eAIP Core Toolbox Documentation

eAIP Core Toolbox Documentation

Table of Contents

List of Tables

List of Examples

Executive Summary

The Electronic AIP (eAIP) Specification developed by EUROCONTROL provides a standard way to:

- publish the content of an AIP (including Amendments (AMDT), Supplements (SUP) and Circulars (AIC)) in a structured electronic format;
- visualise the content of an AIP on a computer screen, using Web technology.

This documentation accompanies the eAIP Core Toolbox and contains information for **eAIP system developers and implementers** regarding Extensible Markup Language (XML) to Hyper-Text Mark-up Language (HTML) conversion, checking compliance of the eAIP, eAIP directory structures, the publication process, MakeAIP-core, creating a new eAIP and Extensible Stylesheet Language Transformations (XSLT) stylesheet documentation.

Many parts of this manual refer to the "MakeAIP-core" provided by EUROCONTROL. However, the MakeAIP-core tool provided by EUROCONTROL shall be considered as a proof of concept only. No effort was spent on fine-tuning the tool for better performance. There was no formal validation (software audit) of the HTML and Portable Document Format (PDF) output to prove that it corresponds entirely with the content of the XML files. The tool should be revisited, improved and validated before being used in a production environment regarding XML to HTML conversion, checking compliance, eAIP directory structures, the publication process, MakeAIP-core, creating a new eAIP and XSLT stylesheet documentation.

Chapter 1. AIP Supplements

1.1. Introduction

This chapter provides specific information for the production of AIP Supplements in the context of an eAIP.

1.2. Automatic Generation of GEN 0.3

As of December 2004 the AIP section GEN 0.3 (List of Supplements) shall not be edited manually but generated automatically.

This transformation uses the eAIP definition file in order to generate the content of section GEN 0.3. This information is also used to generate the list of Supplements in force, which is part of the eAIP HTML interface.

Rationale

Until December 2004, GEN 0.3 was edited manually and the list of Supplements in force was generated from GEN 0.3. The reason for the change is that some AIS offices do not update section GEN 0.3 each time there is a new Supplement to publish. It would be beneficial to States not to have to make an official Amendment if only GEN 0.3 needs to be updated. Hence, the requirement arose to separate the publication of a new Supplement from the publication of GEN 0.3.

The solution involves having a list of Supplements as part of the eAIP description and GEN 0.3 shall only be generated from this list when required.

Impact on Operations

This solution has a smaller impact on AIS offices' operations than the manual editing of GEN 0.3 and generating the list of Supplements from it. Users of the electronic version of eAIPs shall have the complete list of Supplements in force at that time, as part of the eAIP HTML interface.

Users of the paper version of the AIP shall receive the paper version of the eAIP with only the newly published Supplement. The user shall be required to attach the new Supplement to the appropriate volume of their paper copy of the AIP.

Both groups of users will have the same version of GEN 0.3 to hand.

Chapter 2. eAIP Compliance

2.1. Compliance

2.1.1. Definition

Compliance with the eAIP Specification may be claimed for eAIP instances that:

- are valid against the eAIP DTD *and*
- follow the additional rules.

A software product may be declared compliant with the EUROCONTROL eAIP Specification if it can handle/produce/edit/etc. compliant eAIP documents.

2.1.2. How to Validate

2.1.2.1. Informal Compliance

The eAIP may be informally validated using the tools available in the eAIP Core Toolbox.

2.1.2.2. Formal Validation

To formally validate an eAIP comprises validating it against the official DTD published on EUROCONTROL's website, at the address defined in the eAIP Specification section and then checking the eAIP for additional rules compliance.

2.1.2.3. Informal Compliance

The eAIP may be formally validated using the tools available in the eAIP Core Toolbox:

2.1.2.3.1. Using MakeAIP-core

MakeAIP-core is a batch script provided within the eAIP Core Toolbox. It allows compliance to both the eAIP Document Type Definition (DTD) and the additional rules to be checked. Please refer to the MakeAIP-core Documentation for more details. If MakeAIP-core is configured correctly, typing **`MakeAIP-core validator`** will check compliance with the DTD and report any errors; it will then automatically run **`MakeAIP-core schematron`**, which is described below. The Validator will not explicitly report compliance.

Note that MakeAIP-core validates an eAIP against the DTD declared in the eAIP itself, which is generally a local copy of the DTD.

2.1.2.3.2. Interactive Validation

Most XML editing software is able to validate an eAIP against its DTD, but not against the additional rules. Therefore, the eAIP Core Toolbox includes an interactive tool, based on Schematron, which is able to list additional rules validation errors. To use it, follow these two steps:

1. Run **`MakeAIP-core schematron;`**
2. Open or reload the file ../../tools/Validator/schematron-frame.html. [../../tools/Validator/schematron-frame.html]

This tool shows the list of errors in the upper part of the browser window and the whole XML source of the eAIP in the bottom part. When an error is clicked on, the bottom part is scrolled to the exact location where the error occurs. Each error is presented with a link to the eAIP Specification, and the exact rule related to the error.

To view such a report please see the eAIP Specimen's additional rules validation report [../../tools/Validator/schematron-frame.html]. When downloading the eAIP Core Toolbox, it is necessary to generate the report for the first time: open a command-prompt window, change the directory to the eAIP Core Toolbox installed directory and type **`MakeAIP-core schematron`**. The eAIP Specimen contains many errors as a result of the additional rules; most of them are there to check the validation process.

2.1.3. Additional Rules

2.1.3.1. Implementation

The eAIP Specification defines the additional rules in plain English. An implementation of these rules, in Schematron language, is provided in the eAIP Core Toolbox, in the file ../../tools/Validator/eAIP-schematron.xml, which is located in the tools/Validator directory in the eAIP Core Toolbox. Schematron is another XML grammar language, like DTD and XML Schema. It is undergoing International Organisation for Standardisation (ISO) standardisation to become "ISO/IEC 19757 - DSDL Document Schema Definition Language - Part 3: Rule-based validation - Schematron". A Schematron grammar is written in XML and is able to express certain rules that DTDs and XML Schemas cannot. For more information about Schematron, please refer to the Schematron home page [http://www.ascc.net/xml/resource/schematron/].

Chapter 3. eAIP Directories Organisation

3.1. Introduction

This chapter suggests a directory structure for eAIP-related files when managing the eAIP production process manually. This does not concern automatic production systems, which may use a different structure.

3.2. The Issue

Based on feed-back from eAIP Pilot Countries, it was decided that a default structure should be recommended which could help the organisation of various eAIPs, different eAIP DTD versions and eAIP amendments with their graphics.

3.3. Requirements

What has to be organised and how?

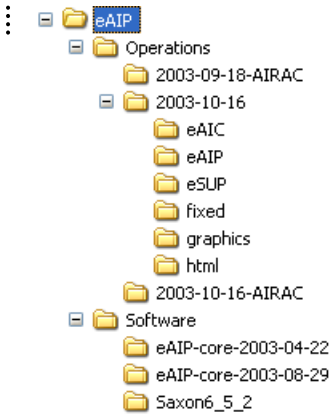
- Operational documents:
  - Grouped by publication date for AMDTs;
  - Different document types should not be mixed (because AIP AMDTs do not have the same life cycle as AICs and SUPs);
  - Grouped by file formats: XML, HTML, PDF and graphics (to ease publication: we do not need to publish Extensible Stylesheet Language - Formatting Object (XSL-FO) and AT files).
- Supporting documents:
  - EUROCONTROL eAIPs, including DTDs;
  - eAIP Core and Extended Toolbox software, such as Saxon.

The general objective of this proposed directory structure is to separate the operational documents (eAIP amendments, circulars and eAIP supplements) from the eAIP Core and Extended Toolbox software and supporting files. Also, it is recommended that the various versions of these documents and supporting files are stored separately. There are strong links between some files, by publication date. For example, eAIPs, eAICs and eSUPs all make reference to a specific "locales" file (for more information, see Multilingual eAIP) and a specific DTD, and they may change with time. It is important that references in documents published in the past are still valid.

