ECCAIRS 5 Reporting System

ECCAIRS 5 Common Framework
RIT/E5X Data Bridge

White Paper

Contents

License Agreement

As used in this Agreement, "JRC" shall mean "The Joint Research Centre of the European Commission". This License Agreement is a legal agreement between you (either an individual or a single entity) and JRC for the ECCAIRS software product identified above, which may include associated software components, media, printed materials, and "online" or electronic documentation ("Software"). By installing, copying, or otherwise using the Software, you agree to be bound by the terms of this Agreement. If you do not agree to the terms of this Agreement, do not install or use the Software.

SOFTWARE LICENSE

The ECCAIRS Reporting System is being distributed by JRC to competent authorities (Civil Aviation Authorities, Accident Investigation Bureaus and Air Navigation Service Providers) of EU Member States, Iceland, Norway and Switzerland who are entitled to use the software for implementing EU Directive 4006/42/EC on Occurrence Reporting in Civil Aviation.

An ECCAIRS National Point of Contact (see Directive 4006/42/EC) can provide the ECCAIRS Reporting System to operators and service providers under specific conditions:

- A National Point of Contact shall provide the software only to operators and service providers placed under their responsibility.
- Usage shall be limited to those situations where it will enable a more efficient circulation of occurrences information between reporting bodies and national regulators/investigators in the scope of Directive 4006/42/EC on Occurrence Reporting in Civil Aviation.
- In accordance with the rules governing the dissemination of information related to occurrence reporting, these operators and service providers will not be allowed to have direct access to the central repository of ECCAIRS.
- Each National Point of Contact shall ensure the necessary technical support to operators and service providers to which the software has been provided.
- Each National Point of Contact shall inform the Commission on operators and service providers to which the ECCAIRS software is given.
- National Points of Contact are not allowed to make a charge for distributing this Software to operators and service providers.

Non EU authorities and non-profit international organisations involved in aviation safety should contact JRC if they are interested in using the ECCAIRS Reporting System.

By using this Software, you agree to these terms and the terms of the Disclaimer below:

DISCLAIMER

THIS SOFTWARE AND ANY SUPPORT FROM JRC ARE PROVIDED "AS IS" AND WITHOUT WARRANTY, EXPRESS OR IMPLIED. JRC SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL JRC BE LIABLE FOR ANY DAMAGES, INCLUDING BUT NOT LIMITED TO ANY LOST PROFITS, LOST SAVINGS OR ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, WHETHER RESULTING FROM IMPAIRED OR LOST DATA, SOFTWARE OR COMPUTER FAILURE OR ANY OTHER CAUSE, OR FOR ANY OTHER CLAIM BY THE USER OR FOR ANY THIRD PARTY CLAIM.

ECCAIRS on the Internet: http://eccairsportal.jrc.ec.europa.eu

1 Abstract

The ECCAIRS *Data Bridge* is an XML based and XSD compliant way to feed data into an ECCAIRS system. The transfer format consists of one or more XML files, each containing the data of a single occurrence in a format defined by a specific taxonomy. These XML files are validated against an XSD schema that describes the involved taxonomy structure and the related data.

The XML files are packaged into a compressed file — a ZIP file with the E5X extension — which is imported into the ECCAIRS system. The ECCAIRS software converts the contents of the compressed file into the proprietary E5F format so that the occurrence(s) can be processed by the software suite and stored in the native containers (file or repository database).

This document describes the XML format and explains how to create the package. Purpose of this document is to provide 3rd parties with a W3C compliant specification so that these parties can produce and verify correctness of occurrence data. The related XSD schema is provided with each release of an ECCAIRS extension. The XML and the accompanying XSD schema are linked to one and only version of the extension; multiple versions of an extension are supported by the provision of an equivalent number of XSD schemas.

The RIT/E5X data format is a file format implementing the Reduced Information Taxonomy (RIT) defined by the aviation community in Europe. RIT/E5X is based on a subset of the ECCAIRS Aviation Taxonomy, the so-called RIT domain, and uses the Data Bridge to process data from industry towards an ECCAIRS environment. RIT Data Bridge is available starting release 5.4.1.16 of the ECCAIRS Common Framework.

This White paper is targeted at developers interested in creating XML files, packaged in an E5X file, to be fed into an ECCAIRS instance. Good knowledge of the XML language and the XSD Schema Definitions is highly recommended.

2 Introduction

The ECCAIRS Reporting System software stores the occurrence data records in the database associated to the repository. The format used is based on a proprietary ECCAIRS 5 Data File format called E5F[1]. This native storage format is supported in the database associated to the repository and in the E5F files and obviously read as well as written by the various applications of the ECCAIRS Reporting System software suite.

A so-called *Reduced Interface Taxonomy* (RIT) has been defined in the Aviation Taxonomy extension for ECCAIRS, which consists of a reduced set of attributes (data fields) of the ECCAIRS Aviation Taxonomy. To implement Regulation 376/2014 of the European Union[2] it was agreed to use this subset of attributes to allow organisations (i.e. the industry) to report occurrences to the National Point of Contact (NPC). In general, such organisations do not have an instance of ECCAIRS, which they could have used to report, but their own. It was agreed that organisations report electronically to the

[1] See White Paper *ECF E5F FILE FORMAT* (Version 2.0 February 2013)

[2] http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1447326759604&uri=CELEX:32014R0376

NPC, using an open, XML-based and W3C compliant data format. This format is named ECCAIRS 5 eXchange format, in short E5X.

In this document the term RIT is used to refer to the data contents and the term E5X for the physical format of these files. The combination RIT/E5X is used to indicate an E5X of a determined RIT.

Starting with version 5.4.1.16 of the ECCAIRS 5 Reporting System, the application able to open and convert the E5X files are the Data Manager, the Windows 3rd party API (E5API) and the Web Services API. Starting with version 5.5.1.5, this same functionality was added to the Browser, albeit limited to conversion of E5X to E5F files.

Though the E5X format is universal — its definition is identical for any taxonomy used — the domain part, RIT in this case, is specific for each deployment of a taxonomy extension. This implies that the ECCAIRS system must be configured properly otherwise it is unable to process any particular E5X for that RIT.

3 Implementation

The conversion of E5X files into the native E5F format is performed with a *Data Bridge*. This Data Bridge validates the contents of the E5X and generates an equivalent E5F[3] file. The configuration of the Data Bridge in the ECCAIRS system is documented in a separate White Paper[4].

In the next figure the ECCAIRS architecture is depicted. The ECCAIRS Common framework is the standard software used in any ECCAIRS instance.

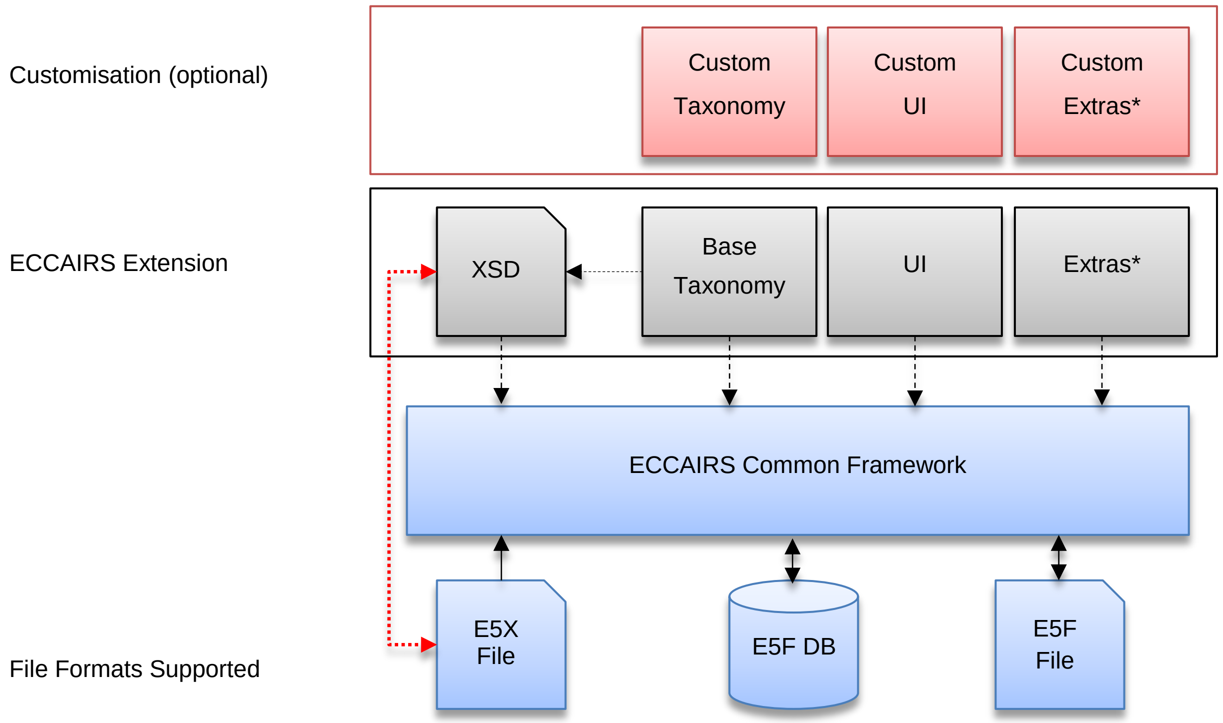

Figure 1 – ECCAIRS 5 Architecture

[3] See the White Paper: *ECF E5F FILE FORMAT* (Version 2.0 February 2013)

[4] See White Paper *ECCAIRS DATA BRIDGE CONFIGURATION* (Version 4.0 March 2018)

A standard ECCAIRS Extension normally provides 3 fixed items: the [base] taxonomy, the user interfaces (UI) and the XSD schema derived from the taxonomy. Depending on the taxonomy, optional extras may be supplied, for instance a special data entry wizard or vessel registry interface.

