceeds defined range</FieldValue>
          </Field>
          <Field msb="15" lsb="1" encode="int">
            <FieldShortName>BVR</FieldShortName>
            <FieldName>Barometric Vertical Rate</FieldName>
            <FieldUnit scale="6.25">feet/minute</FieldUnit>
          </Field>
        </DataItemStructure>
        <DataItemNote>The RE-Bit, if set, indicates that the value to be
transmitted is beyond the range defined for this specific data item and the
applied technology. In this case the Barometric Vertical Rate contains the
maximum value that can be downloaded from the aircraft avionics and the RE-
bit indicates that the actual value is greater than the value contained in the
field.</DataItemNote>
    </DataItem>
<!--Data Item I021/157, Geometric Vertical Rate-->
    <DataItem id="157" rule="optional" frn="25">
        <DataItemName>Geometric Vertical Rate</DataItemName>
        <DataItemDefinition>Geometric Vertical Rate, in two's complement form,
with reference to WGS-84.</DataItemDefinition>
        <DataItemStructure format="fixed" length="2" desc="Two-Octet fixed
length data item.">
          <Field bit="16">
            <FieldShortName>RE</FieldShortName>
            <FieldName>Range Exceeded Indicator</FieldName>
            <FieldValue val="0">Value in defined range</FieldValue>
            <FieldValue val="1">Value exceeds defined range</FieldValue>
          </Field>
          <Field msb="15" lsb="1" encode="int">
            <FieldShortName>GVR</FieldShortName>
            <FieldName>Geometric Vertical Rate</FieldName>
            <FieldUnit scale="6.25">feet/minute</FieldUnit>
          </Field>
        </DataItemStructure>
        <DataItemNote>The RE-Bit, if set, indicates that the value to be
transmitted is beyond the range defined for this specific data item and the
applied technology. In this case the Geometric Vertical Rate contains the
maximum value that can be downloaded from the aircraft avionics and the RE-
bit indicates that the actual value is greater than the value contained in the
field.</DataItemNote>
    </DataItem>
<!--Data Item I021/160, Airborne Ground Vector-->
    <DataItem id="160" rule="optional" frn="26">
        <DataItemName>Airborne Ground Vector</DataItemName>
        <DataItemDefinition>Ground Speed and Track Angle elements of Airborne
Ground Vector.</DataItemDefinition>
```

```
      <DataItemStructure format="fixed" length="4" desc="Four-Octet fixed
length data item.">
         <Field bit="16">
            <FieldShortName>RE</FieldShortName>
            <FieldName>Range Exceeded Indicator</FieldName>
            <FieldValue val="0">Value in defined range</FieldValue>
            <FieldValue val="1">Value exceeds defined range</FieldValue>
         </Field>
         <Field msb="31" lsb="17">
            <FieldShortName>GS</FieldShortName>
            <FieldName>Ground Speed referenced to WGS-84</FieldName>
            <FieldUnit scale="0.00006103515625">NM/s</FieldUnit>
         </Field>
         <Field msb="16" lsb="1">
            <FieldShortName>TA</FieldShortName>
            <FieldName>Track Angle clockwise reference to True
North</FieldName>
            <FieldUnit scale="0.0054931640625">deg</FieldUnit>
         </Field>
      </DataItemStructure>
      <DataItemNote>
1. The RE-Bit, if set, indicates that the value to be transmitted is beyond the
range defined for this specific data item and the applied technology. In this case
the Ground Speed contains the maximum value that can be downloaded from
the aircraft avionics and the RE- bit indicates that the actual value is greater
than the value contained in the field.
2. The Surface Ground Vector format is defined in the Reserved Expansion
Field in the subfield SGV.
      </DataItemNote>
   </DataItem>
<!--Data Item I021/161, Track Number-->
   <DataItem id="161" rule="optional" frn="3">
      <DataItemName>Track Number</DataItemName>
      <DataItemDefinition>An integer value representing a unique reference to a
track record within a particular track file.</DataItemDefinition>
      <DataItemStructure format="fixed" length="2" desc="Two-octet fixed length
Data Item.">
         <Field msb="12" lsb="1">
            <FieldShortName>TrackN</FieldShortName>
            <FieldName>Track number</FieldName>
         </Field>
      </DataItemStructure>
   </DataItem>
<!--Data Item I021/165, Track Angle Rate-->
   <DataItem id="165" rule="optional" frn="27">
      <DataItemName>Track Angle Rate</DataItemName>
      <DataItemDefinition>Rate of Turn, in two's complement
form.</DataItemDefinition>
      <DataItemStructure format="fixed" length="2" desc="2-Byte Fixed length
data item.">
```