A paramount requirement is that once a document has been published, it is not changed. This implies that files referred to from another file must remain in the same place, so that their relative location remains the same. For example, if it was necessary to move a chart's location, then an eAIP already published would need to be modified so that it references the chart at its new location. This is not acceptable.

As a secondary objective, it should be easy to publish the operational documents on the AIS organisation's website and, therefore, preserve the same directory structure.

3.4. Proposed Solution - Core



The directory structure illustrated in the image above is recommended. There are two main zones: Operations and Software. Please refer to the eAIP Extended Toolbox Documentation for further information about the directory structure applicable to the eAIP Extended Toolbox and PDF format.

3.4.1. Operations

The Operations zone contains one directory for each publication "package". Because Circulars and AIP Supplements are always sent with an Amendment, it was decided to name the eAIP directory by the Amendment type, number and year. That way, one can easily find all the documents that were published together and the directories will be sorted in a way that is familiar to the AIS community.

Each publication "package" contains several directories. The following are those directories which should be published, on a CD-ROM or a website:

- **eAIC** contains the new Circulars, published for the first time on the eAIP's publication date, in XML format;
- **eAIP** contains the complete AIP including the Amendment published on the eAIP's publication date, in XML format; only one AMDT is included in a given eAIP;
- **eSUP** contains the new Supplements, published for the first time on the eAIP's publication date, in XML format;
- **graphics** contains all graphic files (charts) referenced in the eAIP; it is used by both XML and HTML files;
- **html** contains the new Circulars, the complete AIP including the current Amendment and the new Supplements in HTML format, as well as a few related files which are specific to this eAIP (notably the list of AICs in force and the table of contents);
- ~~**pdf** contains the new Circulars, the complete AIP including the current Amendment and the new Supplements in PDF format, as well as the AIP Amendment (i.e. only amended pages) in a single PDF file.~~

Note

Only new AICs and SUPs are included in a eAIP; previous ones are available in previous eAIPs. A list of AICs and SUPs in force is available via the table of contents in HTML.

The other directories are not published as they are used for editing purpose only (see the Paper AIP Publication Procedure for details):

- **fixed** complete AIP in XML with current Amendment fixed;
- **publish** contains the AIP, AICs and SUPs in XSL-FO and AT formats;
- **publish-amdt** contains XSL-FO and AT files used for the Paper AIP Publication Procedure;
- **with-amdt-info** contains XSL-FO and AT files used for the Paper AIP Publication Procedure;
- **without-amdt-info** contains XSL-FO and AT files used for the Paper AIP Publication Procedure.

3.4.2. Software

The Software zone contains one directory for each version of the software needed for the eAIP. Saxon and the EUROCONTROL eAIP Core Toolbox are shown as an example. It is important to keep previous versions of any software used, so that the software environment which may be required to manage older documents still exists.

For example, an older eAIP in XML may declare conformance to an older eAIP DTD. If it is necessary to manipulate this eAIP for any reason (for a search engine or a specific presentation), then the correct DTD has to be used.

3.5. Consequences

3.5.1. Duplicated Files

This suggested directory organisation implies a limited duplication of certain files between directories.

For example, when publishing an Aeronautical Information Regulation and Control (AIRAC) AMDT together with a non-AIRAC one, many graphic files are likely to be the same for both AMDTs. Yet, they are duplicated because each AMDT package is independent and complete. If the duplication was avoided the location of the graphic files would change from one AMDT to the next, which involves too much maintenance and does not satisfy the principal requirement, as explained above.

3.5.2. Dispersed Circulars and Supplements

AICs and SUPs are scattered among several eAIPs, depending on their publication date. Again, this is with the aim of minimising file maintenance. These documents refer to the "locales" file, for instance, which can evolve with time. So it is important that they always refer to the same version of that file. Similarly, the eAIP refers to the Supplements, so they should not move either.

Consequently, when a publication on CD-ROM is prepared, several past eAIPs must be included, as well as the new ones, so that all Circulars and Supplements are available on the same media. However, the AIP (including graphics) published in past eAIPs need not be included if they are too large.

Despite being dispersed, these documents are not difficult to find: the user should use the HTML interface and browse the table of contents, which contains a list of Circulars and Supplements in force. From there, the user can access HTML and PDF versions of these documents.

3.5.3. Web Publication

When publishing an amendment on their website, the AIS office can simply copy the "package" directory (e.g. directory "2003-10-16"), excluding those sub-directories that are not needed (fixed, publish, publish-amdt, with-amdt-info and without-amdt-info). All the files are then in place for publication.

Only the link to the index of amendments needs to be updated.

Note

Currently, other files need to be updated as well: the list of amendments (history.html) and the list of circulars. However, these files may be generated automatically.

Chapter 4. eAIP Publication Process

4.1. Introduction

The content of this chapter is aimed at people who support eAIP Editors in producing eAIPs in HTML format.

4.2. Production of HTML Format

1. Open a terminal or command prompt window (e.g., "MS-DOS");
2. Change the current directory to where you unzipped the eAIP (for example on Windows, type **`cd C:\path\to\eAIP-pack`**);
3. Transform the eAIP file into HTML using MakeAIP-core.bat (read the MakeAIP-core documentation first) Usage examples:
   - simply type "makeaip-core" to display usage information;
   - type "makeaip-core sections -f html" to generate all HTML files for the sections;
   - type "makeaip-core GEN-0.1 -f html" to generate only the HTML file for the GEN-0.1 section;
   - type "makeaip-core AD-2.EADD -f html" to generate only the HTML file for the EADD aerodrome;
   - type "makeaip-core menu" to generate the navigation menu (ToC window).
4. Delete any section file in HTML which is "NIL", unless it becomes NIL in this amendment.

Instead of makeaip-core.bat (which only runs on Windows), you can use your favourite XSLT processor directly.

The XSLT processing takes about 10 seconds per file, using a Pentium III processor at 866MHz with 128MB RAM.

4.3. Internationalisation and Localisation

The XSLT stylesheets that produce HTML and XSL-FO are (almost) language- and locale-independent ("internationalised"). When they need to generate static text, such as en-route tables headings, they use a separate file, which contains translations (localisations) of (almost) all static content. For the EUROCONTROL's eAIP Specimen, this file is named "EC-locales-en-GB.xml".

The locales file is part of an eAIP: the eAIP is not usable without its locales file. It shall always be located in the same directory as the XML eAIP file, and be named "-locales.xml", prefixed with the International Civil Aviation Organisation (ICAO) country code.

The eAIP DTD can be found in the eAIP Specification.

4.3.1. Locales File Structure

The "locales" file is written in a simple XML format. Each translated item is represented by a text element with a unique id attribute. A text element contains at least one translation element, which contains the actual text of the translation. The translation element has an xml:lang attribute, which indicates the lan-

guage tag of the translation. A text element may contain an optional note element to provide information to translators.

4.3.2. Implementation of Internationalisation

When a stylesheet generates static text, it calls the "gettext" template, which is defined in xslt/g-get-text.xslt. This template takes a mandatory parameter "ID", containing the id of the text element to be output. An optional "c1" parameter allows the specification of content that may be placed at a different place in a translation item, depending on the language.

The "gettext" template looks in the "locales" file for a text element with the correct id and, within it, for a translation element with the correct language tag, as explained above.

4.3.3. Customising the eAIP Menu

The eAIP menu may be customised to offer links to other language version. If no other language version exists, then the relevant lines in the file xslt/html-content.xslt shall be removed.

The links to other language versions are simple to develop: depending on the file structure, other filenames may be created by simply changing the language suffix. There is currently no way of determining automatically the other languages available for a given eAIP.

4.4. Software Requirements

An XSLT processor is required. However, as most software of this type is developed in Java, a Java Runtime Environment (JRE) may also be needed. Optionally, MakeAIP-core.bat may be configured.

4.4.1. XSLT Processor

An XSLT processor transforms XML files into other formats, such as HTML and XSL-FO, using XSLT instructions. For more information about XSLT, see the References section of the eAIP Specification.

