Specification Dossier

<?xml version="1.0"?><DocumentFile xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <GraphicCharterDefinitionId>0</GraphicCharterDefinitionId> <TemplateBaseTypeId>0</TemplateBaseTypeId> <CompanyId>1</CompanyId> <ConfidentialId>0</ConfidentialId> <ConfidentialDescription /> <CountryId>0</CountryId> <PageSizeId>1</PageSizeId> <PageOrientationId>1</PageOrientationId> <PrePrintedStationary>false</PrePrintedStationary> <Project>DATA+</Project> <Reference>20190830-160841-BCA</Reference> <TemplateType>3</TemplateType> <CultureId>en-GB</CultureId> <LanguageId>1</LanguageId> <Customer>NAVBLUE</Customer> <DocumentDate>2019-09-03T17:45:17.4850218+02:00</DocumentDate> <Saved>true</Saved> <IsValid>true</IsValid> <FirstPageCover>false</FirstPageCover> <IsNew>true</IsNew> <CurrentVersion>1.00</CurrentVersion> <DocumentType>Spécification Dossier</DocumentType> <DocumentTypeId>-1</DocumentTypeId> <Entity /> <HasDistributionList>false</HasDistributionList> <HasForeword>false</HasForeword> <Title>BMT Specification Dossier</Title> <Status>1</Status> <StatusDescription>Preliminary Draft</StatusDescription> <SetEdition>false</SetEdition> <SetVersion>false</SetVersion></DocumentFile>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  | |  |
|  |  |  |  | |  |
|  |  | NAVBLUE | | |  |
|  |  | DATA+ | | |  |
|  |  | Production Services Specification Dossier | | |  |
|  |  | Version 20.1.0.4 of 24 July | | |  |
|  |  |  | | |  |
|  |  |  | |  |  |

Document History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Update Origin** | **Written by** | **Verified by** |
| 19.4.0.3 | 12/09/2019 | Document initialization 19.4.0.3 | A.BARROSO |  |
| 19.4.0.4 | 07/10/2019 | Update for 19.4.0.4 |  |  |
| 19.5.0.1 | 15/10/2019 | Update for 19.5.0.1 |  |  |
| 19.5.0.2 | 28/10/2019 | Update for 19.5.0.2 | A.MOULIN | A.CHARLES |
| 19.5.0.3 | 26/11/2019 | Update for 19.5.0.3 | A.MOULIN | A.CHARLES |
| 19.5.0.4 | 09/12/2019 | Update for 19.5.0.4 | A.MOULIN | A.CHARLES |
| 20.0.0.1 | 24/01/2020 | Update for 20.0.0.1 | M.SOUBIRAN | A.CHARLES |
| 20.0.0.3 | 02/03/2020 | Update for 20.0.0.3 | A.MOULIN |  |
| 20.0.0.4 | 23/03/2020 | Update for 20.0.0.4 | A.MOULIN |  |
| 20.1.0.1 | 12/05/2020 | Update for 20.1.0.1 | A.MOULIN |  |
| 20.1.0.2 | 27/05/2020 | Update for 20.1.0.2 | A.MOULIN |  |
| 20.1.0.3 | 15/06/2020 | Update for 20.1.0.3 | Y. LEGLISE |  |
| 20.1.0.4 | 16/07/2020 | Update for 20.1.0.4 | M. BENCHERIF | A.MOULIN |

Summary

[1.1 Introduction 5](#_Toc19526837)

[1.2 Purpose of the document 5](#_Toc19526838)

[1.3 Scope of the document 5](#_Toc19526839)

[1.4 Terminology 5](#_Toc19526840)

[1.5 Aerodatabase production requirements 6](#_Toc19526841)

[1.1. Overview 6](#_Toc19526842)

[1.2. Database consolidation 6](#_Toc19526843)

[1.2.1. Web service Interfaces 6](#_Toc19526844)

[1.2.2. Data models 38](#_Toc19526845)

[1.2.3. Consolidation Map 49](#_Toc19526846)

[1.3. Data conversion 50](#_Toc19526847)

[1.3.1. Conversion engine 51](#_Toc19526848)

[1.3.2. Conversion engine service 66](#_Toc19526849)

[1.3.3. Convert Delivery 69](#_Toc19526850)

[1.3.4. AIXM Façade 70](#_Toc19526851)

[1.3.5. A816 Façade 70](#_Toc19526852)

[1.3.6. AODB Façade 75](#_Toc19526853)

[1.3.7. A424 Façade 75](#_Toc19526854)

[1.3.8. Dynamo Façade 87](#_Toc19526855)

[1.3.9. Minima Façade 96](#_Toc19526856)

[1.4. Data packaging 97](#_Toc19526857)

[1.4.1. Product Packaging 97](#_Toc19526858)

[1.5. Data qualification 100](#_Toc19526859)

[1.5.1. Cross Check Treatment 100](#_Toc19526860)

[1.5.2. Evaluate Obstacles Traceability 107](#_Toc19526861)

[1.5.3. Adjust Mora From Obstacles 109](#_Toc19526862)

[1.6. Data server 112](#_Toc19526863)

[1.6.1. Overview 112](#_Toc19526864)

[1.6.2. Data server datamodel 112](#_Toc19526865)

[1.6.3. GEO Json format 112](#_Toc19526866)

[1.6.4. Security 114](#_Toc19526867)

[1.6.5. Services 114](#_Toc19526868)

[1.6.6. Mapping file 122](#_Toc19526869)

[1.7. Data production server 124](#_Toc19526870)

[1.7.1. Overview 124](#_Toc19526871)

[1.7.2. Data model 125](#_Toc19526872)

[1.7.3. Services 127](#_Toc19526873)

[1.6 SDA424 Delta tool 144](#_Toc19526874)

[1.7 Overview 144](#_Toc19526875)

[1.8 Business rules 144](#_Toc19526876)

# Introduction

## Purpose of the document

The Specification Dossier:

* Generates a comprehensive description of the operational scope,
* Defines both functional and technical constraints, e.g. functionalities, user interface design…

This document is written during the Solution Definition phase and specifies in detail the solution:

* Responding to the Business Requirements, described in the Business Requirements Dossier,
* According to the architecture and solution described by the Architecture Dossier,
* For the whole system, or for a subsystem defined in the Architecture Dossier,
* It refines the System Requirements from the Architecture Dossier into Component Requirements for a specific subsystem or set of subsystems.

The Specification Dossier is compulsory and it is dedicated to milestone M7, Development launching.

## Scope of the document

The purpose of this document is to detail the DATA+ project specifications for Production Services development.

## Terminology

|  |  |
| --- | --- |
| TL | Team Leader |
| ATL | Assistant Team Leader |
| RCA | Root Cause Analysis |

# Aerodatabase production requirements

## Overview

This component offers services to import convert and produce aeronautical databases.

Aerodatabse production system

Aeronautical database

import

Aeronautical database

Convert and export

This system’s main functionalities are:

* Import data and store it
* Convert to formats

## Database consolidation

The database consolidation component is a set of web services and databases which allow importing aeronautical data in various formats and storing it.

### Web service Interfaces

#### Load and store RDB Shapefile

This web service allows to import a RDB Shapefile.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LSR-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #9.1-10 |
| SD-ALB-LSR-020 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #9.1-10 |
| SD-ALB-LSR-030 | GDAL | This service uses GDAL to import input Shapefile in database. | #9.1-10 |
| SD-ALB-LSR-050 | Uniqueness | If any line is found any table for the tuple:   * ICAO * AIRAC * Version   Then the service doesn’t import the Shapefile and an error message is returned. | #9.1-10 |
| SD-ALB-LSR-060 | Name | LoadAndStoreRDBData | #9.1-10 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LSR-100 | Parameter icaoCode | Type: string  The icao code of the imported airport.  Must not be null or empty. | #9.1-10 |
| SD-ALB-LSR-110 | Parameter  airacCycle | Type: string  The airac cycle of the imported airport.  Must not be null or empty. | #9.1-10 |
| SD-ALB-LSR-120 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #9.1-10 |
| SD-ALB-LSR-130 | Parameter  filePath | Type: string  The path to the directory containing files to import.  Must not be null or empty.  The directory must exist. | #9.1-10 |
| SD-ALB-LSR-140 | Output | Type: string  In case of success the string is empty.  In case of error the string starts with an error code followed by a description message.  TBC | #9.1-10 |
| SD-ALB-LSR-150 | GDAL command | The GDAL command used to import the Shapefile is:  ogr2ogr –append –f MSSQLSpatial “MSSQL:server=<connection string>” <ShapefileInputPath> -explodecollections | #8.1-11 |

#### Load and store AMDB PSW Shapefile

This web service allows to import an AMDB PSW Shapefile.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LSA-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #8.2-05 |
| SD-ALB-LSA-020 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #8.2-05 |
| SD-ALB-LSA-030 | GDAL | This service uses GDAL to import input Shapefile in database. | #8.2-05 |
| SD-ALB-LSA-050 | Uniqueness | If any line is found in any table for the tuple:   * ICAO * AIRAC * Version   Then the service doesn’t import the Shapefile and an error message is returned. | #8.2-05 |
| SD-ALB-LSA-060 | Name | LoadAndStoreAMDBData | #8.2-05 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LSA-100 | Parameter icaoCode | Type: string  The icao code of the imported airport.  Must not be null or empty. | #8.2-05 |
| SD-ALB-LSA-110 | Parameter  airacCycle | Type: string  The airac cycle of the imported airport.  Must not be null or empty. | #8.2-05 |
| SD-ALB-LSA-120 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #8.2-05 |
| SD-ALB-LSA-130 | Parameter  filePath | Type: string  The path to the directory containing files to import.  Must not be null or empty.  The directory must exist. | #8.2-05 |
| SD-ALB-LSA-140 | Output | Type: string  In case of success the string is empty.  In case of error the string starts with an error code followed by a description message.  TBC | #8.2-05 |
| SD-ALB-LSA-150 | GDAL command | The GDAL command used to import the Shapefile is:  ogr2ogr –append –f MSSQLSpatial “MSSQL:server=<connection string>” <ShapefileInputPath> -explodecollections | #8.1-11 |

#### Load and store AMDB GEO Shapefile

This web service allows to import an AMDB GEO Shapefile.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LSG-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #12.1-02 |
| SD-ALB-LSG-020 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #12.1-02 |
| SD-ALB-LSG-030 | GDAL | This service uses GDAL to import input Shapefile in database. | #12.1-02 |
| SD-ALB-LSG-050 | Uniqueness | If any line is found in any table for the tuple:   * ICAO * AIRAC * Version   Then the service doesn’t import the Shapefile and an error message is returned. | #12.1-02 |
| SD-ALB-LSG-060 | Name | LoadAndStoreAMDBData | #12.1-02 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LSG-100 | Parameter icaoCode | Type: string  The icao code of the imported airport.  Must not be null or empty. | #12.1-02 |
| SD-ALB-LSG-110 | Parameter  airacCycle | Type: string  The airac cycle of the imported airport.  Must not be null or empty. | #12.1-02 |
| SD-ALB-LSG-120 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #12.1-02 |
| SD-ALB-LSG-130 | Parameter  filePath | Type: string  The path to the directory containing files to import.  Must not be null or empty.  The directory must exist. | #12.1-02 |
| SD-ALB-LSG-140 | Output | Type: string  In case of success the string is empty.  In case of error the string starts with an error code followed by a description message.  TBC | #12.1-02 |
| SD-ALB-LSG-150 | GDAL command | The GDAL command used to import the Shapefile is:  ogr2ogr –append –f MSSQLSpatial “MSSQL:server=<connection string>” <ShapefileInputPath> -explodecollections | #12.1-02 |

#### Load and store AIP AIXM

This web service allows importing an AIXM file.

The AIXM file is converted in a Shapefile and then imported with GDAL.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LSX-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #8.1-14 |
| SD-ALB-LSX-020 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #8.1-14 |
| SD-ALB-LSX-030 | GDAL | This service uses GDAL to import input Shapefile in database.  If ACN value of parkingStandLocation have a length greater than 255 characters, we must fill the database field CAN with AIXM value and not the shapefile value. | #8.1-14  DLT-154 |
| SD-ALB-LSX-050 | Uniqueness | If any line is found in any table for the tuple:   * ICAO * AIRAC * Version   Then the service doesn’t import the Shapefile and an error message is returned. | #8.1-14 |
| SD-ALB-LSX-060 | Name | LoadAndStoreAIPData | #8.1-14 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LSX-100 | Parameter icaoCode | Type: string  The icao code of the imported airport.  Must not be null or empty. | #8.1-14 |
| SD-ALB-LSX-110 | Parameter  airacCycle | Type: string  The airac cycle of the imported airport.  Must not be null or empty. | #8.1-14 |
| SD-ALB-LSX-120 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #8.1-14 |
| SD-ALB-LSX-130 | Parameter  filePath | Type: string  The path to the file to import.  Must not be null or empty.  The file must exist. | #8.1-14 |
| SD-ALB-LSX-140 | Output | Type: string  In case of success the string is empty.  In case of error the string starts with an error code followed by a description message.  TBC | #8.1-14 |
| SD-ALB-LSX-150 | GDAL command | The GDAL command used to import the Shapefile is:  ogr2ogr –append –f MSSQLSpatial “MSSQL:server=<connection string>” <ShapefileInputPath> -explodecollections | #8.1-11 |
| SD-ALB-LSX-151 | GDAL Command AIXM to Shapefile | The GDAL command used to transform the AIXM in Shapefile is:  ogr2ogr –overwrite –fieldTypeToString StringList,RealList,IntegerList [Target\_Path] [Source\_File\_Path] | #8.1-14 |

#### Load and store ETOD Shapefile

This web service allows to import an eTOD Shapefile.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LSE-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #8.1-15 |
| SD-ALB-LSE-020 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #8.1-15 |
| SD-ALB-LSE-030 | GDAL | This service uses GDAL to import input Shapefile in database. | #8.1-15 |
| SD-ALB-LSE-050 | Uniqueness | If any line is found in any table for the tuple:   * ICAO * AIRAC * Version   Then the service doesn’t import the Shapefile and an error message is returned. | #8.1-15 |
| SD-ALB-LSE-060 | Name | LoadAndStoreETODData | #8.1-15 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LSE-100 | Parameter icaoCode | Type: string  The icao code of the imported airport.  Must not be null or empty. | #8.1-15 |
| SD-ALB-LSE-110 | Parameter  airacCycle | Type: string  The airac cycle of the imported airport.  Must not be null or empty. | #8.1-15 |
| SD-ALB-LSE-120 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #8.1-15 |
| SD-ALB-LSE-130 | Parameter  filePath | Type: string  The path to the directory containing files to import.  Must not be null or empty.  The directory must exist. | #8.1-15 |
| SD-ALB-LSE-140 | Output | Type: string  In case of success the string is empty.  In case of error the string starts with an error code followed by a description message.  TBC | #8.1-15 |
| SD-ALB-LSE-150 | GDAL command | The GDAL command used to import the Shapefile is:  ogr2ogr –append –f MSSQLSpatial “MSSQL:server=<connection string>” <ShapefileInputPath> -explodecollections | #8.1-11 |

#### Load and store AODB file

This web service allows to import an AODB file.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAO-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #10.2-02 |
| SD-ALB-LAO-020 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #10.2-02 |
| SD-ALB-LAO-050 | Uniqueness | If any line is found any table for the tuple::   * ICAO (for any ICAO in the file) * AIRAC * Version   Then the service doesn’t import the AODB file and an error message is returned. | #10.2-02 |
| SD-ALB-LAO-060 | Name | LoadAndStoreAODBData | #10.2-02 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAO-110 | Parameter  airacCycle | Type: string  The airac cycle of the imported file.  Must not be null or empty. | #10.2-02 |
| SD-ALB-LAO-120 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #10.2-02 |
| SD-ALB-LAO-130 | Parameter  filePath | Type: string  The path to the file to import.  Must not be null or empty.  The file must exist. | #10.2-02 |
| SD-ALB-LAO-140 | Output | Type: string  In case of success the string is empty.  In case of error the string starts with an error code followed by a description message. | #10.2-02 |

#### Load and store A424 file

This web service allows importing an A424 file.

It manages multi-model data loading. It is possible to load item contained in different data models from a single file and persist each item independently in the associated data base:



| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAN-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #12.1-05 |
| SD-ALB-LAN-020 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #12.1-05 |
| ~~xSD-ALB-LAN-050~~ | ~~Uniqueness~~ | ~~If any line is found any table for the tuple::~~   * ~~ICAO (for any ICAO in the file)~~ * ~~AIRAC~~ * ~~Version~~   ~~Then the service doesn’t import the NavDB file and an error message is returned.~~ | ~~#12.1-05~~  #32.2-02 |
| SD-ALB-LAN-060 | Name | LoadAndStoreA424Data (A424 model)  LoadAndStoreRDBA424Data (RDB A424 model) | #12.1-05  #31.2-03  #32.1-02 |
| SD-ALB-LAN-070 | Data Model | The web service manages all A424 features of all the defined A424 data model (see §10.2.2.1) including tailored and continuation records. | #31.2-03  #32.1-02  #32.2-02  #35.1-01 |
| SD-ALB-LAN-080 | Header | The web service does not expect the presence of the header records.  Header records is identified and managed if present.  The absence of header records does not raise an exception and interrupt the loading; the header not found information is only logged as a warning. | #31.2-03 |
| SD-ALB-LAN-090 | Record identification process | The record identification process can be described by this algorithm: | #31.2-03  #32.1-02 |
| SD-ALB-LAN-030 | multi-model data loading | The web service is able to load item contained in different data models from a single file and persist each item independently in the associated data base. | #32.2-02 |
| SD-ALB-LAN-040 | Data Inventory | A new entry is created in DataInventory table for each DataSource/Icao(could be $NE)/Airac/version/ElementTypes loaded, only if not yet existing.  The ‘ElementTypes’ is the list of Record Type (section + sub-section code) loaded for an Icao. This list is updated after a new loading for same Airac/Version.  The data inventory management (insert / update avoids unnecessary multiple access:  The **navaid\_vhf** records (D) entries in data inventory is created:   * As A424 **icao** source when the airport identifier (icao) **is not** empty for the loaded record * As A424 **non icao** source when the airport identifier (icao) **is** empty for the loaded record | #CM-151  #32.2-02  #36.2-15 |
| SD-ALB-LAN-050 | Manage transaction and pagination | The data loading method for A424 manages transaction and pagination by loading and persisting a number of rows at a time to avoid filling memory.  The number of primary record parsed for each “page” is configurable to allow adjustment if needed.  Data model objects parsed and created from a page (i.e. number of primary record) are persisted into database and disposed so that memory is free for the next page.  The **A424 data consolidation** uses bulk insert/save methods instead of standard save. | #35.1-01  #36.2-15  #19.1.0.1-02 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAN-110 | Parameter  airacCycle | Type: string  The airac cycle of the imported nav db (may be different from the one specified in the database).  Must not be null or empty. | #12.1-05 |
| SD-ALB-LAN-120 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #12.1-05 |
| SD-ALB-LAN-130 | Parameter  filePath | Type: string  The path to the file to import.  Must not be null or empty.  The file must exist. | #12.1-05 |
| SD-ALB-LAN-140 | Output | Type: string  In case of success the string is empty.  In case of error the string starts with an error code followed by a description message. | #12.1-05 |
| SD-ALB-LAN-150 | Parameter  consolidationIdentifier | Type: string  Value: the name of the consolidation map. This is the name of the file without the .xml suffix.  Constraints: Optional. If file is not present, the data loading uses the A424-20 data model. | #32.2-02 |

#### Load and store A816 file

This web service allows importing an A816 file.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LA8-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #13.2-08 |
| SD-ALB-LA8-020 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #13.2-08 |
| SD-ALB-LA8-050 | Uniqueness | If any line is found any table for the tuple::   * ICAO (for any ICAO in the file) * AIRAC * Version   Then the service doesn’t import the A816 file and an error message is returned. | #13.2-08 |
| SD-ALB-LA8-060 | Name | LoadAndStoreA816 | #13.2-08 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LA8-110 | Parameter  airacCycle | Type: string  The airac cycle of the imported A816 (may be different from the one specified in the database).  Must not be null or empty. | #13.2-08 |
| SD-ALB-LA8-120 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #13.2-08 |
| SD-ALB-LA8-130 | Parameter  filePath | Type: string  The path to the file to import.  Must not be null or empty.  The file must exist. | #13.2-08 |
| SD-ALB-LA8-131 | Parameter source | Type: string  The type of the source data, RDB or AMDB | #14.1-10 |
| SD-ALB-LA8-140 | Output | Type: string  In case of success the string is empty.  In case of error the string starts with an error code followed by a description message. | #13.2-08 |

#### Load and store Reference Data file

This web service allows importing an .ini file containing reference data into ADBLucem data base.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAR-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #23.1-02 |
| SD-ALB-LAR-020 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #23.1-02 |
| SD-ALB-LAR-030 | File Format | The web service takes as input an .ini file containing reference data and extracts information from it. | #23.1-02 |
| SD-ALB-LAR-040 | Name | LoadAndStoreReferenceData | #23.1-02 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAR-100 | Parameter  filePath | Type: string  The path to the .ini file to import.  Must not be null or empty.  The file must exist. | #23.1-02 |
| SD-ALB-LAR-140 | Output | Type: string  In case of success the string is empty.  In case of error the string starts with an error code followed by a description message. | #23.1-02 |

Business Rules:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAR-200 | ADBLucem version | If data already exists in ADB Lucem for the same ICAO/AIRAC couple, the existing version is incremented by one (+1) to get the actual version for creating the features. | #23.1-02 |
| SD-ALB-LAR-210 | Features creation | From the .ini file, the web service creates features in ADBLucem DB from following rules:   * Creates an aerodromereferencepoint feature. Properties are fetched from corresponding AirportHeliport AIP data: geometry, iata, elevation, name. * For each different “Runway” section, it creates a runwayelement feature and use the “width” column for width property. Other required property are filled using Runway/RunwayDirection AIP data. * For each threshold entry set group (same id in .ini file), it creates a runwaythreshold feature:   + thrtype = 0 using “Extremity” coordinates entry if filled else “Threshold” coordinates entry   + thrtype = 1 using “Threshold” coordinates entry columns if “Extremity” coordinates entry is filled   + threshold properties are filled using “Threshold Elevation” or “Extremity Elevation” depending on previous rule   + availPavedSurfFromThr property is filled for all created runwaythreshold with “Hard Surface Length” entry   + The true bearing is computed for each runwaythreshold using the vincenty inverse formula.   + If “Extremity” coordinates entry is filled, it creates a *runwaydisplacedarea* feature and computes displacementLength property with the distance between “Extremity” coordinates entry and “Threshold” coordinates entry. * Imagery table is filled from the “General Data” section of the .ini file using Date of RID imagery, Ortho Absolute horizontal accuracy, DTM Absolute vertical accuracy, DTM Absolute horizontal accuracy * Source is “GEO INT” for all created features. | #23.1-02 |

Tester Tool:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAR-300 | tester | The Consolidation Service tester tool allows testing LoadAndStoreReferenceData web service interface. | #23.1-03 |

#### Load and store Grid MORA Data file

This web service allows importing an A424 grid mora file into grid MORA+ data base.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAM-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #28.1-02 |
| SD-ALB-LAM-020 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #28.1-02 |
| SD-ALB-LAM-030 | File Format | The web service takes as input an A424 grid mora file containing the moras and extracts information from it. | #28.1-02 |
| SD-ALB-LAM-040 | Name | LoadAndStoreGridMoraData | #28.1-02 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAM-100 | Parameter  filePath | Type: string  The path to the A424 grid mora file to import.  Must not be null or empty.  The file must exist. | #28.1-02 |
| SD-ALB-LAM-110 | Parameter  airacCycle | Type: string  The airac cycle of the import.  Must not be null or empty. | #28.1-02 |
| SD-ALB-LAM-120 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #28.1-02 |

