

**Airline Industry Data Model**

*Technology Logical Model Guidelines*

*Date: 24-Mar 2020*

Document Version: V 0.5

*For further information and updates please contact* *Jean-Christophe Cornu (*[*cornuj@iata.org*](mailto:cornuj@iata.org)*) at IATA Airline Industry Data Model Secretariat (*[*aidm@iata.org*](mailto:aidm@iata.org)*)*

Table of Contents

[1 Introduction 4](#_Toc11131686)

[1.1 Document Purpose and Intended Audience 4](#_Toc11131687)

[1.2 Document Context 4](#_Toc11131688)

[2 Overall Approach to Logical Message Modeling 5](#_Toc11131689)

[2.1 Logical Message Model – Relationship with other Models 5](#_Toc11131690)

[2.2 Logical Message Model – Approach and Profile 5](#_Toc11131691)

[3 Modeling Logical Message Schemas 6](#_Toc11131692)

[3.1 Logical Message Schema Usage 6](#_Toc11131693)

[3.2 BBIE Size Restrictions 7](#_Toc11131694)

[3.3 Logical Message Schema Derivation from the LDM – Initial 8](#_Toc11131695)

[3.4 Logical Message Schema Properties – Restrictions and Collapsing 10](#_Toc11131696)

[3.5 Logical Message Schema Derivation from the LDM – Synchronization 14](#_Toc11131697)

[3.6 Logical Message BIE details – Types of ABIEs 17](#_Toc11131698)

[3.7 Logical Message Schema Rules and Quality Assurance 19](#_Toc11131699)

[3.8 Logical Message Schema Diagrams 19](#_Toc11131700)

[3.9 Logical Message Schema Package Structure 19](#_Toc11131701)

[3.10 Logical Message Schema Forward Engineering 19](#_Toc11131702)

[4 IT Component Catalog 20](#_Toc11131703)

[5 IT Context Diagram 20](#_Toc11131704)

[6 State Machine Diagram 20](#_Toc11131705)

[7 Sequence Diagram 20](#_Toc11131706)

[8 Aspects to be refined during initial modeling 20](#_Toc11131707)

[Annexes 21](#_Toc11131708)

[Annex A : Glossary of this document 21](#_Toc11131709)

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Name** | **Description of change** |
| 0.1 | 29 Feb 2016 | Michael Thomas | First version : Skeleton document , with Usage of Synchronize tool |
| 0.2 | 08 Mar 2016 | Michael Thomas | Minor corrections & clarifications after TAG review |
| 0.3 draft1 | 25 Aug 2016 | Michael Thomas | Section 3.5 Logical Msg ABIEs Details, 1st version |
| 0.3 draft2 | 27 Jun 2017 | Matthew Keiller | Minor clarification to avoid unbounded cardinalities and apply more restrictive data types in section 3.3 |
| 0.3 | 30 Aug 2017 | Michael Thomas | Collapsing guidelines added to Logical Message  Schema Properties, sections 3.3+3.4 |
| 0.4 | 10 Jun 2019 | Graham Ferguson | Added section 3.2 BBIE Size Restrictions |
| 0.5 | 24 Jun 2020 | Jean-Christophe Cornu | Minor correction on chapter 3.3 to remove parameter box from synchronization process in section “*Understanding what elements are created by Synchronize* |

# Introduction

## Document Purpose and Intended Audience

The purpose of this document is to describe how to model the artifacts in the logical layer of the technology pillar (partition “T3”), of the airline industry data model.

These artifacts include :

* Logical Message Schemas,
* IT Components Catalog,
* IT Context Diagrams,
* State Machine Diagrams,
* Sequence Diagrams.

The intended audience of this document are all individuals involved in modeling the above artifacts :

* members of PSC (Passenger Services Conference) Standards work-groups developing or changing message schemas,
* PADIS DDSCC TAG (Technical Advisory Group) members developing or changing message schemas.

These individuals have a variety of profiles including Airline and IT supplier Business Analysts, Enterprise Architects, Data Modelers.

## Document Context

The Airline industry data model is to be published by IATA as a foundational layer for the development of airline messaging standards in XML or any other data format that may emerge in the future.