Popular XSLT processors include:

- Saxon [http://saxon.sourceforge.net], an Open Source Software written in Java, by Michael Key;
- Xalan [http://xml.apache.org/xalan-j/index.html], another Open Source Software written in Java by the Apache Software Foundation;
- MSXML [http://msdn.microsoft.com/xml], a closed source implementation by Microsoft, available free of charge.

There are many other processors available. MSXML uses a particular implementation of namespaces making it more appropriate for XML experts. Note that MSXML only runs on Windows platforms, while Xalan and Saxon run on any Java 2 platform (Windows, Unix, Mac and others).

If you are using MSXML, check which version is to be used and specify this in the MakeAIP-core file (read MakeAIP-core.html). This allows any version of MSXML that you have installed to be used, even in "side-by-side mode". Please note that Internet Explorer will only use the version that is in "replace mode". See http://www.netcrucible.com/xslt/msxml-faq.htm for more details.

The "eAIP Splitter", one of the tools provided in the eAIP Core Toolbox, only works with Xalan. It is anticipated that it would be easy to adapt it to Saxon, however. MSXML is incompatible with "eAIP Splitter".

4.4.2. MakeAIP-core.bat

The MakeAIP-core.bat batch file provided in the eAIP Core Toolbox shall be configured before first use. It shall be necessary to identify the location of the XSLT software. Please read MakeAIP-core documentation for more information.

4.5. Software Installation Guidelines

These guidelines may be followed in order to install all the software needed for the eAIP. If these guidelines are followed the result will be a working eAIP production system based on MakeAIP-core.

Please note that these guidelines only deal with Java software. There are other options available. Installing Java and configuring Xalan may seem complex at first glance, even for experts in this field. However, it only needs to be done once (unless upgrading).

4.5.1. Java

It is recommended that the IBM Java Runtime Environment (JRE) version 1.3.x or above is installed. Other JREs should also work but these guidelines may need to be adapted.

1. **Check if JRE is already installed**: open a terminal or command prompt window ("MS-DOS") and type `java -version` and then enter. If Java is correctly installed, the version number (for example: 1.3.0) should be displayed as well as some other information, including the provider (for example, IBM). It is recommended that If you do not have version 1.3.0 that you upgrade unless you are an expert in this field. If you do have JRE already installed, you can skip this section and go to the Xalan section;

2. **Download**: browse to http://www.ibm.com/java/jdk; register on the website (or log in if you are already registered); you will then be presented with a list of packages: click on the JRE 1.3.0 corresponding to your platform (Windows NT, Windows 98, Linux, etc.); you will then see another list: download the installer and the JRE (carefully following the instructions on the site);

3. **Install**: carefully follow the instructions provided on the JRE download page; the installation itself is simple and requires little interaction;

4. **Update your PATH**: Add the java directory to your PATH environment variable: the way to do this depends on the platform you are working on and so you should refer to the installation instructions (they are located in the "docs" directory, where the JRE was installed).

4.5.2. Xalan

1. **Download**: browse to http://xml.apache.org/xalan-j/index.html. Under the title "How do I get it?", further down the page, download the binary distribution;

2. **Install**: create a directory named "c:\dev" (on Unix, name it "~/dev") and unzip the package into this new directory. You may select another directory but make sure there is no space in the path;

3. **Update your CLASSPATH**: Add Xalan and its components to your CLASSPATH environment variable (create this variable if needed). Your CLASSPATH should contain: `c:\dev\xalan-j_2_4_0\bin\xercesImpl.jar;c:\dev\xalan-j_2_4_0\bin\xml-apis.jar;c:\dev\xalan-j_2_4_0\bin\xalan.jar` (for Unix, this path will need to be amended appropriately). CLASSPATH is like the PATH environment variable but is reserved for Java. This variable may be set in the same way as PATH above.

Note

IBM's JRE comes with an old version of Xerces which should be removed. Simply delete the file `xerces.jar`, located in [IBM JRE installation directory]\jre\lib\ext (merely renaming the file is not sufficient).

4.5.3. Formatting Objects Processor (FOP)

1. **Download**: browse to http://xmlgraphics.apache.org/fop/ and click on "Download" in the menu. Click on "distribution directory". Download the latest version (currently, Fop-0.20.5-bin.tar.gz);

2. **Install**: Unzip the package, for example in "c:\dev\fop";

3. Read the documentation for MakeAIP-core.bat from our eAIP Core Toolbox to configure MakeAIP-core.bat for your environment.

Chapter 5. MakeAIP-core.bat Documentation

5.1. Introduction

MakeAIP-core.bat is a batch script for Windows NT/2000/XP. It eases production of eAIP related files, that is conversion of eAIP in XML format to HTML, XSL-FO and PDF.

Note

MakeAIP-core.bat shall be considered as a proof of concept only. No effort was spent on fine-tuning the tool for better performance. There was no formal validation (software audit) of the HTML and PDF output, to prove that it corresponds entirely with the content of the XML files. The tool should be revisited, improved and validated before being used in a production environment.

This documentation applies to MakeAIP-core.bat version 2.6, revision 1.25, but it is not complete: more parameters are explained in AMDT procedures.

5.2. Files

Table 5.1.

MakeAIP-core.bat	Main file, which used for any command
doc\MakeAIP-core.html	This file: the documentation
tools\MakeAIP-core\msxsl.bat	Batch file for msxsl.exe
tools\MakeAIP-core\saxon.bat	Batch file for Saxon
tools\MakeAIP-core\xalan.bat	Batch file for Xalan
tools\MakeAIP-core\xt.bat	Batch file for XT

The file saxon.bat is pre-configured to work with Saxon, provided with the eAIP Core Toolbox.

5.3. Usage

1. Open a command prompt window ("MS-DOS");

2. Change the directory to the one where the eAIP Core Toolbox is installed.

Then simply type makeaip-core.bat to display a summary of usage information, or use one of the sub-commands described below.

Note

Tip: Microsoft offers several "power toys" for their various Windows versions. One of them, called "Command Prompt Here" will prove very useful, as it allows the launch of a command prompt window from File Explorer, directly in a specific directory, by right-clicking on a directory entry. For Windows XP, it is available at http://www.microsoft.com/windowsxp/downloads/powertoys/xppowertoys.mspx.

5.3.1. Command-line Parameters

5.3.1.1. Conventions Used in this Document

Throughout this document the following syntax will be used to describe the command-line commands: `command` { option1 | *`option2`* } {mandatory_parameter *`replaceable_value`*} [optional_parameter *`replaceable_value`*] [other_optional_parameter value]

command
: The batch file or executable file to execute.

options
: Alternatives are separated by a vertical bar. Alternatives may be straight values or replaceable values.

parameters
: Parameters brackets indicate if the parameter is optional or mandatory.

replaceable_value
: Most parameters take an argument, the styling of which indicates that the word should be replaced by an actual value. The possible values are then explained in the command summary.

value
: Fixed values (for a given command) have a different format. These indicate the actual value that must be typed for the command.

5.3.1.2. MakeAIP-core Command General Synopsis

MakeAIP-core.bat accepts several parameters. All parameters are optional and running MakeAIP-core without any parameter outputs a short usage summary.

If any parameter is entered, then the target parameter is mandatory: it indicates the sub-command to run. Depending on the target, other parameters may subsequently become mandatory as well. Here is the command synopsis for general parameters:
`makeaip-core { sub-command | section_id | AIC_or_SUP_name } [parameters...]`

where:

- *sub-command* is one of the following:
  - **sections**: generates all eAIP sections such as GEN-2.2, ENR-3.1 and AD-2.EADD;
  - **menu**: generates the eAIP table of contents[1]: see Section 5.3.4.1, “MakeAIP-core Menu”;[1]
  - **AMDT**: generates the list of changes[1]: see Section 5.3.4.2, “MakeAIP-core AMDT”;
  - **eSUPs**: generates the list of AIP Supplements in force[1]: see Section 5.3.4.2, “MakeAIP-core AMDT”;
  - **eAICs**: generates the list of Circulars in force[1]: see Section 5.3.4.2, “MakeAIP-core AMDT”;
  - **cover**: generates the eAIP cover page[1]: see Section 5.3.4.2, “MakeAIP-core AMDT”;
  - **interface**: copies all static files needed by the HTML interface: indexes, frameset definitions, JavaScript, CSS and images;
  - **all**: generates all the above, in html format only;
  - **history**: generates the list of published eAIPs[1]: see Section 5.3.4.2, “MakeAIP-core AMDT”;
  - **validator**: reports any DTD validation errors and runs MakeAIP-core schematron (see below);

[1]These sub-commands generate components of the eAIP HTML interface. See also the eAIP Specification.