The taxonomy version of the input E5X file and the resulting E5F file are identical. For example, an E5X produced for Aviation Taxonomy version '1.2.3.4' will result in an E5F for the same taxonomy and the same version '1.2.3.4'.

Below is the conversion workflow that illustrates the conversion workflow of E5X into E5F, as well as the applications being able to perform the conversion:

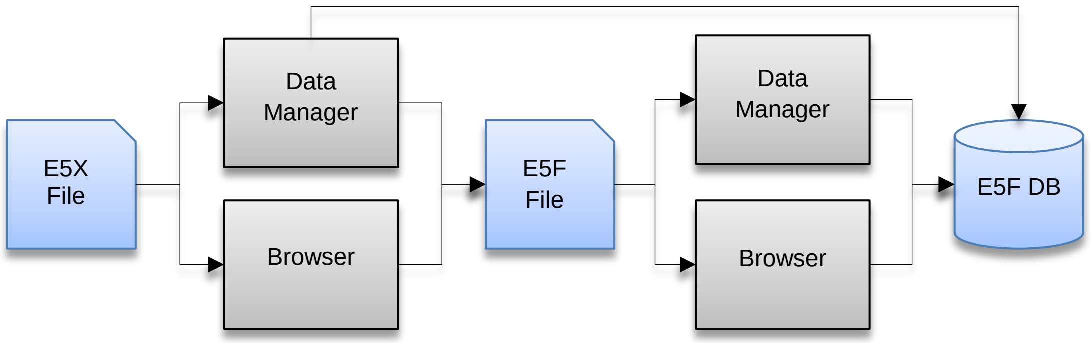

Figure 2 – E5X to E5F/DB conversion workflow

The Data Manager is able to convert to E5F file but also to save the converted occurrence directly in the repository database. The Browser can only convert to E5F file. The second phase of importing the obtained E5F file into repository database is carried out using standard configuration. Some remarks about this phase will be addressed later in this document.

Before the data is accepted by an ECCAIRS system, it must be successfully validated against the taxonomy deployed in the repository. This validation process verifies the semantics of the XML file, the existence of attributes and entities and the correctness of the provided attribute values following the definitions given in an XSD schema pertaining to the taxonomy.

By verifying the XML files against the known XSD schema, a Data Provider can be sure that the data delivered to an ECCAIRS system will be accepted by the Data Integrator deploying the same taxonomy as that from or for which the XSD schema was derived. The verification process can take place at the site of the Data Provider. Since the XSD schema is made available independent from the ECCAIRS software, and the supported XML files and the related XSD schema are W3C compliant, XML files can be generated using standard tools and technologies and occurrence data be produced without the need to install any ECCAIRS software.

The E5X format is implemented as a core function of the ECCAIRS ecosystem.

In this White Paper, examples are presented for both the XSD as well as for the XML. It has been chosen to base some of the peculiar settings on the Aviation Extension but the principle and overall guidelines are valid for any taxonomy extension. Also note that the examples should not be taken as a specification: in the end, it is the XSD that provides the full specification of the format of the data.

To clearly distinguish the XML examples from the XSD definitions, the XSD definitions are written in blue text; the XML examples are in black text.

The XML or XSD examples or code snippets sometimes include the ellipsis '…' which is an intentional omission of sections from a larger text for highlight fragments which are relevant to the text mentioned in the surrounding paragraphs.

4 The Data Bridge

4.1 Technologies and Standards

The XML format supported by the ECCAIRS Data Bridge follows the W3C standard for XML files[5]. The XSD (XML Schema Definition language) format supported by the ECCAIRS Data Bridge follows the W3C standard for XSD files[6]. The XML file can be generated and validated inside different software environments. The following environments have been used for compliance testing:

- XML Spy
- Microsoft .NET Framework with C# language
- Eclipse

4.2 The Data Bridge Package

An ECCAIRS Data Bridge package is a compressed file based on the ZIP standard but with the E5X extension. Occurrences are located in the root folder of the Data Bridge package as individual XML files, and it is indeed possible to include multiple XML files in a single Data Bridge package. The actual filename of the individual XML is not significant and is anyway discarded during the conversion into E5F format. The Data Bridge package should preferably have the recommended E5X extension, but the occurrence files must have the XML extension.

Sample E5X file containing just one occurrence:

```
🗀 DataBridgeContainingOneOccurrence.e5x
   🗎 Occurrence_1.xml
```

Sample E5X file containing multiple occurrences:

```
🗀 DataBridgeContainingMultipleOccurrences.e5x
   🗎 Occurrence_1.xml
   🗎 Occurrence_2.xml
   🗎 ...
   🗎 Occurrence_X.xml
```

The ZIP-format E5X file can be created using the standard ZIP compression functions available in practically all IT platforms. It is recommended not to use extreme compression settings for obtaining the smallest possible file size, because these may not be supported by the decompression module of the ECCAIRS system. A simple test to see if the packaging has been done correctly is to rename the E5X

[5] http://www.w3.org/2001/XMLSchema

[6] http://www.w3.org/TR/xmlschema11-1/

extension to ZIP and open this file using Windows' Explorer ([⊞+E]). By the way, Windows Explorer can be used for creating the E5X files with the correct format (the algorithm is 'deflate' and the compression 'medium').

4.2.1 Occurrence Attachments

The ECCAIRS Data Bridge package support also attachments to ECCAIRS occurrences. Attachments are stored in a dedicated *attachments folder* located at the root level of the compressed file, and the folder name is that of the occurrence to which the attachments belong to.

Above it is indicated that the filename of the occurrence XML is not significant but this is not true for the attachments stored the attachments folder. In here, each item shall be stored with its real or original filename because on one hand this is how these attachments are mapped in the occurrence XML and on the other hand it is with that filename that they are embedded when producing the E5F format.

> **NOTE:** The filename of the attachments in the *attachments folder* is significant. During conversion, the filename of the attachment is not discarded but converted into the E5F format with the same name.

The filenames of the attachments are only unique within the same attachment folder but can be present in other attachment folders. It is not necessary to create an attachment folder for every occurrence but only for those that indeed carry attachments.

Sample E5X file with multiple occurrences and attachments for a limited set of occurrences:

```
DataBridgeContainingMultipleOccurrencesAndSomeAttachments.e5x
   Occurrence_1.xml
   Occurrence_2.xml              ┐
   Occurrence_2                  │
      MaintenceReport.pdf        │    Occurrence with attachments
      AwsomePicture.png          ┘
   Occurrence_3.xml
   Occurrence_4.xml
   Occurrence_5.xml              ┐
   Occurrence_5                  │
      MaintenceReport.pdf        │    Occurrence with attachments
      EngineCowling.jpg          │
      BrokenTailPipe.jpg         ┘
```

In the above sample, '`MaintenceReport.pdf`' is found in 2 different attachment folders. Though the name chosen is the same, they are definitely not the same. Attachments are linked to the occurrence that has the same name as the attachment folder. It is not possible to use attachments from different attachments folder.

4.3 The XSD Schema files

Each ECCAIRS extension will provide an XML Schema Definition (XSD), which is used by the ECCAIRS software to validate the XML files inside an E5X Data Bridge package. The same XSD can be used by Data Providers to validate their output before sending it to their Data-Integrator. It is important to be aware that:

- The provided XSD schema is made up of several files:
  - `Schema.xsd` – The Master Schema containing also references to the other XSD schemas. This XSD file is the one to be used when validating an XML occurrence file
  - `ECCAIRS_datatypes.xsd` – This file contains the general definitions for the XML structures used to store attribute values
  - `VLx_x_x_x_x.xsd` – These XSD files contain the list of the admitted values for attributes with a 'predefined value list', and are included by '`Schema.xsd`'
- The XSD files contain 'annotations' that describe the meaning of each element. These annotations only serve to offer better comprehensibility, they are not used for verification and/or validation
- The XSD Schemas are generated from an ECCAIRS specific taxonomy version. This guarantees that all the validated XML occurrences can be converted into an ECCAIRS Occurrence related to the same taxonomy version
- XML occurrences will also be accepted if the ECCAIRS target system uses a newer version of the same taxonomy[7].

4.4 The XML Occurrence File

The XML Occurrence File actually is the only element to be produced by the Data Provider using the data collected in their proprietary system. This file is packaged in the Data Bridge E5X file.

The ECCAIRS system used '`utf-8`' encoding so that it can support a large character set. Hence, the XML file that contains the occurrence data has to be encoded using the same '`utf-8`'; this encoding type must be specified in the XML file prologue. XSD's are encoded '`utf-8`' as well.

The next code snippet illustrates the basic structure of the XML file:

```
<?xml version="1.0" encoding="utf-8"?>
<SET>
    <OCCURRENCE>
    ...
    </OCCURRENCE>
</SET>
```

The XML uses several tags. Some of these are taxonomy independent (*) and are used to identify specific information or to group homogeneous elements. These are:

[7] Requires a correct configuration of the target repository; in particular the Data Bridge profile(s) and the Conversion Rules need to be added and/or configured.

- SET
- Occurrence (*)
  - ENTITIES
  - ATTRIBUTES
  - LINKS

(*)The name 'Occurrence' depends on the taxonomy in use. See §5.2 entitled OCCURRENCE on page 13 for more detail. The others are valid for any taxonomy.

The rest of the tags in the XML are fully taxonomy dependent and generated from the names given to entities and attributes in a specific ECCAIRS taxonomy.

Please remind that all tags are case sensitive.

5 Supported Elements

5.1 SET

The SET tag is the root node of the XML Occurrence file. The SET tag contains several attributes defining the applicable namespace and the target ECCAIRS taxonomy.

- Applicable namespace and schema attributes
  - xmlns – Data Bridge main namespace (required)
  - xmlns:dt(*) – ECCAIRS data types definition namespace (optional)

    (*) the 'dt' prefix is chosen for representing the initials of 'data' and 'types', but in the XML you can define your own prefix as long as it unambiguously identifies a single namespace. The inclusion of the ECCAIRS data types definition namespace is optional but if omitted, requires to explicitly declare each proprietary datatype with its proper namespace.
- ECCAIRS system, taxonomy and domain attributes (all required)
  - TaxonomyName – Name of current taxonomy
  - TaxonomyVersion – Version of current taxonomy
  - Domain – Named subset of taxonomy attributes for the XML Data Bridge
  - Version – XSD structure version; fixed value '1.0.0.0' for the current structure

The values for the above attributes are defined in Schema.xsd in the XSD block that defines the SET. Highlighted in the SET is the 'Occurrence' element described above:

```
<xs:element name="SET">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:sequence>
            <xs:element name="Occurrence" type="db:Occurrence" minOccurs="1" maxOccurs="1" />
        </xs:sequence>
        <xs:attribute name="TaxonomyName" type="xs:string" use="required" fixed="ECCAIRS Aviation" />
        <xs:attribute name="TaxonomyVersion" type="xs:string" use="required" fixed="4.1.0.3" />
        <xs:attribute name="Domain" type="xs:string" use="required" fixed="RIT" />
        <xs:attribute name="Version" type="xs:string" use="required" fixed="1.0.0.0" />
    </xs:complexType>
    ...
```

Sample XML simple occurrence using Data Bridge and ECCAIRS data type namespaces:

```
<?xml version="1.0" encoding="UTF-8"?>
<SET xmlns="http://eccairsportal.jrc.ec.europa.eu/ECCAIRS5_dataBridge.xsd"
    xmlns:dt="http://eccairsportal.jrc.ec.europa.eu/ECCAIRS5_dataTypes.xsd"
    TaxonomyName="ECCAIRS Aviation" TaxonomyVersion="4.1.0.3" Domain="RIT" Version="1.0.0.0">
    <Occurrence>
        ...
        <Report attributeId="802">
            <dt:FileName>report.pdf</FileName>
        </Report>
        ...
    </Occurrence>
</SET>
```

Sample XML simple occurrence using only Data Bridge namespace:

```
<?xml version="1.0" encoding="UTF-8"?>
<SET xmlns="http://eccairsportal.jrc.ec.europa.eu/ECCAIRS5_dataBridge.xsd"
    TaxonomyName="ECCAIRS Aviation" TaxonomyVersion="4.1.0.3" Domain="RIT" Version="1.0.0.0">
    <Occurrence>
        ...
        <Report attributeId="802">
            <FileName xmlns="http://eccairsportal.jrc.ec.europa.eu/ECCAIRS5_dataTypes.xsd">
                report.pdf</FileName>
        </Report>
        ...
    </Occurrence>
</SET>
```

5.2 Occurrence

The 'Occurrence' node is the container for the actual data to be converted into the E5F formatted occurrence. The node element corresponds to the taxonomy's root entity and is represented by the `Occurrence` tag. The name 'Occurrence' comes from the Aviation taxonomy; for other taxonomies it could be different (e.g. 'Casualty', 'Incident', 'Notification'…).

The Occurrence is a set of logically structured and interrelated Entities, Attributes and Links (using the `ENTITIES`, `ATTRIBUTES` and `LINKS` tags) following the definition of the current taxonomy. Each physical entity is composed by three main nodes:

- `ATTRIBUTES` – contains the definition of attributes that belong to the current physical entity; if the current physical entity doesn't have any attributes then this node doesn't exist
- `ENTITIES` – contains the physical child entities that belong to the current physical entity; if the current physical entity doesn't have any physical child entities then this node doesn't exist
- `LINKS` – contains the linked child entities belonging to the current physical entity; if the current physical entity doesn't have any linked child entities then this node doesn't exist

A physical entity is an entity containing data elements (i.e. attributes specified under the `ATTRIBUTES` node). A linked entity does not contain data elements but specifies a link to a physical entity.

Physical entities are described in §5.4 ENTITIES on page 29, linked entities in §5.5 LINKS on page 31.

5.3 Attributes

The `ATTRIBUTES` node inside an entity contains all the available attributes under that entity. Next is an extract of the XSD attribute definition inside the 'Occurrence' entity, the example only lists the first 7 attributes:

```
<xs:complexType name="Occurrence">
    <xs:all>
        <xs:element name="ATTRIBUTES" minOccurs="0" maxOccurs="1">
            <xs:complexType>
                <xs:sequence minOccurs="1" maxOccurs="1">
                    <xs:element name="Dew_Point" minOccurs="0" maxOccurs="1"> ... </xs:element>
                    <xs:element name="Wx_Conditions" minOccurs="0" maxOccurs="1"> ... </xs:element>
                    <xs:element name="Dang_Goods_Involved" minOccurs="0"
                        maxOccurs="1"> ... </xs:element>
                    <xs:element name="Height_Of_Cloud_Base" minOccurs="0"
                        maxOccurs="1"> ... </xs:element>
                    <xs:element name="Light_Conditions" minOccurs="0"
                        maxOccurs="1"> ... </xs:element>
                    <xs:element name="Maximum_Gust" minOccurs="0" maxOccurs="1"> ... </xs:element>
                    <xs:element name="Cloud_Amount" minOccurs="0" maxOccurs="1"> ... </xs:element>
                    ...
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        ...
    </xs:all>
    <xs:attribute name="entityId" type="xs:string" fixed="24" />
</xs:complexType>
```

Attributes in the XML shall be inserted in the order in which they are defined inside the `sequence` node (highlighted above) of the XSD schema. The XSD generation algorithm uses the attribute identifier assigned in the taxonomy and sorts these in ascending order. This can be seen in the above example, where the first attributes are not listed alphabetically but by their taxonomy ID which is '85', '127', '129', '140', 168', '176' and '266', respectively. Not respecting this same order in the XML will fail the integrity check.

It is not required to insert all the defined attributes when crafting the XML with the live data, you can restrict to including only those for which you actually supply figures and those which are compulsory.

The attribute's unique identifier `attributeId` that is contained inside each attribute definition has a default but fixed value that corresponds to the attribute identifier inside the taxonomy. It is not mandatory and can be omitted but, if used, must match that of the fixed value.

5.3.1 Attribute Types

The XSD schema defines for each attribute the name, type, cardinality and a unique identifier. All those properties are derived from the attribute definition in the taxonomy.

5.3.1.1 ECCAIRS Data Types

The taxonomy foresees a certain number of data types, each with peculiar characteristics, which allow submitting through the XML the variety of data that the ECCAIRS system can manage, which are:

ECCAIRS Data Type	Description

ECCAIRS Data Type	Description
Alphanumeric	Alphanumeric strings with a maximum length of 255 characters
Code	Value which represents a single entry of a predefined list
Code and additional text	Value which represents a single entry of a predefined list plus an optional additional text
Code or alternative text	Value which represents a single entry of a predefined list or an alternative text, but not both
Date	Date
Date and Time	Date and Time
Decimal	Numbers with fractional digits
ECCAIRS Data Link	Reference to an instance out of a defined ECCAIRS repository
Latitude	Latitude coordinate
Longitude	Longitude coordinate
Number	Numbers without fractional digits (aka Integer)
Resource Locator	Structure for holding references to files (PDF, images, videos…) or to other types of resources (URLs)
Text	Text with a possible length of over 255 characters
Time	Time

5.3.1.2 XSD Data Types

To support the above data types offered by the ECCAIRS ecosystem, some dedicated configurations have been created managing specific types that are not natively supported by XSD or that require adequate treatment before data can be imported, for instance by establishing length, accepted characters or digits, and so on. In the root node named ‘`schema`’ in `Schema.xsd`, a link is set to `ECCAIRS5_dataTypes.xsd` containing such dedicated configurations:

```
<?xml version="1.0" encoding="utf-8"?>
    <xs:schema xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning"
        xmlns:db="http://eccairsportal.jrc.ec.europa.eu/ECCAIRS5_dataBridge.xsd"
        xmlns:dt="http://eccairsportal.jrc.ec.europa.eu/ECCAIRS5_dataTypes.xsd"
        targetNamespace="http://eccairsportal.jrc.ec.europa.eu/ECCAIRS5_dataBridge.xsd"
        elementFormDefault="qualified" attributeFormDefault="unqualified" vc:minVersion="1.1"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
        ...
```

A number of defaults are defined, with the following limits:

XSD Default Data Type	Properties
`DataLink`	Complex structure used for linking the XML record to an instance of an ECCAIRS occurrence in a separate repository
`DateTime`	As standard `datetime` but without indication of time zones

XSD Default Data Type	Properties
`Decimal`	Number with fractional digits between -9,999,999,999,999 and +9,999,999,999,999
`EmbeddedData`	Complex structure used for embedding an ECCAIRS occurrence in the Data Bridge package and linking it to the XML record
`keyString`	String with exact length of 32 characters accepting only capital letters (A-Z) and numeric digits (0-9)
`LatitudeDecimal`	Number with fractional digits between -90 and +90
`LongitudeDecimal`	Number with fractional digits between -180 and +180
`ResourceLocator`	Complex structure for storing Universal Resource Locator (URL) and optional description
`string20max`	String with up to 20 characters
`string30max`	String with up to 30 characters
`string50max`	String with up to 50 characters
`string255max`	String with up to 255 characters
`Text`	Complex structure for storing plain text OR binary encoded text
`Time`	As standard `time` but without indication of time zones

Please note that the above data types are generated and valid for any taxonomy but that a particular taxonomy, or a sub-set of attributes thereof, does not automatically require using them, like is the case for the current RIT/E5X.

In the following paragraphs, the data types listed above will be explained and an example of corresponding valid XML will be given, using corresponding attributes from the Aviation taxonomy.

5.3.2 Alphanumeric

Alphanumeric attributes can hold strings of text with a maximum length of up to 255 characters.

Example 1 – Headline

The Headline attribute with ID 601 is an alphanumeric field with a maximum accepted size of 255 characters. It is defined in the XSD as:

```
<xs:element name="Headline" minOccurs="0" maxOccurs="1">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleContent>
            <xs:extension base="dt:string255max">
                <xs:attribute name="attributeId" type="xs:string” fixed="601" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

The attribute definition is based on the default data type `string255max` (see previous paragraph).

A valid XML entry is:

```
<Headline attributeId="601" >Helicopter crashed during emergency landing</Headline>
```

Example 2 – Airspace Name

The Airspace name attribute (ID 14) is also Alphanumeric but it has a different maximum size limit (up to 20 characters) so it is defined in the XSD as:

```
<xs:element name="Airspace_Name" minOccurs="0" maxOccurs="1">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleContent>
            <xs:extension base="db:Base_Airspace_Name">
                <xs:attribute name="attributeId" type="xs:string" fixed="14" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

To support strings lower than 255 characters, an ad-hoc created *base* type is created in the XSD instead of using a default data type:

```
<xs:simpleType name="Base_Airspace_Name" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:restriction base="xs:string">
        <xs:maxLength value="20" />
    </xs:restriction>
</xs:simpleType>
```

The base type name is given by prefixing the attribute name with '`Base_`': '`Base_Airspace_Name`'. Base type definitions are grouped after the entity structure definition.

A valid XML entry is:

```
<Airspace_Name>KOTVRDOVICE</Airspace_Name>
```

Example 3 – Runway Identifier

The Runway Identifier (ID 499) is an alphanumeric string of up to 3 characters and shall accept only uppercase characters or digits. This contain is enforced by an ad-hoc base type:

```
<xs:element name="Runway_Identifier" minOccurs="0" maxOccurs="1">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleContent>
            <xs:extension base="db:Base_Runway_Identifier">
                <xs:attribute name="attributeId" type="xs:string" fixed="499" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

The attribute definition is based on the following base type through `maxLength` and `pattern`:

```
<xs:simpleType name="Base_Runway_Identifier" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:restriction base="xs:string">
        <xs:maxLength value="3" />
        <xs:pattern value="([A-Z0-9])*" />
    </xs:restriction>
</xs:simpleType>
```

A valid XML entry is:

```
<Runway_Identifier attributeId="499">03L</ Runway_Identifier>
```

NOTE: Special characters must be written in their HTML-safe equivalents. For instance, the 'less than' character '<' must be written as '`&lt;`'.

5.3.3 Coded Values

Coded attributes are associated to predefined values lists, and the only accepted values are the ones defined in the associated predefined value list. Each predefined value list contains the value identifiers as defined in the taxonomy and the list is not sorted when generated (e.g. alphabetically), following the order in which they are enumerated in the taxonomy.

Some predefined value lists are shared between attributes and the accepted values are consequently the same for each attribute. This is for instance the case with Location Indicator (ID 5), Last Departure Point (ID 167) and Planned Destination (ID 228).

The documented schema[8] includes corresponding description, detailed description and explanation for each value contained in a predefined value list.

The naming convention of the XSD files containing the predefined value lists uses the following format: '`VLaa_bb_cc_dd_ee.xsd`', made up of these parts separated by the '_' character:

Part	Properties
`'VL'`	Fixed prefix for Value Lists
`aa`	Unique Value List identifier
`bb`	Data type of attribute(s) using the value list. Can be one of the three possible values: • 5 – Code • 12 – Code and Additional Text • 13 – Code or Alternative Text
`cc`	Branches of the value list included (multi-level lists only): • 0 – All branches are included • *nn* – Only the branch having value '*nn*' and all of its child-value are included
`dd`	Levels of the value list included (multi-level lists only): • 1 – All levels are included • 0 – Only the levels indicated at '`ee`' are included
`ee`	Number of levels* included (independent of the value at '`dd`')

[8] The documented schema contains XML '`annotations`' which are not generated for the normal schema in order to minimise the size of the various XSD files. It can be downloaded from the ECCAIRS Community Portal.

Part	Properties
`'.xsd'`	Fixed extension

(*) Levels are indicated starting at '1' with increasing number for deeper levels (aka child levels). Level 1 is the highest level; this number is relative to the starting point '`cc`' and not the absolute level indicator.

5.3.3.1 Code

Code is based on a predefined value list so that the allowed values are those defined in the corresponding value list.

Example – Occurrence Class

The Occurrence Class (ID 431) is of type Code and associated to a predefined value list. The XSD definition which links it to a particular predefined value list is:

```
<xs:element name="Occurrence_Class" minOccurs="0" maxOccurs="1">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleContent>
            <xs:extension base="db:VL1015_5_0_1_2">
                <xs:attribute name="attributeId" type="xs:string" fixed="431" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

Highlighted above is association between attribute and predefined value list identifier.

The predefined value list is stored in a separated file called `VL1015_5_0_1_2.xsd` that is mapped in the XSD through the `xs:include` statement. The '`db:`' prefix is defined inside the root node named '`schema`' in `Schema.xsd`. It indicates the standard *namespace* which is '`ECCAIRS5_dataBridge.xsd`' and that is also the main namespace for the predefined value lists definition:

```
<?xml version="1.0" encoding="utf-8"?>
    <xs:schema xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning"
        xmlns:db="http://eccairsportal.jrc.ec.europa.eu/ECCAIRS5_dataBridge.xsd"
        xmlns:dt="http://eccairsportal.jrc.ec.europa.eu/ECCAIRS5_dataTypes.xsd"
        targetNamespace="http://eccairsportal.jrc.ec.europa.eu/ECCAIRS5_dataBridge.xsd"
        elementFormDefault="qualified" attributeFormDefault="unqualified" vc:minVersion="1.1"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">
    ...
    <xs:include schemaLocation="VL1015_5_0_1_2.xsd" />
    ...
```

A valid XML entry is:

```
<Occurrence_Class attributeId="431">100</Occurrence_Class>
```

5.3.3.2 Code and Additional Text

Code and Additional Text is based, like Code, on a predefined value list so that the allowed values are those defined in the corresponding value list. Optional text can be included inside a related attribute called `AdditionalText`. The text encoding, indicated with `AdditionalTextEncoding`, can be either a standard `xs:string` or `xs:base64binary`[9].