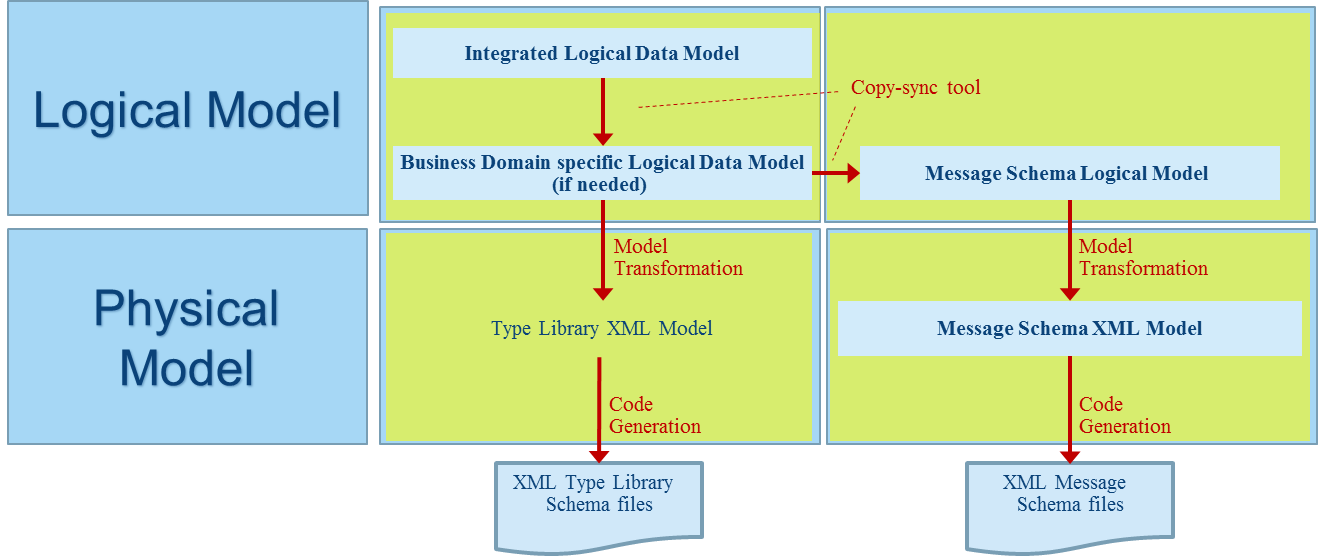
The data model is structured in 3 pillars (Business, Information, Technology), 4 layers (Contextual, Conceptual, Logical, Physical), and operational stakeholder views. A separate guideline document will exist for each of the 12 partitions defined by the pillar and layer.

The data model uses UML and as a tool Sparx Enterprise Architect (EA). The first 3 layers (amongst which “T3”) are platform-independent. The present guidelines will therefore be (partly) EA specific, but agnostic to the target messaging standard (e.g. XML).

# Overall Approach to Logical Message Modeling

## Logical Message Model – Relationship with other Models

|  |  |  |
| --- | --- | --- |
| The below diagram shows the steps from integrated LDM to Message schema, using the four partitions of the cube highlighted on the right.  The scope of the present document only includes the Logical Message Schema Model. The other steps are described in the I3 and T4 Guideline documents. |  |  |



\*\*\* yet to be completed for relationship with Business Messages, Sequence Diagrams, State Machine Diagrams \*\*\*

## Logical Message Model – Approach and Profile

\*\*\* section yet to be completed \*\*\*

# Modeling Logical Message Schemas

## Logical Message Schema Usage

The objective of the Logical Message Schema is to model the data contents of a specific message schema, in a platform-independent manner.

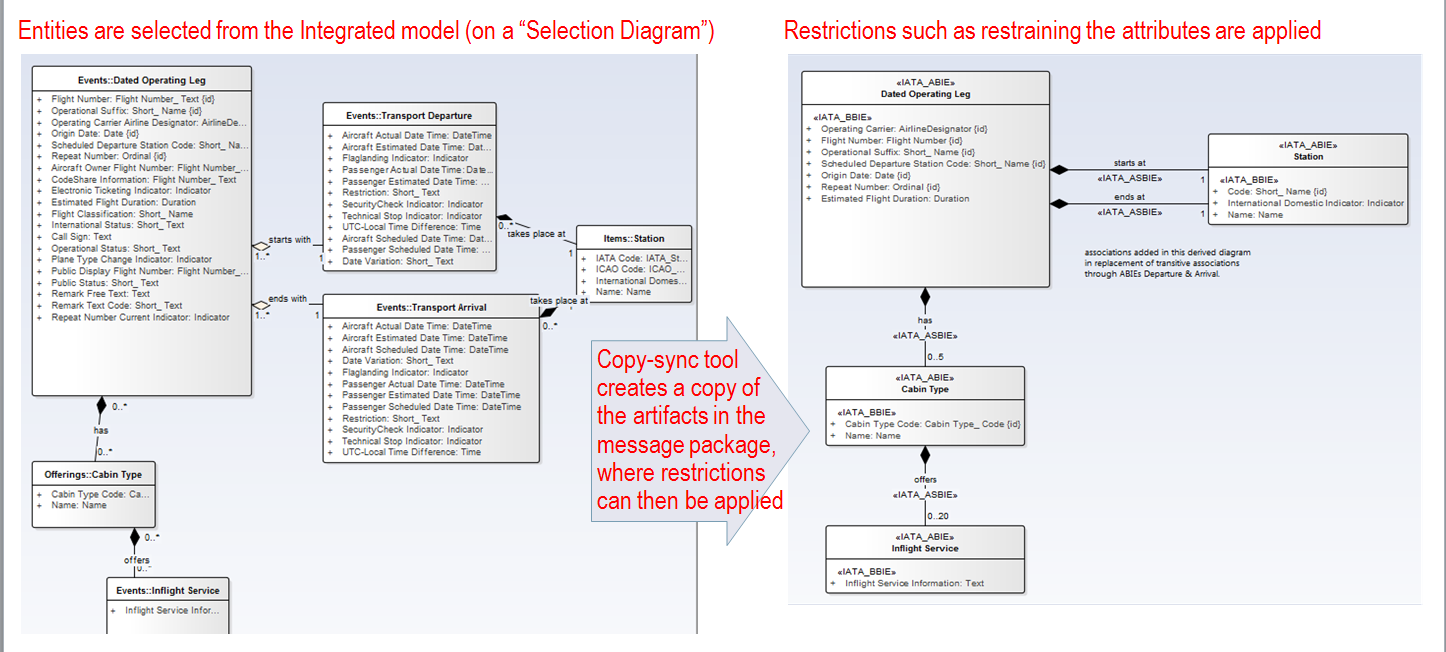
Depending on their nature and source, we can differentiate three types of data entities / ABIEs :