    - **schematron**: generates a Schematron report of Additional Rules validation (see eAIP Specification);
    - **packmap**: generates the `packmap.xml` file.

- *section_id* is a **part, chapter or section id** such as `GEN-2.2`: see Section 5.3.2, “AIP Section Publication”;
- *AIC_or_SUP_name* is an **eAIC or eSUP name** such as eSUP-2001-01-01 or eAIC-2001-01-01-C: see Section 5.3.3, “Circulars and Supplements”;
- *parameters*: common parameters are detailed below and each sub-command.

5.3.1.3. Common Parameters

The following parameters are used for several sub-commands:

`-d` *directory*
: Directory where the eAIP XML file is located. The eAIP XML file must be named XX-eAIP-yy-YY.xml, where XX is the country code and yy-YY the language code.

`-s` *directory*
: Directory where source (input) files to the sub-command are located.

`-t` *directory*
: Directory where target (output) files from the sub-command will be created.

`-l` *language*
: Language selection (your eAIP may be in several languages, identified by the suffix to the eAIP XML file, as in EC-eAIP-**en-GB**.xml).

`-c` *country*
: ICAO country code (in case you have several eAIPs from different countries; identified by the prefix to the eAIP XML file, as in **EC**-eAIP-en.xml).

`-p` *XSLT_processor*
: Replace *XSLT_processor* with one of the following: "saxon", "xalan", "msxsl", "xt" (without the quotes).

`-debug`
: Useful for debugging stylesheets or MakeAIP-core itself, by providing additional output messages.

5.3.2. AIP Section Publication

This is the general process when an editor wishes to generate an AIP section in HTML.

5.3.2.1. HTML Format

`makeaip-core {`*`section_id`*`}` [-d *`eAIP_directory`*] [-t *`target_directory`*] [-l *`language`*] [-c *`country-code`*] [-p *`XSLT_processor`*]

Generates a single HTML file for the eAIP element with the given id. The file will be named `XX-section_id-yy-YY.html`, where XX is the country code and yy-YY the language code.

section_id
: The value of the id attribute of an eAIP element. The element must be a part-, chapter- or section-level element, or an Aerodrome or Heliport element.

Example 5.1. MakeAIP-core section_id sub-command for HTML format

Generate one HTML file named `EC-GEN-en-GB.html` for the whole GEN part:

```
makeaip-core GEN -d ..\..\Operations\2004-10-28\eAIP -t ..\..\Operations\2004-10-28\html\eAIP
```

Generate one HTML file named `EB-ENR-3-fr-FR.html` for the whole ENR 3 chapter:

```
makeaip-core ENR-3 -d ..\..\Operations\2004-10-28\eAIP -t ..\..\Operations\2004-10-28\html\eAIP -c EB -l fr-FR
```

Generate one HTML file for the whole AD 1.3 section:

```
makeaip-core AD-1.3 -d ..\..\Operations\2004-10-28\eAIP -t ..\..\Operations\2004-10-28\html\eAIP
```

Generate one HTML file for the whole EADD Aerodrome:

```
makeaip-core AD-2.EADD -d ..\..\Operations\2004-10-28\eAIP -t ..\..\Operations\2004-10-28\html\eAIP
```

5.3.3. Circulars and Supplements

Generating a Supplement or Circular is very similar to generating an eAIP section. Simply replace the eAIP section id by the document name.

Warning

A limitation of MakeAIP-core's command-line syntax means that the names of all Supplements and Circulars begin with “eSUP” and “eAIC” respectively. Other eAIP publication tools may remove this limitation.

Example 5.2. MakeAIP-core AIC_or_SUP_name sub-command for PDF format

Generate one HTML file for a Supplement:

```
makeaip-core eSUP-2001-02 -d ..\..\Operations\2004-10-28\eSUP -t ..\..\Operations\2004-10-28\html\eSUP
```

Generate one AT file for the whole ENR 3.1 section, without Amendment indications (the PDF file is normally not produced in this case, but it can be generated if needed):

```
makeaip-core ENR-3.1 -f fo -d ..\..\Operations\2004-10-28\eAIP -t ..\..\Operations\2004-10-28\without-amdt-info -a false
```

```
makeaip-core ENR-3.1 -f at -s ..\..\Operations\2004-10-28\without-amdt-info -t ..\..\Operations\2004-10-28\without-amdt-info
```

5.3.4. eAIP HTML Interface

The eAIP has a convenient HTML interface to browse the information it contains. This section describes the MakeAIP-core commands to generate the following components of this interface:

- eAIP Table of Contents: Section 5.3.4.1, “MakeAIP-core Menu”
- eAIP Amendment list of changes: Section 5.3.4.2, “MakeAIP-core AMDT”
- List of AIP supplements in force: Section 5.3.4.3, “MakeAIP-core eSUPs”
- List of Circulars in force: Section 5.3.4.4, “MakeAIP-core eAICs”
- eAIP cover page: Section 5.3.4.5, “MakeAIP-core Cover”
- History (Archive) page: Section 5.3.4.6, “MakeAIP-core History”

5.3.4.1. MakeAIP-core Menu

`makeaip-core` {menu} [-s *`source_directory`*] [-t *`target_directory`*] [-l *`language`*] [-c *`country-code`*] [-p *`XSLT_processor`*]

Generates the AIP table of contents in the file `XX-menu-yy-YY.html`, where XX is the country code and yy-YY the language code. If there is no Amendment in the eAIP, a table of contents will be generated anyway, based on the previous AIP indicated in the eAIP. For more information, refer to the How to Create an eAIP.

source_directory
: Path to the directory where the `eAIS-package.xml` file is located.

Example 5.3. MakeAIP-core menu sub-command

```
makeaip-core menu -s ..\..\Operations\2004-10-28 -t ..\..\Operations\2004-10-28\html\eAIP
```

5.3.4.2. MakeAIP-core AMDT

`makeaip-core` {AMDT} [-s *`source_directory`*] [-t *`target_directory`*] [-l *`language`*] [-c *`country-code`*] [-p *`XSLT_processor`*]

Generates the list of changes, for the current AIP Amendment, in the file `XX-AMDT-yy-YY.html`, where XX is the country code and yy-YY the language code. If there is no Amendment in the eAIP, the file will contain an explanatory note.

source_directory
: Path to the directory where the `eAIS-package.xml` file is located.

Example 5.4. MakeAIP-core AMDT sub-command

```
makeaip-core AMDT -s ..\..\Operations\2004-10-28 -t ..\..\Operations\2004-10-28\html\eAIP
```

5.3.4.3. MakeAIP-core eSUPs

`makeaip-core` {eSUPs} [-s *`source_directory`*] [-t *`target_directory`*] [-l *`language`*] [-c *`country-code`*] [-p *`XSLT_processor`*]

Generates the list of AIP Supplements in force in the file `XX-eSUPs-yy-YY.html`, where XX is the country code and yy-YY the language code.

source_directory
: Path to the directory where the `eAIS-package.xml` file is located.

Example 5.5. MakeAIP-core eSUPs sub-command

```
makeaip-core eSUPs -s ..\..\Operations\2004-10-28 -t ..\..\Operations\2004-10-28\html\eSUP
```

5.3.4.4. MakeAIP-core eAICs

`makeaip-core` {eAICs} [-s *`source_directory`*] [-t *`target_directory`*] [-l *`language`*] [-c *`country-code`*] [-p *`XSLT_processor`*]

Generates the list of Circulars in force in the file `XX-eAICs-yy-YY.html`, where XX is the country code and yy-YY the language code.

source_directory
: Path to the directory where the `eAIS-package.xml` file is located.