Business Rules:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAM-200 | Grid Mora version | If data already exists in Data Inventory for the same Airac/version couple, the process stops and an error is logged. | #28.1-02 |
| SD-ALB-LAM-210 | Mora creation | From the A424 grid mora file, the web service creates features in Grid Mora+ DB from following rules:   * Each record of the a424 grid mora file is processed and data extracted to build mora objects and persist them into database. * 1° x 1° Polygon geometry for each mora is built from starting lat/lon values and index in the record (+/- depending on E/W). * The construction ensures that the lower left corner is the first point of the polygon geometry. * Elevation value is in hundreds of feet in a424, it is to feet in mora object. If mora value is “UNK” in A424 file, elevation value shall be null. * Airac and version values comes from the consolidation service parameters. * Source is “AD&S” for all created features. * Comment is left empty | #28.1-02 |

Tester Tool:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAM-300 | tester | The Consolidation Service tester tool allows testing LoadAndStoreGridMoraData web service interface. | #28.1-02 |

#### Load and store CompanyRoute Data file

This web service allows importing a CompanyRoute file into CompanyRoute data base.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAC-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #32.1-04 |
| SD-ALB-LAC-020 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #32.1-04 |
| SD-ALB-LAC-030 | File Format | The web service takes as input a CompanyRoute file containing the company routes and extracts information from it. | #32.1-04 |
| SD-ALB-LAC-040 | Name | LoadAndStoreCompanyRoute | #32.1-04 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAC-100 | Parameter  filePath | Type: string  The path to the CompanyRoute file to import.  Must not be null or empty.  The file must exist. | #32.1-04 |
| SD-ALB-LAC-110 | Parameter  airacCycle | Type: string  The airac cycle of the import.  Must not be null or empty. | #32.1-04 |
| SD-ALB-LAC-120 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #32.1-04 |
| SD-ALB-LAC-130 | Parameter Customer Code | Type: 3 character string  The customer code of the import.  Must not be null or empty | #33.2-04 |

Business Rules:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAC-200 | CompanyRoute inventory | The data loading of a company route file implies the creation of an entry in data inventory table for the company route data source with the given airac and version, icao code is filled with ‘$NE’.  The customer code value is added to the element list column of data inventory entry.  If data already exists in Data Inventory for the same Airac/version couple, the element list is completed with the new customer code. If the customer code is already in element list the process stops and an error is logged. | #32.1-04  #33.2-04 |
| SD-ALB-LAC-210 | CompanyRoute creation | The company routes imported files are in a structured text file format (called book format) that is parsed and loaded into the CompanyRoute data model and stored into the consolidated database.  The imported file is parsed using following rules:  Routes  Each route consists of the following items, in the order below :  <Name><S>[<Parameter>]<Origin>[<Via><Fix>]<S><Destination>  Where:   * The separator <S> can be 1 or more spaces, new-lines or tabs. * There can only be 1 <name>, <origin> and <destination>. * The number of <parameter> can be zero, 1 or more. * There must be at least 1 <fix>. * Every <fix> must be preceded by a <via>.   Comments  If a “#” character occurs on a line, the rest of the line is taken to be a comment and ignored.  Route Name  The format for the <name> field is :  <Route-name>:<cycle>  Where   * <route-name> is a single word up to 10 characters long that may contain any combination of the characters A-Z or 0-9. * <cycle> must be a valid AIRAC cycle date (4 digits).   Parameters  They apply to the whole route and are in the form: -<letter><value>  They can be :   * -A<alternate\_airport> EX. -AEGLL:EG * -C<cruise\_altitude> EX. -CFL350 * -D<alternate\_distance> EX. -D220 * -I<cost\_index> EX. -I110 * -K<key\_airport> EX. -KEGLL:EG   Origin, Destination and Fix  These all share the same format :  <identifier>:<icao\_code>:<section\_code><subsection\_code>  Where   * The ICAO country code is used for <icao\_code> on all enroute fixes and for airport reference points. * The ICAO airport identifier is used for <icao\_code> on all terminal fixes, such as runways, markers and Terminal waypoints. * If <subsection\_code> is a space, it will be omitted (as for VHF navaids "D").   Examples: EGLL:EG:PA , MAY:EG:D , HOLLY:EG:EA , OM27:EGNX:PM  The Fix rank is given by its position in the list of fixes for a Company Route.  Via  The format for via is :  .<name>[:<type>:<start\_transition>:<end\_transition>].  For SID and STAR the actual transitions signification depends on the via type :   * .<name>:SID:<runway\_transition>:<enroute\_transition>. * .<name>:STR:<enroute\_transition>:<runway\_transition>.   Examples :   * Direct: .. * Airway: .UA34. * SID : .BEJ2U:SID:RW29:BIG. * STAR: .CLSALT:STR:BIG:RW29.   All of the fields are optional, but ":" characters must be placed around fields to be skipped, as shown below (skipping the enroute transition for a STAR): .CLSALT:STR::RW29.    All via's must start and end with a "." character. | #32.1-04 |
| SD-ALB-LAC-220 | CompanyRoute customer code | The customer code attribute of company route objects is filled from the customer code parameter value for all company routes contained in a given file. | #33.2-04 |

Tester Tool:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAC-300 | tester | The Consolidation Service tester tool allows testing LoadAndStoreCompanyRouteData web service interface. | #32.1-04 |

#### Load and store Obstacle Data file

This web service allows importing an Obstacle excel file into Obstacle data base.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LSO-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #29.1-02 |
| SD-ALB-LSO-020 | Logs levels | The web service generates a log file for tracing the loading process and possible errors for each execution. The error(s) log file name contains the name of the country considered by the current Excel capture file.  Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #29.1-02 |
| SD-ALB-LSO-030 | File Format | The web service takes as input an Excel file containing the obstacles and extracts information from it. | #29.1-02 |
| SD-ALB-LSO-040 | Name | LoadAndStoreObstacleData | #29.1-02 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LSO-100 | Parameter  filePath | Type: string  The path to the Excel file to import.  Must not be null or empty.  The file must exist. | #29.1-02 |
| SD-ALB-LSO-110 | Parameter  Country Code | Type: string  The country code of the import.  Must not be null or empty. | #29.1-02 |
| SD-ALB-LSO-120 | Parameter  Airac Cycle | Type: string  The airac cycle of the import.  Must not be null or empty. | #29.1-02 |
| SD-ALB-LSO-130 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #29.1-02 |
| SD-ALB-LSO-140 | Parameter Latitude pattern | Type: string  The pattern format for latitude coordinate in input file.  Must not be null or empty. | #29.1-02 |
| SD-ALB-LSO-150 | Parameter Longitude pattern | Type: string  The pattern format for longitude coordinate in input file.  Must not be null or empty. | #29.1-02 |

Business Rules:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LSO-200 | Obstacle version | If data already exists in Data Inventory for the same CountryCode/Airac/version couple, the process stops and an error is logged. | #29.1-02 |
| SD-ALB-LSO-210 | Obstacle checks | The consolidation service performs consistency checks before loading the file contents:   * Semantic consistency: check that the value data types and value domains are respected for each attribute. The automatic validation semantic consistency checks is performed on all obstacles' attributes of the Excel capture file before raising the success or error log. Empty field acceptable and are not detected like errors. See file Obstacle\_capture\_definition.xlsx for details on dat type and domain:   + In case of error on the value data type of an obstacle, the error raised is "LINE xxx - ATTRIBUTE xxx - DATA TYPE ERROR" with xxx the number of the Excel line and yyy the attribute with the value data type error.   + In case of error on the value domain of an obstacle, the error raised is "LINE xxx - ATTRIBUTE xxx - OUT-OF-RANGE ERROR" with xxx the number of the Excel line and yyy the attribute with the value domain error. * Logical consistency: check that some attributes are consistent between each other). 4 types of logical consistency checks are defined:   + **feattype and latitude/longitude**: check that if feattype value is equal to ""point"" (resp. ""line"" or ""polygon""), then latitude and longitude attributes contains one single coordinate (resp. at least two coordinates or three coordinates).   + **lat and long**: check that the number of latitude coordinates provided for each obstacle is the same than the number of longitude coordinates.   + **latitude/longitude and latlongunit**: check that if latlongunit value is equal to "DD", then latitude (resp. longitude) is between -90 and 90 (resp. between -180 and 180); check that if latlongunit value is equal to "DMS", then degrees / minutes / seconds are between [0;90] for latitude & [0;180] for longitude / [0;60] / [0;60].   + Geometry checks on constructed polygons to verify “is valid” and “is simple” criteria. * The automatic validation logical consistency checks is performed on all obstacles' attributes of the Excel capture file before raising the success or error log:   + In case of error on feattype vs lat/long, the error raised is "LINE xxx - FEATTYPE NOT CONSISTENT WITH NUMBER OF LAT/LONG COORDINATES" with xxx the number of the Excel line.   + In case of error on lat vs long, the error raised is "LINE xxx - NUMBER OF COORDINATES NOT CONSISTENT BETWEEN LATITUDE AND LONGITUDE" with xxx the number of the Excel line.   + In case of error on lat/long vs latlongunit, the error raised is "LINE xxx - LAT/LONG COORDINATES VALUES NOT CONSISTENT WITH LATLONGUNIT" with xxx the number of the Excel line.   + In case of error on geometry check, the error to be raised shall be “LINE xxx – INVALID GEOMETRY” with xxx the number of the Excel line.   Failure on checks prevent the loading. | #29.1-02 |
| SD-ALB-LSO-220 | Obstacle creation | From the obstacle capture Excel file, the web service creates features in Obstacle DB from following rules:   * Identify the obstacle type (point, line, polygon) to create the proper obstacle data model object * Compute the geometry from the contents of the WKT column if not empty or from the lat & long columns. The points list making up the geometry is rebuilt based on sequential lat & long values from each cell. For polygons, the geometry construction process adds the first point at the last position of the polygon outline (if not present) to make the polygon geometry valid later on. * Latitude and longitude are decoded from file columns using the patterns provided as parameters. This pattern is a regular expression matching the DMS coordinates:   + Group1 = degrees   + Group2 = minutes   + Group3 = seconds * Fill other attributes in data model based on excel file columns. See file Obstacle\_model\_definition.xlsx for details. | #29.1-02 |
| SD-ALB-LSO-230 | Obstacle Data inventory | After persistence of obstacles in database, a new entry is created in Data Inventory with following values :   * DataSource = obstacles * Icao = $NE (meaning not associated to any airport icao) * Country = country code from consolidation request * Airac = airac cycle from consolidation request * Version = version from consolidation request | #29.1-02 |

Tester Tool:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LSO-300 | tester | The Consolidation Service tester tool allows testing LoadAndStoreObstacleData web service interface. | #29.1-02 |

#### Load and store Minima file

This web service allows importing a Minima Json file into Minima database.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LSM-001 | Load and Store Minima | URL : https://<root>/loadstore\_minima/  Method : POST  Body: Current Minima Json file  Process:   * Parse Json file and load transmitted data into the Minima data model of the consolidated database * Create an entry into the data inventory for this icao, airac and version and the minima data source   HTTP status code :   * 200 – Ok when Minima persisted into database * 400 – When Json file format is not as expected * 409 - if conflict, Minima already exist in Data inventory for same Icao/Airac/Version * 500 otherwise | #35.1-07  #36.1-05 |

#### Load and store Airport Runway Data file

This web service allows importing airportdata.csv and runwaydata.csv files provided by Gearshift into AirportRunway data base.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAA-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #35.2-11 |
| SD-ALB-LAA-020 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #35.2-11 |
| SD-ALB-LAA-030 | File Format | The web service takes as input airportdata.csv and runwaydata.csv files provided by Gearshift and extracts information from it. | #35.2-11 |
| SD-ALB-LAA-040 | Name | LoadAndStoreAirportRunwayData | #35.2-11 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAA-100 | Parameter  Airport filePath | Type: string  The path to the airportdata.csv file to import.  Can be null or empty.  If value different than null or empty, the file must exist. | #35.2-11 |
| SD-ALB-LAA-110 | Parameter  Runway filePath | Type: string  The path to the runwaydata.csv file to import.  Can be null or empty.  If value different than null or empty, the file must exist. | #35.2-11 |
| SD-ALB-LAA-120 | Parameter  airacCycle | Type: string  The airac cycle of the import.  Must not be null or empty. | #35.2-11 |
| SD-ALB-LAA-130 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #35.2-11 |

Business Rules:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAA-200 | Airac, version management | The airac and version parameters are used as source for the airac and version attributes of all the objects created. | #35.2-11 |
| SD-ALB-LAA-210 | Airport Runway creation | The gearshift airports and runways are in coma separated format that are parsed and loaded into the associated data model and stored into the consolidated database. | #35.2-11 |
| SD-ALB-LAA-220 | Data Inventory management | The data loading of gearshift airports and/or runways imply creation of the data inventory entries.  A new line is created in data inventory table for each icao loaded from airport or runway source file. | #35.2-11 |

Tester Tool:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LAA-300 | tester | The Consolidation Service tester tool allows testing LoadAndStoreAirportRunwayData web service interface. | #35.2-11 |

#### Load and store Enroute Data file

This web service allows importing EnRoute shapefiles into the database.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LER-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #MD-125 |
| SD-ALB-LER-020 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #MD-125 |
| SD-ALB-LER-030 | File Format | The web service takes as input EnRoute shapefiles and extracts information from it. | #MD-125  #MD-538 |
| SD-ALB-LER-040 | Name | LoadAndStoreEnRouteData | #MD-125  #MD-538 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LER-110 | Parameter  Enroute folder Path | Type: string  The path to the Enroute folder containing the .shp files to import.  Can be null or empty.  If value different than null or empty, the file must exist. | #MD-125 |
| SD-ALB-LER-120 | Parameter  airacCycle | Type: string  The airac cycle of the import.  Must not be null or empty. | #MD-125 |
| SD-ALB-LER-130 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #MD-125 |

Business Rules:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LER-200 | Airac, version management | The airac and version parameters are used as source for the airac and version attributes of all the objects created. | #MD-125 |
| SD-ALB-LER-210 | Airport Runway creation | Enroute source files are shape files loaded into the associated data model and stored into the consolidated database. | #MD-125 |
| SD-ALB-LER-220 | Data Inventory management | The data loading of Enroute data imply creation of the data inventory entries.  A new line is created in data inventory table for each icao loaded from airport source file and one line for all objects not related to an Airport. | #MD-125 |

Tester Tool:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LER-300 | tester | The Consolidation Service tester tool allows testing LoadAndStoreEnRouteData web service interface. | #MD-125 |

#### Load and store ADI Data file

This web service allows importing ADI airport and runway data into the database.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-ADI-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #MD-126 |
| SD-ALB-ADI-020 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #MD-126 |
| SD-ALB-ADI-030 | File Format | The web service takes as input ADI.txt and extracts information from it. | #MD-126 |
| SD-ALB-ADI-040 | Name | LoadAndStoreADIData | #MD-126 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-ADI-110 | Parameter  ADI filePath | Type: string  The path to th ADI.txt file to import.  Can be null or empty.  If value different than null or empty, the file must exist. | #MD-126 |
| SD-ALB-ADI-120 | Parameter  Airac Cycle | Type: string  The airac cycle of the import.  Must not be null or empty. | #MD-126 |
| SD-ALB-ADI-130 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #MD-126 |

Business Rules:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-ADI-200 | Airac, version management | The airac and version parameters are used as source for the airac and version attributes of all the objects created. | #MD-126 |
| SD-ALB-ADI-210 | Airport Runway creation | The airports and runways are in pipe separated format that are parsed and loaded into the associated data model and stored into the consolidated database. | #MD-126 |
| SD-ALB-ADI-220 | Data Inventory management | The data loading of airports and/or runways imply creation of the data inventory entries.  A new line is created in data inventory table for each icao loaded from airport or runway source file. | #MD-126 |

Tester Tool:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-ADI-300 | tester | The Consolidation Service tester tool allows testing LoadAndStoreADIData web service interface. | #MD-126 |

#### Load and store NaturalEarth Data file

This web service allows importing NaturalEarth shapefiles into the database.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LEA-010 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #MD-400 |
| SD-ALB-LEA-020 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #MD-400 |
| SD-ALB-LEA-030 | File Format | The web service takes as input NaturalEarth shapefiles and extracts information from it. | #MD-400 |
| SD-ALB-LEA-040 | Name | LoadAndStoreNaturalEarthData | #MD-400 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LEA-110 | Parameter  NaturalEarth folder Path | Type: string  The path to the NaturalEarth folder containing the .shp files to import.  Can be null or empty.  If value different than null or empty, the file must exist. | #MD-400 |
| SD-ALB-LEA-120 | Parameter  airacCycle | Type: string  The airac cycle of the import.  Must not be null or empty. | #MD-400 |
| SD-ALB-LEA-130 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #MD-400 |

Business Rules:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LEA-200 | Airac, version management | The airac and version parameters are used as source for the airac and version attributes of all the objects created. | #MD-400 |
| SD-ALB-LEA-210 | NaturalEarth creation | NaturalEarth source files are shape files loaded into the associated data model and stored into the consolidated database. | #MD-400 |
| SD-ALB-LEA-220 | Data Inventory management | The data loading of NaturalEarth data imply creation of the data inventory entries.  A new line is created in data inventory table for each airac and version. | #MD-400 |

Tester Tool:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-LEA-300 | tester | The Consolidation Service tester tool allows testing LoadAndStoreNaturalEarthData web service interface. | #MD-400 |

### Data models

#### General Model requirements

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-GMR-001 | GMR properties | All models should have common properties in order to be manipulated in a generic way from the conversion engine of future tools. Properties are defined below. | #9.1-09 |
| SD-ALB-GMR-005 | GMR SQL | A model is stored in a SQL database | #9.1-09 |

#### Model organized by ICAO

|  |  |  |  |
| --- | --- | --- | --- |
| SD-ALB-GMR-010 | Additional columns | In addition to columns specific to a model, each table should add these columns:   * Icao * Airac * Version   They are used to uniquely identify an element of a model.  They are populated by the load services which take these values in parameters or extract them from the model.  These columns should never be null. | #9.1-09 |
| SD-ALB-GMR-020 | Data inventory creation | Once the load and store method is completed without errors, the Consolidation Service shall update the DataInventory table (in Qualification database) with the following information for each airport loaded:   * + Data source id (reference to DataSource table)   + ICAO (parameter of the load and store method or from the input data)   + AIRAC cycle (parameter of the load and store method)   + Version (parameter of the load and store method)   + LoadDateTime (execution date and time of the load and store processing | #14.2-13 |

##### RDB model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-RDM-001 | RDB Model name | Fully qualified name of the data source: **ConversionEngine.Product.RdbDataSource** | #9.1-09 |
| SD-ALB-RDM-005 | RDB Model diagram | see RDB.png | #9.1-09 |
| SD-ALB-RDM-010 | RDB Model | RDB model is an SQL database which schema has been deduced by GDAL from a Shapefile provided by Lucem.  This database can store shapefiles which respect the same schema. | #9.1-09 |
| SD-ALB-RDM-020 | EF layer | An entity framework model is built upon this model using the “database first” design. | #9.1-09 |

##### AMDB PSW model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-AMM-001 | Amdb PSW Model name | Fully qualified name of the data source: **ConversionEngine.Product.AmdbDataSource** | #9.2-01 |
| SD-ALB-AMM-005 | AMDB PSW Model diagram | see AMDB\_PSW.png | #9.2-01 |
| SD-ALB-AMM -010 | AMDB PSW Model | AMDB PSW model is an SQL database which schema has been deduced by GDAL from a Shapefile provided by Lucem.  This database can store shapefiles which respect the same schema. | #9.2-01 |
| SD-ALB-AMM -020 | AMDB PSW EF layer | An entity framework model is built upon this model using the “database first” design. | #9.2-01 |

##### AMDB GEO model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-AGM-001 | Amdb GEO Model name | Fully qualified name of the data source: **ConversionEngine.Product.AmgeoDataSource** | #12.1-01 |
| SD-ALB-AGM-005 | AMDB GEO Model diagram | see AMDB\_GEO.jpg | #12.1-01 |
| SD-ALB-AGM-010 | AMDB GEO Model | AMDB GEO model is an SQL database which schema has been deduced by GDAL from a Shapefile provided by Lucem.  This database can store shapefiles which respect the same schema. | #12.1-01 |
| SD-ALB-AGM-020 | AMDB GEO EF layer | An entity framework model is built upon this model using the “database first” design. | #12.1-01 |

##### AIP model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-AIM-001 | AIP Model name | Fully qualified name of the data source: **ConversionEngine.Product.AipDataSource** | #8.1.12 |
| SD-ALB-AIM-005 | AIP Model diagram | see AIP.png | #8.1.12  #19.2.0.2-01  #19.2.0.2-07 |
| SD-ALB-AIM-010 | AIP Model | AIP model is an SQL database which schema has been deduced by GDAL from an AIXM provided by Lucem transformed to a Shapefile.  This database can store AIXM transformed into shapefiles which respect the same schema.  GDAL command to import Shapefiles includes the *–splitlistfields argument.* | #8.1.12  #11.2-12 |
| SD-ALB-AIM-020 | EF layer | An entity framework model is built upon this model using the “database first” design. | #8.1.12 |
| SD-ALB-AIM-025 | AIXM Generatrion compatibility | The model and the consolidation service are compatible with the AIXM files generated by the AIXM Generation function of Albatross. | #17.2-10  #19.2.0.2-01  #19.2.0.2-07 |

##### ETOD model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-ETM-001 | ETOD Model name | Fully qualified name of the data source: **ConversionEngine.Product.EtodDataSource** | #8.1-13 |
| SD-ALB-ETM-005 | ETOD Model diagram | see TOD.png | #8.1-13 |
| SD-ALB-ETM-010 | ETOD Model | ETOD model is an SQL database which schema has been deduced by GDAL from a Shapefile provided by Lucem.  This database can store shapefiles which respect the same schema. | #8.1-13 |
| SD-ALB-ETM-020 | EF layer | An entity framework model is built upon this model using the “database first” design. | #8.1-13 |

##### AODB Model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-AOM-001 | AODB Model name | Fully qualified name of the data source: **ConversionEngine.Product.EodbDataSource** | #10.2-01 |
| SD-ALB-AOM-005 | AODB Model diagram | see AODB.jpg | #10.2-01 |
| SD-ALB-AOM-040 | Parser | A parser allows parsing an AODB file. | #10.2-01 |

##### A424-20 Model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-A4M-005 | A424 Model diagram | see A424.jpg | #11.1-01  #31.2-01  #31.2-02  #32.1-01  #33.2-05  #33.2-07  #34.2-01  #19.1.0.1-01  #19.1.0.1-08  #CM-162  #CM-158 |
| SD-ALB-A4M-010 | A424 Model | Airport Approach has a string column for the Primary Flag used as a filter. | #CM-114 |
| SD-ALB-A4M-040 | A424 Parser | A parser allows parsing an A424-20 file for the content described above including tailored and continuation record. | #11.1-01  #31.2-01  #31.2-02  #32.1-01  #33.2-05  #33.2-07  #34.2-01  #19.1.0.1-01  #19.1.0.1-08 |
| SD-ALB-A4M-045 | A424 P2 record | The model contains a P2 record as described in the document “references/Thalès A424 requirements.pdf”.  The P2 value is associated to ICAO, AIRAC and version for discrimination. | #16.1-17 |
| SD-ALB-A4M-050 | A424 Model Datasource | Two datasources components for this model exists:   * Manage by Icao * Manage without Icao   The datasources are compatible with:   * Consolidation * Conversion | #32.2-01 |
| SD-ALB-A4M-060 | A424-20 Enroute Waypoint flight planning continuation record | The Enroute Waypoint flight planning continuation record (Cont P) shall contain an attribute Free Route Airspace Codes.  It is a 6 characters string that can be extracted from character 44 to 49 in the gearshift.ari database. | CM-162 |
| SD-ALB-A4M-070 | A424-20 Airport SBAS Path Point continuation record | The SBAS Path Point continuation record (Cont A) shall contain an attribute SBAS final approach course.  It is a 4 characters string that can be extracted from character 62 to 65 in the gearshift.ari database. | CM-158 |