|  |  |  |
| --- | --- | --- |
| 1. ABIEs from the integrated or domain Logical Data Models ; this should be the vast majority of ABIEs | => | are reused from the I3 LDM,  with a copy in the message schema package being made by Synchronize |
| 1. ABIEs related to processing or message flow, and specific to this message, such as action codes etc. | => | are created directly in the Logical Message schema package |
| 1. Standard Message ABIEs which are similar to the above but common to many messages | => | are reused from the T3 / Governance View Standard Message Header library,  with a copy in the message schema package being made by Synchronize |

ABIEs of type 1 and 3 are first assembled on a “Message Selection Diagram”, referencing the source ABIEs from I3 or T3/Governance.

Running the Synchronize tool will then create copies of these ABIEs in the Message Schema package, with a “Derived Diagram” referencing those copies, and with the possibility to apply restrictions specific to the message.

Here below is a sample Selection Diagram and Derived Diagram. One of the restrictions applied on the Derived Diagram is that only a subset of the BBIEs has been kept, the others having been deleted from the Message Schema ABIEs.



If the source of the copies changes later on, the message schema model can be synchronized with   
the source, which can again be done by the Synchronize tool. Rules of synchronization when both the source and the message models may have changed are complex and require good understanding of what restrictions a modeler is allowed to apply in the message schema model, i.e. what changes s/he is allowed to make.

The next sections will therefore provide guidelines on three interdependent steps in natural sequence:

* Logical Message Schema Derivation from the LDM – Initial Copy,
* Logical Message Schema Properties (and what changes are allowed at this stage),
* Logical Message Schema Derivation from the LDM – Synchronization.

## BBIE Size Restrictions

The size or length of the allowable values for a BBIE are generally determined by the source model, which might be a Functional Domain Model or the Integrated Data Model. However, if there is no business constraint to the length of an ABIE, such as a Description, it is necessary for the Technology Logical Model to set an upper limit to the ABIE’s length. This is to address security issues where huge amounts of data are inserted as in a Denial of Service attack or to insert illicit program code, and to address issues of malformed XML.

Certain business data types have been ascribed a default length that the modeler must ensure are applied to associated BBIEs. The current default lengths are shown in the table below.

Table 1 - Maximum Allowable Lengths of Business Data Type

|  |  |
| --- | --- |
| Business Data Type | Maximum Length Allowed |
| Identifier | 35 |

## Logical Message Schema Derivation from the LDM – Initial

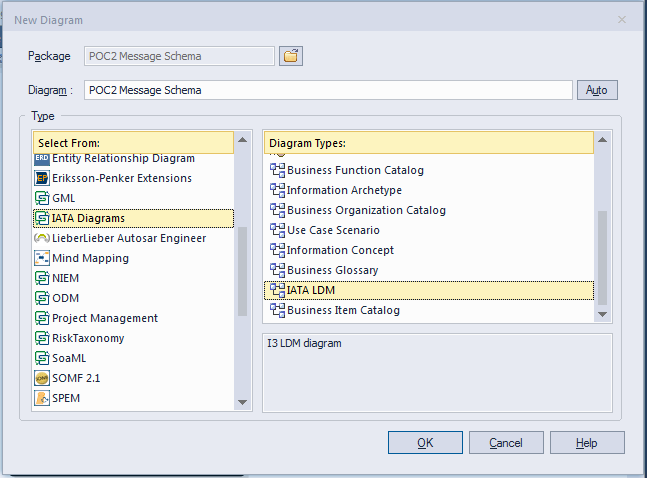
A Logical Message Schema Model is created in 3 big steps:

1. Create a Selection Diagram,
2. Run Synchronize to create a copy of the artifacts in the message schema package,
3. Apply restrictions and collapsing if needed.

This section describes how to perform 1 and 2.

***Creating a Selection Diagram***

Make sure you have created a package under T3 dedicated to the message schema model.

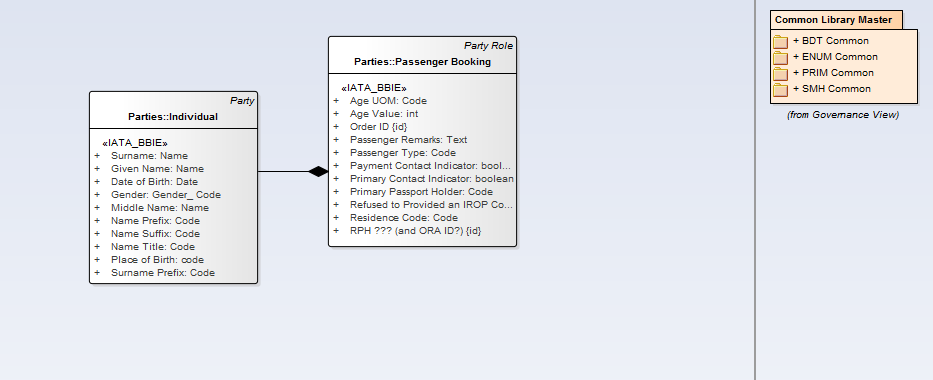


In this package, create an empty diagram of type IATA Logical Data Model : project browser :  
- top menu icon “New Diagram” /  
- Select from “IATA Diagrams”,  
- Diagram Type “IATA LDM”.

The name of the diagram  
must end with “- SELECTION”,   
or else the Synchronize tool will not recognize it.

Pull all ABIEs to be part of the message into the diagram, from their source location, i.e. normally from I3 (integrated or domain Logical Data Model) or T3/Governance (standard message ABIEs). An easy way to achieve this is often to open a diagram with the source ABIEs and drag and drop them into the Selection Diagram, with very importantly choosing the drop option “Create Link on this diagram”. Any associations that exist between the source ABIEs will automatically show as well. You may arrange the layout of the diagram, however must not alter the contents, since this would alter the underlying LDM.

Decide from what source package the BDTs, ENUMs, PRIMs should be copied. While ABIEs can   
be assembled from many source packages (e.g. from I3 Governance plus from a project’s work in progress package), the BDTs, ENUMs, PRIMs need to come from one Common Library package  
(e.g. from a project’s work in progress package). Drag and drop that package from the project browser into the Selection Diagram, using drop option “as a Package Element”.



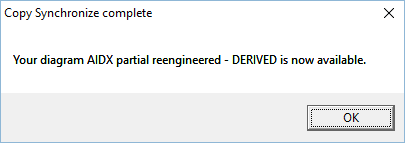
To the right is a sample Selection diagram with the Common Library package referenced from the right side.

There must be one and only one package, the Common Library, referenced from the selection diagram.

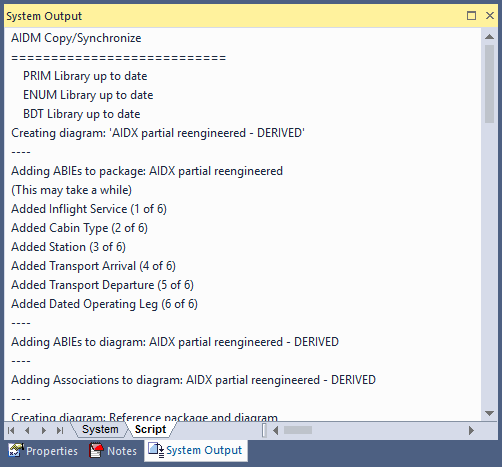
***Running Synchronize to create a copy of the artifacts in the message schema package***

Synchronize is a diagram-driven tool created specifically as part of the AIDM.

To run Synchronize :

1. Make sure you have locked the message schema package with all its contents
2. Select the schema package in the Project Browser, then select “Scripts” / “Synchronize”
3. Wait until you get the completion message on your screen (shown to the right). For a complex Message Schema, this can take   
   1 to 120 min depending on its size and on your workstation performance.   
   After having pressed “OK”, the Derived diagram created will automatically open.

Hints :

* Synchronize will display progress information (as will other AIDM tools) in the “System Output” window”. Hence it may be appropriate to have that window be part of your usual EA desktop configuration.   
  It can be opened from top menu “View”.

A “System Output” Window with sample contents is shown to the right.

* The “System Output” window will indicate stages and progress (e.g. “Creating common objects”, “creating derived diagram”, “adding ABIEs”), and issue any potential warnings at the end.
* In case of a Cloud connection error, the tool will bring up a small dialog with a message referring to the “cloud”. As most of these issues are intermittent, you can then typically press “OK”, and the tool will proceed and complete successfully.
* Any error message other than temporary cloud connection issues should be reported to the AIDM Administrator.