Example 5.6. MakeAIP-core eAICs sub-command

```
makeaip-core eAICs -s ..\..\Operations\2004-10-28 -t ..\..\Operations\2004-10-28\html\eAIC
```

5.3.4.5. MakeAIP-core Cover

`makeaip-core` {cover} [-s *`source_directory`*] [-t *`target_directory`*] [-l *`language`*] [-c *`country-code`*] [-p *`XSLT_processor`*]

Generates the eAIP cover page in the file `XX-cover-yy-YY.html`, where XX is the country code and yy-YY the language code.

source_directory
: Path to the directory where the `eAIS-package.xml` file is located.

Example 5.7. MakeAIP-core eAICs sub-command

```
makeaip-core cover -s ..\..\Operations\2004-10-28 -t ..\..\Operations\2004-10-28\html
```

5.3.4.6. MakeAIP-core History

`makeaip-core` {history} [-s *`source_directory`*] [-t *`target_directory`*] [-l *`language`*] [-c *`country-code`*] [-p *`XSLT_processor`*]

Generates the list of published eAIPs in the file `XX-history-yy-YY.html`, where XX is the country code and yy-YY the language code. This file must be re-generated each time a new eAIP is published. The newly-generated file replaces the previous one.

source_directory
: Path to the directory where the file `eAIS-packages.xml`, containing the list of published eAIPs in XML format, is located.

Example 5.8. MakeAIP-core eAICs sub-command

```
makeaip-core history -s ..\..\Operations -t ..\..\Operations
```

5.3.5. eAIP For EAD/PAMS

An eAIP prepared for the European AIS Database (EAD) (more precisely, for the Published AIP Management System (PAMS)) needs a file named packmap.xml. For more information about this file, please see eAIP for EAD PAMS.
`makeaip-core` {packmap} [-s *`source_directory`*] [-p *`XSLT_processor`*]

Generates the packmap.xml file for the eAIS description file found in the source directory. The generated file is saved in the source directory.

source_directory
: Path to the directory where the file `eAIS-package.xml` is located.

Example 5.9. MakeAIP-core eAICs sub-command

```
makeaip-core packmap -s ..\..\Operations\2004-10-28
```

5.3.6. Validation Sub-Commands

More information about validation can be found in the eAIP Compliance chapter .

5.3.6.1. MakeAIP-core Validator
`makeaip-core` {validator} [-d *`eAIP_directory`*] [-l *`language`*] [-c *`country-code`*]

The validator does not convert the eAIP into any format. Instead, it validates the eAIP against the declared DTD.

5.3.6.2. MakeAIP-core schematron
`makeaip-core` {schematron} [-d *`eAIP_directory`*] [-l *`language`*] [-c *`country-code`*]

The schematron sub-command validates an eAIP against the additional rules. The validation result is output in the tools\validator directory.

5.4. Errors during Execution

When a program called by MakeAIP-core terminates with a proper error code, MakeAIP-core will ask the user if it should continue. At that time, one can press ctrl-C to stop MakeAIP-core or any other key to continue. This can be useful when generating all sections: the generation can take a couple of minutes, and a lot of information is output to the screen, so it is difficult to know if each file is generated without error.

This functionality can be switched off: refer to the last few lines of MakeAIP-core.bat.

5.5. Configuration

Before using MakeAIP-core.bat it should be configured. There are two aspects to configure: the defaults parameters and the location and parameters of the XSLT processors.

5.5.1. Default Parameters

MakeAIP-core.bat command line parameters can be set to default values, so that you do not need to specify them at each use:

Table 5.2.

Command line option	Signification	Initial value	Other possible values
-d	eAIP directory	eAIP	any
-l	language	en-GB	any
-c	ICAO country code	EC	any
-p	XSLT processor	saxon	xalan, msxsl, xt

To change the default values, edit MakeAIP-core.bat in a text editor such as Notepad, locate the section "Set defaults" near the top of the file and edit the parameters as required. Note that some parameters only accept a limited set of values.

5.5.2. XSLT Processors

We provide ready-made batch files for Saxon, Xalan, msxsl.exe, xt, FOP and XEP.

Saxon.bat is already configured to work with the supplied saxon.jar binary. Please note that in order to save space only this binary file is provided and not the whole Saxon distribution. The complete distribution is available at http://saxon.sourceforge.net/, including documentation and source code.

These files (or at least those for the processors/renderers that you will use) need to be edited with a text editor in order to specify the location of these programs, or more generally, how they should be executed. If a standard installation procedure is followed for each of them, their home directory is probably the only change required. To configure XEP, you may also need to change the name of XEP's jar file, which depends on the version you are using (client, server, etc.). Refer to XEP's own batch files for an example.

These batch scripts always take the same set of parameters:

- For XSLT processors (xalan.bat, msxsl.bat, xt.bat):

  ```
  xxx.bat XML XSLT OUTPUT param1-name param1-value param2-name param2-value param3-name param3-value
  ```

  where

  - XML is the source XML file;
  - XSLT is the stylesheet to use;
  - OUTPUT is the resulting file (html or fo);
  - paramX-name and -value are optional XSLT parameters name/value pairs.

5.5.3. XSL-FO processors

Ready-made batch files for Saxon, Xalan, msxsl.exe, xt, FOP and XEP are provided.

Saxon.bat is already configured to work with the supplied saxon.jar binary. Please note that only the binary file is provided and not the whole Saxon distribution, in order to save space. The complete distribution is available at http://saxon.sourceforge.net/, including documentation and source code.

These files need to be edited (or at least those for the processors/renderers that will be used) with a text editor in order to specify the location of these programs, or more generally, how they should be executed. If a standard installation procedure for each of them was followed, the only their home directory may need changing. To configure XEP, the name of XEP's jar file may need changing which depends on the version being used (client, server, etc.). Refer to XEP's own batch files for an example.

These batch scripts always take the same set of parameters:

- For XSL-FO renderers (FOP.bat, XEP.bat): **`xxx.bat FO OUTPUT`** where:
  - FO is the source XSL-FO file;
  - OUTPUT is the resulting file (pdf).

5.5.4. Validator and Schematron

The validator.bat file is different to the others. It first calls a parser to validate the eAIP XML file. This validation is done in the classic XML meaning: the XML file is validated against the DTD that it declares.

Schematron.bat runs an XSLT processor to produce a Schematron validation report. This validates the eAIP against the Additional Rules.

5.5.5. Advanced Configuration

Other processors may also be used: there are many other XSLT processors. To do this, simply copy an existing batch file, rename it and edit it following the program's command line syntax. Then, edit MakeAIP-core.bat and add your program's name to the variable xsltproc_list (for an XSLT processor) in the section "CONFIGURATION". Note that the program name you specify should only contain letters and numbers (no spaces or other special characters).

Chapter 6. How to Create a New eAIP

6.1. Introduction

This chapter is aimed at people who support eAIP Editors in creating their first eAIP.
Note

In order to publish an eAIP, it is necessary to also write an eAIP description file. Please see the Create eAIP documentation for more information.

6.2. Procedure

For the purpose of this example, we will assume that we are creating a new eAIP for Belgium, in French. The ICAO country code is EB and the ISO language code is fr-BE. Also, the eAIP Core Toolbox has been installed in a directory named eAIP-pack and all commands are issued from that directory.

6.2.1. First Method: The Hard (but Clean) Way

In this method, we start a new eAIP from the beginning.

1. Create a directory named EB-eAIP;

2. Create a new XML file named EB-eAIP-fr-BE.xml (it is assumed that good XML editor software is being used from now on);

3. Assign the DTD: the system path should be "../dtd/eAIP.dtd" or the official URL of the DTD (for example: (http://www.eurocontrol.int/ais/ahead/eAIP/dtd/1.0.3/eAIP.dtd"); note that setting a complete path such as "C:\eAIP-pack\dtd\eAIP.dtd" is wrong because the system path must be a valid URL, not a Windows path;

4. At this point, your software should have automatically generated the mandatory structure of an eAIP in your XML file. However, the eAIP will not be valid yet;

5. Edit MakeAIP-core.bat to change the default parameters and add references to the new aerodromes and heliports;

6. Edit EB-eAIP-fr-BE.xml to update all attributes on the "e:eAIP" element;

7. Edit the other XML files to replace text from the Specimen AIP with your AIP text. At this point, you may require further assistance. Visit EUROCONTROL's website [http://www.eurocontrol.int/eaip] and read the eAIP Manuals for more information;

8. When your eAIP is valid, it may be split by using the eAIP Splitter (see below for an example).

6.2.2. Alternative Method Using the Specimen eAIP as a Foundation

This method has the advantage of having complete eAIP content from the beginning. The inconvenience is that this content will eventually need to be replaced or removed.