Example – State or Area of Occurrence

The State or Area of Occurrence attribute (ID 454) is associated to the 'Geographical areas' predefined value list and optionally accepts some free text for providing more detail about a particular landmark in the chosen area (e.g. "Lago di Garda" in "Italy"). It is defined in the XSD as follows:

```
<xs:element name="State_Area_Of_Occ" minOccurs="0" maxOccurs="1">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleContent>
            <xs:extension base="db:VL1090_12_0_1_3">
                <xs:attribute name="attributeId" type="xs:string" fixed="454" />
                <xs:attribute name="AdditionalText" />
                <xs:attribute name="AdditionalTextEncoding">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="xs:string" />
                            <xs:enumeration value="xs:base64Binary" />
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

An example of valid XML using `string` for the additional text is:

```
<State_Area_Of_Occ attributeId="454" AdditionalText="Lago di Garda"
    AdditionalTextEncoding="xs:string">122</State_Area_Of_Occ>
```

An example of valid XML using `base64binary` encoding for the additional text is:

```
<State_Area_Of_Occ AdditionalText="TGFnbyBkaSBHYXJkYQ=="
    AdditionalTextEncoding="xs:base64Binary">122</State_Area_Of_Occ>
```

5.3.3.3 Code or Alternative Text

The Code or Alternative Text attribute is similar to Code and Additional Text illustrated above, the difference being the handling of the Text part. The Code part is based on a predefined value list so that the allowed values are those defined in the corresponding value list. Instead of the code, it is possible to specify text inside a related attribute called `AlternativeText`. The text encoding, indicated with `AlternativeTextEncoding`, can be either a standard `xs:string` or `xs:base64binary`. If an alternative text is set then any code value inserted will be discarded, since the text receives priority over the code.

[9] The conversion of a string in format base64 should be done in the following way:

- Conversion of the string in a byte array using UTF-8 encoding
- Conversion of the byte array in base64 using base64 transfer encoding for MIME (RFC 2045)

Example – FIR/UIR Name

The FIR/UIR Name attribute (ID 16) is associated to a predefined value list but allows setting an alternative text if a FIR/UIR name is not available. It is defined in the XSD as follows:

```
<xs:element name="FIR_UIR_Name" minOccurs="0" maxOccurs="1">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleContent>
            <xs:extension base="db:VL1084_13_0_1_2">
                <xs:attribute name="attributeId" type="xs:string" fixed="16" />
                <xs:attribute name="AlternativeText" />
                <xs:attribute name="AlternativeTextEncoding">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="xs:string" />
                            <xs:enumeration value="xs:base64Binary" />
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

An example of valid XML using the code:

```
<FIR_UIR_Name attributeId="16" AlternativeText=""
    AlternativeTextEncoding="xs:string">219</FIR_UIR_Name>
```

An example of valid XML using only an alternative text with encoding of type `string`:

```
<FIR_UIR_Name attributeId="16" AlternativeText="LIMC: Milano Malpensa (FIR)"
    AlternativeTextEncoding="xs:string"></FIR_UIR_Name>
```

The same example with alternative text, but also with a code (`219`) which is discarded:

```
<FIR_UIR_Name attributeId="16" AlternativeText="LIMC: Milano Malpensa (FIR)"
    AlternativeTextEncoding="xs:string">219</FIR_UIR_Name>
```

5.3.4 Date

Date attributes accept dates. This data type follows standard XML/XSD format, a Gregorian calendar date in the format `CCYY-MM-DD` where `CC` represents the century, `YY` the year, `MM` the month and `DD` the day. No left truncation is allowed for any part of the date. Dates shall not include time zones as these are not managed.

Example – Local Date

The Local Date (ID 433) is defined inside the XSD file with the standard type `xs:date`:

```
<xs:element name="Local_Date" minOccurs="0" maxOccurs="1">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleContent>
            <xs:extension base="xs:date">
                <xs:attribute name="attributeId" type="xs:string" fixed="433" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
```

```
</xs:element>
```

An example of valid XML is:

```
<Local_Date attributeId="433">2001-09-11</Local_Date>
```

5.3.5 DateTime

Dates that also include a Time part are called DateTime. This data type follows standard XML/XSD format, representing a specific date and time in the format `CCYY-MM-DDThh:mm:ss.sss` which is a concatenation of the date and time forms, separated by a literal letter '`T`'. All of the same rules that apply to the date and time types are applicable to DateTime as well. Time zones are explicitly not accepted.

Example – Modification Date

The attribute Modification Date (ID 422) has type DateTime and is defined inside the XSD file as follows:

```
<xs:element name="Modification _Date" minOccurs="0" maxOccurs="1">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleContent>
            <xs:extension base="dt:DateTime">
                <xs:attribute name="attributeId" type="xs:string" fixed="422" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

The attribute definition is based on the following default data type:

```
<xs:simpleType name="DateTime">
    <xs:restriction base="xs:dateTime">
        <xs:pattern value=".+T[^Z+-]+"/>
    </xs:restriction>
</xs:simpleType>
```

An example of valid XML is:

```
<Modification_Date attributeId="422">2001-12-17T09:30:47</Modification_Date>
```

5.3.6 Decimal

Attributes of type Decimal accept numbers with fractional digits. Decimal is also used for attributes that have associated a measurement unit. Supplementary restrictions are defined for some of them, like for instance lower and/or upper limit or maximum decimal digits, and have therefore a dedicated XSD configuration. The XML standard decimal separator is the point '.' character.

Example 1 – Risk Level

Attribute Risk Level (ID 940) has specific lower and upper limits (0 and 100) and is defined in the XSD file as follows:

```
<xs:element name="Risk_Level" minOccurs="0" maxOccurs="1">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleContent>
            <xs:extension base="db:Base_Risk_Level">
                <xs:attribute name="attributeId" type="xs:string" fixed="940" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

The attribute definition is based on the following dedicated base type:

```
<xs:simpleType name="Base_Risk_Level" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:restriction base="xs:decimal">
        <xs:minInclusive value="0" />
        <xs:maxInclusive value="100" />
        <xs:fractionDigits value="50" />
    </xs:restriction>
</xs:simpleType>
```

The maximum number of decimal digits that can be used is defined by `xs:fractionDigits`; this value defaults to 50.

Please note that the actual number of decimal digits that are accepted by an ECCAIRS ecosystem depend largely on the database that is linked to the repository. For instance Microsoft SQL Server[10] limits the number of decimal digits to 10.

An example of valid XML is:

```
<Risk_Level attributeId="940">52.31565</Risk_Level>
```

Example 2 – Dew Point

Attribute Dew point (ID 85) is also of a Decimal type with specific limits but has in addition a measurement unit. A base type is created defining the limits while unit of measurement is specified in attribute definition. The following XSD block defines the Dew Point attribute:

```
<xs:element name="Dew_Point" minOccurs="0" maxOccurs="1">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleContent>
            <xs:extension base="db:Base_Dew_Point">
                <xs:attribute name="Unit" use="required" fixed ="C" />
                <xs:attribute name="attributeId" type="xs:string" fixed="85" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

An XML attribute called `Unit` is defined to set the unit of measurement. Its fixed value corresponds to attribute default unit. The attribute definition is based on the following base type:

```
<xs:simpleType name="Base_Dew_Point" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:restriction base="xs:decimal">
        <xs:minInclusive value="-100" />
        <xs:maxInclusive value="100" />
        <xs:fractionDigits value="50" />
    </xs:restriction>
</xs:simpleType>
```

An example of valid XML is:

[10] https://msdn.microsoft.com/en-us/library/ms190665(v=sql.100).aspx

```
<Dew_Point attributeId="85" Unit="C">-8.453</Dew_Point>
```

NOTE: The measurement unit indication is required and it must correspond to the value given at `fixed`. This is to assure that the information collected is consistent with the lower and upper limits which are indeed set and enforced for the only accepted measurement unit.

5.3.7 Geo-coordinates

Geographical coordinates are entered as degrees using the decimal datatype, using the fractional part for indicating the minutes and seconds of the degree with the following formula:

```
DD + (mm/60) + (ss/(60*60))
```

The XML standard decimal separator is the point '.' character. The geographical coordinates are inserted in the same format as for instance with Google Maps.

5.3.7.1 Latitude

Latitudes accept a value between -90 and +90, inclusive. Negative values represent coordinates in the Southern hemisphere, positive values the Northern hemisphere.

Example – Latitude of Occurrence

The attribute Latitude of Occ (ID 439) has type `LatitudeDecimal` and is defined inside the XSD file as follows:

```
<xs:element name="Latitude_Of_Occ" minOccurs="0" maxOccurs="1">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleContent>
            <xs:extension base="dt:LatitudeDecimal">
                <xs:attribute name="attributeId" type="xs:string" fixed="439" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

The attribute definition is based on the following default data type:

```
<xs:simpleType name="LatitudeDecimal">
    <xs:restriction base="xs:double">
        <xs:minInclusive value="-90.0"/>
        <xs:maxInclusive value="90.0"/>
    </xs:restriction>
</xs:simpleType>
```

An example of valid XML, with the latitude of the main entrance of the JRC site in Ispra, Italy, is:

```
<Latitude_Of_Occ attributeId="439">45.80383</Latitude_Of_Occ>
```

5.3.7.2 Longitude

Longitudes are indicated with a value between -180 and +180, inclusive. Negative values represent coordinates in the Western hemisphere, positive values the Eastern hemisphere. The zero corresponds to the IERS Reference Meridian.

Example – Longitude of Occurrence

The attribute Longitude of Occ (ID 444) has type `LongitudeDecimal` and is defined inside the XSD file as follows:

```
<xs:element name="Longitude_Of_Occ" minOccurs="0" maxOccurs="1">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleContent>
            <xs:extension base="dt:LongitudeDecimal">
                <xs:attribute name="attributeId" type="xs:string" fixed="444" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

The attribute definition is based on the following default data type:

```
<xs:simpleType name="LongitudeDecimal">
    <xs:restriction base="xs:double">
        <xs:minInclusive value="-180.0"/>
        <xs:maxInclusive value="180.0"/>
    </xs:restriction>
</xs:simpleType>
```

An example of valid XML, with the longitude of the main entrance of the JRC site in Ispra, Italy, is:

```
<Longitude_Of_Occ attributeId="444">8.62965</Longitude_Of_Occ>
```

5.3.8 Number

The Number data type accepts integer, that is, without fractional digits.

Example – Total Number of Persons

Attribute Total Number of Persons on board of the aircraft (ID 152), which can only store whole numbers, has specific lower and upper limits (0 and 9999) and is defined in the XSD file as follows:

```
<xs:element name="Total_Number_Of_Persons" minOccurs="0" maxOccurs="1">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleContent>
            <xs:extension base="db:Base_Total_Number_Of_Persons">
                <xs:attribute name="attributeId" type="xs:string" fixed="152" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

The attribute definition is based on the following dedicated base type:

```
<xs:simpleType name="Base_Total_Number_Of_Persons" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:restriction base="xs:int">
        <xs:minInclusive value="0" />
        <xs:maxInclusive value="9999" />
    </xs:restriction>
</xs:simpleType>
```

An example of valid XML is:

```
<Total_Number_Of_Persons attributeId="152">27</Total_Number_Of_Persons>
```

5.3.9 Resource Locator

The Resource Locator is a data type used for linking non-ECCAIRS data to an occurrence. With the Resource Locator one can 'store and embed' digital documents, such as images and PDF files, or resources available on the Internet by means of a URL, into the XML. The inclusion of the digital documents in the Data Bridge package is implemented maintaining the original format of the files. See also §4.2.1 OCCURRENCE ATTACHMENTS on page 10.

Example – Attached Reports

The Reports (ID 802) attribute is an attribute based on resource locator data type, and defined as:

```
<xs:element name="Report" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:complexContent>
            <xs:extension base="dt:ResourceLocator">
                <xs:attribute name="attributeId" type="xs:string" fixed="802" />
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
```

The attribute definition is based on the following default data type:

```
<xs:complexType name="ResourceLocator">
    <xs:sequence>
        <xs:element name="FileName" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="Description" type="xs:string" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
</xs:complexType>
```

A valid XML entry is:

```
<Attachments attributeId="793">
    <dt:FileName>Eurocopter 2800 front window (12-03-2017).png</dt:FileName>
    <dt:Description>Huge crack over entire height</dt:Description>
</Attachments>
```

Note – The `FileName` attribute does not require leading and trailing quotes when containing spaces.

5.3.10 Text

Text data type is similar to Alphanumeric but the size limit of a maximum of 255 characters does not apply, therefore often this data type is also called 'long text'. Due to the particular nature of text data type in ECCAIRS, which is able to store also so-called 'rich text' having text formatting marks (bold, indents, bullets…), the Text definition foresees 2 compulsory and mutually exclusive attributes named `PlainText` and `EncodedText`, based on the XSD data type `xs:string` and `xs:base64binary`, respectively.

The `PlainText` attribute is to be used for unformatted text, and is taken "as is" including any line break or indent placed between the tags. The only remark about plain text is to use HTML-safe equivalents for special characters.

To provide 'rich text' as data for a Text attribute in ECCAIRS you must use the `EncodedText` attribute and provide the data as RTF text encoded in `base64binary`.

Example 1 – Plain Narrative Text

The Narrative Text attribute (ID 425) is a 'long text' element, and defined as:

```
<xs:element name="Narrative_Text" minOccurs="0" maxOccurs="1">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:complexContent>
            <xs:extension base="dt:Text">
                <xs:attribute name="attributeId" type="xs:string" fixed="425" />
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
```

The attribute definition is based on the following default data type:

```
<xs:complexType name="Text">
    <xs:choice minOccurs="1" maxOccurs="1">
        <xs:element name="PlainText" type="xs:string"/>
        <xs:element name="EncodedText" type="xs:base64Binary"/>
    </xs:choice>
</xs:complexType>
```

The `PlainText` and `EncodedText` are mutual choices in the definition which means that they cannot be used together in the same node.

Examples of valid XML are:

```
<Narrative_Text attributeId="425">
    <dt:PlainText>This is a plain text narrative.</dt:PlainText>
</Narrative_Text>
```

The same text as above, but in `base64binary` encoding:

```
<Narrative_Text attributeId="425">
    <dt:EncodedText>VGhpcyBpcyBhIHBsYWluIHRleHQgbmFycmF0aXZlLg==</dt:EncodedText>
</Narrative_Text>
```

Example 2 – Rich Narrative Text

Examples of a 'rich text':

Narrative text

This is **BOLD** text and this is **BOLD RED**.

A list of bullets:
- Bullet 1
- Bullet 2
- Bullet 3

In RTF, this translates to:

```
{\rtf1\ansi\ansicpg1252\deff0\deflang1040{\fonttbl{\f0\fswiss\fcharset0 Tahoma;}{\f1\fnil\fcharset2
Symbol;}}
{\colortbl ;\red255\green0\blue0;}
\viewkind4\uc1\pard\ul\b\i\f0\fs28 Narrative text\par
\ulnone\b0\i0\fs16\par
This is \b BOLD \b0 text and this is \cf1\b BOLD RED\cf0\b0 .\par
\par
A list of bullets:\par
\pard{\pntext\f1\'B7\tab}{\*\pn\pnlvlblt\pnf1\pnindent0{\pntxtb\'B7}}\li495 Bullet 1\fs18\par
\fs16{\pntext\f1\'B7\tab}Bullet 2\fs18\par
\fs16{\pntext\f1\'B7\tab}Bullet 3\fs18\par
}
```

Examples of valid XML based on the above RTF text, in `base64binary` encoding:

```
<Narrative_Text attributeId="425">
    <dt:EncodedText>e1xydGYxXGFuc2lcYW5zaWNwZzEyNTJcZGVmZjBcZGVmbGFuZzEwNDB7XGZvbnR0Ymx7XGYwXGZzd2lzc
    1xmY2hhcnNldDAgVGFob21hO317XGYxXGZuaWxcZmNoYXJzZXQyIFN5bWJvbDt9fQ0Ke1xjb2xvcnRibCA7XHJlZDI1NVxncm
    VlbjBcYmx1ZTA7fQ0KXHZpZXdraW5kNFx1YzFccGFyZFx1bFxiXGlcZjBcZnMyOCBOYXJyYXRpdmUgdGV4dFxwYXINClx1bG5
    vbmVcYjBcaTBcZnMxNlxwYXINClRoaXMgaXMgXGIgQk9MRCBcYjAgdGV4dCBhbmQgdGhpcyBpcyBcY2YxXGIgQk9MRCBSRURc
    Y2YwXGIwIC5ccGFyDQpccGFyDQpBIGxpc3Qgb2YgYnVsbGV0czpccGFyDQpccGFyZHtccG50ZXh0XGYxXCdCN1x0YWJ9e1wqX
    HBuXHBubHZsYmx0XHBuZjFccG5pbmRlbnQwe1xwbnR4dGJcJ0I3fX1cbGk0OTUgQnVsbGV0IDFcZnMxOFxwYXINClxmczE2e1
    xwbnRleHRcZjFcJ0I3XHRhYn1CdWxsZXQgMlxmczE4XHBhcg0KXGZzMTZ7XHBudGV4dFxmMVwnQjdcdGFifUJ1bGxldCAzXGZ
    zMThccGFyDQp9</dt:EncodedText>
</Narrative_Text>
```

5.3.11 Time

Time accepts times, a recurring point in any day. This data type follows standard XML/XSD format, representing a specific time in the format `hh:mm:ss`. Time zones are explicitly not accepted.