***Understanding what elements are created by Synchronize***

The initial run of Synchronize for a Message Schema model will create:

* BDT, ENUM, and PRIM Library packages underneath the Message Schema package,
* A copy in the Message Schema package of all ABIEs referenced from the Selection Diagram, and of their associations, with BBIE “type” links redirected to the BDTs in the above library,
* A “Derived Diagram” which is a copy of the Selection Diagram, with “- DERIVED” at the end of its name, and pointing to the copies of the ABIEs in the Message Schema package (with basic “box” auto-layout applied),
* A Dependency association between source and copied ABIEs (the association can only be seen in the traceability window)
* A “\*Reference\*” package underneath the Message Schema package, holding a copy of all source elements as of time of latest execution of Synchronize (i.e. initial Synchronize in this case).

## Logical Message Schema Properties – Restrictions and Collapsing

This section describes step 3 of the three steps mentioned earlier (1. Create a Selection Diagram,   
2. Run Synchronize to create a copy of the artifacts in the message schema package, 3. Apply restrictions and collapsing if needed).

Logical Message Schemas have the same types of artifacts as the integrated and domain LDMs  
(see I3 Guidelines). No new information entities or attributes must be created at this stage. The only changes to information entities allowed are additional restrictions to the model and – in some cases – collapsing. These changes are described in the present section. Besides this, new ABIEs related to message processing can be added, as described in section 3.5.

***Applying additional restrictions***

The list of allowed restrictions is :

* Removal of individual ABIEs, BBIEs, ASBIEs,
* Restrictions on the cardinality of the BBIEs and ASBIEs. Where possible restrictions should be applied to avoid unbounded cardinality i.e. replace unbounded (\*) with a real integer.
* Declaring BBIEs as mandatory, which is done through setting their cardinality to 1:1 (or 1:more than 1),
* Restrictions on the data type of a BIE (subset of value domain), through reference to a different data type, e.g. changing one BDT for another BDT with more restrictive facets defined.
* Restrictions on the concept of the ASBIEs or BBIEs though “tighter” usage rules (in Constraints Tab),
* Overrides to Notes : All BIE Descriptions as well as ASBIE Role Descriptions,
* Potential change to “IATA Reference to Source” in a BBIE (e.g. if we would now like to reference a usage in a domain-specific IATA resolution),
* Overrides to Status or Author.

***Collapsing***

Associated ABIEs complementing their associating ABIE will become – when transformed into XSD – nested Complex Types. Chains of multiple associations can translate into many levels of nesting. The possibility to collapse allows modelers to simplify XML schemas and documents by collapsing a hierarchy of Complex Types into a single Complex Type, when in the context of the message schema there is only one occurrence needed of each collapsed complex type. While these considerations are somewhat XML specific, it was determined that the easiest approach to model for this is in the T3 message models.

Here below are two example hierarchies where in certain messages one may want to collapse :

* City, Country Sub-Division, and Country into Postal Address,
* Dated Marketing Segment, Dated Operating Segment, and Carrier into Passenger Segment.

|  |  |
| --- | --- |
|  |  |

Looking at the above example, to collapse City into Postal Address, the modeler will “move” the BBIEs from City into Postal Address, then delete the ABIE City from the Message Model package. Next, he can do the same for Country Sub-Division and Country. So far, these are native EA features. Having the possibility to collapse very concretely means that these “move” and “delete’s are authorized and that upon next synchronization with the domain model from which the message model is derived, the Synchronize tool will keep synchronizing the BBIEs despite of their changed location, and will not recreate the deleted ABIE.

There are a number of supported scenarios around the “move” :

* If not all BBIEs of the collapsed ABIE are needed in the associating ABIE, then only a subset may be moved, the remaining ones being typically deleted along with their collapsed ABIE being deleted.
* One special case of the previous point is moving only identifier fields, which results in the associating ABIE having a foreign key reference.
* If the collapsed ABIE is the associating ABIE to a third ABIE (e.g. City to Country Sub-Division), the modeler would typically also collapse this third and any potential further ABIEs in the hierarchy.
* If the associated ABIE – say City – is to be collapsed into the associating ABIE (Postal Address), but in the same message model there is yet another ABIE (say OriginDestination) that associates City but should not have the City BBIEs collapsed into it, then one would obviously keep City with all its BBIEs, and rather than “moving” these BBIEs into Postal Address “copy’ them into Postal Address. These BBIEs will then have multiple copies in the message model that all get synchronized with their source in the domain model, during future Synchronize’s. In such a case the modeler should logically delete any ASBIEs no longer needed, in our example the ASBIE from Postal Address to City.

“Moving” and “Copying” BBIEs in EA can be achieved as follows :

* Copy a BBIE by drag-and-drop from ABIE in project browser towards ABIE in diagram.
* Move a BBIE either through drag-and-drop from old ABIE in project browser to new ABIE in project browser, or : this being less graphic than Copying as per previous point, you can achieve the same result by copying the BBIE into the new place then deleting it from the old place.

As part of collapsing, the following manual actions generally allowed in message model will be particularly useful :

* Change name of a BBIE adding a prefix (e.g. when a BBIE copied in from another BBIE has the same name as one of the pre-existing BBIEs, or would not have a self-explanatory name any more outside the context of its natural ABIE),
* Adjust the sequence of BBIEs within an ABIE as desired (e.g. after having copied or moved additional BBIEs from associated ABIEs),
* Change the name of an ABIE (e.g. if after having received BBIEs from collapsed ABIE, its scope it felt to be larger).

Related to Collapsing, the following rules must be observed :

* BBIEs can only be moved or copied into an ABIE that has a direct ASBIE or a chain of ASBIEs towards the original ABIE. The direction of ASBIEs does not matter, however each of the target cardinalities going from new ABIE to original ABIE should not exceed “1”. Note that Synchronize will not notice potential infringements to this rule, however compliance check will control the existence and visibility of the ASBIE(s) on the message model Selection Diagram (pointing to the domain model).
* When a collapsed ABIE is deleted from the message model, its source counterpart in the domain model must still be referenced from the message model Selection Diagram. If not present, the next Synchronize would consider BBIEs – even in their new ABIEs – removed from the scope of the message and delete them.

*Caution :*

Collapsing should only be used when strongly needed, as it presents a number of drawbacks :

* Makes it harder to understand what domain model components the complex types in a message model are derived from. By the same token makes it harder to change these components if needed, as they are maintained though the domain model.
* Breaks the principle of UN/CEFACT that one defines semantically meaningful “Core Components” in the domain model which will be further restricted but preserved as such in  
  the message models. Collapsing will amalgamate the semantic meaning of the components.
* Collapsing leads to complex synchronization (Synchronize) scenarios and eventually to some limitations explained in the last paragraph (special scenario details worthwhile to be aware of) of the next section (Synchronization).

***Summary of possible changes to the derived model***

In addition to the changes listed with restricting and collapsing, it is also allowed to :

* Add ABIEs by copy-pasting derived ABIEs (with the goal of then applying different restrictions). The additional copies will subsequently be synchronized against the same source as the copy they were made from.
* Rename ABIEs.
* Add new ABIEs related to processing or message flow, and specific to this message, such as action codes etc.
* Redirect or add ASBIEs.
* Change the order of BBIEs within an ABIE (adjusting visual sequence in project browser / diagrams).

The following table recaps all possible changes in the derived model :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Restrict** | **Move to other ABIE** | **Copy** | **Delete** | **Rename** | **Redirect** | **Add new** |
| ABIE |  |  | **✓** | **✓** | **✓** |  | **✓** |
| BBIE | **✓** | **✓** (along ASBIEs) | **✓** | **✓** | **✓** |  |  |
| ASBIE | **✓** |  |  | **✓** | **✓** | **✓** | **✓** |

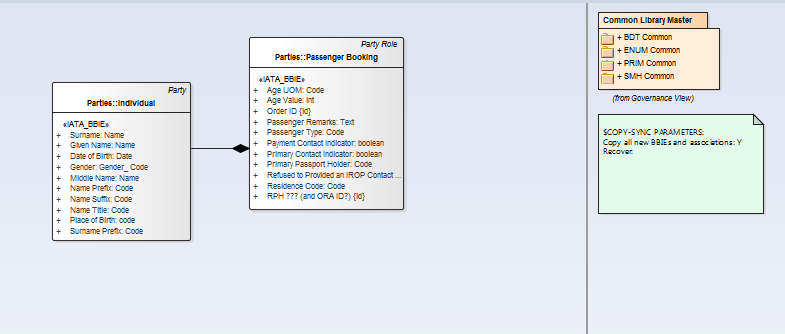
## Logical Message Schema Derivation from the LDM – Synchronization

In case of changes in the source data models (integrated or domain data models, standard message ABIEs), the Synchronize tool can be executed again to cascade these changes into the message model.

Also, in case one or more ABIEs need to be added to the message schema, these ABIEs can be added to the Selection Diagram, which will trigger creation of their copies in the message schema package and addition to the Derived Diagram upon next execution of Synchronize. For instructions on how to populate a Selection Diagram, refer back to section “Logical Message Schema Derivation from the LDM - Initial”.

***Running Synchronize to synchronize post-initial-run***

Before executing Synchronize again, you should check appropriateness of the “Parameter” note box added automatically to the Selection diagram by the initial execution of Synchronize. Here below is an example showing the parameter note box in green.



The parameter note box is in free text format but must contain the exact syntax for parameters :

|  |  |  |
| --- | --- | --- |
| **Line** | **Possible occurrences** **of such line** | **Explanation** |
| $**COPY-SYNC PARAMETERS:**  **Copy all new BBIEs and associations: Y**  **Recover:**  **<<ABIE>>ABIEname1**  **<<ABIE>>ABIEname2<<BBIE>>BBIEname**  **<<ABIE>>ABIEname3<<ASBIE>>ASBIEname** | 1  1  1  0 … multiple  0 … multiple  0 … multiple | *n/a*  put Y or N to copy BBIEs and associations created in source since last Copy/Sync  *n/a*  Recover an ABIE previously deleted from the message schema model  Recover a BBIE previously deleted from the message schema model (within its ABIE)  Recover an ASBIE previously deleted, with a certain name, under an associating ABIE |

To run Synchronize :

1. Make sure the Parameter note box has the values desired.
2. Proceed as described in section “Logical Message Schema Derivation from the LDM - Initial“.   
   Synchronize will automatically detect that this is a synchronization run as opposed to initial run.
3. Watch for any warning messages in the System Output window.

***Understanding the results of Synchronize synchronization***

The synchronization run of Synchronize for a Message Schema will :

* Refresh the BDT, ENUM and PRIM libraries with the latest version from their source,
* Copy any ABIEs newly added in the Selection Diagram,
* Append “TO BE DELETED” to the name of any ABIEs that have been removed from the source model or Selection Diagram,
* Delete from the Message Schema model any ASBIEs that have been removed from the source model or Selection Diagram,
* Copy any BBIEs and Associations newly created in the source models if the first parameter in the parameter note box is set to “Y”,
* Delete from the Message Schema model any BBIEs that have been removed from the source model,
* Copy any ABIEs, BBIEs, ASBIEs specified in the Recovery section of the Parameter note box,
* Align the Message Schema model with the source model where :
  + The differing property *is not* a property allowed to be restricted in a message model, i.e. not part of the list in section “Logical Message Schema Properties”,
  + or the differing property *is* a property allowed to be restricted, but the value in the Message model was either unchanged by the user (i.e. same as in the reference copy) or is not a valid restriction (e.g. a cardinality greater than in the source model),
* Update Dependency associations between source and copied ABIEs as appropriate,
* Set the EA “Derived Flag” in the BBIE – so a “/” is displayed in the diagram in front of the name of the BBIE – for any BBIEs located in a derived model ABIE different from the source model ABIE (see previous section § Collapsing) ; this is to visually indicate and remember which BBIEs originally come from another ABIE.
* Apply yellow-green color on the derived diagram to all ABIEs that have been created manually in the Message Schema model, without having been copied from an underlying source. This will allow the user to verify that they indeed relate to processing or message flow, hence do not need to be synchronized against any source,
* Refresh the \*Reference\* package underneath the Message Schema package.

Note that Synchronize also issues a comprehensive set of information and warning messages, especially in case of deletions made or invalid changes encountered and overridden.

Finally, here are some special scenario details worthwhile to be aware of :

* If a BBIE is copied into a different ABIE, and both copies are kept in the derived model, then very logically both copies are synchronized in future Synchronize’s.
* When a BBIE is moved to a different ABIE in the source model, the next Synchronize will process the “move” as “delete” + “add”. This implies several limitations :
  + Potential restrictions previously added in the derived model need to be re-applied,
  + If in the derived model, the BBIE had been moved (as part of collapsing) into another ABIE (independently and in parallel with the domain model change), it will now be re-added in the derived ABIE corresponding to the new source ABIE – provided that ABIE exists in the derived model,
  + If in the derived model, the BBIE has been cloned into multiple copies, all copies will be deleted (since their source was removed) and only one new derived BBIE will be re-created, as per previous point.

## Logical Message BIE details – Types of ABIEs

*Note : this section is an initial version of guidelines, to be enhanced as we gain more experience.*

The approach to Logical Message ABIEs is partly inspired from UN/CEFACT Core Components Business Document Assemblies which define :

* Message header artifacts (Business Document / Envelope, Business Document Header, Message Assembly)
* Message BIEs derived from the information model BIEs.

Depending on their nature and source, the AIDM differentiates three types of data entities / ABIEs. The following table – which was already introduced in section 3.1 Logical Message Schema Usage – recalls the three types, before the rest of this section provides detailed guidelines on their BIEs.

|  |  |  |
| --- | --- | --- |
| 1. ABIEs derived from the integrated or domain Logical Data Models ; this should be the vast majority of ABIEs | => | are reused from the I3 LDM,  with a copy in the message schema package being made by Synchronize |
| 1. ABIEs related to processing a specific message | => | are created directly in the Logical Message schema package |
| 1. Standard Message ABIEs which are similar to the above but common to many messages | => | are reused from the T3 / Governance View Standard Message Header library,  with a copy in the message schema package being made by Synchronize |

***Type 2 & 3 - ABIEs qualifying as message ABIEs***

These ABIEs – originally modeled in T3 as opposed to I3 – should relate to message processing, message flow, message headers, message action or response codes, or message-specific grouping of information entities (e.g. data list headers).