1. Copy eAIP-pack/eAIP to a new directory named eAIP-pack/EB-eAIP; from now on, we will only work in this new directory;

2. Rename all EC-*-en-GB.xml files, replacing "EC" by "EB" and "en-GB" by "fr-BE";

3. Copy EB-AD-2.EADD-fr-BE.xml for each of your airports, renaming the files with the relevant ICAO airport code;

4. Repeat previous step with EB-AD-3.EADH-fr-BE.xml for each heliport;

5. Edit EB-eAIP-fr-BE.xml to add references to your new airports and heliports files (based on the example with EADD and EADH), in two places:

   - At the end of the DOCTYPE section;

   - At the end of the file, within e:AD-2 and e:AD-3 elements.

6. Edit MakeAIP-core.bat to change the default parameters and add references to the new aerodromes and heliports;

7. Edit EB-eAIP-fr-BE.xml to update all attributes on the "e:eAIP" element;

8. Edit the other XML files to replace text from the Specimen AIP with your AIP text. At this point, you may require further assistance. Visit EUROCONTROL's website [http://www.eurocontrol.int/eaip] and read the eAIP Manuals for more information.

6.2.2.1. Alternative Method Using XSLT

This method makes use of the XSLT identity transformation provided within the eAIP Core Toolbox, in the tools/Splitter directory.

1. Create a new directory named eAIP-split in the eAIP "package" directory;

2. From within the eAIP "package" directory, run an identity transformation on the Specimen eAIP: "**`java -jar tools/saxon/saxon.jar -o eAIP-split/eAIP-full.xml eAIP/EC-eAIP-en-GB.xml tools/Splitter/ns-identity-dtd.xslt`**". Note: the new eAIP will declare the official eAIP DTD on EUROCONTROL's website in its DOCTYPE section; if you don't have direct access to the internet, you should change it to "../dtd/eAIP.dtd";

3. Edit eAIP-full.xml:

   a. update the e:eAIP element (root element): set ICAO-country-code and xml:lang attributes to your country code and AIP language;

   b. update the content of the e:GEN-2.4 element: remove all existing Location-definition elements and add new ones for all your aerodromes and heliports;

   c. copy the e:Aerodrome and e:Heliport elements of EADZ and EADH as many times as required for all your aerodromes and heliports. Remove the redundant ones (EADD, EAZZ). You do not need to have real content in these elements: only two things are important: (1) the Ref attribute must refer to the id attribute of the corresponding Location-definition element, and (2) eAIP-full.xml must be valid.

4. Split eAIP-full.xml: "**`java -jar tools/saxon/saxon.jar eAIP-split/eAIP-full.xml tools/Splitter/ns-eAIP-splitter.xslt`**". eAIP-split directory now contains your split eAIP. Rename this directory to EB-eAIP, for example. Edit the default parameters of MakeAIP-core.bat to use this directory by default;

5. Edit MakeAIP-core.bat to change the default parameters and add references to the new aerodromes and heliports;

6. Edit EB-eAIP-fr-BE.xml to update all attributes on the "e:eAIP" element;

7. Edit the other XML files to replace text from the Specimen AIP with your AIP text. At this point, you may require further assistance. Visit EUROCONTROL's website [http://www.eurocontrol.int/eaip] and read the eAIP Manuals for more information.

Chapter 7. How to Create an eAIP

7.1. eAIP Description Document

7.1.1. Production

During preparation of an eAIP, the AIS office shall be responsible for maintaining the eAIP description document. This is an XML document containing all the information needed to automatically publish an eAIP. It is used by several eAIP tools transformations to generate the eAIP navigation interface in HTML.

This document may be edited directly using a text editor or with an XML editor. It may also be generated by dedicated eAIP management software.

7.1.2. Location

The eAIP description document shall be located in the directory of the eAIP it describes, and it shall be named `eAIS-package.xml`.

If the eAIP includes the XML files, the description document shall be published as well, in the directory of the eAIP. Otherwise, it shall not be provided.

7.1.3. Validation

The eAIP description document shall be valid (in the XML sense) against the eAIP DTD. This DTD is part of the eAIP Specification. A summary of this DTD is provided for reference in the next section..

7.1.4. Document Structure

An eAIP shall be described using the following XML elements:

`eAIS-package`
: This is the root element, containing `Description`, `eAIP-package`, `eSUPs` and `eAICs` elements. It shall have the following attributes:

  `ICAO-country-code`
  : The ICAO country code of the publishing country.

  `State`
  : The full name of the State concerned by the eAIP.

  `Effective-date`
  : The effective date of the information contained in the eAIP. In general, a single effective date applies for any information contained. Exceptionally, it is possible that one or more documents included in the eAIP have a later effective date. For example, an eAIP SUP may have an effective date which is different from the AIRAC cycle date of the eAIP in which it is included.

  `Publication-date`
  : The date on which the eAIP is published.

`Package-name`
The eAIP name, normally used to name the eAIP's base directory. This name shall be unique for any eAIP published for a given ICAO-country-code, in a given series.[1]This name shall normally be composed of the effective date in the form year-month-day. For AIRAC eAIPs, the suffix "-AIRAC" shall be added.

`Publishing-organisation`
The name of the organisation responsible for editing or publication.

`Description`
Each `Description` element shall contain textual information about the eAIP, in one language, in HTML format. This information shall be presented on the eAIP cover page.

`eAIP-package`
This element shall indicate which eAIP contains the relevant eAIP Amendment. An eAIP which does not contain an Amendment itself shall refer to a previous eAIP, so that user applications may find the latest relevant Amendment. If the current eAIP contains an Amendment, then it shall refer to itself. This element shall contain at least one `Language-version` element, which points to a specific version of the Amendment in one language.

`eSUPs`
This element shall contain the list of all AIP Supplements in force on the eAIP publication date. Each Supplement shall be defined in an `eSUP-reference` element.

`eSUP-reference`
This element shall indicate which eAIP contains the relevant Supplement. The reference may be to a previous eAIP or the current one. This element shall contain at least one `Language-version` element, which points to a specific version of the Supplement in one language.

`eAICs`
This element shall contain the list of all Circulars in force on the eAIP publication date. Each Circular shall be defined in an `eAIC-reference` element.

`eAIC-reference`
This element shall indicate which eAIP contains the relevant Circular. The reference may be to a previous eAIP or the current one. This element shall contain at least one `Language-version` element, which points to a specific version of the Circular in 1 language.

7.1.5. Usage

The description document is used by several XSLT stylesheets of the eAIP Specification. It may be used to generate various HTML files:

- eAIP table of contents;
- eAIP list of changes;
- List of AIP Supplements;
- List of Circulars;
- Cover page.

[1]There can be several eAIP "series" for the same country: civil/military, distant geographical regions.

Additionally, the eAIP description document may be used to generate an EAD PAMS-compatible eAIP mapping file (`packmap.xml`). See appendix eAIP for EAD PAMS.

Important

These stylesheets rely on the eAIP description document being valid and complete. AIS editors shall be responsible for the management of the definition file.

7.2. Quick Start: How to Make an eAIP

7.2.1. Procedure for a General eAIP

1. Create an eAIP description file (as a minimum, set the root element's attributes and the content of the `eAIP-package` element); save this file in the base directory of the eAIP Core Toolbox;

2. Open a command prompt, go to the base directory of the eAIP tools and run the following MakeAIP-core command: **makeaip-core all -s *path\to\package\base\directory* -t *path\to\package\base\directory;***

3. Create a directory directly in the root of the C: drive (or any other drive), named in a similar way to the eAIP base directory (for example, 2004-10-04), copy to it the directories `html` and `graphics` from the eAIP;

   [2]

4. Create a zip file; add to all of the contents of the directory which has just been created.

The zip file contains an eAIP with all the required HTML files.

eAIP's History

Optionally, you can add the eAIP's history to the zip file:

1. Create or update the eAIS-packages.xml file for the eAIP's history;

2. Again at the command prompt, run the following MakeAIP-core command: **makeaip-core history -s *path\to\package\parent\directory* -t *path\to\package\parent\directory***;

3. Copy all files (not directories) in the directory used for -s and -t to the root of the zip file.

[2] The objective of this operation is to have a separate directory for the eAIP inside the zip file. This is important if more than one eAIP is to be grouped in a zip file and if the eAIPs' history is to be added.

Chapter 8. XSLT Stylesheet Core Documentation

8.1. Introduction

The EUROCONTROL Electronic AIP Specification is provided with a set of XSLT stylesheets as an example of how an eAIP XML document can be converted into HTML format. The HTML stylesheet is designed to present an AIP on a website, be it an intranet or a public site; it follows most of the eAIP Usability Guidelines [usability_study.pdf]. This document describes how the XSLT stylesheets are structured and explains how the "content logic" is separated from the presentation logic.

8.1.1. Prerequisites

Readers are expected to have at least a basic knowledge of XSLT. The following resources may be a good place to start: XSLT 1.0 W3C Recommendation [http://www.w3.org/TR/xslt]; D. Pawson's XSLT FAQ. [http://www.dpawson.co.uk/xsl/sect2/sect21.html].

8.2. Files and Templates Organisation

8.2.1. Content Logic vs. Presentation Logic

By **content logic**, we refer to the XSLT code that produces the actual text and data (the content) which compose an AIP. For example, a section title is marked up as such in the XML eAIP document and is copied into the generated HTML documents, possibly with automatically-generated numbering. In this case, some content logic may be needed to compute the title numbering.

By **presentation logic**, we refer to the XSLT code that produces the formatting and styling necessary to present the content on the target format (HTML in this case). Continuing with the title example, the title content is typically surrounded by formatting mark-up such as a h1-h6 element in HTML. These elements may contain styling mark-up such as a class in HTML. Output of these elements and attributes is part of the presentation logic.

Presentation logic is specific to each target format, while content logic can generally be re-used for any format. Even though the AIP editors marked the title to be automatically numbered, the number produced must be the same for all target formats.

8.2.2. File Organisation

The stylesheets are split into three groups:

- Generic stylesheets are prefixed with "g-"; example: g-block.xslt;
- HTML-specific stylesheets are prefixed with "html-"; example: html-block.xslt.

8.2.2.1. Generic Stylesheets

Generic stylesheets contain the "content logic". They are normally imported by formatting-specific stylesheets. Generic stylesheets do not generate any element. They generate the content (the text) that must be produced automatically (such as auto-numbering) and combine different content together, where needed.

Mostly eAIP elements are matched by templates within generic stylesheets. These templates deal with content logic that shall be applied before calling a formatting-specific template. The formatting template should be overridden by a calling stylesheet if any formatting is needed, but it is not mandatory.

8.2.2.2. HTML-specific Stylesheets

These stylesheets contain formatting templates specific to HTML. Templates in these stylesheets override named presentation templates defined in generic stylesheets.

8.2.2.3. Table of XSLT files

This table lists all XSLT files alphabetically with a short description of their purpose. See also the elements to XSLT files mapping table, which indicates where each element is processed.

Table 8.1.

Generic	HTML	XSL-FO	Description
-	html-amdt-list.xslt	-	Generates a list of all changes in a given AMDT (in development)
g-amendments.xslt	html-amendments.xslt	fo-amendments.xslt	Amendment elements and attributes, including 0.2 sections
-	-	fo-att-set.xslt	Generic templates for dynamic class name/attribute-set processing (see below, Dynamic Attribute-set Processing for details)
g-block.xslt	html-block.xslt	fo-block.xslt	Processing of block elements such as Address and div
-	-	fo-checklist.xslt	Generation of the checklist of pages in GEN-0.4
g-content.xslt	html-content.xslt	fo-content.xslt	Initialisation of common variables and parameters, plus eAIP element processing for HTML
g-eAIC-eSUP.xslt	html-eAIC-eSUP.xslt	fo-eAIC-eSUP.xslt	Elements and attributes related to Circulars and Supplements documents (but not the Supplement element used in an eAIP document)
g-generic.xslt	html-generic.xslt	fo-generic.xslt	Common templates used for many elements; default templates
g-gettext.xslt	-	-	A generic set of templates to retrieve the localisation of words and sentences
g-graphics.xslt	html-graphics.xslt	fo-graphics.xslt	Templates related to Figure and Graphic-file elements
g-inline.xslt	html-inline.xslt	fo-inline.xslt	Processing of inline elements such as Abbreviation and strong
g-links-s.xslt	html-links-s.xslt	fo-links-s.xslt	Generation of links within an eAIP document
g-lists.xslt	html-lists.xslt	fo-lists.xslt	Processing of list elements such as Location-definition and ul
-	html-menu.xslt	-	Generation of the table of contents for the HTML navigation interface
g-numbering.xslt	html-numbering.xslt	fo-numbering.xslt	Auto-numbering templates
-	-	fo-pages.xslt	Page layouts, page headers and footers
g-structure.xslt	html-structure.xslt	fo-structure.xslt	Processing of structural elements such as Sub-section, sections, chapters and Title elements
-	-	fo-styles.xslt	Definition of style classes for XSL-FO; conceptually equivalent to the eAIP.css file in HTML
g-supplements.xslt	html-supplements.xslt	fo-supplements.xslt	The Supplement in an eAIP document and related templates
-	html-table.xslt	fo-table.xslt	Processing of tables-related elements (especially for XSL-FO)
g-tables-ENR.xslt	html-tables-ENR.xslt	fo-tables-ENR.xslt	All ENR part specific elements, such as Route, Designated-point and Navaid
g-toc.xslt	html-toc.xslt	fo-toc.xslt	Generation of tables of contents (0.6 sections)

8.2.3. Templates Organisation

In order to separate the content from presentation as much as possible, the XSLT code was split into three parts. Most elements are handled by two or three templates. For the purpose of this documentation, we will refer to them as "the matching template", "the presentation template" and "the content template".

In general, templates will only process one element or one attribute (though presentation templates may generate several elements and attributes as needed). Therefore, attributes may have their own set of three templates.

8.2.3.1. Matching Template

Matching templates are generally defined in generic stylesheets. Most often, they will simply call the relevant presentation template. In some cases, they contain some content logic that needs to be applied before any formatting. The table of content is an example of such a case. Matching templates are normally re-used for any target format.

When content cannot be fully separated from presentation, matching templates are defined in formatting-specific stylesheets as well and they then override generic matching templates. Two XSLT mechanisms can then be used: the fact that templates defined within the importing stylesheet will always override an imported template with the same priority; or by using a more specific matching expression in the importing stylesheet.

8.2.3.2. Presentation Template

In most cases, presentation templates generate a formatting-specific element or attribute, which depends on the specific formatting targeted. They call the content template within the generated element. Default presentation templates (in generic stylesheets) only call the relevant content template.

8.2.3.3. Content Template

Content templates only generate the actual AIP content: text and other data. They are normally re-used for any target format.