##### A424-17 Model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-A47-010 | A424-17 Model | SQL and C# model exist.  See A424-17.jpg | #CM-146  #32.2-01  #33.1-02  #33.2-05  #33.2-07  #34.2-01  #36.2-14  #19.1.0.1-01 |
| SD-ALB-A47-020 | A424-17 Parser | A parser allows parsing an A424-17 file for the content described above including tailored and continuation record.  Create a runway continuation record cont A to generate GE database correctly. | #CM-113  #CM-146  #32.2-01  #33.1-02  #33.2-05  #33.2-07  #34.2-01  #36.2-14 |
| SD-ALB-A47-030 | A424-17 Model Datasource | Two datasources components for this model exists:   * Manage by Icao * Manage without Icao   The datasources are compatible with:   * Consolidation * Conversion | #32.2-01 |

##### A424-15 Model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-A45-010 | A424-15 Model | SQL and C# model exist.  See A424-15.jpg | #CM-147 |
| SD-ALB-A45-020 | A424-15 Parser | A parser allows parsing an A424-15 file for the content described above including tailored and continuation record.  Need to generate MSA records in -15 A424 specification to provide Honeywell database following the requirements. | #CM-147 |

##### RDB A424

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-R4M-005 | RDB A424 Model | The RDB A424 model is an A424 model with a scope reduced to:   * Airport * Runway * Localizer\_GlideScope * Navaid\_VHF * P2 | #16.1-18 |
| SD-ALB-R4M-010 | RDB A424 Model Datasource | A datasource for this model exists and is compatible with Qualification and Conversion services | #16.1-18 |

##### A816 Model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-A8M-005 | A816 Model diagram | see A816.png | #13.2-08 |
| SD-ALB-A8M-040 | A816 Parser | A parser allows parsing an A816 file for the content described above.  Secondary | #13.2-08 |
| SD-ALB-A8M-041 | A816 Source | A data field allows to retrieve the source of the A816, RDB or AMDB. | #14.1-10 |
| SD-ALB-A8M-042 | A816 extensions for JEPPESSEN | Add “status” attribute to TaxiwayElement as defined in A816-2 in A816-0 model.  Allow value 15 to FeatBaseType.  Allow value 100 for PntstTypType. | #14.2-10 |
| SD-ALB-A8M-043 | A816 versions | Two models are implemented:   * A816-0 * A816-2 | #21.1-04 |

##### ADB LUCEM data model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-ADBLUCEM-001 | DPS – ADB Lucem Database | ADB Lucem is a datamodel containing same features and attributes as AMGEO minus:   * *aerodromesurfacelighting,* * *arrestingsystemlocation,* * *asrnedge,* * *asrnnode,* * *blastpad,* * *runwaycenterlinepoint,* * *surveycontrolpoint*   Plus:   * *airport area* * *ROPSLandingSpecificLength for runwaythreshold* | #16.1-05  #28.2-05  #19.2.0.2-02  #19.2.0.2-12 |
| SD-ALB-ADBLUCEM-002 | DPS – ADB Lucem C# | C# model exists for in memory manipulations | #16.1-05 |
| SD-ALB-ADBLUCEM-003 | DPS – ADB Lucem Datasource | The datasource component for this model exists. | #16.1-05 |
| SD-ALB-ADBLUCEM-004 | DPS – ADB Lucem usage | The datasource is compatible with:   * Qualification * Conversion * Data server | #16.1-05 |
| SD-ALB-ADBLUCEM-005 | DPS – Layer attribute | Each object of the ADB model contains a “Layer” attribute.  The attribute contains one of the following the values:   * Null * RDB * AMDB | #16.2-04 |
| SD-ALB-ADBLUCEM-010 | DPS – checkStatus attribute | AirportDatabase.checjkStatus from Opale model is not published to ADB Lucem model. | #17.1-04 |
| SD-ALB-ADBLUCEM-011 | DPS – RunwayThreshold properties | RunwayThreshold properties contains:   * **availPavedSurfFromThr**: real (meters) [0.00, 9999.99] * **measuredLda**: real (meters) [0.00, 9999.99] * **elevation**: integer (meters)   Properties are handled by:   * publish * merge * qualification * conversion | #18.1-01 |

##### Minima data model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-MID-001 | Minima Model diagram | see Minima.jpg | #35.1-06  #36.1-04 |
| SD-ALB-MID-010 | Minima Model C# | C# model exists for memory manipulations with Entity framework | #35.1-06 |
| SD-ALB-MID-020 | Minima Model Datasource | The datasource component for this model exists. | #35.1-06 |
| SD-ALB-MID-030 | Minima Model usage | The datasource is compatible with:   * Qualification * Conversionpu * Data server | #35.1-06 |
| SD-ALB-MID-040 | Minima Model Json compatibility | The model and the consolidation service are compatible with the Json files generated by the computation in the Minima Capture Tool. | #35.1-06 |

##### Airport Runway data model

Gearshift provides files runwaydata.csv and airportdata.csv. They are Comma separated values files containing airport and runway data, including some data not captured by 424.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-ARW-001 | Airport Runway model diagram | see AirportRunway.jpg | #35.2-10 |
| SD-ALB-ARW-010 | Airport Runway model  C# | C# model exists for memory manipulations with Entity framework | #35.2-10 |
| SD-ALB-ARW-020 | Airport Runway model  Datasource | The datasource component for this model exists. | #35.2-10 |
| SD-ALB-ARW-030 | Airport Runway model  usage | The datasource is compatible with:   * Qualification * Conversion * Data server | #35.2-10 |

##### ADI data model

Gearshift provides ADI textfile.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-ERD-001 | ADI diagram | see ADI datamodel.jpg | #MD-126 |
| SD-ALB-ERD-010 | ADI Model C# | C# model exists for memory manipulations with Entity framework | # MD-126 |
| SD-ALB-ERD-020 | ADI Model Datasource | The datasource component for this model exists. | #MD-126 |
| SD-ALB-ERD-030 | ADI Model usage | The datasource is compatible with:   * Qualification * Conversion * Data server | #MD-126 |

#### Model organized without ICAO

|  |  |  |  |
| --- | --- | --- | --- |
| SD-ALB-GMR-100 | Additional columns | In addition to columns specific to a model, each table should add these columns:   * Airac * Version   They are used to uniquely identify an element of a model.  They are populated by the load services which take these values in parameters or extract them from the model.  These columns should never be null. | #28.1-01 |

##### Grid MORA+ Model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-MORA-001 | Grid Mora + Database | See GridMora+.JPG | #28.1-01 |
| SD-ALB-MORA-002 | Grid Mora + C# | C# model exists for memory manipulations | #28.1-01 |
| SD-ALB-MORA-003 | Grid Mora + Datasource | The datasource component for this model exists. | #28.1-01 |
| SD-ALB-MORA-004 | Grid Mora + usage | The datasource is compatible with:   * Consolidation * Conversion | #28.1-01 |

##### CompanyRoute Model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-COR-001 | CompanyRoute Database | See CompanyRoute.jpg | #32.1-03  #33.2-04  #35.2-06 |
| SD-ALB-COR-002 | CompanyRoute C# | C# model exists for memory manipulations | #32.1-03  #33.2-04  #35.2-06 |
| SD-ALB-COR-003 | CompanyRoute Datasource | The datasource component for this model exists. | #32.1-03  #32.1-04  #35.2-06 |
| SD-ALB-COR-004 | CompanyRoute usage | The datasource is compatible with:   * Consolidation * Conversion | #32.1-03  #32.1-04  #35.2-06 |

##### EnRoute Model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-ENR-001 | EnRoute Database | See enroute datamodel.JPG | # MD-125 |
| SD-ALB-ENR-002 | EnRoute C# | C# model exists for memory manipulations | # MD-125 |
| SD-ALB-ENR-003 | EnRoute Datasource | The datasource component for this model exist for icao and non\_icao | # MD-125 |
| SD-ALB-ENR-004 | EnRoute usage | The datasource is compatible with:   * Consolidation * Conversion | # MD-125 |

##### NaturalEarth Model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-EAR-001 | NaturalEarth Database | See naturalEarth datamodel.JPG | # MD-400 |
| SD-ALB-EAR-002 | NaturalEarth C# | C# model exists for memory manipulations. | # MD-400 |
| SD-ALB-EAR-003 | NaturalEarth Datasource | The datasource component for this model exists. | # MD-400 |
| SD-ALB-EAR-004 | NaturalEarth usage | The datasource is compatible with:   * Consolidation * Conversion | # MD-400 |

#### Model organized by country

|  |  |  |  |
| --- | --- | --- | --- |
| SD-ALB-GMR-200 | Additional columns | In addition to columns specific to a model, each table should add these columns:   * Country Code * Airac * Version   They are used to uniquely identify an element of a model.  They are populated by the load services which take these values in parameters or extract them from the model.  These columns should never be null. | #29.1-01 |

##### Obstacles Model

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-OBST-001 | Obstacle Database | See Obstacles.JPG | #29.1-01 |
| SD-ALB-OBST-002 | Obstacle C# | C# model exists for memory manipulations | #29.1-01 |
| SD-ALB-OBST-003 | Obstacle Datasource | The datasource component for this model exists. | #29.1-01 |
| SD-ALB-OBST-004 | Obstacle usage | The datasource is compatible with:   * Consolidation * Conversion | #29.1-01 |

### Consolidation Map

The consolidation map gives the link between a feature and the associated data model used for the data loading input file parsing and the DB persistence.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-CSM-010 | XSD | XML product files respect an XSD schema.  See ConsolidationMap.xsd | #32.2-02 |
| SD-ALB-CSM-020 | Name | The name attribute in ConsolidationMap element defines the name of a consolidation map. | #32.2-02 |
| SD-ALB-CSM-030 | defaultDataModel | The defaultDataModel attribute in ConsolidationMap element defines the default model used by the data loading. | #32.2-02 |
| SD-ALB-CSM-040 | Entity | Entities managed in the data loading are defined with:   * name : the element unique functional name * class : the associated class in the data model * model : the associated data model. If the model is not given the default model will be use for the entity. | #32.2-02 |
| SD-ALB-CSM-050 | Parameters | Additional parameters managed in consolidation engine:   * pageSize: number of features parsed by loop * persistanceSize: number of features persited in DB by loop | #35.2-08 |

#### Consolidation map sample

See GearshiftA424.xml for mapping definition between gearshift output file and A424 data models.

## Data conversion

### Conversion engine

#### Overview

source1

Conversion engine

Product map

output

source2

client

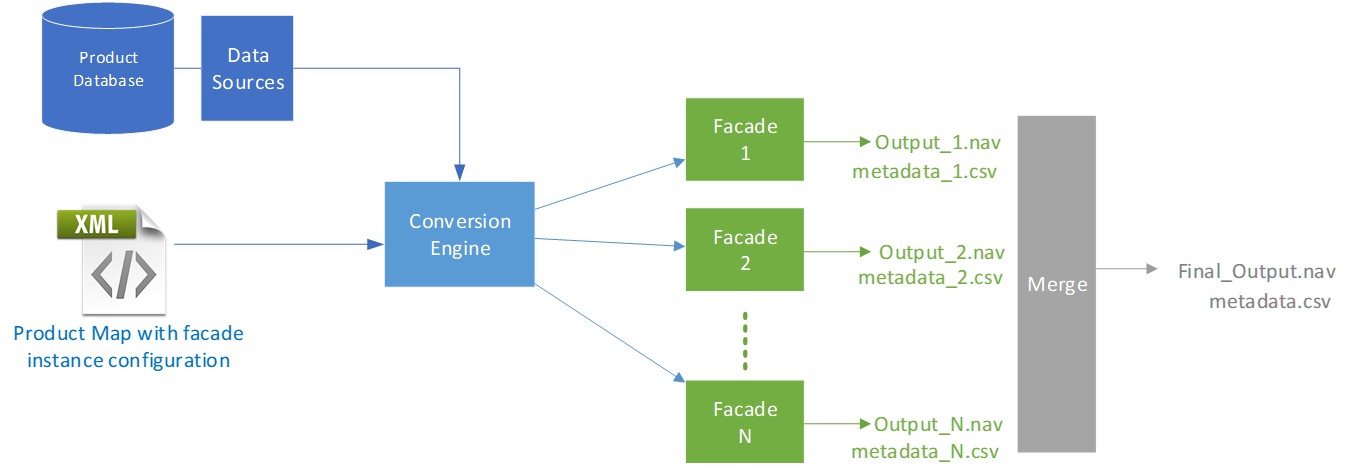
The schema above shows how the conversion engine is working:

* Source1 and source2 are the input data which are converted
* Output is the file produced by the conversion engine
* Product map is the configuration file of the conversion. It defines:
  + Sources
  + Output format
  + Mapping between sources fields and output fields

The conversion works as this:

* It analyses the product map and deduces from it
  + Sources
  + Output format
* Creates one or several façade instance for the output
  + Initialize the façade
  + It loads data from sources
  + For each field in the product map it extracts data from sources and adds data to the output through the right façade. Note that when a collection is used in facade, the use of a special field creates a new element and the facade keeps this newly created element in a state and any new access to elements of this collection will be done on the last created element.
  + Finalize the façade
  + Generates output with façade
* Merge output data base to a single data base

To reduce memory usage during the conversion process. The conversion engine is able to start several instances of the destination façade. Each façade instance generates output files that are finally merged to a single file.



#### XML Format version 1

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-CVE-410 | XSD | XML product files respect an XSD schema.  See ProductMap.xsd | #11.1-08 |
| SD-ALB-CVE-411 | Name | The name attributIoductMap elementIdes a way to give a name to a product map. | #11.1-08 |
| SD-ALB-CVE-412 | Facade | The façade element defines the facade to use for the conversion. It contains the qualified name of the class. | #11.1-08 |
| SD-ALB-CVE-413 | Initialize | The method of the facade to initialize the facade. | #11.1-08 |
| SD-ALB-CVE-414 | Mapping | The top element which define a mapping between a source and the facade. A product map can contain several mappings. | #11.1-08 |
| SD-ALB-CVE-415 | Datasource | The data source of a mapping. It is a fully qualified name of class which provides data and implements the datasource interface. | #11.1-08 |
| SD-ALB-CVE-416 | FeatureMapping | An element which contains all features mappings for a feature.  **Source** : the layer of the datasource  **Destination** : an element of the model created by the façade  **updateRule** : indicates if the element must be created and/or updated by the façade with following rules:   |  |  | | --- | --- | | **updateRule** | **Mapping behavior** | | Default | Create new feature and update it.  *Default behavior* | | UpdateOnly | Only update an existing feature. | | NeverUpdate | Create a new feature but never update an existing one. |   If parameter is not present in product map or if the value is different than enumerate values then the updateRule value shall be default.  **facadeInstance** : allow the management on a set of feature mapping in a dedicated façade instance. It is an optional attribute, when not present it shall be considered as equal 0 for the associated feature mapping.  When feature mapping instantiation is activated the conversion engine performs following actions for each façade instance declared in product map:   * Create a new façade instance * Call Initialize method of current façade * For each Icao in the conversion (including $NE): * Apply all mappings for all Data Sources but only for Feature Mappings that have a façade instance number equal to current façade instance. * Call Finalize method of current façade * Call Output method of current façade * Destroy façade instance   **updateWithOrigination** : indicates if the source element can be originated. If parameter is not present in product map or if the value is different than ‘true’ then the originations are not checked. | #11.1-08  #35.2-09  #36.1-10 |
| SD-ALB-CVE-417 | FeatureID | Allows to define the identity of a feature.  The feature Id can be the combination of several attributes merge with a separator at conversion engine level. | #11.1-08  #33.1-03 |
| SD-ALB-CVE-418 | propertyMapping | Allows to map the property of a feature to a property of the element of the façade | #11.1-08 |
| SD-ALB-CVE-419 | Transform | Apply a modification to the value extracted from the source before sending it to the façade.  Name : name of the transformation  Parameters : a comma separated list of parameters for the transformation  Default : the default value to be used if the transformation fails  {0} can be used in parameters and default value. This chain is reply the input value of the transformation. | #11.1-08 |
| SD-ALB-CVE-424 | Special attributes | Special attributes:  “\*” indicates that the corresponding attribute in the source is mandatory but is not mapped to any parameter in the facade. | #11.1-08 |
| SD-ALB-CVE-425 | Finalize | Finalize to define the name of the finalization method of the facade | #11.1-08 |
| SD-ALB-CVE-426 | Output | Allows to define the name of the output method of the facade | #11.1-08 |
| SD-ALB-CVE-427 | Parameters ordering | In a layer mapping the order of attributes and parameters are important. There is the same number of attributes and parameters in a layer mapping. The nth attribute is mapped to the nthparameter. | #11.1-08 |
| SD-ALB-CVE-430 | Merge | Allows to define the name of the merge method of the façade. | #36.1-10 |
| SD-ALB-CVE-450 | LinkTo | The LinkTo feature allows to retrieve linked element from a feature by navigation.  <LinkTo feature=”F” with=”ID” using=”ID\_REF”>  </LinkTo>  “F” is the feature to be reached by navigation.  “ID” if the field name of “F” id.  “ID\_REF” is the field name referencing F id in the current feature  A transform element inside <LinkTo> is applied to the Id of the current feature.  Several LinkTo elements can be used sequentially to navigate through several links.  When LinkTo is applied in a propertyMapping, the <Input> element applies to the last reached feature.  When LinkTo is applied in a Filter, the filter applies on the last linked element. The filter works on the data source elements by keeping records only if a linked record is found for a given filter.  If ID is null, the LinkTo feature returns null | #11.2-03  #25.1-03  #35.2-02 |
| SD-ALB-CVE-460 | Filter | The “filter” element is used to filter features inside a FeatureMapping element.  <Filter field=”F” type =”T” value=”X” />  The filter selects only features for which the filter is true. Available comparison operators are:   * Equal * Not Equal * Lower Than * Lower Than Or Equal * Greater Than * Greater Than Or Equal * Like * Not Like * In * Not In * Geo   The filter allows filtering on attributes with different types than String and filtering on NULL value in database.  Like and Not Like allows to compare string properties with a pattern that can contain wildcards (similar to SQL LIKE). These wildcards can only be positioned at one extremity of the string. % 🡪 0 or any number of random characters \_ 🡪 one random character  Possible patterns:  %AA / AA% / %AA% / \_AA / AA\_ / \_AA\_ / %AA\_ / \_AA%  In and Not In filters value field contains the possible values list separated by a comma (“,”).  Geo filter works on the data source elements by filtering records contained in a geographical area. The value field contains a geographical area defined as WKT polygon. | #11.2-03  #15.1-09  #23.1-04  #25.1-05  #33.2-08  #35.2-01 |
| SD-ALB-CVE-470 | OrderBy | The “order” element allows ordering the features from the datasource in a “featuremapping” element.  <OrderBy field=”F” ascending=”true/false”/>  The features extracted from the datasource will be ordered following their field “F” in the order defined by “ascending”.  It is possible to define multiple “OrderBy” items in a feature mapping. In this case all consecutive “order by” clauses are combined and applied in the same order to the query done on the data source to retrieve the mapping elements. | #11.2-10  #33.2-10 |
| SD-ALB-CVE-480 | Parameters | Allows to map additional parameters used by the façade.  Presence of requested parameters is checked in the façade. | #28.2-07 |
| SD-ALB-CVE-481 | Parameter output file name | The parameters ‘outputFileName’ allows the façade constructing the filename based on a pattern. (e.g. “{icao}\_{airac}{version:1}\_rdb.bml”) | #28.2-07 |
| SD-ALB-CVE-482 | Parameter Empty Data Source | The parameter ‘ignoreEmptyDataSource’ set to value ‘true’ allows the conversion engine to continue the conversion of an Icao even if entries do not exist for each DataSource in the product map.  When parameter ‘ignoreEmptyDataSource’ is not defined in product map or its values is different than ‘true’ the conversion is performed only for the Icaos having entries for all Data Sources in the product map. | #32.2-03 |
| SD-ALB-CVE-483 | Parameter  facadeIcaoInstantiation | The parameter ‘facadeIcaoInstantiation’ set to value ‘true’ allows the conversion engine to create a new façade instance for each Icao requested in tejh conversion. If parameter is not present in product map the default value shall be false (do not create a new façade instance by icao).  When icao instantiation option is activated the conversion engine performs following actions for each icao in the conversion (including $NE):   * Create a new façade instance * Call Initialize method of current façade * Apply all mappings : for all Data Sources and all Feature Mappings * Call Finalize method of current façade * Call Output method of current façade * Destroy façade instance   When a facade multi instance is activated, the instance value and/or the icao are passed to the façade to be used as Prefix to name the output file. e.g: {icao}{instanceNumber}\_{outputfilename} | #36.1-10 |
| SD-ALB-CVE-484 | Parameter  convertOnlyRequestedAirac | The parameter ‘convertOnlyRequestedAirac’ set to ‘true’ allows the conversion only to take instances with only the requested airac, instead of the usuall behavior that takes the highest airac lower than the requested one that’s available.  I.e. if the instance doesn’t exist with the requested airac, it won’t be corrected. | #CM -177 |
| SD-ALB-CVE-490 | Condition | The “Condition” element is used to define conditional application of a filter or a transformation.  <Condition field=”F” type =”T” value=”X” />  The condition is described with same fields and behavior than the existing filter section (field, value, type), see SD-ALB-CVE-460.  Several Condition elements can be used sequentially, in this case a ‘AND’ logic is applied between the conditions. | #35.2-03  #35.2-04 |
| SD-ALB-CVE-495 | Header | The “Header” element is used to define a header for the output database.  <Header>  <HeaderMapping destination="H" >  <ParameterMapping>  <Transform name="Static" parameters="X" />  <Output>O</Output>  </ParameterMapping>  ….  </HeaderMapping>  </Header> | 19.0.0.1-11 |

#### Facades

A façade is a class which can produce an output (A816, A424, AIXM …). An instance of this class is used by the conversion engine for each conversion.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-CVE-100 | Initialize method | An initialize method prepares the underlying model used to store data provided by the conversion engine.  It has no parameters.  Return: void | #8.1-18 |
| SD-ALB-CVE-101 | Finalize method | A finalize method is called by the conversion engine before the output. This method should do whatever has not been done during mapping processing in order to produce a valid output.  It has no parameters.  Return: void | #8.1-18 |
| SD-ALB-CVE-102 | Output method | The method which produces the output.  Parameters:   * String : icao * String airac cycle   Return :   * Type : string * Value : the folder where the output has been generated | #8.1-18 |
| ~~X SD-ALB-CVE-103~~ | ~~Parameter method V0~~ | ~~A parameter method must have the following parameters:~~   * ~~Inputs~~   + ~~Type: List<DynamicClass>~~   + ~~Value: the list of entities provided by the conversion engine from the datasource.~~ * ~~Source~~   + ~~Type string[]~~   + ~~Value: all attributes names for this layer~~ * ~~Destination~~   + ~~Type; string[]~~   + ~~Value: all parameters names for this layer~~ * ~~Layer~~   + ~~Type: string~~   + ~~Value: the name of this layer~~ | ~~#8.1-18~~ |
| SD-ALB-CVE-104 | Parameter method V1 | For each entity XYZ managed by the façade, the façade must define 3 methods:   * Get<XYZ> : returns an entity by its ID * Create<XYZ> : creates an entity * Update<XYZ> : update an existing entity | #11.1-08 |
| SD-ALB-CVE-105 | Merge method | The method that merges the outputs of the façade instances to a single output.  Parameters:   * String : airac cycle * String : output path   When merge tag is present, the conversion engine executes the associated method after all façade instances conversion process.  A new façade instance is created to execute the merge method. | #36.1-10 |
| SD-ALB-CVE-110 | Filter on LinkTo | LinkTo (see SD-ALB-CVE-450) supports filtering like this:  <LinkTo feature=”RunwayDiredtion” with=”identifier” using=”runwayDirection\_href”>  <Filter feld=”magneticBearing” value=”0” type=”NOT\_EQUAL”/>  </LinkTo>  LinkTo result is filtered by the enclosed Filter element. | #22.1-02 |
| SD-ALB-CVE-120 | Header Method | A header method is called by the conversion engine after the merge method. This method produce a header for the output database.  It has no parameters.  Return: void | 19.0.0.1-11 |