Examples include :

\*\*\* tbc \*\*\*

These ABIEs should *not* include :

* Information entities reflecting the core Business Pillar concepts (Process, Business Function, Business Action, Actor, Organization, Location, Event), which should be modeled in I3
* Any true business information BBIE, especially one that would need to be persistent and carried across messages, which again should be modeled in I3.

The idea is to have a relatively simple structure for modeling in T3, comprising message-process specific ABIEs and common message ABIEs that could be useful across all messages and domains.

The above Business Pillar concepts would probably be shared across messages of a business domain and would already be modeled in I3.

Message ABIEs (type 2 and type 3) can have – within the message model – associations with Type 1 ABIEs Synchronizeh’ed from I3. I3 integrated and domain models must not reference message ABIEs.

***Type 3 - ABIEs qualifying as standard message ABIEs***

Any of the above ABIEs deemed to be reusable across messages of multiple business domains should be categorized as standard message ABIEs.

They can be modeled under T3 Operational View, and when finalized promoted to T3 Governance View. Exactly as for I3, the T3 Governance View is under control of the AIDM Administrator.

ABIEs in the T3 Governance View cannot have associations with any ABIEs outside the T3 Governance View.

***Type 2 & 3 – BDTs & ENUMs***

BDTs and ENUMs specifically needed for type 2 ABIEs are created directly in each T3 message package, adding them to the BDTs and ENUMs created through Synchronize.

BDTs and ENUMs needed for type 3 ABIEs are defined in I3 Governance. Exactly as for BDTs and ENUMs needed for the I3 models :

* They will be originally modeled under the first project’s I3 Gov Work in Progress Package.
* Once in I3 Governance, I3 Operational View domain models still need to have a copy. In this way, the BDTs and ENUMs can be Synchronize’ed into the message model together with the other I3 BDTs.
* If needed, they can be replaced in the message model by more restrictive BDTs and ENUMs.

***Type 2 & 3 – Quality and Compliance Checks***

Before releasing Type 2 ABIEs as a standard, or before promoting Type 3 ABIEs to T3 Governance View, a compliance check will be performed similar to I3, except that :

* no I1 and I2 items (terms, taxonomies, concepts) will typically exist for T3 message ABIEs,
* no shared identification BBIEs are expected to be identified for T3 message ABIEs,
* rules on derivation are irrelevant for type 2 ABIEs, which are not derived from any previous source ABIEs.

Type 3 ABIEs can receive restrictions in the message model vs the T3 Governance View, exactly as ABIEs sourced from I3. We are not yet sure at this stage if the possibility to apply restrictions to type 3 ABIEs is actually needed.

## Logical Message Schema Rules and Quality Assurance

\*\*\* section yet to be completed \*\*\*

## Logical Message Schema Diagrams

\*\*\* section yet to be completed \*\*\*

## Logical Message Schema Package Structure

\*\*\* section yet to be completed \*\*\*

## Logical Message Schema Forward Engineering

\*\*\* section yet to be completed \*\*\*

# IT Component Catalog

\*\*\* section yet to be completed \*\*\*

# IT Context Diagram

\*\*\* section yet to be completed \*\*\*

# State Machine Diagram

\*\*\* section yet to be completed \*\*\*

# Sequence Diagram

\*\*\* section yet to be completed \*\*\*

# Aspects to be refined during initial modeling

\*\*\* section yet to be completed \*\*\*

# Annexes

## Annex A : Glossary of this document

|  |  |
| --- | --- |
| Term | Description |
| ABIE | Aggregated Business Information Entity \* |
| ASBIE | Association Business Information Entity \* |
| BBIE | Basic Business Information Entities \* |
| BIE | Business Information Entity : includes all of ABIE, BBIE, ASBIE |
| BDT | Business Data Type \* |
| Core Components | Concept defined by UN/CEFACT. Core Components are the syntax-neutral and technology-independent building blocks that can be used for data modeling as well as / interoperations message modeling. |
| EA | Sparx Enterprise Architect (the software product we use for modeling) |
| ENUM | Enumeration \* |
| LDM | Logical Data Model |
| PIM | Platform Independent Model |
| PRIM | Primitive Data Type \* |
| Tagged Value | “Custom” properties added to the standard set of information stored with each artifact in EA, either through adding a certain “profile” (e.g. UPCC) or defined and added by the data model team. |
| UN/CEFACT | United Nations Centre for Trade Facilitation and Electronic Business |
| UPCC | UML profile for UN/CEFACT Core Components |

\* See section “2.2 Usage of UN/CEFACT Core Components UML Profile” for artifact definitions.