```
            <Field msb="10" lsb="1" encode="int">
               <FieldShortName>TA_rate</FieldShortName>
               <FieldName>Track Angle Rate</FieldName>
               <FieldUnit scale="0.03125" min="-16" max="16">deg/s</FieldUnit>
            </Field>
         </DataItemStructure>
         <DataItemNote>
1. A positive value represents a right turn, whereas a negative value represents
a left turn.
2. "Maximum value" means Maximum value or above.
3. This item will not be transmitted for the technology "1090 MHz Extended
Squitter".
      </DataItemNote>
      </DataItem>
<!--Data Item I021/170, Target Identification-->
      <DataItem id="170" rule="optional" frn="29">
         <DataItemName>Target Identification</DataItemName>
         <DataItemDefinition>Target (aircraft or vehicle) identification in 8
characters, as reported by the target.</DataItemDefinition>
         <DataItemStructure format="fixed" length="6" desc="Six-octet fixed length
Data Item.">
            <Field msb="48" lsb="1" encode="sixbitschar">
               <FieldShortName>TId</FieldShortName>
               <FieldName>Characters 1-8 (coded on 6 Bits each) defining target
identification when flight plan is available or the registration marking when no
flight plan is available. Coding rules are provided in [6] Section 3.1.2.9.1.2 and
Table 3-9.</FieldName>
            </Field>
         </DataItemStructure>
      </DataItem>
<!--Data Item I021/200, Target Status-->
      <DataItem id="200" rule="optional" frn="23">
         <DataItemName>Target Status</DataItemName>
         <DataItemDefinition>Status of the target</DataItemDefinition>
         <DataItemStructure format="fixed" length="1" desc="One-octet fixed length
Data Item">
            <Field bit="8">
               <FieldShortName>ICF</FieldShortName>
               <FieldName>Intent Change Flag</FieldName>
               <FieldValue val="0">No intent change active</FieldValue>
               <FieldValue val="1">Intent change flag raised</FieldValue>
            </Field>
            <Field bit="7">
               <FieldShortName>LNAV</FieldShortName>
               <FieldName>LNAV Mode</FieldName>
               <FieldValue val="0">LNAV Mode engaged</FieldValue>
               <FieldValue val="1">LNAV Mode not engaged</FieldValue>
            </Field>
            <Field msb="5" lsb="3">
               <FieldShortName>PS</FieldShortName>
```

```
            <FieldName>Priority Status</FieldName>
            <FieldValue val="0">No emergency / not reported</FieldValue>
            <FieldValue val="1">General emergency</FieldValue>
            <FieldValue val="2">Lifeguard / medical emergency</FieldValue>
            <FieldValue val="3">Minimum fuel</FieldValue>
            <FieldValue val="4">No communications</FieldValue>
            <FieldValue val="5">Unlawful interference</FieldValue>
            <FieldValue val="6">Downed Aircraft</FieldValue>
         </Field>
         <Field msb="2" lsb="1">
            <FieldShortName>SS</FieldShortName>
            <FieldName>Surveillance Status</FieldName>
            <FieldValue val="0">No condition reported</FieldValue>
            <FieldValue val="1">Permanent Alert (Emergency
condition)</FieldValue>
            <FieldValue val="2">Temporary Alert (change in Mode 3/A Code
other than emergency)</FieldValue>
            <FieldValue val="3">SPI set</FieldValue>
         </Field>
      </DataItemStructure>
      <DataItemNote>
Bit-8 (ICF), when set to "1" indicates that new information is available in the
Mode S GICB registers 40, 41 or 42.
      </DataItemNote>
   </DataItem>
<!--Data Item I021/210, MOPS Version-->
   <DataItem id="210" rule="optional" frn="18">
      <DataItemName>MOPS Version</DataItemName>
      <DataItemDefinition>Identification of the MOPS version used by a/c to
supply ADS-B information.</DataItemDefinition>
      <DataItemStructure format="fixed" length="1" desc="One-octet fixed length
Data Item">
         <Field bit="7">
            <FieldShortName>VNS</FieldShortName>
            <FieldName>Version Not Supported</FieldName>
            <FieldValue val="0">The MOPS Version is supported by the
GS</FieldValue>
            <FieldValue val="1">The MOPS Version is not supported by the
GS</FieldValue>
         </Field>
         <Field msb="6" lsb="4">
            <FieldShortName>VN</FieldShortName>
            <FieldName>Version Number</FieldName>
            <FieldValue val="0">DO-260 [Ref. 8]</FieldValue>
            <FieldValue val="1">DO-260A [Ref. 9]</FieldValue>
            <FieldValue val="2">DO-260B [Ref. 11]</FieldValue>
         </Field>
         <Field msb="3" lsb="1">
            <FieldShortName>LTT</FieldShortName>
            <FieldName>Link Technology Type</FieldName>
```