#### Sources

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-CVE-200 | Source class  Icao management | A source with Icao management is a class which provides entities for a given tuple:   * Icao * Airac   It identifies the last version import for the ICAO and get all entities for:   * ICAO * Airac * Last Version   It is referenced in ProductMap DataSource element. | #8.1-18 |
| SD-ALB-CVE-201 | Source class  Non Icao management | A source with non Icao management is a class which provides entities for a given Airac.  It identifies the last version import for the Airac and get all entities with unique geometry for:   * Airac * Last Version   It is referenced in ProductMap DataSource element. | #28.1-01 |
| SD-ALB-CVE-205 | Origination | When a datasource is used in a conversion, the result of the application of the origination is returned to the conversion engine if updateWithOrigination attribute is true in the associated feature mapping.  The application of an origination is logged during the conversion.  If the feature is not found in associated feature map, any origination is discarded and it does not prevent the conversion. | #17.1-07  #36.1-10 |

#### Enumeration v1

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-CVE-220 | Enumerations v1 | A database contains enumerations translation. Each line defines a value for an enumeration in a format.  Columns are:   * Name\_enum : the name of the enumeration (ex : surface type) * Code : the code for the value (ex : sand) * Format : the format for the value (ex A816) * Value : the value for the format (ex : 0x14)   This structure allows retrieving a value for an enumeration, a code and a format. It also allows translating a code between various formats.  This database is used by the enumeration transformation. | #11.1-10 |

#### Transformations

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-CVE-510 | Transformation GetX | Extract the X value from a dbgeometry  Input : dbgemorty  Output : decimal  Params : none | #11.1-09 |
| SD-ALB-CVE-520 | Transformation GetY | Extract the Y value from a dbgeometry  Input : dbgemorty  Output : decimal  Params : none | #11.1-09 |
| SD-ALB-CVE-530 | Transformation Split | Split a string with a separator and return the nth element  Input : string  Output : string  Params : separator, index of the element to be returned | #11.1-09 |
| SD-ALB-CVE-540 | Transformation toLower | Lower case transformation  Input : string  Output : string  Params : none | #11.1-09 |
| SD-ALB-CVE-550 | Transformation toUpper | Upper case transformation  Input : string  Output : string  Params : none | #11.1-09 |
| SD-ALB-CVE-560 | Transformation ToString | Transform any value to a string using C# default transformation  Input : any  Output : string  Params : none | #11.1-09 |
| SD-ALB-CVE-570 | Transformation static | Returns a static value  Input : any  Output : string  Params : the value to be returned | #11.1-09 |
| SD-ALB-CVE-580 | Transformation trim | Removes trailing spaces at the beginning and the end of a string.  Input : string  Output : string  Params : none | #11.1-09 |
| SD-ALB-CVE-581 | Transformation enumeration | Translates an enumeration value from one format to another format using enumeration database v1  Input : string  Output : string  Params : the enumeration name, the input format, the output format  For the NAVDB generation, the country code can be transformed to region code.  The enumengine table is completed with associated values between country code and icao code.  See country\_region.xls file | #11.1-10  #36.2-05 |
| SD-ALB-CVE-582 | Transformation ExtractUUID | Extracts the uuid with the following regexp: ([a-f0-9]{8}-[a-f0-9]{4}-4[a-f0-9]{3}-[89aAbB][a-f0-9]{3}-[a-f0-9]{12}) from a string.  Input : string  Output : string  Params : none | #11.2-11 |
| SD-ALB-CVE-583 | Transformation GenerateNumberId | Generates a value from a counter incremented at each call of the transformation. The counter starts at 1.  At each ICAO generation request, the counter is reset to 1. | #16.1-16 |
| SD-ALB-CVE-584 | Transformation  Truncate | Truncate a string to the given number of characters  Input : string  Output : string  Params : number of characters | #ALB-544 |
| SD-ALB-CVE-585 | Transformation Math | Performs simple operations on attribute values like:   * Plus (+) * Minus (-) * Multiply by (\*) * Divide by (/)   Parameters include the operator and operand separated by a comma “,”.  Decimal operand are noted with point (“.”) decimal separator. | #19.0.0.1-01 |

### Conversion engine service

The conversion web service allows calling the conversion engine for a set of airports.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-CVE-250 | Parameters | Name: icaoCodeList  Type: List<String>  Value: the list of icao for which to produce a database.  Constraints: Required.  Name: airacCycle  Type: string  Value: the airac cycle of the airports  Constraints: Required  Name: productIdentifier  Type: string  Value: the name of the product map. This is the name of the file without the .xml suffix.  Constraints: Required  Name: outputPath  Type: string  Value: the path where to store the output of the conversion  Constraint: required | #8.1-18  #14.2.3 |
| SD-ALB-CVE-260 | Output | Type: string  Value: a string describing the process of the conversion, all methods which have been called. | #8.1-18 |
| SD-ALB-CVE-265 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #8.1-18 |
| SD-ALB-CVE-270 | Logs levels | Use log levels as this:   * INFO: cessing st–ps * DEBUG: detailed info   ERROR: errors | #8.1-18 |
| SD-ALB-CVE-280 | Name | ConversionService | #8.1-18 |
| SD-ALB-CVE-290 | Data inventory check | The Conversion Service shall check the presence of an entry in *DataInventory* table (Qualification database) for all data used in the production request (i.e. ICAO, AIRAC cycle and ProductMap data sources)  The Conversion Service shall:   * + list all data sources used in requested ProductMap   + for each of the data sources with icao management, try to find an entry in *DataInventory* for     - all ICAO airport codes requested     - the AIRAC cycle requested     - Any version (take the latest)   + for each of the data sources with non icao management, try to find an entry in *DataInventory* for     - the AIRAC cycle requested     - Any version (take the latest)   If no matching entry is found in *DataInventory* for one of the requested data, the Conversion Service shall raise an Exception (with details of the missing information) and abort the conversion process for all airports. | #14.2-14  #28.1-03 |
| SD-ALB-CVE-300 | AIRAC used | When a product is searched in the datasource, if the product doesn’t exist for the provided AIRAC, then the newer AIRAC older than the provided AIRAC is used. If it doesn’t exist then the older AIRAC is used.  If no AIRAC exist for the provided ICAO then no product is generated without error. | #18.1-07 |
| SD-ALB-CVE-310 | All Airports | When icaoCodeList is filled with value [“\*”], it activates the all airport request option.  When all airports option is activated the conversion service builds the icao list by:   * Retrieving the icao datasources in the product map and the requested airac in the conversion request * Fetching from the data inventory table content the icao values available for these data sources with an airac lower or equal than requested airac * An airport icao appears only one time in the output list | #35.2-05 |
| SD-ALB-CVE-320 | Façade introspection | The façade methods inspection by introspection from conversion engine is done by storing destination façade methods information in a dictionary. This dictionary is then used on each element loop to get façade methods. | #35.2-13 |
| SD-ALB-CVE-330 | Multi Thread | The conversion engine is able to execute in parallel the façade instances in several threads.  Concurrent access caused by Multi thread implementation are managed.  It is possible to limit the number of thread executed in parallel from a parameter in the product map.  When the parameter is not present in product map, number of thread are managed automatically by the system. | #19.0.0.1-13 |

### Convert Delivery

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-CVD-001 | Convert delivery service | ConvertDelivery function receives an Airac list and a dictionary (ProductMap, icao list).  It loops on airac list then on on productmap list to launch as many conversion as necessary.  ConvertDelivery function creates the directories architecture for the packaging from airac list and productmap list:  Delivery  Product 1  Airac n  Airac n-1  Product 2  Airac n  Airac n-1    ConvertDelivery manage following processing statuses on Customer delivery entity:   * Conversion In Progress * Conversion Done * Conversion Error | #21.2-21 |

### AIXM Façade

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-AXF-001 | Name façade name | AIXM façade - IThe fully qualified name of this façade is **Converfafaçadene.Facade.AIXMFacade** | #8.2-02  #9.2-01  #9.2-02 |
| SD-ALB-AXF-270 | AIXM Facade | The heirsow describes the AIXM facade parameters for each feature. The order of theIres and the order of parameters inside the feature is important and must respect the order fafaçadetable. | #8.2-02  #9.2-01  #9.2-02 |
| SD-ALB-AXF-300 | AIXM Packaging | The AIXM facade output produces a valid AIXM file containing all elements and attributes defined in SD-ALB-AXF-270. | #8.2-01 |
| SD-ALB-AXF-310 | AIXM Methods | Initialize method name: **Initialize**  Finalize method name: **Finalize**  Output method name: **Generate** | #8.2-01 |

#### Product map sample

Product map sample façade: see ETOI.xml

### A816 Façade

| **ID** | **Description** | | **US** |
| --- | --- | --- | --- |
| SD-ALB-A81-001 | A816 Facade | The fully qualiffafaçade of this facace is: **ConversionEngine.Facade.A816Facade** | #9.1-01 |
| SD-ALB-A81-270 | A816 Facade | See docufaçafaçadein DocumentationFacade.docx. | #9.1-01  #12.1-04 |
| SD-ALB-A81-300 | A816 Packaging | The A816-0 facade output produces a valid A816-0 file containing all elements and attributes defined in SD-ALB-A81-270. | #9.1-01 |
| SD-ALB-A81-310 | A816 Methods | Initialize method name: **Initialize**  Finalize method name: **Finalize**  Output method name: **Generate** | #9.1-01 |
| SD-ALB-A81-320 | A816 output file | The AIRAC parameter is used to fill header dates.  Airport files are named from this pattern:  [ICAO]\_[AIRAC+version]\_[type of DB].bml  with :   * [ICAO] : ICAO code of the airport in lowercase * [AIRAC + version] = 5 characters with 4 for the AIRAC Cycle and one for the version. The version in the bml file name is the same than the last published version of the airport. The published version is managed modulo 10 (i.e. if 10 then 0, 11 then 1, etc…) * [type of DB] : if the delivered airport is a RDB, it is "rdb". When the airport is delivered in AMDB, it is "amdb". | #18.1-07  #28.2-07 |
| SD-ALB-A81-321 | A816 versions | Two façades exist for two versions of A816:   * A façade A816-0 * A façade A816-2 | #20.1-05 |
| SD-ALB-A81-500 | A816 bounding box | The airport bounding box limits (minx, miny, maxx, maxy) is computed from airport area geometry relative to arp:   * Min x, Max x : from the westernmost and easternmost longitude respectively * Min y, Max y : from the southernmost and northernmost latitude respectively * X and Y values are computed as projection regarding the arp coordinates like other x/y values in A816 database.   If airport area geometry is not available, the bounding box of an airport includes a padding around the smallest bounding box including all points of all features. The padding value is 100m on each direction. | #20.2-04  #28.2-06 |
| SD-ALB-A81-510 | A816 Stand containers | Each Stand Container shall contain only one Stand Guidance Line with same 'idobject'  In case of multiple stand lines / locations on a same parking stand area, several stand containers shall be defined for each sub identifier of the stand area. | #24.2-01 |
| SD-ALB-A81-520 | A816 parking containers | Parking Container shall be created following ARINC 816-0 specifications :   * Parking container should contain all stand containers linked to a parking stand according to following rule:   A parking container should contain all, and at least two, stands that belong to a same logical parking entity. A stand container is part of a parking container if:   * + Its children share the same *terminalref* attribute as the children of all other stand containers in the parking container.   + Its anchor point is separated by less than 200 meters from another anchor point in the parking container. * The *idgen* attribute of the parking container shall provide general identification for parking container (i.e. generic label) which is the common characters (numeric or alphabetic) at the beginning of the identifiers for a group of features of the same type * When stand containers that make up a parking container do not share common characters, the *terminalref* attribute of the children should be used as generic identification | #24.2-02  #26.2-14 |
| SD-ALB-A81-530 | A816 bridge points | The bridge points elements for TaxiwayElement feature in A816 shall be created in A816 façade.  These points shall be created if the TaxiwayElement feature has attribute bridge set to Overpass(1) or Underpass(2) (i.e. different from None(0)).  See A816-0 specification §2.4.3 :  Bridge points should be provided for each taxiway element which overpasses a bridge. Bridge points should consist of two pairs of points. A set of points should represent the side of the taxiway element where it intersects the bridge. No specific priority is required for these pairs of points.    Bridge points computation is done before gathering and taxiways features with bridge attribute different from None (0) are never gathered. | #27.1-02 |
| SD-ALB-A81-540 | A816 runway threshold elevation | The A816 façade manages the ‘elev’ attribute for feature runway threshold.  The elev attribute appears in the output bml. The elev attribute is optional and not generated if not filled during the conversion. | #30.2-08 |
| SD-ALB-A81-550 | A816 Invalid Acceptable Geometries | Invalid geometries are ignored during Anchor point process in conversion A816.  Flat triangles are ignored during triangulation process in conversion A816. | #30.2-15 |
| SD-ALB-A81-560 | A816 Limit the number of Elements for RWY displaces Area and Stopway features | The maximum number of **Runway Displaced Area** features per runway is limited to 2.  The maximum number of **stopway** features per runway is limited to 2.  For some airports, the starter extension part has been captured as a runway displaced area feature.  During the **conversion** of adb lucem data source to A816 , the system identifies possible multiple runway displaced area features with same identifier and store them as a single feature with multiple geometries.  This is done during the features aggregation phase.  When aggregating the elements, if attributes are different:   * "hacc" is the highest one from each element. * "status" is the one of the closest to the runway (inner one) * "surftype" is the one of the closest to the runway (inner one)   The system does the same with stopway features, i.e. identify possible multiple stopway features with same identifier and store them as a single feature with multiple geometries. | #36.2-11 |
| SD-ALB-A81-570 | A816 Order the polygons of runway displaced are and Stopway features | In case of **Runway Displaced Area** or **stopway** polygons separated by at least 2 Runway Intersection features, the two polygons the most far from each other are provided first (i.e. the extremities of the feature)  During the **Conversion** of adb lucem data source to A816, the system identifies runway displaced area or stopway features made of more than 2 polygons and make sure that geometries are ordered in a way that the 2 polygons more distant from each other are added first. | #36.2-12 |

#### Product map sample

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-A81-600 | AMDBLUCEM2A816 Product Map | Product map sample for A816 façade: see AMDBLUCEM2A816.xml | #27.1-18  #28.2-06  #28.2-07  #30.2-02 |
| SD-ALB-A81-610 | RDBLUCEM2A816 Product Map | Product map sample for A816 façade: see RDBLUCEM2A816.xml | #25.2-07  #28.2-06  #28.2-07  #30.2-02 |
| SD-ALB-A81-620 | AMDBLUCEM2A8160+ Product Map | Product map sample for A816 façade: see AMDBLUCEM2A8160+.xml | #30.2-08  #30.2-02 |
| SD-ALB-A81-630 | RDBLUCEM2A8160+ Product Map | Product map sample for A816 façade: see RDBLUCEM2A8160+.xml | #30.2-08  #ALB-548 |

### AODB Façade

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-AOF-010 | AODB Facade | AODB Façade is a façade produc–ng–a single AODB file.  See documentation in DocumentationFacade.docx. | #10.2-03 |
| SD-ALB-AOF-020 | AODB Façade – –ame | Fully qualified name of AODB façade is : **AODBDataModel.AodbFacade** | #10.2-03 |
| SD-ALB-AOF-030 | AODB Fa–a–e - output | AODB Façade outputs a single file containing all airports. | #10.2-03 |
| SD-ALB-AOF-035 | AODB Façade - version | AOBD Façade is compatible with engine version 1. | #15.1-01 |

#### Product map sample

Product façfaçademple for this façade: see A–P2AODB.xml

### A424 Façade

It is possible to generate a product with a façade that uses different data models to instantiate objects and serialize them to a single file:



| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-A4F-010 | A424 Facade | A424 Façade is a producing a single A424 file.  See documentation in DocumentationFacade.docx. | #11.1-02 |
| SD-ALB-A4F-020 | A424 Façade – name | Fully qualified name of A424 façade is: **A424DataModel. A424Facade** | #11.1-02 |
| SD-ALB-A4F-030 | A424 Façade - output | A424 Façade outputs a single file containing all records. | #11.1-02  #32.2-03 |
| SD-ALB-A4F-040 | A424 Façade - version | A424 Façade is compatible with version 1 the conversion engine. | #11.1-02 |
| SD-ALB-A4F-050 | A424 Façade - mora | A424 Façade manages Grid Mora feature:   * Grid mora records are created in memory based on the parameters in input of the façade methods and serialized to a424 format according to arinc A424-20 specification * The façade can generate a file without header, named mora.ari * Output gridmora record is identified from the coordinates of the first point of Mora object geometry * Mora value is converted to hundreds of feet (rounded up) * Airac cycle of the record is the latest airac of all mora objects in the record. | #28.1-03 |
| SD-ALB-A4F-060 | A424 Façade –  multi-model conversion | The web service is able to generate a product with the façade that uses different data models to instantiate objects and serialize them to a single file. | #CM-147  #CM-146  #32.2-03  #33.1-03  #33.2-06 |
| SD-ALB-A4F-070 | A424 Façade –  Primary continuation conversion | The facade is able to generate an A424 product with primary and continuation records created from A424 DataSource or any others. | #32.2-03  #33.1-03  #33.2-06  #19.1.0.1-03  #19.1.0.1-08  #CM-191  #CM-193  #CM-194  #CM-195  #CM-196  #CM-199  #CM-202  #CM-200  #CM-198  #CM-201  #CM-205  #CM-222  #CM-223  #CM-242 |
| SD-ALB-A4F-075 | A424 Façade –  Continuation renumbering | If a continuation record is skipped, the number of the potential next continuation records shall be modified in order to preserve the consistency of numbering.  More generally, continuation record numbering shall be consistent in the generated product. | #CM-192  #CM-246 |
| SD-ALB-A4F-080 | A424 Façade –  Metadata report | The A424 façade generates a report containing record counts concerning the A424 output file generated. The report result file is in csv format.  The record counts to include in the report are:   * Record count by record type (without distinction between A424 models) * Record count by region (areaCode) * Record count by country (first icaoCode on each record)   Report is created in the generation method of the A424 façade after the Data base generation. Report creation uses counters that are incremented directly from the record’s create/update methods and the finalize method in the façade (no post treatment on generated data base).  Report output csv file is generated in the same directory than the A424 data base with name Metadata.CSV. | #35.2-07 |
| SD-ALB-A4F-090 | A424 Façade – generateStatReport | A424 Façade manages an optional parameter generateStatReport.  The parameter value can be true or false.  If the parameter is present with the value equal true, the metadata report is generated. | #35.2-07 |
| SD-ALB-A4F-091 | A424 Façade transformApproachHoneywell | A424 Façade manages an optional parameter transformApproachHoneywell of the product map  The parameter value can be true or false.  If no parameter the default value of the parameter is false.  If false it does not apply transformation on approach  If true it does apply the transformation as the finalize method conversion as describe in the attached exel file. | #CM-150  #CM-129 |
| SD-ALB-A4F-092 | A424 Façade transformApproachGE | A424 Façade manages an optional parameter transformApproachGE of the product map.  The parameter value can be true or false.  If no parameter the default value of the parameter is false.  If false it does not apply transformation on approach  If true it does apply the transformation as the finalize method conversion as describe in the attached exel fi. | #CM-149  #CM-129 |
| SD-ALB-A4F-093 | A424 Façade transformation Enroute Airways records | A424 Façade manages an optional parameter transformAirways of the product map.  The parameter value can be true or false.  If no parameter the default value is set to false.  If the value is false then, do not apply transformation on Enroute Airways. If the value is true apply the transformation and the finalize method conversion rule as describe in the attached exel file. | #CM-148 |
| SD-ALB-A4F-094 | A424 Façade transformation Runway records | Special transformation for the customer Thales TAWS  A424 Façade manage a parameter which is false by default.  If the parameter is false, do not apply the transformation.  If the parameter is set to true then transform all records PG present. In the finalize method.  A424 Façade manages a parameter which is false by default and true if the customer is GE or Honeywell. If this parameter is false, columns Runway Accuracy Compliance Flag and Landing Threshold Elevation Accuracy Compliance Flag are empty. If this parameter is set to true, in the finalize method there is a comparison with ADBLucem database. According the result of the comparison column Runway Accuracy Compliance Flag and Landing Threshold Elevation Accuracy Compliance Flag are filled  Comparison rules are described in the following file : “RW Accuracy Flags - WAPP comparison requirements.docx” | #CM-65  #CM-113 |
| SD-ALB-A4F-095 | A424 façade transformation missing route type Z | A424 Façade manages an optional parameter transformAirways of the product map.  The parameter value can be true or false.  If no parameter, the default value is set to false.  If the value is true apply the transformation Missed approach route type Z | #CM-172 |
| SD-ALB-A4F-100 | A424 Façade – generateHeader | A424 Façade manages an optional parameter generateHeader.  The parameter value can be true or false.  If the parameter is present with the value equal true, the A424 header is generated and added in output database. | #35.2-12 |
| SD-ALB-A4F-110 | A424 Façade – merge | This method analyzes the output directory content and merge all A424 files to a single file and all metadata.csv files to a single file.  For A424 files :   * Concatenate all lines from all file in a single file * Order all lines in the final file alphabetically * Increment File Record Number on each line (character 124 to 128) * Generate Header if option activated in product map * Name the result file as defined in the Product Map (parameter ‘outputFileName’)   For metadata files:   * Sum the record count for each record type, region and country to a single file * Name the result file metadata.csv | #36.1-10 |
| SD-ALB-A4F-120 | A424 Façade – Convert markers to terminal waypoints | For NAVDB generation markers record (PM) can be converted to terminal waypoints record (PC) upon option activation.  This option is activated via a parameter “convertMarkerToWaypoint”.  If parameter is not present in product map the default value in facade is false (do not convert the markers).  When option is activated all record PM present in the A424 façade are converted in record PC following conversion rule described in file: conversion\_rules.png  The conversion is done in the finalize method of A424 façade | #36.2-04 |
| SD-ALB-A4F-129 | A424 Façade – GE customizations | A424 Façade allows to generate runway continuation type A records based on an input csv file.  Route type R should be changed to H where value of column 89 of approach continuation record (application type W) is **A**. | #CM-146  #CM-245 |
| SD-ALB-A4F-130 | A424 Façade – Headers | The A424 facade handles multiple possible header options. Some of these options use A424-15 header objects, some A424-17 header objects, and one is full custom object.  In the output file each line of header contains 132 column  All options are described bellow:  Old WAPP Option 3  Option 3 is identical to option 17 (standard -17 see below). The only difference is that the file extension does not form part of the file name in columns 6-20. The CRC is calculated as per -17  EffectiveDate and ExpirationDate are filled from Tools Database if there are not value in the mapping  The supplierTextField value is NAVBLUE  OLD WAPP Option 4  Option 4 is based on -15 (Standard -15 below). Differences below  EOF and EOV lines are as per -15 but are located directly below the header lines and not at the bottom of the file  Old WAPP Option 5  Option 5 is based on -15 (Standard -15 below). Differences below  Header Supplied in separate text file (headSA)  EOF and EOV lines supplied in separate file (tailSA) and include a repeat of standard -15 header line 2  Col 51-78 are blank  CMC Option 6  Option 6 is based on -15 (Standard -15 below). Differences are that  EOF is at the beginning just after the 2 line of Header and there is no EOV lines  OLD WAPP Option 7  1 line custom header  Col 1-27 Database supplier name  Col 28 Blank  Col 29 – 52 Day, date and time and year of generation e.g. Fri Oct 19 22:19:49 2018  Col 53 Blank  Col 54-57 – Cycle date  Col 58-132 – Blank  OLD WAPP Option 15  The ownersIdentifier value is NAVBLUE  Standard -15 : This follows the specified ARINC 424 exactly and is self-descriptive apart from the following.  Line 1  Col 4-9- Volume Serial Number – Set to 000001 in all examples but we may need facility not define this manually.  Col 37-50 Owner’s ident – Must be defined manually  Line 2  Col 21-26 – Manually defined by us (could be defaulted to be derived from database name and truncated from the right accordingly)  Line 3 As standard  EOF and EOV lines – At the bottom of the file – exactly as standard -15  OLD WAPP Option 17  The supplierTextField value is NAVBLUE  Line 1 Standard -17 This follows the specified ARINC 424 exactly and is self-descriptive apart from the following.  Col 62-77- Data Supplier ident – Currently EAG but we will need the ability to define this manually when required.  Col 78-93 – Target Customer – This is not always populated for this option, so this must be optional and defined by us manually.  Col 94-113 – Database Part number – This is not populated in any current examples and is defined as optional. We may want to retain the ability to populate this if necessary.  Line 2 Standard -17  Col 29-58 and 59-88 are optional free text and should be definable by us but none are currently populated.  See ARINC 424-17 6.10.1 and 6.10.2 for CRC calculation. | #19.0.0.1-11  #CM-221  #CM-220  #CM-231  #CM-239  #CM-241 |
| SD-ALB-A4F-140 | A424 Façade – Clean and merge A424 | The A424 façade contains a method to enforce the integrity on excluded records.  First this method executes a nominal merge to have a single A424 file and a single metadata.csv (if needed) in the output directory. (see SD-ALB-A4F-110)  Then the method splits the A424 file by record type. A file is created per record type.  Next the method cleans the records in the split files. To do so, the method is able to load an XML file which is the integrity configuration file :  <IntegrityConfig name="IntegrityConfigFile" version="1">  <Integrity parentRecord="A424DataModel.Heliport">  <LinkedRecord name="A424\_17DataModel.HeliportSid">  <LinkMapping>  <Parent>HeliportIdentifier</Parent>  <Linked>HeliportIdentifier</Linked>  </LinkMapping>  </LinkedRecord>  <LinkedRecord name="A424DataModel.HeliportApproach">  <LinkMapping>  <Parent>HeliportIdentifier</Parent>  <Linked>HeliportIdentifier</Linked>  </LinkMapping>  <LinkMapping>  <Parent>TransitionAltitude</Parent>  <Linked>TransitionAltitude</Linked>  </LinkMapping>  </LinkedRecord>  </Integrity>  </IntegrityConfig>  To enable the access to this XML file by the method, the A424 Façade manages an optional parameter IntegrityPath. The parameter value contains the path to the integrity configuration file.  This XML file describes links between records that have to be respected. The function is able to read the links defined in the XML file and to apply them on the corresponding records of the split files :   * If a link is respected by a record, the record is kept. * If a link is not respected by a record, the record is removed from the split file. If the generate state report option is activated, the metadata file is updated (corresponding lines have to be decreased)   The rules must be applied to the primary records as well as for the continuation records.  Once all the links of the XML file have been applied on the corresponding records, records have been cleaned. The function merges all split files into one single A424 file. | #19.0.0.1-12  #CM-247 |
| SD-ALB-A4F-150 | A424 Façade – Cycle data | The A424 façade maps the cycle date attribute from each A424 data source object to the cycle date attributes of the A424 entries. | #19.0.0.2-13 |

Product map sample for this façade: see A424TOA424.xml.