8.2.4. Example

The designated-point element gives us a complete example:

8.2.4.1. Matching Template

A designated-point may be listed in the designated-points table. If the Listed attribute is set to "Yes", then the designated-point's entry in the table is generated. This if-clause is common to both HTML and other presentations, so the relevant code is located in the generic stylesheet (g-tables-ENR.xslt), in this matching template.

```
1. <xsl:template match="e:Designated-point">
2.     <xsl:if test="@Listed='Yes'"><!-- foreign points are not listed -->
3.         <xsl:call-template name="Designated-pointStyle"/>
4.     </xsl:if>
5. </xsl:template>
```

8.2.4.2. Presentation Template

In HTML a table row is generated for each (listed) designated-point:

```
1. <xsl:template name="Designated-pointStyle">
2.     <tr>
3.         <xsl:apply-templates select="@*"/>
4.         <xsl:call-template name="Designated-pointContent"/>
5.     </tr>
6. </xsl:template>
```

8.2.4.3. Content Template

Back to the generic stylesheet, where the "Designated-pointContent" template calls the relevant templates to process the content of a Designated-point element:

```
1. <xsl:template name="Designated-pointContent">
2.     <xsl:apply-templates select="e:Designated-point-ident"/>
3.     <xsl:apply-templates select="e:Latitude"/>
4.     <xsl:call-template name="Designated-pointRouteStyle"/>
5. </xsl:template>
```

8.3. Elements Organisation

8.3.1. Where does it Start?

There are two "main" stylesheets: html-content.xslt for HTML pages and html-menu.xslt for the HTML Navigation menu (ToC). These three files import many other files, which handle specific groups of eAIP elements.

8.3.2. Elements to Files Mapping

The following table shows which XSLT files handle each element.

Table 8.2.

Element	For XHTML	For XSL-FO	Element	For XHTML	For XSL-FO
/ (document root)	-	pages	Abbreviation	inline	inline
Abbreviation-description	lists	lists	Abbreviation-details	lists	lists
Abbreviation-ident	lists	lists	AD, AD-x, AD-x.y	structure	structure
Address	block	block	Address-part	block	block
Aerodrome	structure	structure	Affects	supplements	supplements
Amendment	supplements	supplements	Date-time	inline	inline
Deleted	amendments	amendments	Designated-point	tables-ENR	tables-ENR
Designated-point-ident	tables-ENR	tables-ENR	Designated-point-table	tables-ENR	tables-ENR
eAIC	eAIC-eSUP	-	eAIP	content	-
eAIP-reference	eAIC-eSUP	eAIC-eSUP	ENR, ENR-x, ENR-x.y	structure	structure
eSUP	eAIC-eSUP	-	Figure	graphics	graphics
GEN, GEN-x, GEN-x.y	structure	structure	Generated (in AD 2.1 & 3.1)	structure	structure
Generated (in sections 0.6)	toc	toc	Graphic-file	graphics	graphics
Heliport	structure	structure	Inserted	amendments	amendments
Latitude (Designated-point)	tables-ENR	tables-ENR	Latitude (Navaid)	tables-ENR	tables-ENR
Location	inline	inline	Location-definition	lists	lists
Location-ident	lists	lists	Location-name	lists	lists
Location-table	lists	lists	Longitude (Designated-point)	tables-ENR	tables-ENR
Longitude (Navaid)	tables-ENR	tables-ENR	Navaid	tables-ENR	tables-ENR
Navaid-declination	tables-ENR	tables-ENR	Navaid-elevation	tables-ENR	tables-ENR
Navaid-frequency	tables-ENR	tables-ENR	Navaid-hours	tables-ENR	tables-ENR
Navaid-ident	tables-ENR	tables-ENR	Navaid-indication	tables-ENR	tables-ENR
Navaid-indication-distance	tables-ENR	tables-ENR	Navaid-indication-radial	tables-ENR	tables-ENR
Navaid-magnetic-variation	tables-ENR	tables-ENR	Navaid-name	tables-ENR	tables-ENR
Navaid-remarks	tables-ENR	tables-ENR	Navaid-table	tables-ENR	tables-ENR
Navaid-type	tables-ENR	tables-ENR	NIL	structure	structure
Not-applicable	block	block	References	eAIC-eSUP	eAIC-eSUP
Route	tables-ENR	tables-ENR	Route-designator	tables-ENR	tables-ENR
Route-remark	tables-ENR	tables-ENR	Route-RNP	tables-ENR	tables-ENR
Route-segment	tables-ENR	tables-ENR	Route-segment-airspace-class	tables-ENR	tables-ENR
Route-segment-ATC	tables-ENR	tables-ENR	Route-segment-COP	tables-ENR	tables-ENR
Route-segment-length	tables-ENR	tables-ENR	Route-segment-lower	tables-ENR	tables-ENR
Route-segment-lower-override	tables-ENR	tables-ENR	Route-segment-mag-track	tables-ENR	tables-ENR
Route-segment-minimum	tables-ENR	tables-ENR	Route-segment-remark	tables-ENR	tables-ENR
Route-segment-remark-reference	tables-ENR	tables-ENR	Route-segment-reverse-mag-track	tables-ENR	tables-ENR
Route-segment-reverse-true-track	tables-ENR	tables-ENR	Route-segment-RNP	tables-ENR	tables-ENR

Element	For XHTML	For XSL-FO	Element	For XHTML	For XSL-FO
Route-segment-true-track	tables-ENR	tables-ENR	Route-segment-upper	tables-ENR	tables-ENR
Route-segment-usage	tables-ENR	tables-ENR	Route-segment-usage-direction	tables-ENR	tables-ENR
Route-segment-usage-level-type	tables-ENR	tables-ENR	Route-segment-usage-reference	tables-ENR	tables-ENR
Route-segment-width	tables-ENR	tables-ENR	See-supplement	supplements	supplements
Significant-point-ATC	tables-ENR	tables-ENR	Significant-point-description	tables-ENR	tables-ENR
Significant-point-reference	tables-ENR	tables-ENR	Significant-point-remark	tables-ENR	tables-ENR
Significant-point-remark-reference	tables-ENR	tables-ENR	Sub-section	structure	structure
Supplement	supplements	supplements	SUP-section	structure	structure
Title	structure	structure	a	generic	inline
br	generic	inline	caption	generic	table
cite	generic	inline	col	generic	table
colgroup	generic	table	div	block	block
em	generic	inline	li	generic	lists
ol	generic	lists	p	generic	block
span	generic	inline	strong	generic	inline
table	table	table	tbody	generic	table
td	generic	table	tfoot	generic	table
th	generic	table	thead	generic	table
tr	generic	table	ul	generic	lists

8.4. Customisation

If you need to re-use most of the provided stylesheets, but need customisation for a few templates, you can avoid modifying the provided stylesheets by::

1. Create your own XSLT file and write your templates (matching or named) in it;
2. In your XSLT file, import one of the provided main files (html-content.xslt, html-menu.xslt or fo-content.xslt), depending on your needs; in that way, your templates will override the imported ones;
3. In your templates, take care to call the correct provided templates or to "apply-templates", where applicable, to make the link to other, provided templates;
4. Alternatively, if your templates allows it, use "apply-imports" to call the default processing for the node your template matches.

8.5. Content Logic Implementation

The objective of this section is to explain how the eAIP content logic is implemented, as it may seem difficult.

8.5.1. Route Tables

The XSLT code which builds route tables in ENR 3.1 to ENR 3.5 was originally developed "by example", i.e. by closely following the ICAO AIP Specimen. This may seem too restrictive sometimes, but can be

changed: These restrictions are only present in the XSLT transformations and not in the DTD. Specifically, the following rules are implemented:

- Route-segment-width and Route-segment-COP are not considered for ENR-3.3 and 3.5;
- Route-segment-minimum and Route-segment-lower-override are not considered for 3.2, 3.3 and 3.5.

Appendix A. Glossary

AIC	Aeronautical Information Circular
AIP	Aeronautical Information Publication
AIRAC	Aeronautical Information Regulation and Control
AIS	Aeronautical Information Services
AMDT	AIP Amendment
DTD	Document Type Definition
EAD	European AIS Database
eAIP	Electronic AIP
FOP	Formatting Objects Processor
HTML	HyperText Mark-up Language
ICAO	International Civil Aviation Organisation
ISO	International Organisation for Standardisation
JRE	Java Runtime Environment
PAMS	Published AIP Management System
PDF	Portable Document Format
SUP	AIP Supplement
XML	Extensible Markup Language
XSL-FO	Extensible Stylesheet Language - Formatting Object
XSLT	Extensible Stylesheet Language Transformations