```
            <FieldValue val="0">Other</FieldValue>
            <FieldValue val="1">UAT</FieldValue>
            <FieldValue val="2">1090 ES</FieldValue>
            <FieldValue val="3">VDL 4</FieldValue>
            <FieldValue val="4">Not asint</FieldValue>
            <FieldValue val="5">Not asint</FieldValue>
            <FieldValue val="6">Not asint</FieldValue>
            <FieldValue val="7">Not asint</FieldValue>
         </Field>
      </DataItemStructure>
      <DataItemNote>Bit 7 (VNS) when set to 1 indicates that the aircraft
transmits a MOPS Version indication that is not supported by the Ground
Station. However, since MOPS versions are supposed to be backwards
compatible, the GS has attempted to interpret the message and achieved a
credible result. The fact that the MOPS version received is not supported by the
GS is submitted as additional information to subsequent processing
systems.</DataItemNote>
   </DataItem>
<!--Data Item I021/220, Met Information-->
   <DataItem id="220" rule="optional" frn="31">
      <DataItemName>Met Information</DataItemName>
      <DataItemDefinition>Meteorological information.</DataItemDefinition>
      <DataItemStructure format="compound" desc="Compound data item
consisting of a one byte primary sub-field, followed by up to four fixed length
data fields.">
         <SubField format="fixed" length="2">
            <Field msb="16" lsb="1">
               <FieldShortName>WindS</FieldShortName>
               <FieldName>Wind Speed</FieldName>
               <FieldUnit scale="1" min="0" max="300">knot</FieldUnit>
            </Field>
         </SubField>
         <SubField format="fixed" length="2">
            <Field msb="16" lsb="1">
               <FieldShortName>WindD</FieldShortName>
               <FieldName>Wind Direction</FieldName>
               <FieldUnit scale="1" min="1" max="360">deg</FieldUnit>
            </Field>
         </SubField>
         <SubField format="fixed" length="2">
            <Field msb="16" lsb="1" encode="int">
               <FieldShortName>Temp</FieldShortName>
               <FieldName>Temperature</FieldName>
               <FieldUnit scale="0.25" min="-100" max="100">C</FieldUnit>
            </Field>
         </SubField>
         <SubField format="fixed" length="1">
            <Field msb="8" lsb="1">
               <FieldShortName>Turb</FieldShortName>
               <FieldName>Turbulence</FieldName>
```

```
                <FieldUnit min="0" max="15"></FieldUnit>
            </Field>
        </SubField>
    </DataItemStructure>
</DataItem>
<!--Data Item I021/230, Roll Angle-->
<DataItem id="230" rule="optional" frn="20">
    <DataItemName>Roll Angle</DataItemName>
    <DataItemDefinition>The roll angle, in two's complement form, of an
aircraft executing a turn.</DataItemDefinition>
    <DataItemStructure format="fixed" length="2" desc="A two byte fixed
length data item.">
        <Field msb="16" lsb="1" encode="int">
            <FieldShortName>RollA</FieldShortName>
            <FieldName>Roll Angle</FieldName>
            <FieldUnit scale="0.01" min="-180" max="180">deg</FieldUnit>
        </Field>
    </DataItemStructure>
    <DataItemNote>1. Negative Value indicates “Left Wing Down”. 2.
Resolution provided by the technology “1090 MHz Extended Squitter” is 1
degree.</DataItemNote>
</DataItem>
<!--Data Item I021/250, Mode S MB Data-->
<DataItem id="250" rule="mandatory" frn="39">
    <DataItemName>Mode S MB Data</DataItemName>
    <DataItemDefinition>Mode S Comm B data as extracted from the aircraft
transponder.</DataItemDefinition>
    <DataItemStructure format="repetitive" length="8" desc="Repetitive Data
Item starting with a one-octet Field Repetition Indicator (REP) followed by at
least one BDS message comprising one seven octet BDS register and one
octet BDS code.">
        <Field msb="64" lsb="9" encode="hex">
            <FieldShortName>MBData</FieldShortName>
            <FieldName>Mode S Comm B message data</FieldName>
        </Field>
        <Field msb="8" lsb="5">
            <FieldShortName>BDS1</FieldShortName>
            <FieldName>Comm B Data Buffer Store 1 Address</FieldName>
        </Field>
        <Field msb="4" lsb="1">
            <FieldShortName>BDS2</FieldShortName>
            <FieldName>Comm B Data Buffer Store 2 Address</FieldName>
        </Field>
    </DataItemStructure>
</DataItem>
<!--Data Item I021/260, ACAS Resolution Advisory Report-->
<DataItem id="260" rule="mandatory" frn="40">
    <DataItemName>ACAS Resolution Advisory Report</DataItemName>
```