#### RDB A424 Façade

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-R4F-010 | RDB A424 Facade | RDB A424 Façade is a façade producing a single A424 file.  See documentation in DocumentationFacade.docx. | #16.2-01 |
| SD-ALB-R4F-020 | RDB A424 Façade–- –ame | Fully qualified name of A424 façade is: **RDBA424DataModel. RDBA424Facade** | #16.2-01 |
| SD-ALB-R4F-030 | RDB A424 Façade - output | RDB A424 Façade outputs a single file containing all airports. | #16.2-01 |
| SD-ALB-R4F-040 | RDB A424 Façade – P2 Record | For P2 calculation, one façade method is called:   * First in “create” mode for all AIP attributes and fields * Then in “update only” mode for geomatic values (not directly stored in P2 fields but used for validation) * In Finalize method, if the “runway validation status” field of the P2 element is set to 0, then proceed to the comparison between geomatic values and AIP values and set the result in the various fields as described in the RDB A424 PDF P2 record specification provided in previous run. * For legacy conversion, the “runway validation status” field shall be set to 1 or 2 depending on the validation outcome on the RDB A424 used as source.   When computing P2 record from geomatics data, the service takes into account the NotValidated feature of the qualification model. If the feature is a runway direction with “not validated” set to “true” then the P2 record is set to “not validated. This means P2 is set to:   * + Runway valid status = 0   + Runway THR position error = 0   + Runway THR elevation error = 0   + LDA valid status = 0   + Runway width error = 0   + Runway bearing error = 0   If the P2 record is linked to a closed runway direction (AIP operationalStatus = CLOSED), the P2 record is set to “not validated” and the same values are assigned. | #16.2-01  #19.2-04  #20.1-03  #24.2-08 |
| SD-ALB-R4F-050 | RDB A424 Façade - AIRAC | The AIRAC of the header is the AIRAC passed as a parameter to the conversion service.  The AIRAC of all other lines is the AIRAC of the retrieved product. | #18.1-07 |
| SD-ALB-R4F-055 | RDB A424 – P2 LDA | LDA validation status is set to true if *landingDistanceAvailable <= geomaticLda + 5 meters* | #20.2-14  #19.2.0.2-14  #DLT-72 |
| SD-ALB-R4F-056 | RDB A424 – P2 imagerydata | The imagery date for all P2 records is filled from input reference data. If the imagery date is not specified in reference data, the A424 conversion fills the imagery date for all P2 records if an *imagerydata* record is found for this airport. | #22.1-05 |
| SD-ALB-R4F-060 | RDB A424 – TCH | If Threshold Crossing Height value is not filled from the ProductMap feature mapping the   * If associated Runway length is greater or equal than 6000 FT, set TCH value to 50 FT and TCH indicator to “D” * If associated Runway length is lower than 6000 FT, set TCH value to 40 FT and TCH indicator to “D” | #21.2-06 |
| SD-ALB-R4F-061 | RDB A424 – Displacement | If the displacedThresholdDistance (i.e. displacement) is not filled from AIP mapping nor from ADBLucem maping (runwaydisplacedarea):   * + If the displacement field of runway record (PG) is empty, the façade shall fetch the extremity coordinates in PG record and compare them to the threshold coordinates in P2 record for same runway id.   + 2 possibilities:     - If threshold coordinates are filled in P2, there is a displacement 🡪 displacement value in PG record is computed by taking the distance between the extremity coordinates (in PG) and the threshold coordinates (in P2)     - If threshold coordinates are not filled in P2, there is no displacement 🡪 the displacement value in PG record is empty | #21.2-06 |
| SD-ALB-R4F-062 | RDB A424 – Airac Cycle | The AIRAC cycle on each record is set only upon record creation and not on record update. | #21.2-06 |
| SD-ALB-R4F-063 | RDB A424 – DISTHR | if in AIP data there is no runway centerline point with role DISTHR, the displacedThresholdDistance value is set to 0 even if there is a value in ADB lucem data displacementLength (as computed from the reference points loading). | #22.1-06 |
| SD-ALB-R4F-070 | RDB A424 – AIRAC Cycle | The AIRAC cycle present on each record of the A424 RDB indicates the production cycle (i.e. the cycle requested in the customer delivery) | #21.2-10  #22.1-04 |
| SD-ALB-R4F-080 | RDB A424 – order | A424 lines are sorted alphanumerically. | #21.2-15 |
| SD-ALB-R4F-090 | RDB A424 – stopway | The stopway length is propagated to the A424 façade along with its metrics so that it is stored into the Runway record.  The value is converted to FT from the metrics given in parameter.  The resulting value is rounded up from 0.5 | #24.2-07 |
| SD-ALB-R4F-100 | RDB A424 – bearing management | The output magnetic bearing value in PG (runway) record is converted to magnetic bearing using true bearing and airport magnetic variation. | #29.2-04 |
| SD-ALB-R4F-110 | RDB A424 – P2 bearing error | For bearing error calculation, the smallest difference value between the 2 angles is always considered.  The computation is done using the following formula : delta=180 - abs(abs( a1 - a2) - 180). | #29.2-04 |
| SD-ALB-R4F-120 | RDB A424 – Parameter runwaySurfaceTypes | RDB A424 Façade manages an optional parameter runwaySurfaceTypes.  The parameter value contains the list of surface types to export (in A424 format) with wildcards:  Ex:   * “H ASPH;H ASGR;H BITU” * “H%” * “H%;S%;U UNKN”   If a given runway surface does not match the parameter value, this runway will not be present in the output database. | #31.1-05 |
| SD-ALB-R4F-130 | RDB A424 – Parameter getCodesFromAirportData | A424 Façade manages an optional parameter getCodesFromAirportData.  The parameter value can be true or false.  If the parameter is present with the value equal true the IcaoCode and AreaCode attributes are set in the Create method for each record linked to icao. | #33.1-03 |
| SD-ALB-R4F-140 | RDB A424 – default metric for origination | To manage origination from AIP to ADBLucem, for the runway and P2 record if the input metric is not given for distance attributes (elevation, length, width…) a conversion meter to feet is applied. | #DLT-70 |
| SD-ALB-R4F-150 | RDB A424 – metric origination | When a data is originated, its metric is also originated so that it is possible to convert the value to the relevant output value (applying a conversion if needed, depending on the metric of the originated data and the metric of the generated one). | #DLT-118 |

Product map sample for this façade: see RDBLUCEM2RDBA424.xml.

### Dynamo Façade

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DYF-010 | Dynamo Facade | Dynamo Facade is a facade producing a Spatialite dump SQL file for the dynamo product.  The dump is produced by GDAL with command:  ogr2ogr -f SQLite -dsco SPATIALITE=YES *<filepath>* -lco GEOMETRY\_NAME=geom -lco SRID=4326 *<SQL connection string with table names>*  Indexes for parameters idarpt, origin and origincpm are added on generated file from GDAL with command:  ogrinfo *<filepath>* -sql "CREATE INDEX am\_polygons\_aggregation\_idarpt\_idx ON am\_polygons\_aggregation <parameter>  See documentation in DocumentationFacade.docx. | ~~#13.2-01~~  ~~#13.2-03~~  #23.2-03 |
| SD-ALB-DYF-020 | Dynamo Facade - name | Fully qualified name of Dynamo façade is:  **DynamoDataModel.DynamoFacade** | #13.2-02  #14.1-01 |
| SD-ALB-DYF-030 | Dynamo Facade - output | Dynamo Façade outputs a single Spatialite dump SQL file containing all airports.  SQL output file path in the form: <conversion engine output path>/dump\_NavBlue<AIRAC>.sqlite | #13.2-02  #14.1-01  #23.2-02 |
| SD-ALB-DYF-040 | Dynamo Facade - version | Dynamo Façade is compatible with version 1 of the conversion engine. | #13.2-02  #14.1-01 |
| SD-ALB-DYF-041 | Dynamo Façade  Business rules | Dynamo façade contains constructors methods to produces objects of dynamo data model (see SD-ALB-DYF-100).  Conversion and attribute mapping rules are detailed in document DQR\_Dynamo.docx chapter 4.4.   |  |  | | --- | --- | | ***origin*** | ***Feature*** | | 101 | Aerodrome Reference Point | | 301 | *Apron Element* | | 302 | *Construction Area* | | 303 | *Deicing Area* | | 304 | *Final Approach And Take Off Area* | | 305 | *Parking Stand Area* | | 306 | *Runway Displacedarea* | | 307 | *Runway Element* | | 308 | *Runway Intersection* | | 309 | *Runway Marking* | | 310 | *Runway Shoulder* | | 311 | *Service Road* | | 312 | *Stopway* | | 313 | *Taxiway Element* | | 314 | *Taxiway Shoulder* | | 315 | *Vertical Polygonal Structure* | | 316 | *Water* | | 317 | *Touch Down Lift Off Area* | | 318 | *Hotspot* |   Am\_version is a table of dynamo database. It is cleared before any dynamo delivery.  Dynamo façade creates a single am\_version element with:   * AIRAC cycle as defined by Conversion Service method parameter * AIRAC cycle start and end dates as retrieved from the Tools AiracCycle view (\*) * Version as defined by Conversion Service method parameter * Comments as a constant string * Delivery date as the conversion date timestamp   (\*) **Tools** is a database and **AiracCycle** is a view | #23.2-02  #19.2.0.2-03  #19.2.0.2-09 |
| SD-ALB-DYF-042 | Dynamo Façade  Anchor Point | The anchor point computation for single polygons (not aggregates) first uses STCentroid() geometry function. If the returned point is not on surface (concave geometry), it then uses STPointOnSurface() geometry function to ensure that the point computed is inside the polygon geometry.  For multi polygons (aggregated), it first get the Centroid point of the multi polygon with STCentroid() function. Then it looks for the closer “single polygon” centroid point from the multipolygon centroid point.  If this point is not in one of the polygons, it then looks for the closer “single polygon” point on surface (with PointOnSurface() function) from the multipolygon centroid point. | #25.2-06  #27.1-05 |
| SD-ALB-DYF-043 | Dynamo Façade PickingInfo | The pickinginfo field (string) contains info about the feature.  For runwaythreshold feature (origin = 108), the TORA, TODA, ASDA, LDA values shall be the same for the two thresholds of the couple normal/displaced. If the value is not set, it shall be replaced by “-“ | #25.2-04  #25.2-05 |
| SD-ALB-DYF-044 | Dynamo Façade Wingspan calculation | The dynamo façade shall compute the actual wingspan value for each individual row of origin “305” (parking stand area) or “313” (taxiway element) using the following algorithm:  If origin is equal to '305’:   * The value has to be the maximum value of wingspan of all the “StandGuidanceLine” features which are captured on the “ParkingStandArea” feature. * If no wingspan is defined in all these “StandGuidanceLine” features, default value has to be used ('-32767’).   If origin is equal to '313’:   * The value has to be the maximum value of wingspan of all the “TaxiwayGuidanceLine” features which are captured on the “TaxiwayElement” features. * If no wingspan is defined in all these ""TaxiwayGuidanceLine"" features, default value has to be used ('-32767’).   In other cases, the value has to be defined as '-32767'". | #19.2.0.2-04 |
| SD-ALB-DYF-045 | Dynamo Façade Surfacetype calculation | The dynamo façade shall compute the actual surfacetype value for each individual row using the following algorithm:   * When origin is equal to '306', '307', '308' or '312', the value has to be the surftype value passed in the feature mapping (see possible values on next slide); If not defined, it has to be set to '-32767’. * In other cases, the value has to be set to '-32767’”. | #19.2.0.2-05 |
| SD-ALB-DYF-046 | Dynamo Façade Status calculation | For rows with origin equal to “305” (parking stand area):   * If at least 1 “StandGuidanceLine” feature exists on the “ParkingStandArea” feature:   + If at least one of the “status” attribute value of all the “StandGuidanceLine” features present on the “ParkingStandArea” feature is different from ‘0’ (closed), then status has to be set to ‘1’ (open)   + Else status has to be set to ‘0’ (closed). * Else status has to be set to default value ('1’).   For rows with origin equal to “313” (taxiway element) :   * If at least 1 “TaxiwayGuidanceLine” feature or 1 “RunwayExitLine” exists on the “TaxiwayElement” feature:   + If at least one of the “status” attribute value of all the “TaxiwayGuidanceLine” and “RunwayExitLine” features present on the “TaxiwayElement” feature is different from ‘0’ (closed), then status has to be set to ‘1’ (open).   + Else status has to be set to ‘0’ (closed). * Else status has to be set to default value ('1’).   In all other cases, there is no modification from current computation. | #19.2.0.2-06 |

#### Dynamo data model

The dynamo façade uses a temporary data model to produce the output spatialite dump.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DYF-100 | Dynamo Model diagram | See models/Dynamo\_DataModel.pptx | #23.2-01  #23.2-02  #19.2.0.2-04  #19.2.0.2-05  #MD64 -> MD179 |
| SD-ALB-DYF-110 | Dynamo EF layer | An entity framework model is built upon this model using the “database first” design. | #23.2-01  #19.2.0.2-04  #19.2.0.2-05 |
| SD-ALB-DYF-115 | Dynamo Airports data | The data model use the methods Get() Create and Update() to produce the output database  The primary key is OGC\_FID  The data model is use to deal with the columns of the table with special requirements:  A method is used to Update the LONGEST\_RWY if there is a highter value.  A method is used to convert the TIMEZONE into a specified format.  A method is used to convert into feet the SPD\_LIM\_ALT.  A method is used with routeType and routeType Qualifier as input to set the LDG\_SYS.  A method is used to update the cycle date if a more recent one is found.  Airport coverage :   * If there is at least one runway which is greater or equal to 1000 meters (meaning that LONGEST\_RWY attribute is greater or equal to 1000 meters),   + For the US and Canada airports (beginning with character 'K', 'C', or 'P'),     - If the ICAO\_ID attribute contains at least one number, the airport does not have to be provided.     - Else,       * If the NAME attribute contains at least one of the full following words or series of words, the airport does not have to be provided:       * Field, FLD, FD, Airpark, APK, Jet park, flight park, Executive, exec, Farm, Ranch, Rancho, Airstrip, Naval auxiliary land field, nalf, Navy outlying field, nolf, NAF, Air Force Base, AFB, Army Air Field, AAF, Memorial, State, Industrial, Industrial Park, Junction       * Else, the airport has to be provided.   + Otherwise, the airport has to be provided * Otherwise, the airport does not have to be provided. | #MD64->90  #MD-75  #MD-76  #MD-78  #MD-82  #MD-89  MD-448  #MD-320  #MD-543  #MD-484 |
| SD-ALB-DYF-120 | Dynamo Waypoints data | The data model use the methods Get() Create and Update() to produce the output database  The primary key is OGC\_FID  The data model is use to deal with the columns of the table with special requirements:  A method is used to concatenate “Waypoint” “IDENT” and “COUNTRY\_ID” into FEATURE\_ID  A method is used to fill the AWY\_BELONG regarding if it belongs to an airport or not  A method is used to update the cycle date if a more recent one is found. | #MD-92 -> 110  #MD-94  #MD-106  #MD-109  MD-410  #MD-320 |
| SD-ALB-DYF-125 | Dynamo Navaids data | The data model use the methods Get() Create and Update() to produce the output database  The primary key is OGC\_FID  The data model is use to deal with the columns of the table with special requirements:  A method is used to concatenate “NavAids”, “NPTYE” “IDENT” and “COUNTRY\_ID” into the FEATURE\_ID attribute  A method is used to combine some Navblue Data Dictionnary parameters into the attribute CLASS  A method is used to combine some Navblue Data Dictionnary parameters into the attribute MAGVAR  A method is used to fill the AWY\_BELONG regarding if it belongs to an airport or not  A method is used to update the cycle date if a more recent one is found. | #MD-112 -> 130  #MD-114  #MD-120  #MD-127  #MD-128  #MD-130  MD-411  #MD-320  #MD-48 |
| SD-ALB-DYF-130 | Dynamo Airway Segments data | The data model use the methods Get() Create and Update() to produce the output database  The primary key is OGC\_FID  The data model is use to deal with the columns of the table with special requirements:  A method is used to concatenate different information into the attribute FEATURE\_ID  A method is used to compute the valuein NM for RNP  A method is used to update the cycle date if a more recent one is found. | #MD- 133 -> 179  #MD-135  #MD-146  MD-412  #MD-320 |
| SD-ALB-DYF-135 | Dynamo MatchWaypointAirway data | The data model create the fill the MatchWaypointAirway table at the finalize step.  The primary key is OGC\_FID  The data is provided by the tables Navaids, Waypoints and AirwaySegments | #MD-181 ->191  MD-507  #MD-320  #MD-279  #MD-415 |
| SD-ALB-DYF-140 | Dynamo MagVarIsogons data | The data model use the methods Get() Create and Update() to produce the output database  The primary key is OGC\_FID  The data model is use to deal with the columns of the table with special requirements:  A method is used to remove the “-“ characters from the attribute EFFDATE  A method is used to concatenate different information into the attribute FEATURE\_ID  A method is used to compute value of the attribute MAGVAR  A method is used to update the cycle date if a more recent one is found. | #MD-32 -> 38  #MD-320 |
| SD-ALB-DYF-145 | Dynamo ControlledAirspaces data | The data model use the methods Get() Create and Update() to produce the output database  The primary key is OGC\_FID  The data model is use to deal with the columns of the table with special requirements:  A method is used to concatenate different information into the attribute FEATURE\_ID  A method is used to match the value of the attribute ARSP\_TYP with the input value of BNDRY\_TYP  A method is used to compute the value of UP\_LIM, UP\_LIM\_TYP, LO\_LIM, LO\_LIM\_TYP and CYCLE\_DATE | #MD-201-> 222  MD-432  MD-434  MD-435  #MD-320 |
| SD-ALB-DYF-150 | Dynamo ContinentBoundaryPol data | The data model use the methods Get() Create and Update() to produce the output database  The primary key is OGC\_FID  The data model is use to deal with the columns of the table with special requirements:  The GEOM column contains the geometry of all continents.and GEOM information is computed from simplified-water-polygons-split-3857 extracted from <https://osmdata.openstreetmap.de/data/land-polygons.html> database.A method is used to concatenate different information into the attribute FEATURE\_ID | #MD-285-> 295  MD-416  #MD-320  #MD-561 |
| SD-ALB-DYF-155 | Dynamo CountryBoundaryLin data | The data model use the methods Get() Create and Update() to produce the output database  The primary key is OGC\_FID  The data model is use to deal with the columns of the table with special requirements:  GEOM does not have to be provided when TYPE attribute of ne\_10m\_admin\_0\_boundary\_lines\_land.shp file extracted from Natural Earth data source is equal to "Water Indicator".  A method is used to concatenate different information into the attribute FEATURE\_ID | #MD-296-> 301  MD-416  #MD-320  #MD-560 |
| SD-ALB-DYF-160 | Dynamo CountryBoundaryPt data | The data model use the methods Get() Create and Update() to produce the output database  The primary key is OGC\_FID  The data model is use to deal with the columns of the table with special requirements:  The GEOM column contains the geometric center of each country and GEOM information is computed from ne\_10m\_admin\_0\_sovereignty.shp extracted from <https://www.naturalearthdata.com/> database.  The NAME is equal to the NAME attribute of ne\_10m\_admin\_0\_sovereignty.shp extracted from https://www.naturalearthdata.com/ database.  A method is used to concatenate different information into the attribute FEATURE\_ID | #MD-302-> 308  MD-416-> 418  #MD-320  #MD-557  #MD-558 |
| SD-ALB-DYF-165 | Dynamo FirUirPol data | The data model use the methods Get() Create and Update() to produce the output database  The primary key is OGC\_FID  The data model is use to deal with the columns of the table with special requirements:  A method is used to concatenate different information into the attribute FEATURE\_ID  A method is used to concatenate CPDLC\_INF1 and CPDLC\_INF2 into the attribute CPDLC  A method is used to compute the value of FIR\_HI\_LIM, FIR\_HI\_TYP, UIR\_LO\_LIM, UIR\_LO\_TYP, UIR\_HI\_LIM and UIR\_HI\_TYP  A method is used to update the cycle date is a more recent one is found. | #MD-310-> 339  MD-431  #MD-320 |
| SD-ALB-DYF-170 | Dynamo Runways data | The data model use the methods Get() Create and Update() to produce the output database  The primary key is OGC\_FID  The data model is use to deal with the columns of the table with special requirements:  A method is used to concatenate different information into the attribute FEATURE\_ID  A method is used to compute the value of GEOM, SURFTYPE and WAYTYPE. The method computing the GEOM shall take into account that threshold name may contain the letter 'T' in the end, that shall match with the opposite threshold if it also contains the letter 'T' in the end (as for the couples 'C'/'C', 'R'/'L' and 'L'/'R').  A method is used to update the cycle date is a more recent one is found.  Runway coverage:  If airport linked is included in the coverage, runway has to be provided. | #MD-347-> 363  MD-448  #MD-544  #MD-320  #MD-47  #MD-543  #MD-573 |
| SD-ALB-DYF-175 | Dynamo Thresholds data | The data model use the methods Get() Create and Update() to produce the output database  The primary key is OGC\_FID  The data model is use to deal with the columns of the table with special requirements:  A method is used to concatenate different information into the attribute FEATURE\_ID  A method is used to compute the values of SLOPE, BRG\_REF, TORA, TODA, ASDA, LDA, ELEV, BRG.  A method is used with routeType and routeType Qualifier as input to set the LDG\_SYS.  Threshold coverage:  If airport linked is included in the coverage, threshold has to be provided. | MD-364 -> 382  MD-446  MD-447  MD-448  #MD-320  #MD-543 |