Example – UTC Time

The attribute UTC Time (ID 478) has type Time and is defined inside the XSD file as follows:

```
<xs:element name="UTC_Time" minOccurs="0" maxOccurs="1">
    <xs:complexType xmlns:xs="http://www.w3.org/2001/XMLSchema">
        <xs:simpleContent>
            <xs:extension base="dt:Time">
                <xs:attribute name="attributeId" type="xs:string" fixed="478" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

The attribute definition is based on the following default data type:

```
<xs:simpleType name="Time">
    <xs:restriction base="xs:time">
      <xs:pattern value=".[^Z+-]+"/>
    </xs:restriction>
</xs:simpleType>
```

An example of valid XML is:

```
<UTC_Time attributeId="478">09:30:47</UTC_Time>
```

5.3.12 Single and Multiple Value Attributes

Single Value attributes represent the majority of ECCAIRS attributes, and accept only 1 (one) value. There are a number of ECCAIRS attributes that support multiple values, independent of their data type. The cardinality is set through the `maxOccurs` attribute in their XSD definition. The accepted amount is any number between 1 and unlimited. Most of the definitions only use the boundaries, either 1 or unlimited. Unlimited is indicated in the XSD as '`unbounded`'.

In the following sample XSD definition, Local Date (ID 433) accepts only 1 value while Occurrence Category (ID 430) accepts multiple values:

```
<xs:element name="Local_Date" minOccurs="0" maxOccurs="1"> ... </xs:element>
<xs:element name="Occurrence_Category" minOccurs="0" maxOccurs="unbounded"> ... </xs:element>
```

Adding multiple values for the same attribute is achieved by entering multiple instances of the XML statement used for specifying a single value. The next statements add a single value for 'Local Date' and 4 values for 'Occurrence Category':

```
<Local_Date>2018-03-22</Local_Date>
<Occurrence_Category attributeId="430">1</Occurrence_Category>
<Occurrence_Category attributeId="430">29</Occurrence_Category>
<Occurrence_Category attributeId="430">5</Occurrence_Category>
<Occurrence_Category attributeId="430">19</Occurrence_Category>
```

5.4 Entities

An `ENTITIES` node contains all the available physical child entities of a particular entity. Entities can be nested inside entities; in this case they are called child entities. With such structure, parent-child-grandchild relations are obtained. In the XML it is not required to insert all entities defined, you can restrict to including only those for which you actually supply data and those which are compulsory.

An example of XSD child entity definition inside the parent entity is:

```
<xs:complexType name="Occurrence">
    <xs:all>
        <xs:element name="ATTRIBUTES" minOccurs="0" maxOccurs="1"> ... </xs:element>
        <xs:element name="ENTITIES" minOccurs="0" maxOccurs="1">
            <xs:complexType>
                <xs:sequence minOccurs="1" maxOccurs="1">
                    ...
                    <xs:element name="Aircraft" type="db:Aircraft" minOccurs="0"
                        maxOccurs="unbounded" />
                    <xs:element name="Events" type="db:Events" minOccurs="0"
                        maxOccurs="unbounded" />
                    <xs:element name="Narrative" type="db:Narrative" minOccurs="0"
                        maxOccurs="unbounded" />
                    ...
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:all>
</xs:complexType>
```

The entities have to be inserted in the order they are defined in the XSD schema since they are contained inside a XSD `sequence` node. According to the XSD generation algorithm, this order is given by the entity identifier assigned in the taxonomy, sorted in ascending order. In the above example the entities appear listed alphabetically on their name; this is purely coincidental, sort is determined by their taxonomy ID which is '4', '14' and '22', respectively. Not respecting this order will fail the integrity check.

Each child entity is described by:

- Name, corresponding to the entity synonym defined in taxonomy
- Type, containing the type name that defines the entity inside the XSD file's entity structure. Each child entity is defined as XSD `complexType` inside the entity structure
- Cardinality

The cardinality indicates with two parameters how many times the child entity can appear under the parent entity. A value of '`0`' under `minOccurs` specifies that the entity may not exist (i.e. zero or more).

If `maxOccurs` is greater than '`1`' then the entity node can be repeated several times inside the `ENTITIES` node in the XML, up to maximum amount; the term '`unbounded`' indicates that there is no upper limit.

The child entities referred to the same parent entity have to be grouped together inside the `ENTITIES` node of the parent entity, for instance:

```
...
<ENTITIES>
    <Aircraft> ... </Aircraft>
    <Aircraft> ... </Aircraft>
    <Events> ... </Events>
    <Narrative> ... </Narrative>
    <Dangerous_Goods> ... </Dangerous_Goods>
    <Risk_Assessment> ... </Risk_Assessment>
</ENTITIES>
...
```

5.4.1 Entity ID

The identifier in the ECCAIRS taxonomy is specified for each entity in the XSD schema using a `xs:attribute` with the name set with `entityId`. Though specified as *string*, the entity ID is actually a number and shall be indicated only using digits. Below is an example for entity 'Occurrence':

```
<xs:complexType name="Occurrence">
    <xs:all> ... </xs:all>
    <xs:attribute name="entityId" type="xs:string" fixed="24" />
</xs:complexType>
```

This attribute is optional and defaults to the only accepted value set with `fixed`. Use of `entityId` can improve readability of an XML but occasionally also intercept a variation, though a quite rare event, in the assignment of an ID in the associated XSD. Omitting the entity ID gives more trust to the XSD and will automatically capture and adjust to a different ID, would this be assigned.

```
...
<ENTITIES>
    <Aircraft entityId="4"> ... </Aircraft>
    <Events entityId="14"> ... </Events>
    <Narrative> ... </Narrative>                ]
    <Dangerous_Goods> ... </Dangerous_Goods>    ]      Entity ID omitted
    <Risk_Assessment> ... </Risk_Assessment>    ]
</ENTITIES>
...
```

5.4.2 Link ID

If an entity can be linked to another entity in the taxonomy structure, its XSD definition also contains an attribute called `ID`. The type `xs:ID` is used for identifiers and has the following characteristics:

- It uniquely identifies an element in an XML document and thus its value must not be repeated within an XML instance

- Its value follows *NCName*[11] rules meaning that it must start with a letter or underscore, and can only contain letters, digits, underscores, hyphens, and periods

Below is an example for entity named 'Aircraft':

```
<xs:complexType name="Aircraft">
    <xs:all> ... </xs:all>
    <xs:attribute name="ID" type="xs:ID" />
    <xs:attribute name="entityId" type="xs:string" fixed="4" />
</xs:complexType>
```

A suggestion for an ID is to use the code '`ID`' followed by the entity's `entityId` followed by entity instance separated with the underscore character. Below is an XML snippet for the entity 'Aircraft', with the suggestion applied:

```
...
<Aircraft ID="ID_4_1" entityId="4">
    <ATTRIBUTES> ... </ATTRIBUTES>
    <ENTITIES> ... </ENTITIES>
    <LINKS> ... </LINKS>
</Aircraft>
<Aircraft ID="ID_4_2">
    <ATTRIBUTES> ... </ATTRIBUTES>
    <ENTITIES> ... </ENTITIES>
    <LINKS> ... </LINKS>
</Aircraft>
...
```

The ID is only declared for entities that can become a target for a link. Indication in the XML of an ID for an entity that is not a link target will result on a validation error. This is further explained in §5.5 LINKS below.

5.5 Links

A `LINKS` node contains all the available *linked* child entities of a particular entity. Links point to a single physical entity but do not inherit any attribute or child entity from that entity. The physical entity is the end-node of the taxonomy structure. Parent-child-grandchild relations based on links are therefore not possible. In the XML it is not required to insert all linked entities defined, you can restrict to including only those for which you actually supply data and those which are compulsory.

An example of XSD linked child entity definition inside the parent entity is:

```
...
<xs:complexType name="Aircraft">
    <xs:all>
        <xs:element name="ATTRIBUTES" minOccurs="0" maxOccurs="1"> ... </xs:element>
        <xs:element name="ENTITIES" minOccurs="0" maxOccurs="1"> ... </xs:element>
        <xs:element name="LINKS" minOccurs="0" maxOccurs="1">
            <xs:complexType>
                <xs:sequence minOccurs="1" maxOccurs="1">
```

[11] https://www.w3.org/TR/1999/WD-xmlschema-2-19990924/#NCName

```
                    <xs:element name="Air_Space" type="db:L-Air_Space" minOccurs="0"
                        maxOccurs="1" />
                    <xs:element name="Events" type="db:L-Events" minOccurs="0"
                        maxOccurs="unbounded" />
                    <xs:element name="Runway" type="db:L-Runway" minOccurs="0" maxOccurs="1" />
                    <xs:element name="Sector" type="db:L-Sector" minOccurs="0" maxOccurs="1" />
                    <xs:element name="Dangerous_Goods" type="db:L-Dangerous_Goods"
                        minOccurs="0" maxOccurs="unbounded" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:all>
    <xs:attribute name="ID" type="xs:ID" />
    <xs:attribute name="entityId" type="xs:string" fixed="4" />
</xs:complexType>
...
```

The linked entities have to be inserted in the order they are defined in the XSD schema since they are contained inside a XSD `sequence` node. According to the XSD generation algorithm, this order is given by the entity identifier assigned in the taxonomy, sorted in ascending order. This can be seen in the above example, where the 5th entity named `Dangerous_Goods` is listed after `Sector`. Not respecting this order will fail the integrity check.

Each linked child entity is described by:

- Name, corresponding to the entity synonym defined in taxonomy
- Type, containing the type name that defines the linked entity inside the XSD file's entity structure. Each child entity is defined as XSD `complexType` inside the entity structure
- Cardinality

The cardinality indicates with two parameters how many times the child entity can appear under the parent entity. A value of '`0`' under `minOccurs` specifies that the entity may not exist (i.e. zero or more). If `maxOccurs` is greater than '`1`' then the entity node can be repeated several times inside the `LINKS` node in the XML, up to maximum amount; the term '`unbounded`' indicates that there is no upper limit.

Below is an example for a linked entity named 'Events':

```
<xs:element name="Events" type="db:L-Events" minOccurs="0" maxOccurs="unbounded" />
```

Using the following ad-hoc type:

```
<xs:complexType name="L-Events">
    <xs:attribute name="REF" type="xs:IDREF" use="required"/>
</xs:complexType>
```

This complex type contains a mandatory attribute called `REF` with `xs:IDREF` type. This is used to refer to the physical entity and it has to contain the linked physical entity identifier that is contained in that entity's `ID` attribute. See §5.4.2 LINK ID on page 30 for more information. It is usually named prefixing with '`L-`' the same name of the linked child entity, for instance '`Events`' and '`L-Events`'

The linked child entities referred to the same parent entity have to be grouped together inside the `LINKS` node of the parent entity, for instance:

```
...
<LINKS>
    <Events ... />
    <Events ... />
    <Runway> ... />
</LINKS>
...
```

Below is an XML snippet with two physical entities 'Aircraft' and a link from somewhere else to one of the two physical entities:

```
...
<ENTITIES>
    <Aircraft ID="ID_4_1">
        <ATTRIBUTES> ... </ATTRIBUTES>
    </Aircraft>
    <Aircraft ID="ID_4_2">
        <ATTRIBUTES> ... </ATTRIBUTES>
    </Aircraft>
</ENTITIES>
...
<LINKS>
    <Aircraft REF="ID_4_2" />
</LINKS>
...
```

Link to physical entity

Note that links do not contain any other data or elements than the reference ID and can be written as so called *self-closing* tag.

Sample XML simple occurrence:

```
<?xml version="1.0" encoding="UTF-8"?>
<SET xmlns="http://eccairsportal.jrc.ec.europa.eu/ECCAIRS5_dataBridge.xsd"
    xmlns:dt="http://eccairsportal.jrc.ec.europa.eu/ECCAIRS5_dataTypes.xsd"
    TaxonomyName="ECCAIRS Aviation" TaxonomyVersion="4.1.0.3" Domain="RIT" Version="1.0.0.0">
    <Occurrence>
        <ATTRIBUTES>
            <Local_Date>2014-08-13</Local_Date>
        </ATTRIBUTES>
        <ENTITIES>
            <Aircraft ID="ID_4_1">
                <ATTRIBUTES>
                    <Manufacturer_Model AdditionalText=""
                        AdditionalTextEncoding="xs:string">18370</Manufacturer_Model>
                    <A_C_Flight_Level>150</A_C_Flight_Level>
                    <Total_Cycles_A_C>10</Total_Cycles_A_C>
                    <Call_Sign>ACR500</Call_Sign>
                    <Serial_Number>EPO879VF</Serial_Number>
                </ATTRIBUTES>
            </Aircraft>
        </ENTITIES>
    </Occurrence>
</SET>
```

6 Unsupported Elements

The following paragraphs list the elements that are not supported in the current RIT/E5X schema. These elements are listed here because they could be supported by other taxonomy schemas or be included in future versions of RIT/E5X.

6.1 Attribute Types

The following attribute types are types that ECCAIRS system can manage, that are included in the definition file but that are not supported by the current E5X RIT schema:

ECCAIRS Data Type	Description
Data Link	Reference to an instance out of a defined ECCAIRS repository
Embedded Data	Binary structure containing a single element from an ECCAIRS repository (comparable to a 'record' from a database)

7 RIT/E5X Validation

It is useful to check the produced XML and validate it against the Data Bridge before submitting a Data Bridge package to a receiving body. Not only because this allows verifying the accuracy of the procedure producing the E5X file, but it also relieves the receiving body from performing any validation tests, which should and can be performed locally; the body should naturally receive a valid data from a provider, and eventually perform tests on the quality of the information (are all fields filled in? is the data consistent?...).

RIT/E5X validation can be performed off-line or on-line, using tools and facilities made available to the ECCAIRS community by the developers of the system. Items are available on the ECCAIRS Community Portal, under *Support* → *Electronic Reporting – E5X* → *E5X Data Files*. You need to register for an account on the ECCAIRS Community Portal before getting access to the pages.

7.1 Off-line Validation

Off-line validation tools and files are available under *Support* → *Electronic Reporting – E5X* → *E5X Data Files* → *E5X Tools and Documentation*. In the *Software* section you find a link that downloads a ZIP file with the *E5 Data Bridge Validator*.

The ZIP file contains 1 folder containing 3 files:

```
🗁 e5databridge_validator.zip
    🗁 E5DataBridge Validator
        🗎 E5DataBridge.dll.remove
        🗎 E5DataBridgeValidator.exe.remove
        🗎 Readme.txt
```

After downloading and unzipping the archive to a folder on your PC, remove the '`.remove`' extension from the files starting with '`E5DataBridge`'. Microsoft Windows will warn that the file might become unusable, but this is exactly what is wanted: the '`.remove`' extension was added to prevent anti-virus software from deleting both files from the ZIP. The Validator is a stand-alone application.

In addition to the Validator you will need to download from the same page the Data Bridge Profiles for the version or versions you are producing E5X files for. These are available in the *Data Bridge Profiles* section and have the EDB file extension (for ECCAIRS Data Bridge). For convenience you can store the Data Bridge profiles in the same folder of the Validator application.

Start the application and proceed as follows.

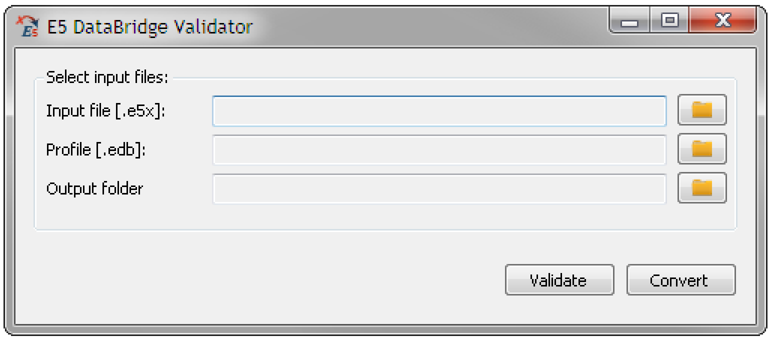

Figure 3 – E5 Data Bridge Validator application

- Select the input E5X file using the folder icon button right to **Input file [.e5x]**
- Select the proper Data Bridge profile using the folder icon button right to **Profile [.edb]**
- Select a folder where the result will be dropped using the folder icon button right to **Output folder**
- Click **Validate** if you want to validate the E5X against the chosen Data Bridge profile
- Click **Convert** if you want to convert the E5X into an E5F occurrence file

Close the application by clicking **X**.

7.2 On-line Validation

7.2.1 Via ECCAIRS Community Portal

On-line validation is available under *Support* → *Electronic Reporting – E5X* → *E5X Data Files* → *Upload and Validate an E5X file*. With this function you upload an E5X file for validation and/or conversion into an E5F occurrence file. The function supports all RIT/E5X versions available.

- On the indicated page, load an E5X from your PC using the button[12] right to **E5X file:**
- Insert the captcha verification text and click **Validate and Convert E5X** to obtain the validation report and the converted occurrence
- Click **Download Log File** if you want a copy of the validation report
- Click **Download E5F File** if you want a copy of the converted occurrence

[12] The button's text depends on the Internet Browser used: Chrome uses **Choose file**, Firefox and IE use **Browse...**

7.2.2 E5X Validation and Conversion Website

A dedicated website is available for developers wanting to validate their E5X files. This website can be accessed without an account on the ECCAIRS Community Portal; functionality is limited to validation and conversion of E5X files: http://eccairs-e5x.jrc.ec.europa.eu/

The website supports all RIT/E5X versions available.

- ➢ Click **Upload E5X** to upload an E5X from your PC
- ➢ On the indicated page, load an E5X from your PC using the button[13] right to **E5X file:**
- ➢ Insert the captcha verification text and click **Validate and Convert E5X** to obtain the validation report and the converted occurrence
- ➢ Click **Download Log File** if you want a copy of the validation report
- ➢ Click **Download E5F File** if you want a copy of the converted occurrence

7.3 E-mail Validation

Email validation is available as an on-line service. You send via e-mail a message where you attach the E5X files to be validated and converted. The function supports all RIT/E5X versions available.

- ➢ Start your e-mail client and create a new message
- ➢ Enter jrc-e5x-validator@ec.europa.eu as destination e-mail address
- ➢ Attach one or more E5X files to the newly created message (required)
- ➢ Send the e-mail

The e-mail message does not require any subject or body text, which is ignored since the relevant item in the message is the attachment. You can in principle attach any number of E5X files to the e-mail message, and mix files with different RIT/E5X taxonomies. There is a limit on the inbox of the above e-mail account, so avoid overloading a single e-mail message with very large attachments.

Non-conform attachments, like for instance pictures or PDFs, will also be ignored.

The e-mail validation function checks the mailbox for incoming messages approximately every 5 minutes. The reply e-mail with the validation result and conversion may therefore not be immediate. If the reply e-mail takes longer to arrive then you may want check your Spam or Junk folder, just in case it got delivered there.

On the **ECCAIRS Community Portal** a shortcut to this function is available under *Support* → *Electronic Reporting – E5X* → *E5X Data Files* within the *E-mail and Validate an E5X file* section.

A similar link exists on the dedicated **E5X Validation and Conversion** website.

~≈{ }≈~

[13] The button's text depends on the Internet Browser used: Chrome uses **Choose file**, Firefox and IE use **Browse...**