```
      <DataItemDefinition>Currently active Resolution Advisory (RA), if any,
generated by the ACAS associated with the transponder transmitting the RA
message and threat identity data.</DataItemDefinition>
      <DataItemStructure format="fixed" length="7" desc="Seven-octet fixed
length Data Item.">
        <Field msb="56" lsb="52">
          <FieldShortName>TYP</FieldShortName>
          <FieldName>Message Type (= 28 for 1090 ES, version
2).</FieldName>
        </Field>
        <Field msb="51" lsb="49">
          <FieldShortName>STYP</FieldShortName>
          <FieldName>Message Sub-type (= 2 for 1090 ES, version
2).</FieldName>
        </Field>
        <Field msb="48" lsb="35">
          <FieldShortName>ARA</FieldShortName>
          <FieldName>Active Resolution Advisories</FieldName>
        </Field>
        <Field msb="34" lsb="31">
          <FieldShortName>RAC</FieldShortName>
          <FieldName>RAC (RA Complement) Record</FieldName>
        </Field>
        <Field bit="30">
          <FieldShortName>RAT</FieldShortName>
          <FieldName>RA Terminated</FieldName>
        </Field>
        <Field bit="29">
          <FieldShortName>MTE</FieldShortName>
          <FieldName>Multiple Threat Encounter</FieldName>
        </Field>
        <Field msb="28" lsb="27">
          <FieldShortName>TTI</FieldShortName>
          <FieldName>Threat Type Indicator</FieldName>
        </Field>
        <Field msb="26" lsb="1">
          <FieldShortName>TID</FieldShortName>
          <FieldName>Threat Identity Data</FieldName>
        </Field>
      </DataItemStructure>
      <DataItemNote>
1. Version denotes the MOPS version as defined in I021/210, bits 6/4.
2. This data items copies the value of BDS register 6,2 for message type 28,
subtype 2.
3. The “TYP” and “STYP” items are implementation (i.e. link technology)
dependent.
4. Refer to ICAO Annex 10 SARPs for detailed explanations [Ref. 10].
      </DataItemNote>
    </DataItem>
<!--Data Item I021/271, Surface Capabilities and Characteristics-->
```

```
    <DataItem id="271" rule="optional" frn="37">
      <DataItemName>Surface Capabilities and
Characteristics</DataItemName>
      <DataItemDefinition>Operational capabilities of the aircraft while on the
ground.</DataItemDefinition>
      <DataItemStructure format="variable" desc="Variable Length Data Item,
comprising a primary subfield of one-octet, followed by an one-octet extents if
necessary.">
        <SubField format="fixed" length="1">
          <Field bit="6">
            <FieldShortName>POA</FieldShortName>
            <FieldName>Position Offset Applied</FieldName>
            <FieldValue val="0">Position transmitted is not ADS-B position
reference point</FieldValue>
            <FieldValue val="1">Position transmitted is the ADS-B position
reference point</FieldValue>
          </Field>
          <Field bit="5">
            <FieldShortName>CDTI_S</FieldShortName>
            <FieldName>Cockpit Display of Traffic Information
Surface</FieldName>
            <FieldValue val="0">CDTI not operational</FieldValue>
            <FieldValue val="1">CDTI operational</FieldValue>
          </Field>
          <Field bit="4">
            <FieldShortName>B2_low</FieldShortName>
            <FieldName>Class B2 transmit power less than 70
Watts</FieldName>
            <FieldValue val="0">&gt;= 70 Watts</FieldValue>
            <FieldValue val="1">&lt; 70 Watts</FieldValue>
          </Field>
          <Field bit="3">
            <FieldShortName>RAS</FieldShortName>
            <FieldName>Receiving ATC Services</FieldName>
            <FieldValue val="0">Aircraft not receiving ATC-
services</FieldValue>
            <FieldValue val="1">Aircraft receiving ATC services</FieldValue>
          </Field>
          <Field bit="2">
            <FieldShortName>IDENT</FieldShortName>
            <FieldName>Setting of “IDENT”-switch</FieldName>
            <FieldValue val="0">IDENT switch not active</FieldValue>
            <FieldValue val="1">IDENT switch active</FieldValue>
          </Field>
        </SubField>
        <SubField format="fixed" length="1">
          <Field msb="4" lsb="1">
            <FieldShortName>length_width</FieldShortName>
            <FieldName>Length and width of the aircraft</FieldName>
          </Field>
```

```
            </SubField>
        </DataItemStructure>
        <DataItemNote>
1. The length and width of the aircraft are encoded according to the following
table.
2. Version 2 (as defined in I021/210, bits 6/4) data technology protocols encode
"No Data or Unknown" with value 0. In this case data item I021/271, first
extension is not generated.
3. This data item is a variant of the "Extended length data field" as described in
ASTERIX part1. The LSB in the first extension is not used as FX-bit.
        </DataItemNote>
    </DataItem>
<!--Data Item I021/295, Data Ages-->
    <DataItem id="295" rule="optional" frn="42">
        <DataItemName>Data Ages</DataItemName>
        <DataItemDefinition>Ages of the data provided.</DataItemDefinition>
        <DataItemStructure format="compound" desc="Compound Data Item,
comprising a primary subfield of up to five octets, followed by the indicated
subfields.">
            <SubField format="fixed" length="1">
                <Field msb="8" lsb="1">
                    <FieldShortName>AOS</FieldShortName>
                    <FieldName>Age of the latest received information transmitted in
item I021/008.</FieldName>
                    <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
                </Field>
            </SubField>
            <SubField format="fixed" length="1">
                <Field msb="8" lsb="1">
                    <FieldShortName>TRD</FieldShortName>
                    <FieldName>Age of the last update of the Target Report
Descriptor, item I021/040</FieldName>
                    <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
                </Field>
            </SubField>
            <SubField format="fixed" length="1">
                <Field msb="8" lsb="1">
                    <FieldShortName>M3A</FieldShortName>
                    <FieldName>Age of the last update of the Mode 3/A Code, item
I021/070</FieldName>
                    <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
                </Field>
            </SubField>
            <SubField format="fixed" length="1">
                <Field msb="8" lsb="1">
                    <FieldShortName>QI</FieldShortName>
                    <FieldName>Age of the latest information received to update the
Quality Indicators, item I021/090</FieldName>
                    <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
                </Field>
```

```
        </SubField>
        <SubField format="fixed" length="1">
          <Field msb="8" lsb="1">
            <FieldShortName>TI</FieldShortName>
            <FieldName>Age of the last update of the Trajectory Intent
information updating item I021/110</FieldName>
            <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
          </Field>
        </SubField>
        <SubField format="fixed" length="1">
          <Field msb="8" lsb="1">
            <FieldShortName>MAM</FieldShortName>
            <FieldName>Age of the latest measurement of the message
amplitude, item I021/132</FieldName>
            <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
          </Field>
        </SubField>
        <SubField format="fixed" length="1">
          <Field msb="8" lsb="1">
            <FieldShortName>GH</FieldShortName>
            <FieldName>Age of the information contained in item
021/140</FieldName>
            <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
          </Field>
        </SubField>
        <SubField format="fixed" length="1">
          <Field msb="8" lsb="1">
            <FieldShortName>FL</FieldShortName>
            <FieldName>Age of the Flight Level information in item
I021/145</FieldName>
            <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
          </Field>
        </SubField>
        <SubField format="fixed" length="1">
          <Field msb="8" lsb="1">
            <FieldShortName>ISA</FieldShortName>
            <FieldName>Age of the Intermediate State Selected Altitude in
item I021/146</FieldName>
            <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
          </Field>
        </SubField>
        <SubField format="fixed" length="1">
          <Field msb="8" lsb="1">
            <FieldShortName>FSA</FieldShortName>
            <FieldName>Age of the Final State Selected Altitude in item
I021/148</FieldName>
            <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
          </Field>
        </SubField>
        <SubField format="fixed" length="1">
```