|  |  |  |  |
| --- | --- | --- | --- |
| SD-ALB-DYF-180 | Dynamo CruiseTable data | The data model use the methods Get() Create and Update() to produce the output database.  The primary key is OGC\_FID  An input from the CruisingTable can lead to to the creation of up to 4 CruiseTable insertions in the output database, depending whether the input field of each cruising block (CruiseLevelFrom1, CruiseLevelFrom2, CruiseLevelFrom3 or CruiseLevelFrom1) is empty or not.  The data model is used to deal with the columns of the table with special requirements:  A method is used to concatenate “CruiseTable”, CT\_ID” “FROM\_CRS” “TO\_CRS” “FROM\_ALT” and “TO\_ALT” into the FEATURE\_ID attribute  A method is used to convert the tenth of degrees from the FROM\_CRS and TO\_CRS attributes into regular degrees and round them to the upper unit. | #MD-385-> 399  #MD-320 |
| SD-ALB-DYF-185 | Dynamo RestrictiveAirspaces data | The data model use the methods Get() Create and Update() to produce the output database.  The primary key is OGC\_FID  The data model is used to deal with the columns of the table with special requirements:  A method is used to concatenate different information into the attribute FEATURE\_ID  A method is used to compute the value of the attribute “TIME\_OF\_OP”. The source of the data can be found in the document at references/ RestrictiveAirspaces-TIME\_OF\_OP.docx | #MD-224-> 244  #MD-320  #MD-458 |
| SD-ALB-DYF-190 | Dynamo CommunicationAreas data | The data model use the methods Get() Create and Update() to produce the output database.  The primary key is OGC\_FID  The data model is used to deal with the columns of the table with special requirements:  A method is used to concatenate different information into the attribute FEATURE\_ID  A method is used to compute the value of the attribute “MULTICODE”. Indeed, when several IDENT are identical, the MULTICODE attribute has to be created:   * If only one feature with a given IDENT attribute exists, associated MULTICODE has to be empty   If several features with the same IDENT attribute exist, a MULTICODE value has to be created, beginning with 'A' character for the first feature, 'B' for the second... | MD-402 -> 409  MD-419 -> 427  #MD-320 |
| SD-ALB-DYF-195 | Dynamo GridMORA data | The data model use the methods Get() Create and Update() to produce the output database.  The primary key is OGC\_FID  The data model is used to deal with the columns of the table with special requirements:  A method is used to concatenate "MORA", the latitude and the longitude of the bottom left corner into the attribute FEATURE\_ID, separated with colons (":").  In finalization we merge GridMora according this rule :   * When the GridMORA is between [71 ; 80] and [-72 ; -81] latitudes then the polygons have to be merged by each 2° longitude pattern, starting from 0°-Greenwich longitude. * When the GridMORA is between [81 ; 84] and [-82 ; -85] latitudes then the polygons have to be merged by each 4° longitude pattern, starting from 0°-Greenwich longitude. * When the GridMORA is between [85 ; 88] and [-86 ; -89] latitudes then the polygons have to be merged by each 12° longitude pattern, starting from 0°-Greenwich longitude. * When the GridMORA is between [89 ; 90] and latitude is equal to -90, then the polygons have to be merged by each 24° longitude pattern, starting from -12° to have Greenwich centered at -12° to +12°. * Otherwise, it will be equal to GEOM attribute value.   Value mora of the new GridMora is maximum of merged GridMora | MD-23 -> MD-30  MD-517  MD-518  #MD-320  #MD-549  #MD-550  #MD-554  #MD-551  #MD-552  #MD-566  #MD-567  #MD-568 |
| SD-ALB-DYF-200 | Dynamo EnrouteHoldings data | The data model use the methods Get() Create and Update() to produce the output database.  The primary key is OGC\_FID  The data model is used to deal with the columns of the table with special requirements:  A method is used to concatenate different information into the attribute FEATURE\_ID | MD-457  MD-461 -> MD-478  MD-515  MD-516  #MD-320 |
| SD-ALB-DYF-205 | Dynamo COP data | The data model use the methods Get() Create and Update() to produce the output database.  The primary key is OGC\_FID  The data model is used to deal with the columns of the table with special requirements:  A method is used to concatenate different information into the attribute FEATURE\_ID | MD-460  MD-482  MD-485 -> MD-497  MD-481  MD-513  MD-514  #MD-320 |
| SD-ALB-DYF-210 | Dynamo AirportCommunications | The data model use the methods Get() Create and Update() to produce the output database.  The primary key is OGC\_FID  The data model is used to deal with the columns of the table with special requirements:  A method is used to concatenate different information into the attribute FEATURE\_ID  A method is used to compute the value of the attribute “TIME\_OF\_OP”. The source of the data can be found in the document at references/ AirportCommunications-TIME\_OF\_OP.docx | #MD-414  #MD-510  #MD-511  #MD-320 |
| SD-ALB-DYF-215 | Dynamo EnrouteCommunications | The data model use the methods Get() Create and Update() to produce the output database.  The primary key is OGC\_FID  The data model is used to deal with the columns of the table with special requirements:  A method is used to concatenate different information into the attribute FEATURE\_ID | #MD-526  #MD-527  #MD-528  #MD-320 |
| SD-ALB-DYF-220 | Dynamo MajorLakes | The data model use the methods Get() Create and Update() to produce the output database.  The primary key is OGC\_FID  The data model is used to deal with the columns of the table with special requirements:  A method is used to concatenate different information into the attribute FEATURE\_ID | #MD-498  #MD-499  #MD-550  #MD-500 |
| SD-ALB-DYF-225 | Dynamo GraticuleLines | The primary key is OGC\_FID  The data model is used to deal with the columns of the table with special requirements:  A method is used to compute GEOM :   * All latitudes have to be provided by 1° interval in range between [89; -89]. * When the Graticule Line latitude is between [71; -71 ] then graticule lines for longitude have to be provided by 1° interval, starting from 0°-Greenwich longitude. * When the Graticule Line latitude is between [71 ; 81] and [-71 ; -81] then graticule lines for longitude have to be provided by 2° interval, starting from 0°-Greenwich longitude. * When the Graticule Line latitude is between [81 ; 85] and [-81 ; -85] then graticule lines for longitude have to be provided by 4° linterval, starting from 0°-Greenwich longitude. * When the Graticule Line latitude is between [85 ; 89] and [-85 ; -89] then graticule lines for longitude have to be provided by 12° linterval, starting from 0°-Greenwich longitude. * When the Graticule Line latitude is between [89 ; 90] and [-89 ; -90]  then graticule lines for longitude have to be provided by 24° linterval, starting from -12° to have Greenwich centered at -12° to +12°.   In addition, when graticule line geometry is created, it has to be divided into several segments whose length corresponds to a section of 10 degrees whatever latitude or longitude ones. | #MD-556  #MD-564  #MD-559  #MD-502  #MD-580 |

#### Product map sample

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DYF-050 | Dynamo Product Map  ADB2DYNAMO | Product map sample for dynamo façade: see ADB2DYNAMO.xml | #23.2-04  #25.2-01  #25.2-02  #19.2.0.2-03  #19.2.0.2-04  #19.2.0.2-05  #19.2.0.2-09  #MD64-179 |
| SD-ALB-DYF-055 | Dynamo Product Map  ENR2DYNAMO | Product map sample for dynamo façade:see ENR2DYNAMO.xml  Table : Airports, Runways, Thresholds, ControlledAirspaces, CommunicationAreas, FirUirPol, RestrictiveAirspaces, Waypoints, Navaids, AirwaySegment, CruisingTable, MatchWaypointAirway, COP, EnrouteHoldings, AirportCommunications, EnrouteCommunications | #MD-64  #MD-92  #MD-112  #MD-133  #MD-181  #MD-201  #MD-310  #MD-347  #MD-364  #MD-414  #MD-526 |

### TerrainElev And TerrainErr GeoTiff Conversion

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-GEC-010 | Create TerrainElev file in GeoTiff format | The TerrainElev GeoTIFF contain elevation as specified below:   * Data has to be provided for each 30 arcsecond. * Characteristics of the package * COMPRESSION = PACKBITS * Only one band in type Int16 * TILED = True * Size = 43200x21600 * Overviews : 21600x10800, 10800x5400 * Coordinate system = WGS84 * Elevation value defined as follows * If the 30 arcseconds point is on land (suggestion is to consider if this point is contained inside a polygon of "ContinentBoundaryPol" Mission+ table), the elevation will be the maximum elevation of all the elevation values of the source not taking into account the default or NoData values. * Else, the elevation will be a default value defined at -32765. | #MD-530  #MD-533 |
| SD-ALB-GEC-015 | Create TerrainErr file in GeoTiff format | The TerrainErr GeoTIFF to contain elevation error as specified below:   * Data has to be provided for each 30 arcsecond. * Characteristics of the package * COMPRESSION = PACKBITS * Only one band in type Int16 * TILED = True * Size = 43200x21600 * Overviews : 21600x10800, 10800x5400 * Coordinate system = WGS84 * Elevation error value defined as follows: * If the 30 arcseconds point is on land (suggestion is to consider if this point is contained inside a polygon of "ContinentBoundaryPol" Mission+ table), the elevation error of this 30 arcseconds point will be the maximum of all the absolute differences between the elevation of the 30 arcseconds point and the elevation not taking into account the default or NoData values of each source point; plus 10 meters. This computation is synthesized in the following formula: * ElevationErr = max(i;j) [|max(i;j) [Elevation(i;j)]- Elevation(i;j)|] with i and j the abscissa & ordinates of the source for one 30 arcseconds point + 10. * Not taking into account the default values. * Else, the elevation error will be a default value defined at -32765. | #MD-531  #MD-533 |

## Data packaging

### Product Packaging

This service packages the content of a folder.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-PPA-010 | Product packaging – dynamo DB | When the product is ADBMISSP the following process is applied:   * datainfo.json * Directory named "data" containing:   + Contents.xml (built as today)   + dump\_NavblueXXXX.md5   + dump\_NavblueXXXX.sqlite   with XXXX corresponding to the AIRAC cycle of the delivery   * Create A665-2 load from directory contents (archive + empty file) by calling LSBM Wrapper library using P/N and target H/W as parameters.   When the product is ENRMISSP the following process is applied:   * datainfo.json * Directory named "data" containing:   + dump\_NavblueXXXX.md5   + dump\_NavblueXXXX.sqlite   with XXXX corresponding to the AIRAC cycle of the delivery  When the product is EARMISSP the following process is applied:   * datainfo.json * Directory named "data" containing:   + LUC\_XXX\_EARMISSP\_YYYY\_Z.md5   + LUC\_XXX\_EARMISSP\_YYYY\_Z.sqlite   with XXX being the LUCEM code of Account in ALBATROSS, YYYY corresponding to the AIRAC cycle of the delivery and Z the increment of the delivery. So they will have the same name with "PackageGroupName" (zip folder)  When the product is TERMISGT the following process is applied:   * datainfo.json * Directory named "data" containing:   + LUC\_XXX\_TERMISGT\_YYYY\_Z.md5   + LUC\_XXX\_TERMISGT\_YYYY\_Z.sqlite   with XXX being the LUCEM code of Account in ALBATROSS, YYYY corresponding to the AIRAC cycle of the delivery and Z the increment of the delivery. So they will have the same name with "PackageGroupName" (zip folder)  Delivery folder must be multi-platform (Windows, iOS, LINUX...) | ~~#14.1-06~~  #23.2-05  #26.1-05  #19.2.0.2-09  #MD-49  #MD-480  #MD-483 |
| SD-ALB-PPA-011 | Product packaging – dynamo DB Meta INF | The META-INF directory shall contain a single MANIFEST.MF text file with contents from this template:  Manifest-Version: 1.0  EFB-LPCNG-Functional-Designation: dynamo-data-base  Bundle-Name: dynamo-data-base  Build-Date: {0}  Bundle-Vendor: NAVBLUE  Bundle-SymbolicName: dynamo-data-base  EFB-LPCNG-Software-FIN: FIN\_DYNAMO\_DATA\_BASE  Bundle-Version: {1}  Bundle-Description: AIRAC Cycle {1}  {0} in the form YYYY/MM/DD HH:MM:SS  {1} AIRAC cycle number (e.g.: 1705) | ~~#14.1-06~~  #23.2-05 |
| SD-ALB-PPA-012 | Product packaging – Contents.xml | The Contents.xml file contains information on the current delivery.  The structure is :  <Contents>  <Product name = “ADBMISSP”/>  <Effectivity cycle=”1706” date=”2017-05-25”/>  <AirportEntries>  <Airport icao=”EHAM” name = “Amsterdam” production\_cycle=”1706” updated=”Updated”>  <Airport icao=”LFPG” name = “Paris CDG” production\_cycle=”1705” updated=”Unchanged”>  <Airport icao=”LFPO” name = “Paris Orly” production\_cycle=”1706” updated=”Added”>  <Airport icao=”KLAX” name = “Los Angeles” production\_cycle=”1705” updated=”Removed”>  </AirportEntries>  </Contents>  For each airport, it shows if data has been updated since last cycle.  This file is for the moment only generated for Dynamo deliveries from the last delivery of same customer with status ‘Delivered’ and the ‘Test delivery’ flag set to false. | #26.1-05  #26.1-06  #30.2-10  #19.2.0.2-09  #19.2.0.2-10 |
| SD-ALB-PPA-014 | Product packaging – Contents.xml | The datainfo.json file contains information on the current delivery for products ENRMISSP, EARMISSP, TERMISGT and ADBMISSP.  datainfo = {  DQR Version :  DataTypeCode :  PackageGroupName :  PackageGroupSourceDate :  PackageGroupFileType :  EffectiveDateStart :  EffectiveDateEnd : *(except for EARMISSP and TERMISGT)*  }  For product TERMISGT (respectively EARMISSP), "DQRVersion" is "TERMISGT\_X.Y.Z" (respectively "EARMISSP\_X.Y.Z"), with X.Y.Z which can be settable in the delivery itself (to be manually added). X.Y corresponds to the DQR document and Z to an increment about the format of the database, and "PackageGroupName" is "LUC\_XXX\_TERMISGT\_YYYY\_Z" (respectiveley "LUC\_XXX\_EARMISSP\_YYYY\_Z") (as described in SD-ALB-PPA-010) | MD-428  MD-437  MD-438  #MD-438  #MD-480  #MD-483 |
| SD-ALB-PPA-020 | Product packaging – SFTP publish | Load packaging service allows publishing a delivery to a distant sftp site.  The interface takes as parameter the delivery reference to allow finding the delivery package on disk for publication (using delivery path configuration).  The sftp connection details are configurable: server url, login, password and path.  The interface logs the processing and returns a clear status so that the publication step are monitored from albatross customer delivery. | #27.2-06 |
| SD-ALB-PPA-030 | Product packaging – Grid Mora + | Load packaging service for Grid Mora+ product consists in:   * Generating a MD5 sum crc file regarding the .ari file. * Storing both .ari and .md5 files in a directory named like the part number (or timestamp)   The grid Mora output file contains UNIX style line terminators (LF only) | #28.1-04  #36.2-10 |
| SD-ALB-PPA-040 | Product Packaging - Default | Load packaging service accepts a default product.  No specific packaging action performed for default product:   * The output directory structure does not change * The processing status synchronism with Customer Delivery is performed | #32.2-04 |

## Data qualification

### Cross Check Treatment

#### Overview

The data qualification service cross check treatment web method compares two data sources, outputs a check report and sets a status.

The comparison is done on two heterogeneous databases and a mapping file is used to describe which properties should be compared, the tolerance, what are the metrics … This file is called configuration check file.

#### Mapping file

The following requirements describe the structure of the mapping configuration file.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-QUA-010 | Data Qualification – sample | The configuration file describes how to map properties to be compared. See “CheckConfiguration.xml” file. | #12.2-03  #19.2.0.2-13 |
| SD-ALB-QUA-020 | Data Qualification - FeatureCheck | A check configuration file is composed of FeatureCheck elements, each one describing the two features compared.  The “name” attribute is the name of the feature. | #12.2-03 |
| SD-ALB-QUA-030 | Data Qualification – VerifiedData | The description of the verified feature.  <Product> is the datasource of the feature.  <Feature> is the verified feature  <Identifier> is the name of the property of the feature used to identify this feature. | #12.2-03 |
| SD-ALB-QUA-040 | Data Qualification –ReferenceData | The description of the reference data.  <Product> is the datasource of the feature.  <Feature> is –he reference feature  <Identifier> is the name of the property of the feature used to identify this feature | #12.2-03 |
| SD-ALB-QUA-050 | Data Qualification - Transformations | An element is used to transform a value before verification or use.  It can be used in:   * VerifiedData * ReferenceData * LinkTo   Several transformations can be used in sequence to produce a transformation chain.  ”name””is the name ”f t”e transformation applied.  “parameters” allows t– pass parameters specific to the transformation.  Sample:  <Transform name="Split" parameters="/,0"/> | #12.2-03 |
| SD-ALB-QUA-060 | Data Qualification - Check | <Check> is the element that definitions which co–parisons are don on two features of <FeatureCheck>.  A <FeatureCheck> element can contain several <Check>. | #12.2-03 |
| SD-ALB-QUA-070 | Data Qualification - Attribute | <VerifiedAttribute> and <ReferenceAttribute> define the compared attributes of a check.  “metric” attribute is optional and can define statically the metric of the compared property.  <Field> is the name of the property compared.  <FieldMetric> is the name of the field containing the metric for <Field>. This element can conflict with “metric” attribute. | #12.2-03 |
| SD-ALB-QUA-090 | Data Qualification – LinkTo | The LinkTo element is used to define an attribute by navigation.  <LinkTo feature="Runway" with="identifier" using="usedRunway\_href">  “feature” is the name of the feature containing the property where navigation leads.  “with” is the name of the property used –or comparison on “Runway”  “using” is the name of the property used for comparison on the current feature. | #12.2-03 |
| SD-ALB-QUA-100 | Data Qualification - Comparator | The comparator defines which comparison will be done on the two attributes of the compared features. | #12.2-03 |
| SD-ALB-QUA-110 | Data Qualification – FieldName | FilesName element is available as a child of <VerifiedAttribute> and <ReferenceAttribute>.  It can contain a constant or a reference to another attribute.  In the report it replaces the name of the attribute. | #14.2-05 |
| SD-ALB-QUA-111 | Data Qualification – FeartureCheckName | “name” is an attribute available in “FeatureCheck” element.  This attribute is mandatory and unique in a mapping file.  The value is included in the FeatureCheckName column of the CrossCheckReport table. | #14.2-06 |

#### Checker behavior

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-QUA-500 | Data Qualification – conversion | A conversion may be done to compare attributes with different metrics. If no metric is defined, no conversion is done.  If a metric is defined on an attribute it has to be defined on both attributes compared and reference. Then a conversion to the reference metric is done before values comparison. | #12.2-08 |
| SD-ALB-QUA-510 | Data Qualification – tolerance | A tolerance may be applied in comparisons. By default a tolerance of 0 is applied. | #12.2-08 |
| SD-ALB-QUA-520 | Data Qualification – Filter | Comparisons are applied on all features a datasource. The filter allows to limit the set of compared features. | #12.2-03 |
| SD-ALB-QUA-530 | Data Qualification – results | Any discrepancy found by the checker is stored in the database | #12.2-06  #12.2-09 |
| SD-ALB-QUA-540 | Data Qualification – datasources | The verified databsource is identified by ICAO, version and AIRAC.  The reference datasource is identified by ICAO. The last version for last airac is retrieved from database. | #12.2-07 |
| SD-ALB-QUA-550 | Data Qualification – status | At the end of the check process, the inventory table status is updated.  If no discrepancy is found the status is set to “validated”.  If at least one discrepancy is found, the status is set to “invalid”. | #12.2-10 |
| SD-ALB-QUA-560 | Data Qualification – service | A service allows to call the check proves with the following parameters:   * Datasourcetype * ICAO * Airac * Version * Level   A cross check is performed on all datasources of type “Datasourcetype”.  Datasources are retrieved as stated in SD-ALB-QUA-540.  Level can take the following values:   * OK * WARNING * ERROR   Level is optional and its default value is ERROR. | #12.2-11  #14.2-02 |
| SD-ALB-QUA-561 | Data Qualification – service level | The level parameter defines the minimum level of checks which are traced in database.  Ascending level is OK, WARNING, ERROR. | #14.2-02 |
| SD-ALB-QUA-565 | Data Qualification –features detections | The checker detects features which exist only in reference data or verified data. | #14.2-03 |
| SD-ALB-QUA-566 | Data Qualification –filter persistence | The filter used on verified data and reference data features mapping is stored in the CrossCheckReport table during the check.  If several filters are set, the filters are separated with “;”. | #14.2-04 |
| SD-ALB-QUA-570 | Data Qualification –identifiers | The identifiers VerifiedIdentifier and ReferenceIdentifier are stored in the CorssCheckReport without application of the transformations. | #14.2-07 |
| SD-ALB-QUA-571 | Data Qualification – delta value | The delta column stored the delta between the verified value and the reference value.  The delta value contains the metric.  For geometries representing a point, the delta is the distance between project points (A816 projection). | #14.2-09 |
| SD-ALB-QUA-580 | Data qualification – trigger on validation | When a product of data inventory is set to validated, perform the following operation:   * Identify all product maps where the product is used a data source * For each of this product map if all data sources are validated for same ICAO / AIRAC then create in albatross the corresponding ProductDatabase for same ICAO / AIRAC. Set the ProductDatabase status to ‘Provided’. | #17.2-09  #30.2-09 |

#### Comparators

| **ID** | **Short ti–le** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-QUA-700 | Data Qualification – eq | Equals comparator. Can be applied on all values. | #12.2-03 |
| SD-ALB-QUA-710 | Data Qualification - lte | Less than or equal comparator.  Can be applied on any numeric value. | #12.2-03 |
| SD-ALB-QUA-720 | Data Qualification – gte | Greater than or equal comparator.  Can be applied on any numeric value. | #12–2-03 |
| SD-ALB-QUA-730 | Data Qualification - gt | Greater than comparator.  Can be applied on any numeric value | #12.2-03 |
| SD-ALB-QUA-740 | Data Qualification - lt | Less than comparator.  Can be applied on any numeric value | #12.2-03 |

#### Check result database

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-QUA-800 | Data Qualifica–ion – database | A database stores all data relative to check results.  See Schema model in “models” folder. | #12.2-05 |
| SD-ALB-QUA-801 | Data Qualifi–ation - CrossCheckResult | A data field allows to store the cross check result with values:   * OK * WARNING * ERROR | #14.2-01 |
| SD-ALB-QUA-802 | Data Qualification - Delta | A delta column allows to store the delta between the reference and the verfifiedvalue. | #14.2-09 |
| SD-ALB-QUA-803 | Delta Qualification – NotValidated Table | A table is used to store the NotValidated information for a feature.  Indicates if the properties can be validated or not during P2 RDB A424 conversion. Shall be false by default, only present if true.  This tables contains:   * Id of product in inventory * Id for feature * id for attribute * Analyst identification * Analyst comment   The presence of the line means the attribute is not validated. | #19.1-05  #19.2-02 |

#### Email notification and CASE creation

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-QUA-850 | Data Qualification – email notification | At the end of the cross check process, if errors exist in the cross check report, an email is sent. | #18.1-06 |
| SD-ALB-QUA-851 | Data Qualification – CASE creation | At the end of the cross check process, if errors exits in the cross check report, a CASE is created in CRM. | #26.1-01 |
| SD-ALB-QUA-855 | Data Qualification – email notification recipients | The recipients list of the mail is configurable. | #18.1-06 |
| SD-ALB-QUA-856 | Data Qualification – CASE accound and requestor | The account and the requestor identifier used to create the CASE are configurable. | #26.1-01 |
| SD-ALB-QUA-860 | Data Qualification – email notification template | The content of the mail is built from a configurable template. | #18.1-06 |
| SD-ALB-QUA-865 | Data Qualification – email notification content | The mail contains the following information:   * The ICAO/AIRAC/version and data source of the product that was checked * The details of the errors raised (extract from cross-check report) * An URL to the Analyst tool allowing to inspect the product(s) and report details ([https://dataserv.lucem-aerodata.com](https://dataserv.lucem-aerodata.com/)) | #18.1-06 |
| SD-ALB-QUA-866 | Data Qualification – CASE content | The CASE shall have the following charaterisitcs:   * The Title is ICAO\_cycle\_version\_[X]\_Qualification Error Report, with [X] empty if data source is AIP or "ADB" if data source is ADB Lucem * The Nature is "Technical" * The Type is "Issue" * The Subtype is "Data" * The Airport is ICAO * The Origin is Internal * The Airac cycle * The associated Cycle * The responsible is "LV2 Production" * The Account is the one specified in configuration * The Requestor is the one specified in configuration * The error report is attached to the CASE. | #26.1-01 |

### Evaluate Obstacles Traceability

#### Overview

The data qualification service evaluate obstacles traceability web method allows triggering the identification of obstacle history on the published obstacle data.

The service loops on published obstacles and compare with previous published version to identify changes and store changes details in the Lineage attribute of the latest obstacle version.

#### Behavior

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-EOT-001 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #29.1-03 |
| SD-ALB-EOT-002 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #29.1-03 |
| SD-ALB-EOT-003 | Name | EvaluateObstaclesTraceability | #29.1-03 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-EOT-010 | Parameter – country code | Type: string  A country code (2 letters ISO 3166-1 alpha-2 as defined in albatross ; ex: “de” for germany)  Must not be null or empty. | #29.1-03 |
| SD-ALB-EOT-020 | Parameter – icao | Type: string  The Airport icao.  Optional; default is $NE. | #29.1-03 |
| SD-ALB-EOT-030 | Parameter  airacCycle | Type: string  The airac cycle of the import.  Must not be null or empty. | #29.1-03 |
| SD-ALB-EOT-040 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #29.1-03 |

Business Rules:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-EOT-100 | History qualification | The history qualification of published obstacles::   * Loop on all obstacles in Obstacles database matching the parameters * For each of them, it identifies a matching obstacle from a previous version (see SD-ALB-EOT-110). * Depending on the comparisons the system writes in lineage attribute of the published obstacle:   + If no previous obstacle version is identified, it is considered as added and lineage is filled with: "OBS [identifier] ADDED ON [current date] BY [operator]" with 'stfeat' and 'stvalid' attributes filled in accordingly with the [current date]   + If a previous obstacle is identified but an end date attribute (endfeat or endvalid) is set on the new version, it is considered as deleted and lineage is filled with: "OBS [identifier] DELETED ON [current date] BY [operator]"   + If a previous obstacle version is identified, it is considered as modified and the system compare the 2 versions attribute by attribute and identify the changes. Lineage is filled with: "OBS [identifier] MODIFIED ON [current date] BY [operator] - MODIFICATION DONE ON ATTRIBUTE [attribute]: PREVIOUS VALUE [previous value of the modified attribute], NEW VALUE [new value of the modified attribute]" (a line shall be created for each attribute modified) | #29.1-03 |
| SD-ALB-EOT-110 | Matching Obstacles | A matching obstacle in a previous version is identified using the following rules:   * the same "ident" attribute (if any), * and the same "feattype" attribute * and the same geometry. | #29.1-03 |

Tester Tool:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-EOT-200 | tester | The Qualification Service tester tool allows testing EvaluateObstaclesTraceability web service interface. | #29.1-03 |

### Adjust Mora From Obstacles

#### Overview

The data qualification service adjust mora from obstacle web method allows triggering the mora elevations adjustment based on latest published obstacles.

The service loads the published obstacles, check their validity and compare their adjusted elevation to any intersecting mora to update the mora value accordingly.

#### Behavior

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-AMO-001 | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #29.2-01 |
| SD-ALB-AMO-002 | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #29.2-01 |
| SD-ALB-AMO-003 | Name | AdjustMoraFromObstacles | #29.2-01 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-AMO-010 | Parameter – country code | Type: string  A country code (2 letters ISO 3166-1 alpha-2 as defined in albatross ; ex: “de” for germany)  Must not be null or empty. | #29.2-01 |
| SD-ALB-AMO-020 | Parameter – icao | Type: string  The Airport icao.  Optional; default is $NE. | #29.2-01 |
| SD-ALB-AMO-030 | Parameter  airacCycle | Type: string  The airac cycle of the import.  Must not be null or empty. | #29.2-01 |
| SD-ALB-AMO-040 | Parameter  version | Type: string  The version of the import.  Must not be null or empty | #29.2-01 |

Business Rules:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-AMO-100 | Obstacle Validity | The obstacle validity for considering as candidate to update the mora is checked against the following tables.  Only obstacles that pass the checks are considered for mora update. | #29.2-01 |
| SD-ALB-AMO-110 | Mora Adjustment | Once the valid obstacle list has been obtained, the system loops on the list to :   * retrieve the intersecting MORA(s) objects from Mora database * Compute and Adjust the obstacle elevation at top in feet depending on actual elevation at top value (or elevation at base + height)   🡪 +1000 ft if under or equal to 5000 ft, +2000 ft if over 5000 ft)   * Add the adjusted obstacle elevation to a memory map indexed with mora id * Compare the higher adjusted obstacle elevation in map with the indexed mora value * If higher, update the mora elevation value with the adjusted obstacle elevation value * If lower or equal and mora source is obstacle, update the mora elevation value with the adjusted obstacle elevation value only if adjusted obstacle elevation is higher than latest MORA version with AD&S source * If lower or equal and mora source is AD&S, do not update the mora elevation value | #29.2-01 |
| SD-ALB-AMO-120 | Mora persistance | Any updated mora object is persisted as a new object with updated attributes as follows:   * The source attribute of the updated mora contains the reference to the obstacle that was the cause of the elevation adjustment: Internal identifier + Obstacle Ident from capture * The airac cycle and version attributes of the updated mora contains the airac cycle and version of the obstacle that was the cause of the elevation adjustment   If at least one mora is updated, add data inventory entry for grid mora data source with airac cycle and version of the obstacle publication that was the source of the changes, only if similar entry does not exist in DataInventory. | #29.2-01 |

Tester Tool:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-AMO-200 | tester | The Qualification Service tester tool allows testing AdjustMoraFromObstacles web service interface. | #29.2-01 |

## Data server

### Overview

The data server is a server component which delivers data sources as GEO JSON format. Various requests allow retrieving airports data from a generic datasource described by a mapping file.

The server also allows to originate data (modify source data). Originated data is stored in a separate database.

### Data server datamodel

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DSV-000 | Data Server – datamodel | The datamodel used to store originations is the qualification model.  See models/QualificationDataModel.jpg | #13.1-02  #29.1-01 |
| SD-ALB-DSV-005 | Data Server – datamodel content | Data stored in qualification model is:   * + Associated product (from inventory table)   + Feature (with identifier) and attribute modified   + New value   + Source product (from inventory table) used as source of modification   + Analyst identification   + Analyst comments | #13.1-02 |
| SD-ALB-DSV-007 | Data Server – origination | An origination is a modified property of an original datasource. This datasource allows to store an origination without modifications to the original datasource. | #13.1-02 |
| SD-ALB-DSV-010 | Data Server – modified model | An airport should be viewed as the original data source with originations modifications applied. | #13.1-02 |
| SD-ALB-DSV-015 | Data Server – Entity Framework | Entity Framework models of datasources are used to access exitsting datasources. | #13.1-03 |

### GEO Json format

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DSV-100 | Data Server – GUID | A GUID is generated for each feature of the GEO JSON document. This GUID identifies a unique feature in the datasource. | #13.1-04 |
| SD-ALB-DSV-105 | Data Server – GEOJSON Standard | The standard used for GEOSON is <http://geojson.org/geojson-spec.html> | #13.1-05 |
| SD-ALB-DSV-110 | Data Server – GEO JSON common properties | All features in the GEO JSON document should have the following properties:   * + **feattype** property shall contain the feature type name (ex. RunwayDirection)   + **label\_id** shall be the label string to display (on map) (configurable by concatenation of attributes using separator)   + **name\_id** shall be the display name in product selection tree (configurable by concatenation of attributes using separator)   + **originated** shall indicate if the feature is from initial product or if it has been created from origination data (see SD-ALB-DSV-007)   + **readonly** shall indicate if the feature is replaceable from the GUI or not (always true when feature is originated)   + **notvalidated** indicate if the feature is NotValidated | #13.1-05  #13.1-09  #30.2-04 |
| SD-ALB-DSV-115 | Data Server – GEO JSON specific properties | A feature in the GEO JSON document contains all properties mapped in the mapping file. Each feature is implemented a structure containing:   * the value * the value Field Name (optional) * the metric (optional) * the metric Field Name (optional) | #13.1-05  #DLT-70  #DLT-118 |
| SD-ALB-DSV-116 | Data Server – Geometries | The geo json document contains geometric properties of the data source features. Geometries are converted from database format to geo json format.  A geo json geometry can be null. | #13.1-05 |
| SD-ALB-DSV-117 | Data Server – feature grouping | The “value” property of “featttype” is prepended with the value of the “group” attribute of the mapping file.  A “group\_by” property is added to the “properties” of the document with a single property named “value” which contains the GroupBy attribute of the corresponding feature in the mapping file. | #14.2-15 |
| SD-ALB-DSV-118 | Data Server - NotValidated | For each feature, if a corresponding line exists in the NotValidated table of the qualification database, then set the property notvalidated=true for this feature. | #19.1-05  #19.2-02  #30.2-04 |
| SD-ALB-DSV-119 | Data Server - Dates | All the dates that could appears in the GEO JSON should be in the format “YYYY-MM-DD HH:MM:SS” | #27.1-01 |

### Security

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DSV-200 | Data Server – HTTPS | HTTPS is used to secure exchanges. | #13.1-01 |
| SD-ALB-DSV-210 | Data Server – Authentication | Authorizations are setup using Windows Integrated Authentication against Active Directory. | #13.1-01 |

### Services

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DSV-300 | Data Server – originations | Modifications of the data source is done without modifying the original source. | #13.1-02 |
| SD-ALB-DSV-310 | Data Server – complete data set | A service allows to retrieve a complete data set for a product defined by its:   * ICAO * AIRAC Cycle * Version   The data set should contain original features and originated features. (SD-ALB-DSV-010) | #13.1-03 |
| SD-ALB-DSV-315 | Data Server – add feature | A service allows adding a feature in the datasource. | #13.1-04 |
| SD-ALB-DSV-320 | Data Server – find feature | A service allows retrieving a feature with a GUID. | #13.1-04 |
| SD-ALB-DSV-325 | Data Server – sources service | The sources service allows retrieving all available sources:   * Method : GET * URI : sources   Returns all data sources present in the inventory table. | #13.1-06 |
| SD-ALB-DSV-330 | Data Server – airports service | The sources service allows retrieving all available airports in a source:   * Method : GET * URI : airports/{source}   Returns all airports of a data source. | #13.1-06 |
| SD-ALB-DSV-335 | Data Server – airac service | The airac service allows retrieving all airac cycles of an airport in a data source:   * Method : GET * URI : cycles/{source}/{icao}   Returns all airac cycles for an airport in a data source.. | #13.1-06 |
| SD-ALB-DSV-340 | Data Server – version service | The version service allows retrieving all versions of an airac cycles of an airport in a data source:   * Method : GET * URI : versions/{source}/{icao}/{airac}   Returns all versions of an airac cycles for an airport in a data source.. | #13.1-06 |
| SD-ALB-DSV-345 | Data Server – status service | The status service allows retrieving all versions of an airac cycles of an airport in a data source:   * Method : GET * URI : status/{source}/{icao}/{airac}/{version}   Returns the status of the cross check of an airport. Values are one of:   * valid * invalid | #13.1-06 |
| SD-ALB-DSV-350 | Data Server – product service | The product service allows retrieving a product in geo json format:   * Method : GET * URI : product/{source}/{icao}/{airac}/{version}   Returns a geo json document respecting rules describes in the dedicated § of this section and using the mapping file.  Returns a 404 response if product is not found in database, | #13.1-07 |
| SD-ALB-DSV-355 | Data Server – cross check report service | The product service allows retrieving all versions of an airac cycles of an airport in a data source:   * Method : GET * URI : report/{source}/{icao}/{airac}/{version}   Returns a CSV document representing the cross check report defined by parameters of the URI.  Returns a 404 response if product is not found in database,  Return an empty document if no cross check report exist for this product. | #13.1-08  #21.1-01 |
| SD-ALB-DSV-360 | Data Server – replace attribute service | The replace attribute service allows replacing an attribute of a feature by the value of another attribute:   * Method : GET * URI : replace   Parameters are passed in the body of the request:  {"src":{"featurekey":“*<guid of feature>*","attributename":"*<property name or “geometry”>*, “unitname”:”<metric attribute name>”(optional)},  "dest":{"featurekey":"*<guid of feature>*","attributename":“*<property name or “geometry”>*", “unitname”:”<metric attribute name>” (optional)},  "comments":“*<analyst comments>*"}  The origination is saved in the database.  Returns a single feature containing the originated feature.  Returns a 404 response if product of feature is not found in database,  If the attribute is “read only” in the feature map, an error is returned.  If the ‘unitname’ is filled in the origination request, the metric attributes are stored in addition of value attributes. When the ‘unitname’ is filled only on src or dst, the default metric from the service configuration is used to create the originations attributes for the missing metric.  When AIP data source is found in the source feature key, the function creates a Qualification entity in Albatross and attaches it to the corresponding Cycle entity.  Failure to find the corresponding entity or cycle in albatross or to create the entity logs an error in data server but not prevent the initial action (create the origination in the database). | #13.1-09  #17.2-07  #30.1-02  #DLT-118  #DLT-131 |
| SD-ALB-DSV-365 | Data Server – validate service | The validate service allows validating a product:   * Method : GET * URI : replace   Body of the request:  {“isvalid”:true/false,"comments":“*<analyst comments>*"}  Processing:  Update the QualificationStatusId attribute of the inventory table.  When a product of data inventory is set to validated, perform the following operation:   * Identify all product maps where the product is used a data source * For each of this product map if all data sources are validated for same ICAO / AIRAC then create in albatross the corresponding ProductDatabase for same ICAO / AIRAC. Set the ProductDatabase status to ‘Provided’. | #13.1-10  #30.2-09 |
| SD-ALB-DSV-366 | Data server – set not validated service | The set not validated service allows to mark a feature as NotValidated.  Set a feature attribute notvalidated status:   * + Method = "POST",   + UriTemplate = “notvalidated"   Request body as JSON:   * + {"featurekey":“*<guid of feature>*", "attributename":"*<property name or “geometry”>*", “status”:true/false, "comments":“*<analyst comments>*"}   Processing:   * If status = true   + Add a line in the NotValidated table for the property * If status = false   + Remove the NotValidated line if it exists   The request body without attribute name (missing, null or empty) is acceptable.  Return HTTP 404 NotFound if product/feature not found in database.  Return concerned feature in GeoJSON (single feature, not FeatureCollection) with NotValidated property set on appropriate feature.  When AIP data source is found in the feature key, the function creates or deletes (depending on status) a Qualification entity in Albatross and attaches it to the corresponding Cycle entity.  Failure to find the corresponding entity or cycle in albatross or to delete/create the entity logs an error in data server but not prevent the initial action (create, delete the notValidated in the database). | #19.1-06  #19.2-03  #20.1-02  #30.1-02  #30.2-04 |
| SD-ALB-DSV-367 | Data server – delete origination service | The delete service allows deleting existing originations:   * Method : POST * URI : delete   Parameters are passed in the body of the request:  {"featurekey":"*<guid of feature>*","comments":“*<analyst comments>*"}  All originations associated to destination feature are removed from the database.  Returns a single feature containing the feature not originated with originated flag set to false.  Returns a 404 response if product of feature is not found in database,  When AIP data source is found in the source feature key, the function deletes a Qualification entity in Albatross on corresponding Cycle entity.  Failure to find the corresponding entity or cycle in albatross or to delete the entity logs an error in data server but not prevent the initial action (delete the origination in the database). | #27.1-10  #30.1-02 |
| SD-ALB-DSV-370 | Data server – Duplicate origination | The duplicate origination service allows duplicating existing originations:   * Method : POST * URI : duplicateOrigination/{originationId}/{icao}/{airac}/{version}   originationId: identifier in qualification database origination table  icao, airac, version: information to identify the published aip data in inventory  The function duplicates the existing origination arborescence (origination / attributes) and link it to the published aip data inventory as identified by the parameters.  It returns the identifier in qualification database origination table of the newly created origination.  Returns a 404 response if product of feature is not found in database, | #30.1-02 |
| SD-ALB-DSV-371 | Data server – Duplicate not validated | The duplicate not validated service allows duplicating existing not validated:   * Method : POST * URI : duplicateNotValidated/{notValidatedId}/{icao}/{airac}/{version}   notValidatedId: identifier in qualification database notValidated table  icao, airac, version: information to identify the published aip data in inventory  The function duplicates the existing notValidated entry and link it to the published aip data inventory as identified by the parameters.  It returns the identifier in qualification database notValidted table of the newly created notValidated.  Returns a 404 response if product of feature is not found in database, | #30.1-02 |
| SD-ALB-DSV-380 | Data server – Get inventory service | The product service allows retrieving all versions of an airac cycles of an airport in a data source:   * Method : GET * URI : inventory   Returns a CSV document representing the contents of the data inventory table.  The get request runs against the database a query similar to the one provided bellow and return the result as a csv file with header row:  *SELECT DI.[Icao]*  *,DS.Name*  *,DI.[AiracCycle]*  *,DI.[Version]*  *,DI.[LoadDateTime]*  *,CR.Name as CrossCheckStatus*  *,DI.[CrossCheckDateTime]*  *,QS.[Name] as QualificationStatus*  *,DI.[QualifiedBy]*  *,DI.[QualificationDateTime]*  *,DI.[Comment]*  *FROM [Qualification].[dbo].[DataInventory] DI,*  *[Qualification].[dbo].[DataSource] DS,*  *[Qualification].[dbo].[CrossCheckResult] CR,*  *[Qualification].[dbo].[QualificationStatus] QS*  *where DI.CrossCheckStatusId=CR.Id and DI.QualificationStatusId=QS.Id*  *and DS.Id=DI.DataSourceId*  *order by DI.[Icao],DS.Name desc,DI.[AiracCycle],DI.[Version]* | #32.1-05 |
| SD-ALB-DSV-390 | Data server – Get Reference Data service | The service allows retrieving all reference data:   * Method : GET * URI : referencedata   Returns a CSV document representing the contents of the reference data tables in a readable format (i.e. runway thresholds geomatical information).  The get request runs against the database a query similar to the one provided bellow and return the result as a csv file with header row:  *SELECT [ogr\_geometry].STY as Lat,[ogr\_geometry].STX as Long, tab.[idthr], [status], [tdze], [tdzslope], [brngtrue], [brngmag], [rwyslope], [tora], [toda], [asda], [lda], [thrtype], [elev], [availPavedSurfFromThr], [measuredLda], [measuredTora], [measuredAsda], [source], tab.[icao], tab.[airac], tab.[version], [DbType], img.[imagery\_date]*  *FROM [AlbatrossService\_ADBLucem].[dbo].[runwaythreshold] tab,*  *[AlbatrossService\_ADBLucem].[dbo].[imagerydata] img*  *where tab.airac = (select max(airac) from [AlbatrossService\_ADBLucem].[dbo].[runwaythreshold] where tab.icao=icao)*  *and tab.version = (select max(version) from [AlbatrossService\_ADBLucem].[dbo].[runwaythreshold] where tab.icao=icao and tab.airac=airac)*  *and img.icao=tab.icao and img.airac=tab.airac and img.version=tab.version and img.idthr=tab.idthr*  *order by tab.icao, tab.airac, tab.version* | #33.1-05 |

### Mapping file

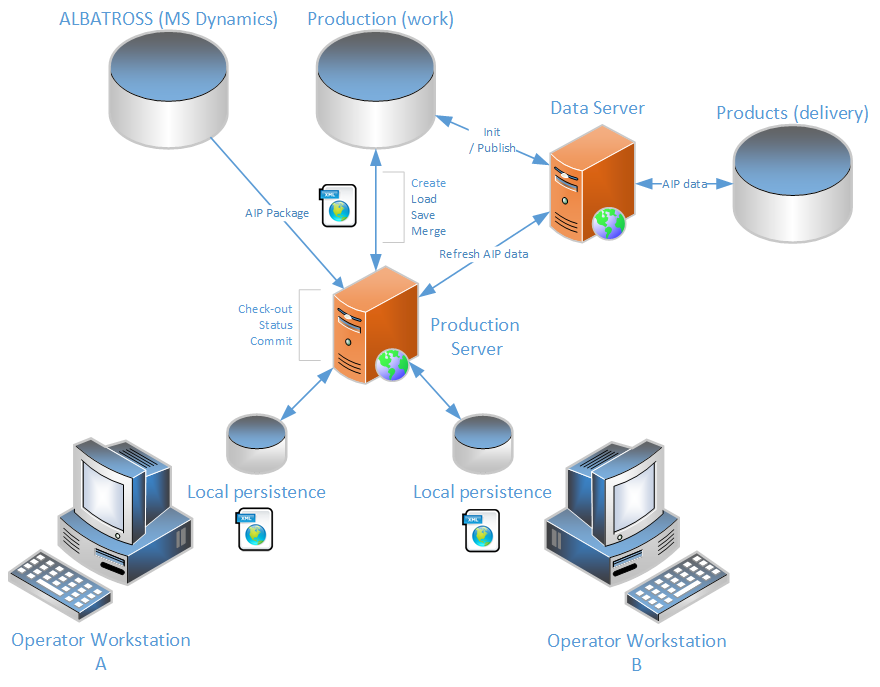
| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DSV-400 | Data Server – mapping file | A mapping file describes how to map a data source to the geo json form | #13.1-01 |
| SD-ALB-DSV-410 | Data Server – format | Format description:  <!—name of the feature (table) in the database ->  <Feature name="ApronElement">  <!—the field used to identify this feature 🡪  <Id>ogr\_fid</Id>  <!—the field used as the name in geo json for this feature 🡪  <Name>idapron</Name>  <!—the field used as the label in geo json for this feature 🡪  <Label>idapron</Label>  <!—the field used as the geometry in geo json for this feature 🡪  <Geometry>ogr\_geometry</Geometry>  <!— a field used as classic property in geo json for this feature 🡪  <Field>idnumber</Field>  </Feature> | #13.1-01 |
| SD-ALB-DSV-415 | Data server – mapping file parameter | All services using a mapping file provide an optional parameter to select a mapping file.  If parameter is not present, the default mapping file is used.  If mapping file doesn’t exist, 404 error code is returned.. | #17.2-05 |
| SD-ALB-DSV-420 | Data server – linkTo | The “linkTo” feature allows to retrieve a property from another feature.  The linkTo function takes into origination on the target feature.  A property filled with the linkTo functionality is read only.  The LinkTo clause manages the case where the linked feature does not exist and return NULL as a value for the property. (Jira ALB-517)  The LinkTo clause manages fields using the ‘IsMetric’ property (Jira ALB-553) | #17.2-06 |
| SD-ALB-DSV-500 | Data server – AIP/AMDB to OPALE feature map | A feature map exists to create a GeoJSON file with features dedicated to OPALE. See OPALE.xml file. | #17.2-13  #19.2.0.2-02  #19.2.0.2-08 |
| SD-ALB-DSV-510 | Data server – Minima feature map | Data server component contains a feature map to allow retrieval of the A424 data source contents:   * Airport * Runway (with continuation record type s)   This feature map is called “minima” to be called from the minima capture tool and used to retrieve data for a given icao, airac and version. | #19.0.0.2-11 |

## Data production server

### Overview

The data production server is a configuration management system for the Opale editor. Opale is a standalone graphical tool used to edit airport maps. It uses its own XML format.

The data production server stores in a relational database the model contained in Opale XML files and offers Web services to get versions, commit versions, merge versions and publish versions the same way a tool like SVN would do.