```
            <Field msb="8" lsb="1">
               <FieldShortName>AS</FieldShortName>
               <FieldName>Age of the Air Speed contained in item
I021/150</FieldName>
               <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
            </Field>
         </SubField>
         <SubField format="fixed" length="1">
            <Field msb="8" lsb="1">
               <FieldShortName>TAS</FieldShortName>
               <FieldName>Age of the value for the True Air Speed in item
I021/151</FieldName>
               <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
            </Field>
         </SubField>
         <SubField format="fixed" length="1">
            <Field msb="8" lsb="1">
               <FieldShortName>MH</FieldShortName>
               <FieldName>Age of the value for the Magnetic Heading in item
I021/152</FieldName>
               <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
            </Field>
         </SubField>
         <SubField format="fixed" length="1">
            <Field msb="8" lsb="1">
               <FieldShortName>BVR</FieldShortName>
               <FieldName>Age of the Barometric Vertical Rate contained in
I021/155</FieldName>
               <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
            </Field>
         </SubField>
         <SubField format="fixed" length="1">
            <Field msb="8" lsb="1">
               <FieldShortName>GVR</FieldShortName>
               <FieldName>Age of the Geometric Vertical Rate in item
I021/157</FieldName>
               <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
            </Field>
         </SubField>
         <SubField format="fixed" length="1">
            <Field msb="8" lsb="1">
               <FieldShortName>GV</FieldShortName>
               <FieldName>Age of the Ground Vector in item
I021/160</FieldName>
               <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
            </Field>
         </SubField>
         <SubField format="fixed" length="1">
            <Field msb="8" lsb="1">
               <FieldShortName>TAR</FieldShortName>
```

```
            <FieldName>Age of item I021/165 Track Angle Rate</FieldName>
            <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
          </Field>
        </SubField>
        <SubField format="fixed" length="1">
          <Field msb="8" lsb="1">
            <FieldShortName>TId</FieldShortName>
            <FieldName>Age of the Target Identification in item
I021/170</FieldName>
            <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
          </Field>
        </SubField>
        <SubField format="fixed" length="1">
          <Field msb="8" lsb="1">
            <FieldShortName>TS</FieldShortName>
            <FieldName>Age of the Target Status as contained in item
I021/200</FieldName>
            <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
          </Field>
        </SubField>
        <SubField format="fixed" length="1">
          <Field msb="8" lsb="1">
            <FieldShortName>MET</FieldShortName>
            <FieldName>Age of the Meteorological data contained in
I021/220</FieldName>
            <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
          </Field>
        </SubField>
        <SubField format="fixed" length="1">
          <Field msb="8" lsb="1">
            <FieldShortName>ROA</FieldShortName>
            <FieldName>Age of the Roll Angle value as in item
I021/230</FieldName>
            <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
          </Field>
        </SubField>
        <SubField format="fixed" length="1">
          <Field msb="8" lsb="1">
            <FieldShortName>ARA</FieldShortName>
            <FieldName>Age of the latest update of an active ACAS
Resolution Advisory</FieldName>
            <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
          </Field>
        </SubField>
        <SubField format="fixed" length="1">
          <Field msb="8" lsb="1">
            <FieldShortName>SCC</FieldShortName>
            <FieldName>Age of the latest information received on the surface
capabilities and characteristics of the respective target</FieldName>
            <FieldUnit scale="0.1" max="25.5">s</FieldUnit>
```

```
            </Field>
         </SubField>
      </DataItemStructure>
      <DataItemNote>1. In all the subfields, the age is the time delay since the
latest update received from the target. 2. In all the subfields, the maximum
value indicates “maximum value or above”.</DataItemNote>
   </DataItem>
<!--Data Item I021/400, Receiver ID-->
   <DataItem id="400" rule="optional" frn="41">
      <DataItemName>Receiver ID</DataItemName>
      <DataItemDefinition>Designator of Ground Station in Distributed
System.</DataItemDefinition>
      <DataItemStructure format="fixed" length="1" desc="One-octet fixed length
Data Item.">
         <Field msb="8" lsb="1">
            <FieldShortName>RID</FieldShortName>
            <FieldName>Receiver ID</FieldName>
         </Field>
      </DataItemStructure>
   </DataItem>
</Category>
```