### Data model

The data production server stores the content of XML files in a database. The XML file respect the schema defined by an XSD file.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-001 | DPS – database | Data model is persisted in a SQL database | #15.2-01 |
| SD-ALB-DPS-010 | DPS - schema | The SQL model can store a model provided as an XML file compatible with the XSD schema. | #15.2-02 |
| SD-ALB-DPS-020 | DPS - AirportArea | The model contains a Feature AirportArea which inherits from abstractAMDBSurfaceFeature with a single property geometry.  This feature has his proper layer “Airport Area”  There can be 0 or 1 of this feature inside the layer. | #16.1-15  #26.2-09 |
| SD-ALB-DPS-030 | DPS – CheckStatus | “AirportDatabase” element contains an attribute “checkStatus” of type unsignedByte.  Values:   * 0 = NOT CHECKED, * 1 = OK * 2 = WARNING, * 3 = ERROR * 4 = ERROR ACKNOWLEDGED   Default value is 0 on dataset creation. | #17.1-04 |
| SD-ALB-DPS-040 | DPS - Status | The status workflow respect the following schema: | #17.2-15  #33.2-01 |
| SD-ALB-DPS-050 | DPS - Lock | Geometry and feature elements from airport database contains an attribute lock which allows to indicate if the geometry/feature is locked in Opale.  Values:   * 0 = NOT LOCKED, * 1 = LOCKED   Default value is 1 on dataset creation. | #26.2-10 |
| SD-ALB-DPS-060 | DPS – aip change | The parking stand location feature elements contains an optional attribute ‘aipchange’.  This attribute is a string enum with the following values: “none”, “created”, “updated”, “deleted”.  This attribute is not considered as a change for the delta computation during update merge. | #30.1-08 |

### Services

Here are described the services published by the DPS server.

#### Publish

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-PUBLISH-001 | DPS –PUBLISH | URL : https://<root>/publish/<ICAO>/<AIRAC>  Method : GET  Load latest publish version / working version with status “Validated” (see US 16.1-14) from database for same ICAO and AIRAC parameters  If AirportDatabase of database model is equal to one of:   * NOT CHECKED * ERROR   Then stop processing and return 403.  Convert elements from OPALE model into ADB LUCEM model by:   * Parsing features elements (RunwayElement, TaxiwayElement, etc. …) and creating similar element in ABD LUCEM model * Copying attributes values * Creating geometries by following the surface, curves and points links in the OPALE model (see SD-ALB-DPS-PUBLISH-002) * Giving a value to the “Layer” attribute with the “name” attribute of the layer containing the feature * Exploding feature with multi geometries to several features with simple geometry. Each new feature has a unique “idnumber”   Persist created ADB LUCEM model into Product Database  Update Data Inventory for ICAO / AIRAC / publish version  Set the dataset status in production database to “Published”  If processing is OK:  Return ICAO, AIRAC, publish\_version, working\_version, status from new version as JSON dataset object  If error: Status is set to Submitted  HTTP status code :   * 400 if problem * 403 if checkStatus not correct * 200 otherwise | #16.1-06  #16.1-07  #16.1-14  #16.2-04  #16.2-05  #16.2-06  #17.1-05  #24.2-11  #26.1-02  #30.2-03 |
| SD-ALB-DPS-PUBLISH-002 | DPS –PUBLISH-BEZIERS | Transform Bezier curves in a line geometry with 20 points.  Number of points reduction  Number of points are reduced to **10** if feature geometry contains only one beziers curve, **5** if feature geometry contains 2 beziers curves or more.  If the RunwayExitLine is made of more than one OPALE geometry (i.e. a beziers curve and a segment, or 2 beziers curves), the resulting points shall be merged in a single polyline geometry in the published ADB Lucem model.  If shape is too complex, abort number of points reduction. | #16.1-06  #19.1-03 |
| SD-ALB-DPS-PUBLISH-003 | DPS –PUBLISH-IMAGERY DATA | A row in the ImageryData table is created for each runway direction/threshold of the published database with the same image information from Opale AirportDabase attributes imagedate, part number and hacc. | #24.1-13  #24.1-14  #27.2-02 |
| SD-ALB-DPS-PUBLISH-004 | DPS –PUBLISH-SIMPLIFY GEOMETRIES | Published geometries are simplified by removing the outline points that connect 2 aligned segments for both Lines and Polygons using SQL geometry Reduce function.  When a geometry is not valid after the reduce function, the original geometry is published.  Tolerance parameter for SQL geometry Reduce function is configurable. | #26.2-02 |
| SD-ALB-DPS-PUBLISH-005 | DPS –PUBLISH-AIRPORT AREA | During opale database publish, the airport area is constructed from opale dataset airport area feature:   * Id is generated at database persistance * Geometry is constructed from opale feature geometry * Area is set to “2” (airport zone) * Icao, airac and version is taken from the published data info. | #28.2-05 |

#### Commit

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-COMMIT-001 | DPS –COMMIT | URL : https://<root>/commit  Method : POST  Body: current working XML file in a ZIP archive  Process:  Retrieve XML file from request body  De-serialize into data model  Retrieve status of airport database in DB. If status is one of:   * Published * Submitted * Validated * Cancelled   Then:   * Don’t commit * Return 409   Check XML working version and latest working version from database  If same :   * Update uploaded version model with new working\_version * Get the “aip version” from the previsous working version. * persist model to database * Return new ICAO, AIRAC, publish\_version, working\_version (updated) from database as JSON dataset object   Else   * return empty   HTTP status code :   * 409 if conflict * 201 otherwise | #15.2-08  #16.1-14  #33.1-08  #33.2-01 |

#### CheckCommit

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-CHKCOMMIT-001 | DPS – CHECKCOMMIT | URL : https://<root>/commit/<ICAO>/<AIRAC>/<pubilish\_version>/<working\_version>  Method : GET  Search the last working version according to ICAO, AIRAC and publish\_version  If found version is greater than the one provided in parameter then return a JSON structure describing the version.  Otherwise return empty.  HTTP status code :   * 200 if database version is more recent * 204 otherwise | #15.2-07 |

#### Checkout

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-CHECKOUT-001 | DPS – CHECKOUT | URL : https://<root>/checkout/<ICAO>/<AIRAC>/<pubilish\_version>/<working\_version>  Method : GET  Retrieve the dataset corresponding to the provided parameters.  Cancelled databases are not available for checkout.  Return a ZIP archive containing:   * Dataset as an XML file * Binary map archive * GeoJSON AIP * AIP documents   HTTP status code :   * 200 * 404 if the dataset hasn’t been found | #15.2-04  #16.1-09  #16.1-14  #26.2-03  #33.2-01 |

#### Update

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS- UPDATE-001 | DPS –UPDATE | URL : https://<root>/update  Method : POST  Body: an XML file in a ZIP archive  Check | #15.2-06 |
| SD-ALB-DPS-MERGE-001 | DPS –MERGE | The merge service is called during the update process, it merges the current model in database and the one provided in XML as described in the document “references/merge.docx”.  Deleted elements are taken into account during delete.  Some attributes are ignored during differences detection.  HTTP status code :  200 Created | #16.2-03  #17.1-03  #19.2-09  #26.2-01 |

#### CkeckUpdate

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-CHECK\_UPDATE-001 | DPS – CHECK UPDATE | URL : https://<root>/update/<ICAO>/<AIRAC>/<pubilish\_version>/<working\_version>  Method : GET  Retrieve the dataset for ICAO/AIRAC/publish\_version with the higher working\_version.  If working\_version of found dataset greater than the provided working\_version on the URL then:   * Return HTTP code 200 * Return ICAO, AIRAC, publish\_version, working\_version from database as JSON dataset object   Else   * Return HTTP code 204 | #15.2-05 |

#### List

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-LIST-001 | DPS – LIST | URL : https://<root>/list  Method : GET  Return all datasets as a JSON document.  For performance reason, the query shall be limited to AirportDatabase contents necessary in the Json.  The JSON contains one line per airport last publish version and last working version with status different than ‘cancelled’.  The JSON contains:   * ICAO * Name * AIRAC Cycle * Publish version * Working version * AIP version   HTTP status code : 200 | #15.2-03  #ALB-543  #33.2-01 |

#### Initialize

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-INIT-001 | DPS –INIT | URL : https://<root>/initialize  Method : POST  Body : a JSON structure with:   * A JSON dataset object with ICAO, AIRAC, publish\_version, working\_version   Process:   * Retrieve data from JSON request body * Create data model and initialize with request details * Get the more recent image archive present in the binary directory (see SD-ALB-DPS-IRP-004).   If no file with ‘.dat’ extension is available in the binary directory, the binary map to associate with the airport database is taken directly from the table BinaryMap in Opale DB if it exists a line with same Image Date than binary directory name.   * Retrieve matching AIP data from Data Server using feature map described in SD-ALB-DSV-500 (as GeoJSON, see US 15.2-12) and initialize airport information for dataset. * Initialize Parking Stand location features with idstd, can, termref from AircraftStand in GeoJson and set ‘aipchange’ attribute to “none”. * Persist model into database * Attach the binary map archive to the dataset in database (e.g. as BLOB object) * Return a JSON dataset object with details of the created dataset or empty if same ICAO / AIRAC already exist * Retrieve AIP documents from Albatross and store them in db for later download * Set the airport database status to “Initialized”.   If existing a cancelled database shall be ignored.  HTTP status code :  201 Created  409 Conflict | #16.1-09  #15.2-10  #16.1-14  #17.2-14  #26.1-08  #27.2-05  #30.1-09  #33.2-01 |

#### AIP Update

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-AIPUDP-001 | DPS –AIPUPD | URL : https://<root>/aip/<ICAO>/<AIRAC>/<publish\_version>/<working\_version>/<aip\_version>  Method : GET  Get last version from data server of the AIP.  If found version is more recent than the one stored in database then update database (including AIP documents from Albatross).  If version in database is more recent than the one provides on the request then return AIP as GeoJSON (including AIP Documents).  HTTP status code :  200 if new version  204 otherwise | #15.2-13  #16.1-09 |

#### Create airport database version

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-CREATEAIRPORT-001 | DPS –CREATEAIRPORT | URL : https://<root>/version/<ICAO>/<AIRAC>>?ignoreComparison=<ignoreComparison>  Method : GET  Load latest publish version from database for same ICAO and AIRAC parameters  If not found   * insert call to “get from products” to take previous version from a published product (AMDB, A816, …).   If found (a cancelled database is ignored and the copy is done from the previous active version in database):   * upgrade the publish version number to +1 * Retrieve, convert metric data into meters and store matching AIP data * check the latest image archive present in the binary directory   + If the current image record in table has the same date as the image archive folder, the service uses the current record.   + If the image archive folder has a more recent date, a new record is created and this new record is used for further datasets   + If no file with ‘.dat’ extension is available in the binary directory, the binary map to associate with the airport database is taken directly from the table BinaryMap in Opale DB if it exists a line with same Image Date than binary directory name. * Update the parking stand location features (and geometry) in opale dataset according to the changes detected in aip data parking stands (see SD-ALB-DPS-CREATECYCLE-010) * Persist the new version dataset to database * Set the airport database status to “Initialized”.   Else:   * Return 400   HTTP status code :   * 201 Created | #16.1-01  #16.1-02  #16.1-14  #26.1-08  #27.2-05  #30.1-09  #33.2-01  #DLT-26  #DLT-153 |
| SD-ALB-DPS-CREATEAIRPORT-02 | Parking stand guidance Update | When Status Wingspan, Terminalref of a stand are updated, all the connected stand guidance attributes are updated automatically.  When the data of a field coming from AIP is empty then the OPALE corresponding field is updated by "$UNK" on strings and -32767 on numerical values. | #DLT-22  #DLT-90 |
| SD-ALB-DPS- CREATEAIRPORT -03 | Parking stand area update | When PCN, Terminalref of a stand location are updated, all the underlapped stand area with the same id are updated automatically.  When the data of a field coming from AIP is empty then the OPALE corresponding field is updated by "$UNK" on strings and -32767 on numerical values. | #DLT-23  #DLT-90 |
| SD-ALB-DPS- CREATEAIRPORT -04 | Update reference data SA | When there is the creation of an Update Reference data Service activity, it is possible to prevent the AMDB update | #DLT-26 |

#### Create cycle interface

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-CREATECYCLE-001 | DPS –CREATECYCLE | URL : https://<root>/cycle/<ICAO>/<AIRAC>?ignoreComparison=<ignoreComparison>  Method : GET  Load latest AIRAC and publish version from database for same ICAO parameter  If not found,   * insert call to “get from products” to take previous version from a published product (AMDB, A816, …)   If found (a cancelled database is ignored and the copy is done from the previous active version in database):   * Set the AIRAC cycle to the AIRAC parameter provided * Set the publish version to 0 and the working version to 1 * Retrieve, convert metric data into meters and store matching AIP data * check the latest image archive present in the binary directory   + If the current image record in table has the same date as the image archive folder, the service uses the current record.   + If the image archive folder has a more recent date, a new record is created and this new record is used for further datasets.   + If no file with ‘.dat’ extension is available in the binary directory, the binary map to associate with the airport database is taken directly from the table BinaryMap in Opale DB if it exists a line with same Image Date than binary directory name. * Update the parking stand location features in opale dataset according to the changes detected in aip data parking stands (see SD-ALB-DPS-CREATECYCLE-010) * Update the parking stand guidance Line features in opale dataset according to the changes detected in aip data parking stand. * Update the parking stand area features in opale dataset according to the changes detected in aip data parking stands.Persist the new cycle dataset (exact copy of the previous one save for the AIRAC cycle and versions) to database * Set the airport database status to “Initialized”.   Else:   * Return 400   HTTP status code :   * 201 Created | #16.1-03  #16.1-04  #16.1-14  #26.1-08  #27.2-05  #30.1-09  #33.2-01  #DLT-21  #DLT-22  #DLT-23  #DLT-26  #DLT-153 |
| SD-ALB-DPS-CREATECYCLE-010 | DPS – AIP changes | The function realize following actions:   * Compare the parkings stands in aip data with the existing parking stand locations in new opale dataset * Update or not the stand location feature in opale depending on the result of the comparison. * Set the aipchange flag value of the stand location feature according to the action identified (created, updated, deleted)   Same parking stands in aip and opale data are identified with following rule:   * The stand features shall have same idstd attribute value * The stand features coordinates shall be distant of 45 meters or less   The following table gives the rules and actions to apply depending on the outcome of the comparison between aip and opale data:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | AIP found | OPALE found | Attributes changed | Coordinates changed | Action | AipChange flag value | | Yes | No | - | - | Add point and feature initialized from AIP data. | created | | Yes | Yes | Yes | No | Update feature attributes except if the AIP attribute value (acn, terminalref) is not captured (i.e. empty or NULL). | updated | | Yes | Yes | No | Yes < 5m |  | none | | Yes | Yes | Yes | Yes <5m | Update feature attribute | updated | | Yes | Yes | No | Yes  between 5m and 45m |  | locationChanged | | Yes | Yes | - | Yes > 45m | Update feature attribute if changed. | updated | | Yes | Yes | Yes | Yes >5m and < 45m | Update feature attributes. | updated | | Yes | Yes | No | No | - | none | | No | Yes | - | - | - | deleted |   The distance ranges:   * 45m for identification, * > 5M for update   are configurable to allow future adjustment. | #30.1-09  #ALB-549 |
| SD-ALB-DPS-CREATECYCLE-012 | Parking stand guidance Update | When Status Wingspan, Terminalref of a stand are updated, all the connected stand guidance attributes are updated automatically.  When the data of a field coming from AIP is empty then the OPALE corresponding field is updated by "$UNK" on strings and -32767 on numerical values. | #DLT-22  #DLT-90 |
| SD-ALB-DPS-CREATECYCLE-014 | Parking stand area update | When PCN, Terminalref of a stand location are updated, all the underlapped stand area with the same id are updated automatically.  When the data of a field coming from AIP is empty then the OPALE corresponding field is updated by "$UNK" on strings and -32767 on numerical values. | #DLT-23  #DLT-90 |
| SD-ALB-DPS-CREATECYCLE-016 | Update reference data SA | When there is the creation of an Update Reference data Service activity, it is possible to prevent the AMDB update | #DLT-26 |

#### Initialize from data source

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-IFDS-001 | DPS –Initialize from data source | URL : https://<root>/initialize/<source>  Method : POST  Body: a JSON structure with:   * A JSON dataset object with ICAO, AIRAC, publish\_version, working\_version   Process:   * Retrieve data from JSON request body * Create data model and initialize with request details * Get the more recent image archive present in the binary directory (see SD-ALB-DPS-IRP-004). Once the archive data is loaded into opale database binarymaps table, it is deleted from disk and replaced with a placeholder file for reference <icao>.txt to save storage space. (<icao>.txt file contains loading date, and airac cycle for history). * Retrieve feature data from the requested data source (A816 only) and initialize the data model contents. * Retrieve matching AIP data from Data Server (as GeoJSON) and PDF package * Attach the binary map archive to the dataset in database (e.g. as BLOB object) * Return a JSON dataset object with details of the created dataset or empty if same ICAO / AIRAC already exist * Merge points as described in SD-ALB-DPS-IFDS-002   HTTP status code :   * 201 Created * HTTP 409 Conflict | #16.2-07  #17.2-08  #26.1-08 |
| SD-ALB-DPS-IFDS-002 | DPS –Initialize from data source – merge points | Merge all points with same coordinates update references to the new merge point. | #17.2-08 |

#### Start dataset capture

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-STC-001 | DPS –start dataset capture | Starting dataset capture:   * + URL: https://<root>/start/<ICAO>/<AIRAC>/<publish\_version>   + Type: GET   + Set status to “InProgress” | #17.2-16 |

#### Submit dataset capture

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-SUC-001 | DPS –start dataset capture | Submit dataset capture:   * URL: https://<root>/submit/<ICAO>/<AIRAC>/<publish\_version> * Type: GET * Set status to “Submitted” | #17.2-16 |

#### Validate dataset capture

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-VC-001 | DPS –validate dataset capture | Validate dataset capture:   * URL: https://<root>/validate/<ICAO>/<AIRAC>/<publish\_version> * Type: GET   Set status to “Validated”  Retrieve status of airport database in DB. If status different than:   * Submitted * Validated   Then:   * Don’t validate * Return 403 * Force checkstatus if ERROR to ERROR\_ACKNOWLEDGED * Lock all geometries and features of the airport database | #17.2-16  #26.2-10 |

#### Clean previous working version

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-CPWV-001 | DPS – clean previous working version | Clean previous working version:   * URL: https://<root>/clean/<ICAO>/<AIRAC>/<publish\_version> * Type: GET * Delete previous working versions for similar publish version (keep latest only). * If the latest dataset status is not “published”, the request for deletion is rejected with a status forbidden. * In case of exception only log an error | #30.2-03 |

#### Cancel an airport database

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-CANCEL-001 | DPS – cancel airport database | Cancel working version:   * URL: https://<root>/cancel/<ICAO>/<AIRAC>/<publish\_version> * Type: GET   Process:   * Change status to “Cancelled” for last working version * Delete previous working versions | #33.2-01 |

#### Image retrieval and processing

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-DPS-IRP-001 | IRP – binary application | The Image Retrieval and processing application is a binary console application developed in C# or .NET language.  It is installable on the production server and called from scheduled task in windows. | #26.1-07 |
| SD-ALB-DPS-IRP-002 | IRP – process | The Image Retrieval and processing application realizes following actions :   * Checks the presence of a geo archive in a dedicated directory * Check the md5sum for zip validy * Check the archive content * Reads the <icao>\_ortho\_metadata.XML file from the archive and extracts needed information : part number, airport name, arp latitude, arp longitude, icao, acquisition date, haccuracy (the latitude and longitude can be DDD°MM'SS"[E/W] or DD°MM'SS"[E/W]). * Extracts the TIFF file and converts it to OPALE archive by calling QUARTZ with the proper parameters * Clean Quartz working directory * Create layer in geoserver (see SD-ALB-DPS-IRP-005) * Discards the geo archive from the incoming directory and stores it for reference in another directory | #26.1-07  #27.2-03  #DLT-186 |
| SD-ALB-DPS-IRP-003 | IRP – Quartz | The Image Retrieval and processing application extracts the TIFF image from DIMAP archive in a dedicated temporary directory and converted to OPALE archive using QUARTZ tool with proper parameters:   * ICAO code (extracted from geo archive) * Airport name (extracted from geo archive) * ARP LAT + LONG (extracted from geo archive) * Horizontal accuracy (extracted from geo archive) * Tiles size ("250“) * TIFF image source directory * OPALE archive output path (see SD-ALB-DPS-IRP-004) * Image date (extracted from geo archive with time set to “00:00:00.00”) | #26.1-07  #27.2-03 |
| SD-ALB-DPS-IRP-004 | IRP – archive directory structure | OPALE archive output path is in the binary folder of the DataProductionService (e.g. “E:\Work\DataProductionService\Binary”), in a folder by ICAO and a sub-folder by part number.  The OPALE archive is named <ICAO>.dat | #26.1-07  #27.2-03 |
| SD-ALB-DPS-IRP-005 | IRP – image layer creation in GeoServer | The Image Retrieval and processing application copy the tif image to a dedicated data storage folder usable by GeoServer (the path is configurable).  Then it sends an http request to geoserver to create the layer based on the copied image file (GeoServer url, workspace name, login and password are configurable.  Request looks like: curl -v -u admin:geoserver -XPUT -H "Content-type: text/plain" -d "file:///<path to copied tif file>" “<geoserver\_url>/rest/workspaces/Navblue/coveragestores/<ICAO>/external.geotiff?configure=first&coverageName=<ICAO>" | #27.2-04 |

## Product Map generation service

### Product Map generation service

This service generates a product map from a configuration file specifying what it shall contain.

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| ***SD-ALB-PMG-001*** | Technology | This web service is a WCF service over net.tcp protocol. It is defined as a CS class. | #CM-189 |
| ***SD-ALB-PMG-002*** | Logs levels | Use log levels as this:   * INFO: cessing steps * DEBUG: detailed info * ERROR: errors | #CM-189 |
| ***SD-ALB-PMG-003*** | Name | ProductMapGenerationService | #CM-189 |

Parameters:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-PGM-010 | Parameter – configuration JSON | Type: string  A JSON content describing the configuration for the product map generation.   * "AreaCode" : the list of area codes to use for filtering * "IcaoCode" : the list of valid ICAO codes to use for filtering (optional) * "Geometry" : a structure giving the latitude ("Latitude") and longitude ("Longitude") for the bottom/left ("BottomLeftPoint") and top/right point ("TopRightPoint") defining a bounding area to use for filtering (optional). Features that have a geometry will only be included if all their geometries match * "Header" : the identifiier of the header to use in the product map ("0" for none) * "outputFileName" : gives the output file name for the produced product map output.   Then, there is for each feature, there is :   * "value" of type boolean to define whether this feature shall be included or not in the product map * "areaCode" to override the general "AreaCode" if specific for this feature, * "IcaoCode" to override the general "IcaoCode" if specific for this feature * "Geometry" to override the general "Geometry" if specific for this feature   Must not be null or empty. | #CM-189 |
| SD-ALB-PGM-020 | Parameter – output file name | Type: string  The base name for the generated product map file (".xml" will be appended automatically)  Must not be empty. | #CM-189 |

Business Rules:

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-PGM-100 | Header inclusion | If the header value is 0, then no header shall be included. Otherwise the header corresponding to the specified one shall be included in the generated product map. | #CM-189 |
| SD-ALB-PGM-110 | Inclusion | The features for which the "Value" is false shall not be included in the generated product map. | #CM-189 |
| SD-ALB-PGM-120 | Filtering | If a filtering is implied from the specification of code areas, ICAO codes, or bounding geometry (using the value defined at general level, or feature level if defined) the relevant Filter shall be inserted for that feature in the generated product map.  The default product map configuration shall be filtered on record type S.  In the case of the filter on the geometry of Navaids, the filter only applies on geometry not null | #CM-189  #CM-219  #CM-237 |

# SDA424 Delta tool

## Overview

This tool is a standalone program which compares two A424 and generates a CSV containing differences.

## Business rules

| **ID** | **Short title** | **Description** | **US** |
| --- | --- | --- | --- |
| SD-ALB-A4D-001 | A4D – output format | The output format should follow the format of the sample references/A424-diff-sample.xlsx | #18.1-05 |
| SD-ALB-A4D-010 | A4D – packaging | The tool is packaged as a standalone program | #18.1-05 |
| SD-ALB-A4D-020 | A4D – comparison | The tool compares all fields of the files for all ICAOs | #18.1-05 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |