

Table of Contents

Sketch 5.5

User Documentation

Revisions

Copyrights and Legal Notice

Preface

1. Introducing Sketch

2. Getting Started

3. Logical Model

4. The Decision Table

5. House Style Templates

6. Data Structure

7. Sketch-specific Building Blocks

8. Fragments

9. Adding logic Conditions

10. Adding logic Loop

11. Adding complexity Nested conditions, loops and fragments

12. Copy Titles

13. Preview your dynamic document

14. Import and Export

15. Using variables in charts

16. Writer

17. Responsive Email

18. Tips and Tricks

19. FAQ

ConnectR REST API Documentation

Revisions

Description

Authorization

Translation File Access

List Templates

Template Details

List Email Templates

Email Template Details

Create Request

[Get Information about Process](#)

[List Documents](#)

[Get Document](#)

[List Tasks](#)

[Get Task File](#)

[Delete Request](#)

[XML Schema for Generation Request](#)

[XML Schema for Decision Table Structure](#)

[XML Schema for Language Definitions](#)

[XML Schema for Template List Details](#)

Sketch 5.5

In this section you'll find the following documentation about Sketch 5.5:

- [User Documentation](#)
- [API Documentation](#)

User Documentation

This section contains the User Documentation of Sketch 5.5.

Revisions

DATE	OWNER	TOPIC
2012-03-23	NTA	Document creation
2012-04-13	THH	First review
2014-03-13	NTA	Document creation
2014-07-31	NVA	Rebranding Connective
2015-02-24	NVA	Update release Sketch 3.3
2015-08-19	NVA	Update release Sketch 4.0
2017-09-20	DGI	Update release Sketch 5.2
2018-02-26	DGI	Update release Sketch 5.4
2018-04-05	DGI	Update release Sketch 5.5
2018-08-22	DGI	Removed overview
2018-10-22	DGI	Correction responsive email

Copyright and Legal Notice

5.5.1-22082018-01

This documentation is provided for informational purposes only, and Connective and its suppliers make no warranties, either express or implied, in this documentation. Information in this documentation, including URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results of the use of this documentation remains with the user.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this documentation may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Connective.

© 2018 Connective. All rights reserved.

Any additional company and product names mentioned in this documentation may be trademarked and/or registered as trademarks. Mention of third-party products constitutes neither an endorsement nor a recommendation. Connective assumes no responsibility with regard to the performance or use of these products.

Preface

This document describes the functionality and terminology of the **Sketch** Document Designer module. Sketch is a component of the Connective Suite.

The **Connective Suite** is a flexible, process-driven and service-oriented application suite for the creation and handling of customized business documents in a multi-platform environment. The flexible approach guarantees a fast time-to-market and the ability to generate various volumes of documents in varying complexity without modifying the existing IT infrastructure.

Sketch allows you to build dynamic models in a Word processing environment. The templates created in Sketch contain an internal logic, giving them the flexibility to generate customized documents with dynamic content, based on provided data. Sketch also lets you preview the result of one or more generated samples of the model design.

1. Introducing Sketch

This chapter explains the application and introduces some elementary concepts and terminology. If you are already familiar with this and want to go ahead, you may skip this chapter.

1.1 What is Sketch?

1.2 The Sketch Graphical User Interface

1.3 Terminology

1.1 What is Sketch?

Sketch is the document designer module of the **Connective Suite**.

The Connective Suite is an application suite for the creation and handling of customized business documents.

In this application suite, various applications execute different tasks. The actual generation of documents is done by the following applications:

- **Sketch**: allows you to create and modify document templates containing text and internal logic.
- **Generator**: produces complex and highly personalized documents at lightning speed, using the templates created in Sketch and provided data.

The system is based on templates and data. Note that templates are called "models" in Sketch.

A **model** is a set of Word processing documents with text combined with the provided variable data. A model also contains the necessary logic to print text portions based on certain data. All text can be formatted to give a coherent layout throughout the document.

The **data** actually decides which parts of the text will be printed in different scenarios. This data is provided at generation time. Example: in some cases the variable FirstName is replaced by "Kristien", in other cases it is replaced by "Stijn".

Sketch makes use of **LibreOffice Writer**, giving you all the possibilities associated with this full-featured Word processing application. This means you can create any document, from simple to complex, with text and graphical elements, tables and columns, inserted objects, document references etc. using the available formatting and styles.

Sketch allows business people to apply legal or marketing driven modifications to documents for instance. No comprehensive IT knowledge or background is necessary to implement those modifications. This makes the whole process fast and flexible for any organization.

The graphical user interface (**GUI**) can individually be set up in the language of each author.

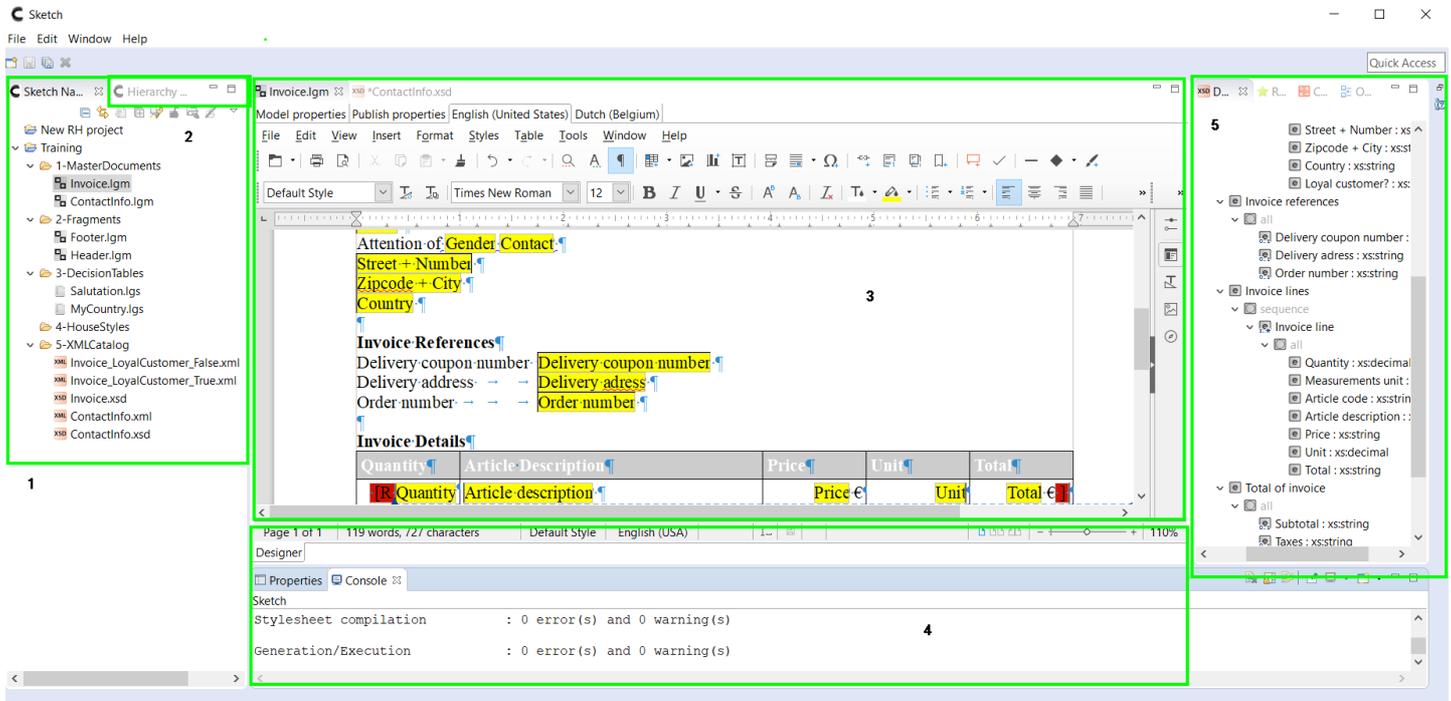
Documents can be created in the languages of your choice. Documents created in different languages will nicely coexist to offer a flawless international marketing approach.

Based on the **Open Document Format** (ODF), the generated documents can easily be passed on to be converted, sent as attachments, printed or archived.

1.2 The Sketch Graphical User Interface

The Sketch application is divided into 4 different regions. Each region allows you to perform specific tasks.

1. Sketch Navigator/ModelStore
2. Hierarchy Navigator
3. Editor region, with Model Editor
4. Information region
5. Objects palette



Schematic overview of the different regions in Sketch

These regions contain tabs with items or information. The tabs can be detached from their region, and/or dragged to any other region on the screen. You can even move or size the regions themselves. All this is to help you define your ideal way to work with Sketch.

We will have a look at those regions as they appear in their default settings.

1.2.1 Sketch Navigator

The Navigator region at the left hand side of the screen gives an overview of all the ModelStore Connections, Projects, models, house styles, decision tables, data structures etc. an author has access to.

Once the user logs in to Sketch, the workspace is displayed. The author can now connect to one or several ModelStores. By clicking on the nodes of the different ModelStores, the author can expand the underlying tree view structure. The Navigator can be seen as a library in which all document templates (or models) are stored.

The different items that are available in the tree view are explained in [1.3. Terminology](#). For more information about the Navigator, see [2.5. Browsing through the Workspace](#).

1.2.2 Hierarchy Navigator

The Hierarchy Navigator provides a different look on the ModelStore. In the Hierarchy Navigator you can easily see which files are used in which model. To work quickly and efficiently, it is important for the authors to get an overview of the linked models, xsds/xmls, house styles and decision tables. Authors using Sketch need an overview of the elements - fragments, decision tables, etc. - that are used in a template model.

The Hierarchy Navigator is displayed as a tree with each block representing a node in the hierarchy. The parental node is the

ModelStore with the list of model templates. Each model template can be expanded, visualizing the underlying blocks to the user. By default, only the first node “ModelStore” will be expanded. The view represents bidirectional references of each element in a group, meaning that also the inverted reference is presented in this view.

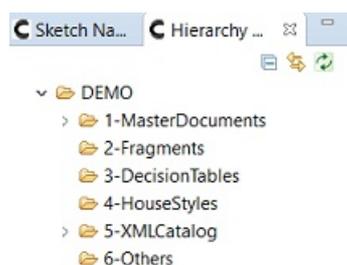
A block represents the following items:

- A group of **models** (.lgm)
- A group of used **fragments** (.lgm)
- A group of **decision tables** (.lgs) with a group of models in which they are used
- A group of **house styles** (.ott) with a group of models in which they are used
- A group of **Writer** files (.lgw) with a group of models in which they are used and with a group of Textblocks used in LetterWriter Files (.lgb)
- A group of **XSDs** (.xsd) with a group of models in which they are used
- A group of **XMLs** (.xml) with a group of models in which they are used
- A group of **responsive emails** (.cem) with a group of models in which they are used

When a user navigates to the Hierarchy Navigator for the first time, the view will be populated for the first time and depending on the number of models and reusability of the underlying elements, this action will take some time.

The user can refresh the view via a refresh button in the Sketch toolbar. This refresh will trigger an update of the view based on the changes done in the model templates.

Example of the Hierarchy Navigator:



1.2.3 Editor region

The Editor region, taking the main part of the screen, is the heart of Sketch. In the Editor region the author creates, modifies and visualizes models, house styles, decision tables, data structures etc. When a language model is opened (to view or modify), the LibreOffice Writer application is opened in this area.

For every item a user is working on, the item’s descriptive name and language is shown in a window tab on top. The active item is shown with dedicated menu and toolbars.

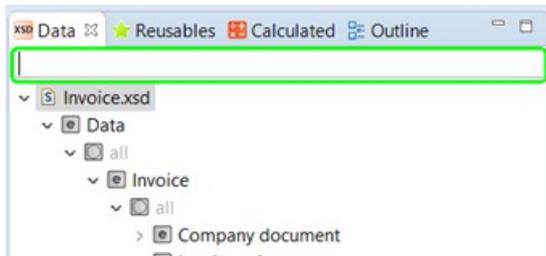
When a model is opened for modification, the author can use the functionalities of LibreOffice Writer to build his document, like fixed text, graphical objects, sections, columns, tables, header, footer, etc. Apart from these standard LibreOffice functionalities, the author can also make use of the specific Sketch functionalities: he can insert variables from the data structure, add logic to the text, making his document flexible and dynamic.

1.2.4 Objects Palette

The objects palette is located on the right side of the screen. While working on models, the available building blocks (variables, conditions, loops ...) appear under the respective tabs and can be inserted in the document.

By default, the **Data** tab is opened. This is a representation of the data structure (XSD) linked to the model. An XSD (short for XML Schema Definition) file defines the structure of the data that will be delivered in the XML format from a specific application to the Generator, in order to generate the document.

Tip: to quickly find elements within an XSD, use the search field beneath the **Data** tab.



1.2.5 Information region

At the bottom of the screen, just below the Editor region, you'll find the Information region. By default, two tabs are available: the Console and the Properties tabs.

- The **Console** tab provides technical information on internal processes running while Sketch is being used. It should help a technician when troubleshooting the program (in case of bad networking connection for instance).
- The **Properties** tab provides information about an item that is selected by the user in the Navigator, Objects Palette or in the Editor. This works for projects, connections, folders, models, house style templates, decision tables or any other files in the Navigator region, or variables, conditions and loops in the Objects Palette or in the Model.

1.3 Terminology

To understand the Sketch application, every author should comprehend the meaning of a few frequently used concepts. Read this carefully, it will definitely help you through a successful exploration of Sketch.

[1.3.1 Workspace](#)

[1.3.2 Project](#)

[1.3.3 Stage](#)

[1.3.4 ModelStore](#)

[1.3.5 PublishStore](#)

[1.3.6 Model](#)

[1.3.7 Fragment](#)

[1.3.8 As is \(no merge\) – Inherit styles \(merge\) – Integrate with context style](#)

[1.3.9 Check out - Save - Check in](#)

[1.3.10 House Style \(Template\)](#)

[1.3.11 Decision table](#)

[1.3.12 Working with data](#)

[1.3.13 XML and XSD \(Data structure\) documents](#)

1.3.1 Workspace

This is the working environment of the Sketch user. More specifically, it is the top level of the Navigator, the collection of all the ModelStores and projects.

1.3.2 Project

 The workspace can be structured by using projects. Within Sketch, a project is a collection of files that are created together to achieve a particular aim or end product. A separate project could be made for everything that concerns a certain company department, a product line or an actual project.

A project always is a top level node. All the files that reside in a project are saved locally on the computer where they were created. The purpose is to be able to create them offline, and to move or copy them to the server when appropriate.

By expanding a project node, the structure of the project is displayed in the tree view. Each project can contain folders, models, house styles, decision tables, XML schemas, XML documents and other files.

1.3.3 Stage

Sketch can work in a system of stages. A stage represents the development status of a document. Each user can be given different roles for different stages. This means that a user may or may not have the right to perform certain tasks, like View, Create, Copy, Delete, Promote, etc.

1.3.4 ModelStore

 The workspace contains one or more ModelStores, which are a special type of projects. A ModelStore is a collection of all the files that are stored on one server.

Normally there is one ModelStore per development stage. In a four stages development system (DTAP), you would have a separate ModelStore for development, test, acceptance and production. Models – as well as house styles, decision tables, XML schemas, XML documents and other files – that have been created in the Development ModelStore, can be promoted to a higher stage, like Test, then Acceptance, to finally be set into Production.

A ModelStore is the top level node of the workspace. All the files in a ModelStore are saved on a server. A user can only access the files in a ModelStore when his computer is connected to the server of that ModelStore.

By expanding a project node, the structure of the project is displayed in the tree view. Each project can contain folders, models, house styles, decision tables, XML schemas, XML documents and other files.

1.3.5 PublishStore

 A PublishStore is a collection of files that are published on a specific generation server.

When a model is ready to be used in production, you can publish this model to a PublishStore. This means that the template will be pre-compiled and stored in that PublishStore. This version of your template will then be used for the generation.

This enables authors to modify templates without having any impact on the documents used for generation in production. When the author has thoroughly tested his newly created or updated template, he can publish the version and only at that moment it will be used at generation time in production.

The workspace can contain one or more PublishStores, so the user can connect to different PublishStores in his workspace.

Individual models, as well as entire folders with models can be published at once.

A user can only publish or access the files in a PublishStore when the computer is connected to the server they are on.

1.3.6 Model

A template is a generic name that can be used for at least two types of documents. That's why the files that are created in Sketch, containing the final document's fixed content along with the variable content and the logic, are called models.

As they can be created in different languages sometimes we make the distinction between a language model and a logical model. A language model refers to a document in a specific language. A logical model is a set containing all language versions of a template, along with the model properties.

When not specified, you should find out which model is meant from the context. Otherwise we assume it is the logical model (which contains one or more language models).

Logical Model

The logical model is a central reference to a business document. It contains information shared or used by all the embedded language models. The logical model properties also include 'Descriptions' for each of the available languages. Logical models consist of a file with different language models. This file will be identified based on a unique file name.

If the model is intended to work with data you should also attach the data structure, one or more sample data files as well as decision tables.

Language Model

The language model represents the actual document in a specific language.

When a language model is added, viewed or checked out, it is shown in the model editor. Simultaneously the Objects palette on the right hand side of the editor populates the Data view among other views.

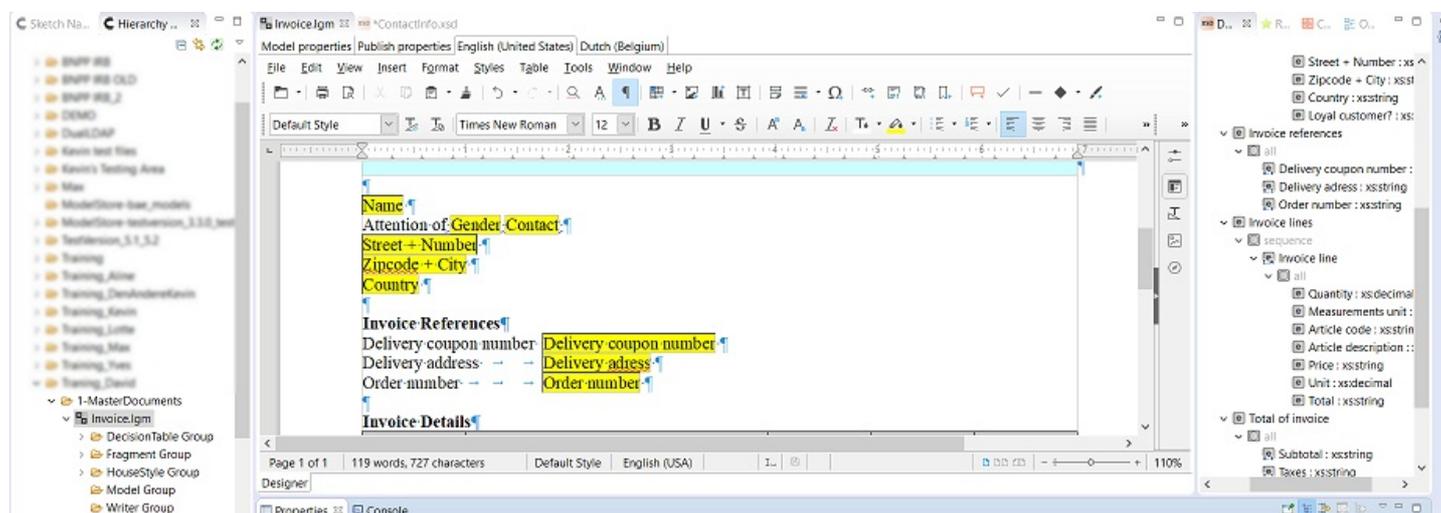
When modifying a language model in the Model editor, the Word processing functionalities of LibreOffice Writer are available. Moreover, the author can add variables and the like using the Objects palette on the right. Finally, logic – like conditions and loops – can be implemented using the attached data structure. By doing all this, the template becomes flexible.

Together with one of the attached sample XMLs, a preview of the personalized document can be generated.

Each language model can be found in multiple ModelStores. Every ModelStore can contain one language model per language that is available. Dutch, French, English, Italian, Spanish, German, Turkish, Czech, Slovak, Portuguese and Arabic are default languages in Sketch, but more languages can be implemented easily.

Note: Arabic is supported as Language Model, but advanced properties such as variable formatting (e.g. number to text) are not supported in Arabic.

Example of language Model



1.3.7 Fragment

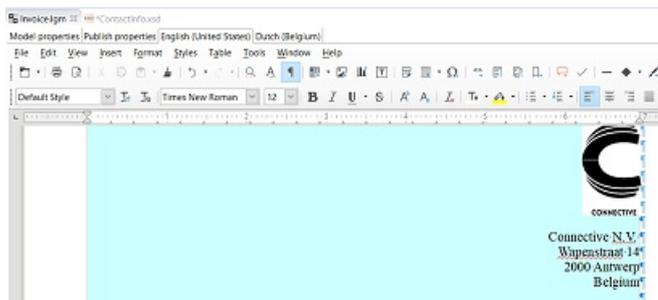
Every model can be 'reusable'. This means that you can insert it in other models. This child document is called a fragment. Reusable means that, once created, a document can be inserted in other documents over and over again. The main advantage of working with reusable fragments is that maintenance only needs to be done in 1 model, instead of in several models. Example: if a telephone number changes in a company footer, it must be adapted on one single location instead of in all the templates where the fragment is used.

A fragment can contain one or more paragraphs, sentences or even characters, tables, logos, signatures or any other element. Like any other model, a fragment can also contain variables, conditions and loops. These can be used as such or in any combination and with intended formatting.

A fragment is indicated by a blue background.

Note: responsive email models can be only used as fragments inside other responsive email models.

Example of fragment, inserted in a language model



1.3.8 As is (no merge) – Inherit styles (merge) – Integrate with context style

A fragment – a language model inserted in another language model – can inherit the style properties of the main document or the fragment can keep its own formatting.

There are 3 formatting options in Sketch:

- **As is (no merge):** the fragment keeps its own formatting and styles definitions, after it has been inserted in the main document.
- **Inherit styles (merge):** the fragment inherits the style properties of the main document, but still keeps its own specifically applied formatting.
- **Integrate with context style:** the formatting and style definitions of the fragment will be replaced by the formatting and style definitions of the previous paragraph, after being inserted in the main model.

1.3.9 Check out - Save - Check in

All models (and their associated files) belonging to a ModelStore are stored in a central library. A ModelStore provides revision control, which is a way of managing changes to documents over multiple releases. To prevent concurrent access problems, a file has to be locked so that only one user at a time has write access to the central copy of that file. It has to be checked out whenever an author wants to make some changes. This means that other users can read the file, but no one else can change that file until the author checks in the updated version (or cancels the checkout with the uncheckout action).

The author who has checked out the model sees a blue circle in front of the model name, showing that he has the permission to make modifications into the document. All other users see a 'lock' icon in front of that model name, showing the model is subject to changes and that they can temporarily only view it.

The user who has checked out the selected model is listed in the **Properties** tab in the lower panel.

When the author makes changes in the model and does a Save there is a red circle in front of the model name.

ICON	DESCRIPTION
	Model is checked out by the user, but no modifications have been saved.
	Model is locked. This means that the model is checked out by another user.
	Model is checked out by the user, and the modifications have been saved locally.
	Model is checked in.

After the modifications, the author checks in the model. The library tree refreshes and the icons are updated. Each user who opens the model now sees the updated version.

If user A already had a language model open for reading before that model was edited by user B, the changes made by user B are not visible to user A as long as he did not close and reopen the model.

The same goes for delete by user B. As long as user A does not close the model, the model stays visible on his screen even after user B has deleted that model.

If user A wants to delete a model that was already deleted by user B, then user A receives a message "The model 'model description' does not exist anymore". The time gap between the delete action of user A and the delete action of user B does not matter.

Even if user B is one millisecond faster, user A receives the above mentioned message. This means that user A and user B can have the same popup window in their screen: "Are you sure you want to delete this model?". The one who confirms first deletes, the latter sees the message "The model doesn't exist anymore!".

1.3.10 House Style (Template)

 House style templates are predefined templates (with .ott extension) that determine the general presentation of the language models. A company can associate a house style template to its templates so that all generated documents follow a corporate identity.

These LibreOffice Writer templates can contain all the style elements of the document: character, paragraph, frame, list and page styles, thus contributing to a typical formatting for text, tables and titles, page layout including watermarks, headers and footers etc. Moreover, the default text for headers and footers can be specified.

House style templates can be created and modified directly in Sketch. Authors should then apply these predefined house styles in the creation process of the language models.

When creating a language model, an author can associate a house style template. Then, all styles from the template are available in that document. The content of headers and footers is also copied from the template.

When a house style template is modified, the altered styles will affect the associated language models next time they are checked out. Theoretically, every language model can be linked to a different house style template.

1.3.11 Decision table

 Decision Tables are sets of available values for variables. They translate the variable content – a system's internal key – into language-dependent (descriptive) values. Example: the value of the variable Gender "F" will become "Miss" and "M" will become "Mr."

In other words, a decision table contains a set of data pairs: a key and the respective value. Every key in the list corresponds to one - and only one - value per language. If a decision table exists for a given variable, the key delivered by the XML or interactively selected by the user is automatically translated into the corresponding value. It is that value that will be displayed on the generated document. The key is the same for all languages, while the corresponding value is language dependent.

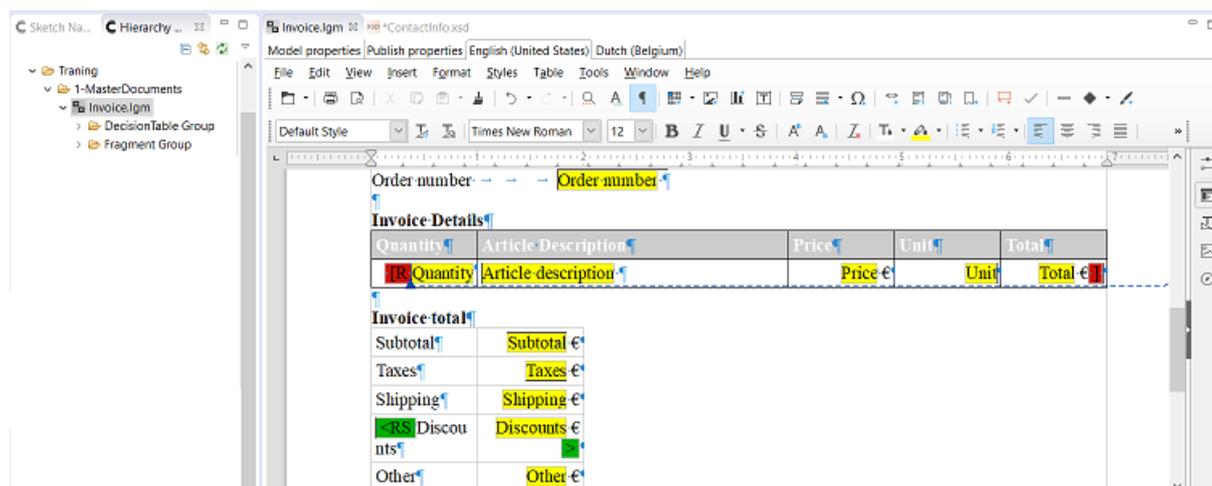
If the variable content is not a key in the decision table, the variable content itself will be displayed on the generated document.

Decision tables evolve through stages, just as models do. You could find distinct versions of them in the different ModelStores.

1.3.12 Working with data

As mentioned before, the author can add variables using the Objects palette. Also logic – such as conditions and loops – can be implemented using the attached data structure. By doing all this, the template becomes dynamic. With more than 40 available Functions for the data structure elements, Sketch is extremely powerful.

Example of variables, a condition and a loop



Variables



Variables are the smallest piece of data to work with. Every variable contains a little piece of information such as a name, an address, a date, a product code, a price etc.

Variables are grouped in one or more blocks of data (E.g. 'Street' is part of 'Address').

These data blocks can be part of a higher level data block etc. All these levels provide 'structure' to the data. You'll find this structure in the XSD schema as well as the sample XML file.

Conditions



Conditions are used to show or hide a word, a sentence, a fragment or even complete parts of a document. A wide range of expressions and operators allow the author to set certain criteria to validate variables and fixed data elements.

Example: depending on the input data, a particular address is shown. If BE is given as input data, the Connective address in Belgium is shown. If NL is given as input data, the Connective address in the Netherlands is shown.

When a condition opening tag displays an 'R' then it will affect the table row it is in. When an opening tag displays a 'T' then the condition will affect the whole table.

Loops



When repeating elements from the data structure have to be printed in a certain way, we use a loop. E.g. in a list of products, instances of the same element like product code, description, price etc. can be printed under each other.

Quantity	Article Description	Price	Unit	Total
R Quantity	Article description	€ Price	Unit	€ Total

The number of times it is repeated depends on the number of times the same element is found in the data. In loops the repeating

data can also be sorted and/or filtered.

When a loop opening tag displays an 'R' then it will affect the whole table row, or perhaps even more rows.

1.3.13 XML and XSD (Data structure) documents

If a model is intended to work with data then we need to attach some extra files. These files contain information on how the delivered data is structured and labelled. When adding variables or logic to a language model it is imperative to use the correct data. The following files must have the same structure as the data that will be provided at generation time.

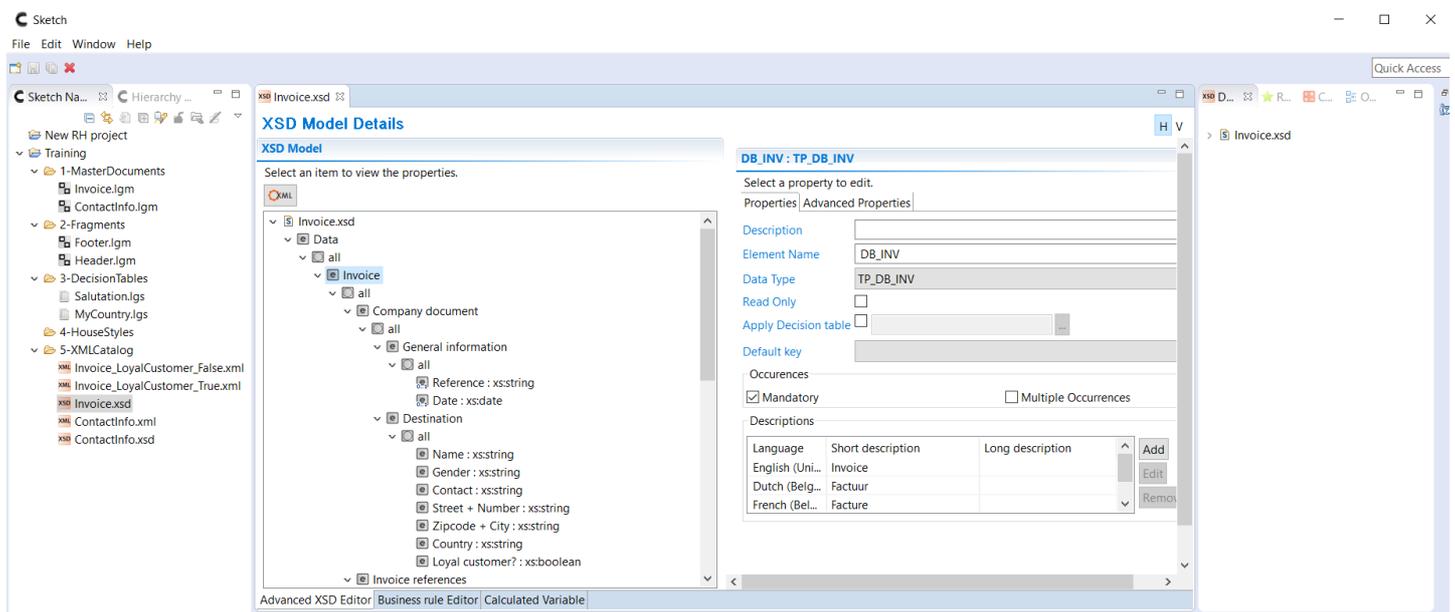
Data Structure (XSD)

XSD stands for 'XML Schema Document'. An XML Schema describes the structure of an XML document. XSD is used to define a set of rules to which an XML document must conform.

The XSD file can also contain description data, which associates the XML identification of each level (data blocks and variables) in the data structure with descriptions in one or more languages.

So instead of working with the variables internal key names (e.g. 'DB_INV'), the author has the option to work with descriptions in different languages.

Example of an XML schema Document (XSD)

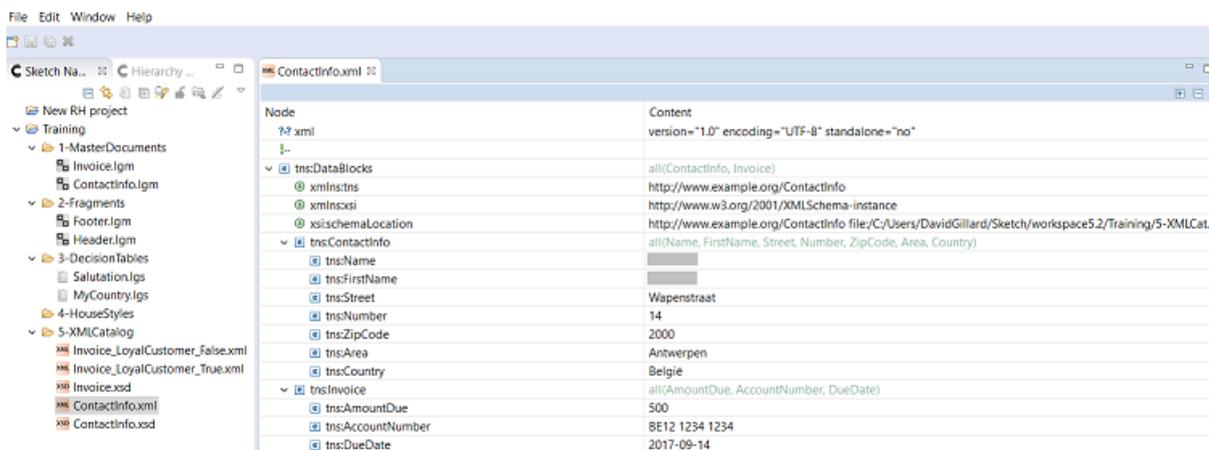


Sample data (XML)

XML stands for 'eXtensible Markup Language' and is designed to transport and store data.

The XML file gives a set of data for test purposes only. It is a sample of what would be provided during generation. Its data follows the structure found in the schema file.

Example of an XML data sample



2. Getting Started

[2.1 How to start Sketch?](#)

[2.2 How to connect to a ModelStore](#)

[2.3 How to create a local project](#)

[2.4 How to manage your account](#)

[2.5 Browsing through the Workspace](#)

[2.6 Searching for an item](#)

2.1 How to start Sketch?

A prerequisite to work with Sketch is that your computer is connected to the network.

There are several ways to launch Sketch:

- Double-click the **Sketch** icon on the desktop.



Or

- Open the **Start** menu and select **Sketch 5.5**.

Or

- Open the **Start** menu and type *Sketch* in the **Search** field.
- Click the **Sketch** icon.

When Sketch is launched, a Windows security alert might appear (depending on the version of Windows and the company security policies). You need to grant access to some features of the LibreOffice Writer application. Click **Unblock** to do so.

When Sketch is launched for the first time, you will see an empty Sketch Navigator.

If somebody would have already done the setup for you, you might see the ModelStores and projects listed in the Sketch Navigator on the left.

2.2 How to connect to a ModelStore

Once you are connected to a ModelStore, you can access the central repository of all the files that have been created in that specific ModelStore.

There are several ways to connect to a ModelStore:

- Click **File > New**.
- Expand the **Sketch** folder and click **ModelStore Connection**.
- Click **Next**.

Or

- Press **[Ctrl+N]** on the keyboard.
- Type *ModelStore Connection*.
- Press **Enter**.

Or

- Right-click in the Sketch Navigator region.
- Click **New > ModelStore Connection**.

The **New ModelStore Connection** window is opened:

- Select one of the **found servers**.

or

- Fill in the **Host** and **Port**.
- Enter your **Username** and **Password**.
- Click **Next**.

New ModelStore Connection

Connection

Specify the connection details for the ModelStore.

Servers found

Host: aslapad15

Port: 15220

Username:

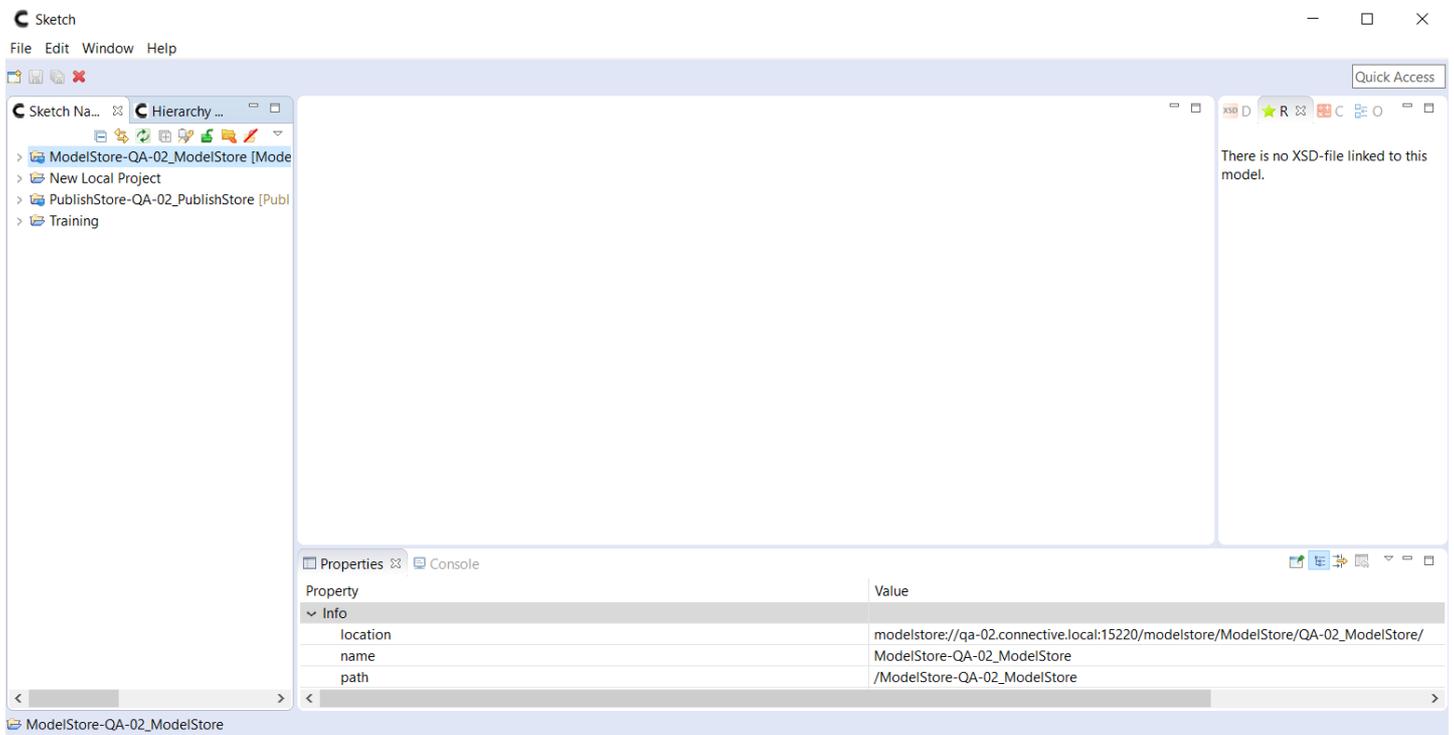
Password:

SSL enabled (https)

< Back Next > Finish Cancel

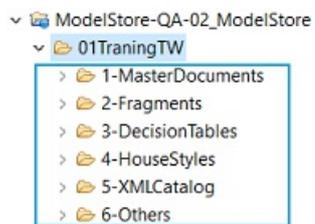
- Select the **ModelStores/PublishStores** you want to connect to.
- Click **Finish**.

After a successful logon to the Sketch server, the new ModelStore connections will be visible in Sketch Navigator at the left part of the screen. At the bottom of the screen, the **Properties** tab and the **Console** tab are visible.



The preferred folder structure of a ModelStore looks as follows:

- 1-MasterDocuments
- 2-Fragments
- 3-DecisionTables
- 4-HouseStyles
- 5-XMLCatalog
- 6-Other

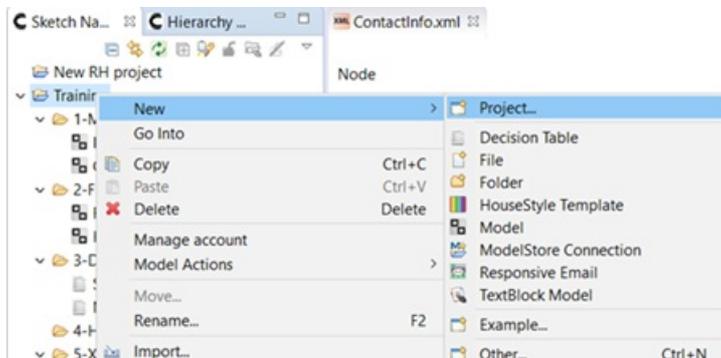


2.3 How to create a local project

A local project is used to work locally, without being connected to a ModelStore.

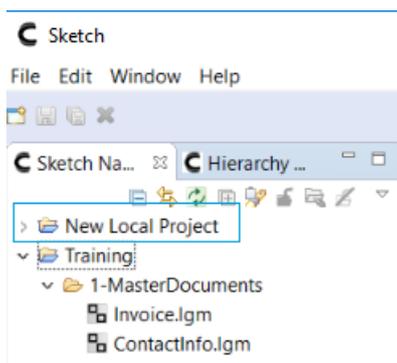
To create a local project:

- Right-click in the Sketch Navigator region.
- Click **New > Project**.



- In the **General** folder select **Project**, and click **Next**.
- Enter a name for your local project.
- Click **Finish**.

The local project will be added in the Sketch Navigator:

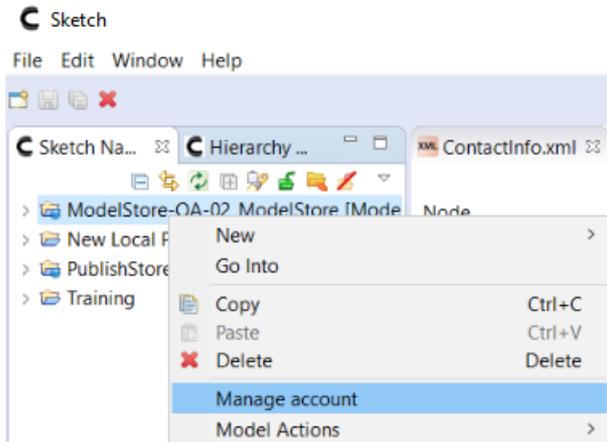


2.4 How to manage your account

Sketch makes it easy to manage your accounts: you can easily switch users for a specific ModelStore or PublishStore.

To switch users:

- Select the ModelStore/PublishStore.
- Right-click and click **Manage account**.



- In the **Manage account** window, enter your **User name** and **Password**.
- Click **Finish**.

2.5 Browsing through the Workspace

2.5.1 Overview

Once you are connected to a ModelStore, the new ModelStore is added to the Sketch Navigator. By default, the ModelStore is minimized, so you can only see the top level of the tree view. By clicking on the nodes, you can expand or collapse the folders, in order to show or hide the underlying content:

ICON	DESCRIPTION
	Clicking the node will show the content of the ModelStore.
	Clicking the node will hide the content of the ModelStore.
	Clicking the node will show the content of the folder.
	Clicking the node will hide the content of the folder.

You can find different types of files in your Workspace:

ICON	DESCRIPTION
	ModelStore Connection See 1.3.4. ModelStore
	Project See 1.3.2. Project
	Folder
	Model See 1.3.6. Model
	XML Schema See 1.3.13. XML and XSD (Data structure) documents.
	XML file See 1.3.13. XML and XSD (Data structure) documents.
	HouseStyle See 1.3.10. House Style (Template)
	Decision Table See 1.3.11. Decision Table
	Text Block Model See 16.1 Sketch TextBlock Models
	Writer Model See 16.2 Sketch Writer Models

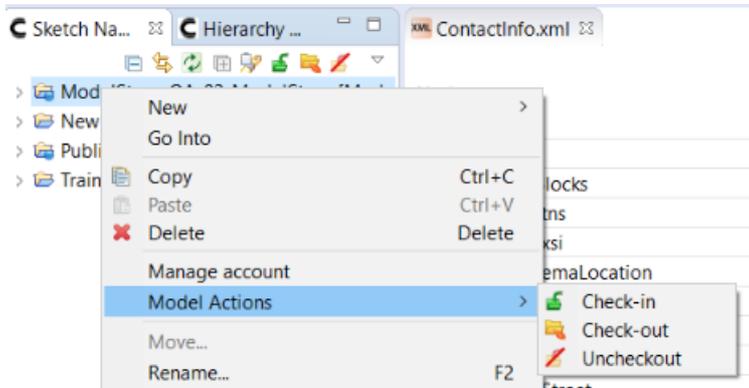
2.5.2 Contextual menu

For all different items in the workspace, a contextual menu is available. This contextual menu lists all the actions that can be

performed on that specific item.

To access the contextual menu:

- Right-click the required item.
- The contextual menu is opened.
- The actions related to the Logical Models are grouped in the submenu **Model Actions**.



2.5.3 Basic Model Actions

The basic Model Actions you can perform on each file in the ModelStore are grouped in a submenu in the contextual menu.

To access the submenu:

- Right-click the required item.
- The contextual menu is opened.
- Select the submenu **Model Actions**.

The 3 most important actions - the actions you will constantly use while working in Sketch - are **Check in**, **Check out** and **Uncheckout**.

All the files that have been created in a specific ModelStore are stored in a central database. To prevent different users from working on the same file at the same time, a system of version control is put in place.

When you want to start working on a model, you need to check out that model. When you do a checkout, a copy is made from the last version that has been saved in the central database. This copy is made available to you, so you can make your changes. At the same time, the original file in the database is locked, so the file can't be checked out by another user. When another user tries to check out the same model, he will only see a read-only copy of the original file.

When your changes are done, you need to **check in** the model. When you do this, the locked copy in the database is replaced by the updated version. Once this is done, the model is available to the other users.

In case you don't want to save the changes you made, you can **uncheckout** the model. This action cancels the checkout of the file and unlocks the locked copy in the database without replacing it with an updated version. All changes made to the model are canceled.

The status of a file is reflected in the ModelStore:

ICON	DESCRIPTION
	The file is checked in . You or other users can check out the file in order to make changes.
	The file is checked out by you. No modifications have yet been saved, so the only possible action is Uncheckout . Other users cannot make changes to this file.

ICON	DESCRIPTION
	The file is checked out by you. Modifications have already been saved locally , so possible actions are Check in (if you want to save your changes) or Uncheckout (if you want to discard your changes). Other users cannot make changes to this file.
	The file is checked out by another user. The file is locked, so you cannot make changes to this file.

2.5.4 Refresh

A ModelStore is a central repository shared by different authors. As an author, you have to keep in mind that the tree view in the Sketch Navigator might not show the latest situation: in between the automatic refreshes of your tree view, another user might have checked in/out another model – this change will not yet be reflected in your tree view.

To make sure you see the latest situation, you can do a refresh. There are 2 ways to do a manual refresh:

- Right-click the folder or file you want to refresh.
- Click **Refresh**.

OR

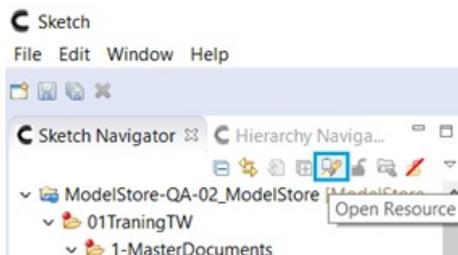
- Select the folder or file you want to refresh.
- Press **F5**.

The refresh action can be done for all items in the workspace.

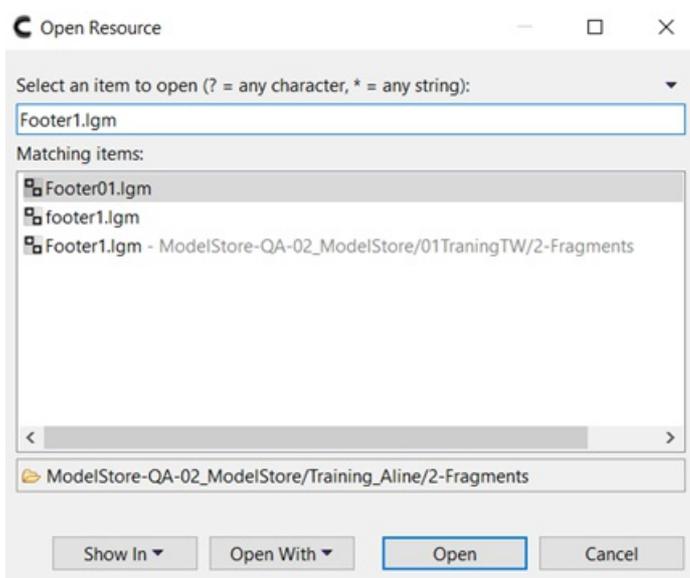
2.6 Searching for an item

To search for an item in the Sketch Workspace:

- Click the **Open Resource** icon.



- Enter the item you are searching for.
- The results are immediately displayed.



- Click **Open** to open the selected search result.

Or

- Click **Open with** and select the required component.

3. Logical Model

The first step in the construction of a dynamic document is the creation of a logical model. A logical model contains all the template properties and contains one or more language models.

[3.1 Properties of the logical model](#)

[3.2 How to create a model](#)

[3.3 How to view a model \(read-only\)](#)

[3.4 How to check out a logical model](#)

[3.5 How to save a logical model locally](#)

[3.6 How to check in a logical model](#)

[3.7 How to cancel modifications to a logical model](#)

[3.8 How to close a logical model](#)

[3.9 How to publish a logical model](#)

[3.10 How to delete a model](#)

[3.11 How to stage a logical model](#)

[3.12 How to restore a logical model](#)

3.1 Properties of the logical model

As mentioned before, the logical model is the central reference to a business document in all languages and in all stages. It contains the following information:

ITEM	DESCRIPTIONS
Model Name	The name of the logical model, which is visible in the tree view in your Sketch Navigator. A model can be renamed afterwards. Note however, that it is strongly recommended not to change the name of models used as fragments in other documents. Renaming the model impacts the link that points to the fragment.
Model Key	The key of the logical model, which must be unique . This name is used to publish the model and to call the document at generation time. The model key can be added in the tab Publish Properties .
Descriptions	A description for the logical model is automatically added in 1 language. This description can be modified, or descriptions in other languages can be added later on.
Data structure (XSD)	A data structure can be linked to the logical model, but this is optional. Note: if you plan on using variables, calculated variables or loops, linking a data structure is required.
Sample XML	Once a sample data file - which needs to be compatible with the attached XSD - is attached to the logical model, you can preview the model.
Language models	A logical model can contain one or more language models.
Copy titles	Copy titles are linked to the logical model in order to create different copies of one generated document to several recipients.
Decision table	Decision tables are linked to the logical model so they can be used in conditions or for formatting specific variables.
Writer	Writer models can be linked to the logical model to enable users to add paragraphs to the generated document in the Writer application.

3.2 How to create a model

3.2.1 Create a logical model

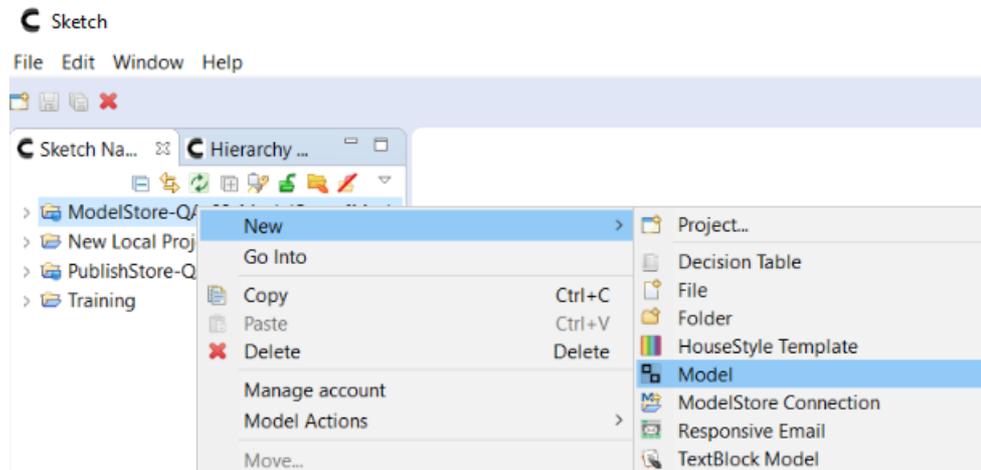
3.2.2 Create a language model

3.2.1 Create a logical model

When you want to add a new logical model, you first have to decide where you want the model to be placed. This is very important. Once a model is created in a specific folder in the tree view, links will be established to other elements that are associated with that model (for example a data structure, decision tables, fragments, etc.). It is recommended not to change the location of models. Moving the model impacts the links that point towards the model.

To add a new logical model:

- Select the folder where you want the logical model to be located.
- Right-click and click **New > Model**.



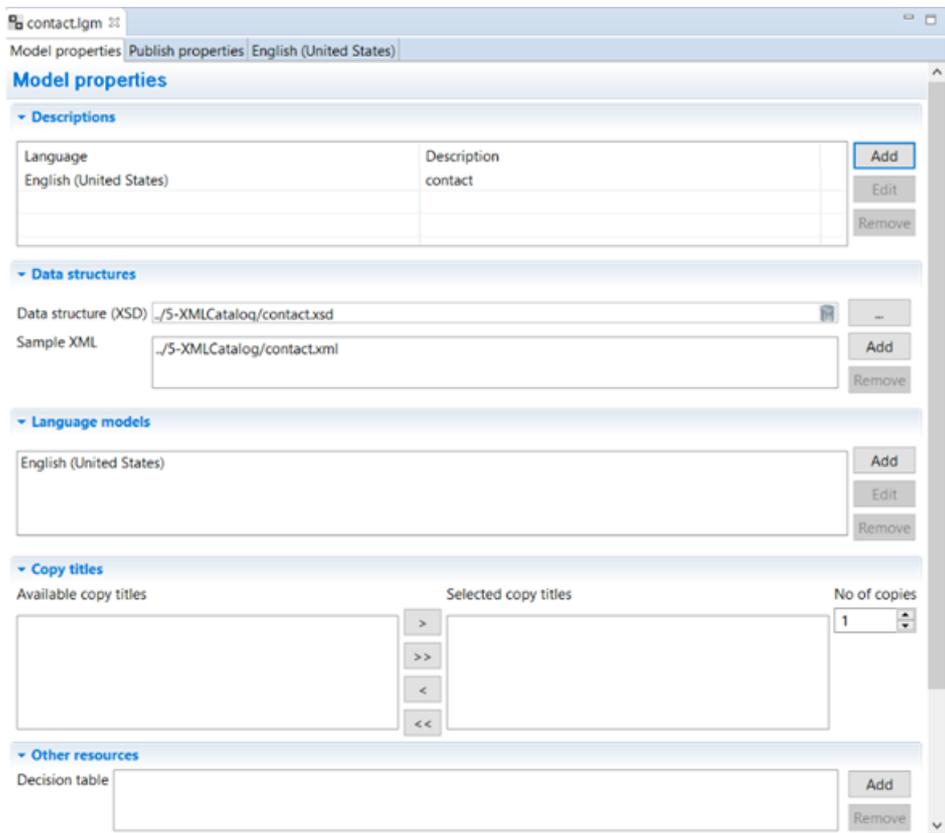
- The **New Model** wizard appears.
- The folder you have selected is by default indicated as **parent folder**. To change the location of the logical model, click on the nodes to expand/collapse the different folders and then select the folder where you want to save the model.
- Enter the **model name** in the **File name** text box. A logical model must have the extension .lgm.
- Click **Finish**.

Your new logical model is created in the selected folder. The model is checked out and opened in the **Editor** region.

Now you can add the **Model Properties**.

To add a Description:

- Under **Descriptions**, click **Add**.
- Select the **language**.
- Add a **description**.
- Click **OK**.



To add an XSD file:

- Under **Data Structures**, click the browse button (...)
- Browse and select the **XSD file** you want to associate with the logical model.
- Click **OK**.

To add a sample XML:

- Under **Data Structures**, click **Add**.
- Browse and select one or more **XML file(s)** you want to add to the logical model.
- Click **OK**.
- If the XML is not compatible with the XSD, an error message will appear.

To add a Language model:

- Under **Language models**, click **Add**.
- Select the **language**.
- Select a **housestyle** (optional).
- Add a **description** (optional).
- Click **OK**.

The language model is now opened in a new tab.

To add a Decision table:

- Under **Decision table**, click **Add**.
- Browse and select one or more **Decision table(s)** you want to add to the logical model.
- Click **OK**.

3.2.2 Create a language model

A logical model can contain several language models. A language model represents the **physical document** in a specific language.

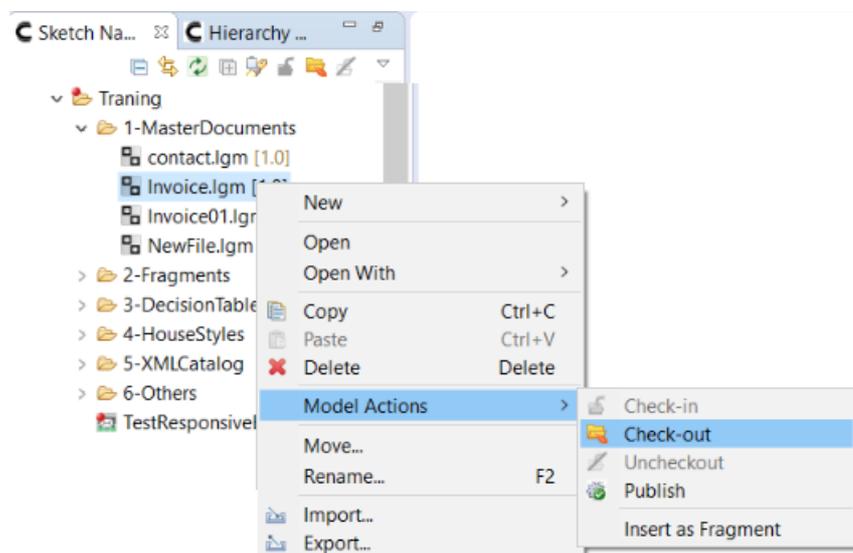
When a language model is opened, the document is opened in LibreOffice. This way you can create your document while using all the standard LibreOffice Writer functionalities, like inserting fixed text, tables, header/footer and customizing the layout of your document. For more information about LibreOffice, visit <http://www.libreoffice.org> and <http://www.libreoffice.org/get-help/documentation>.

Apart from these standard functionalities, you can use the Sketch functionalities like inserting data and adding logic. This way, you make your document dynamic.

Create a new language model from scratch

To add a language model from scratch:

- Check out the logical model.



- Under **Language models**, click **Add...**
- Select the required language from the **Language** list.
- Select a **HouseStyle** template (optional).
- Click **OK**.

The new language model is opened in a new tab and you can start working in the language model.

Create a new language model based on a source language

If you already created a dynamic document in 1 language, you can create a new language model based on this language. The advantage is that you don't have to rebuild all the logic a second time. You can start working with the exact copy of the source language, so all you have to do is translate the fixed text in order to get a working dynamic document in a second language.

To add a language model based on a source language:

- **Check out** the logical model.
- Under **Language models**, click **Add**.
- Select the **Language** from the available languages in the selection list.
- Select the **Source** language.
- Select a **HouseStyle** template (optional).
- Click **OK**.

The new language model is opened in a new tab and the content is a copy of the source language.

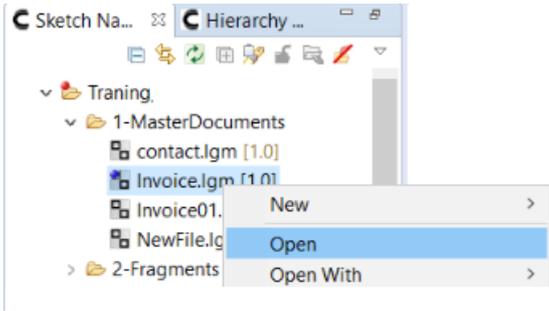
3.3 How to view a model (read-only)

It is possible to view a logical model and the associated language models without checking out the logical model. In this case, a read-only copy of the model will be opened, so it is not possible to make any modifications. You can however preview or print the language model.

3.3.1 View a logical model

To view a logical model:

- Select the logical model in the Sketch Navigator.
- Right-click and click **Open**.



Or

- Double-click on the logical model in the Sketch Navigator.

A read-only copy of the logical model is opened. In read-only mode, no editing can be done. Therefore, no menu bars are displayed.

The logical model is indicated by a grey symbol in the ModelStore.



Tip: in case any elements are missing, e.g. an .xsd, .xml or decision table, a message is displayed showing which elements are missing.



3.3.2 View a language model

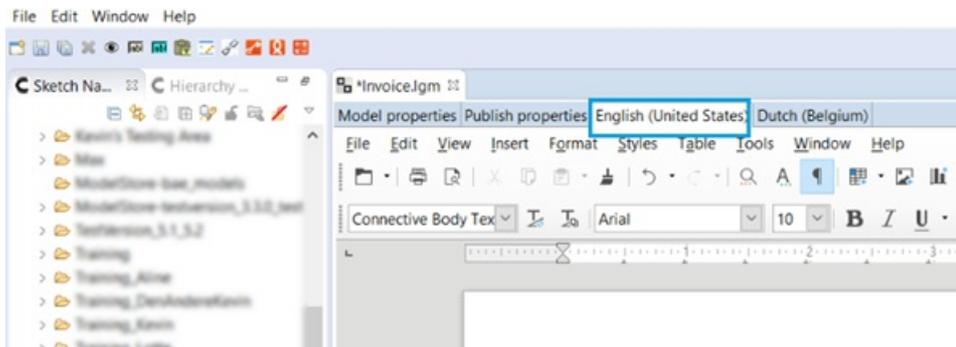
To view a language model:

- Select the logical model.

- Right-click and select **Open**.

Or

- Double-click on the logical model.
- Select the **language model** tab in the Model Editor.



A read-only copy of the language model is opened. In read-only mode, no editing can be done. Therefore, no menu bars are displayed.

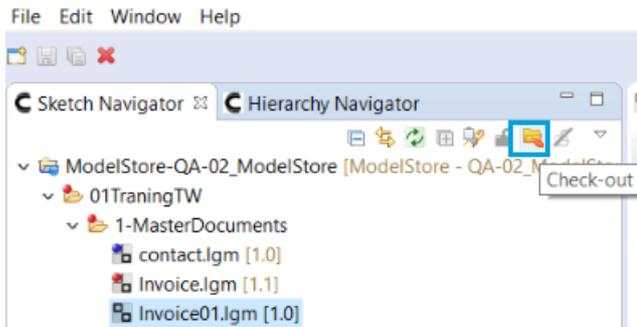
The logical model is indicated by a grey symbol in the ModelStore.

3.4 How to check out a logical model

If you want to make changes to a logical model or the associated language models, you need to **check out** the logical model. This way, the latest version of the model will be available to you for modification and the model will be locked for other users.

To check out a logical model:

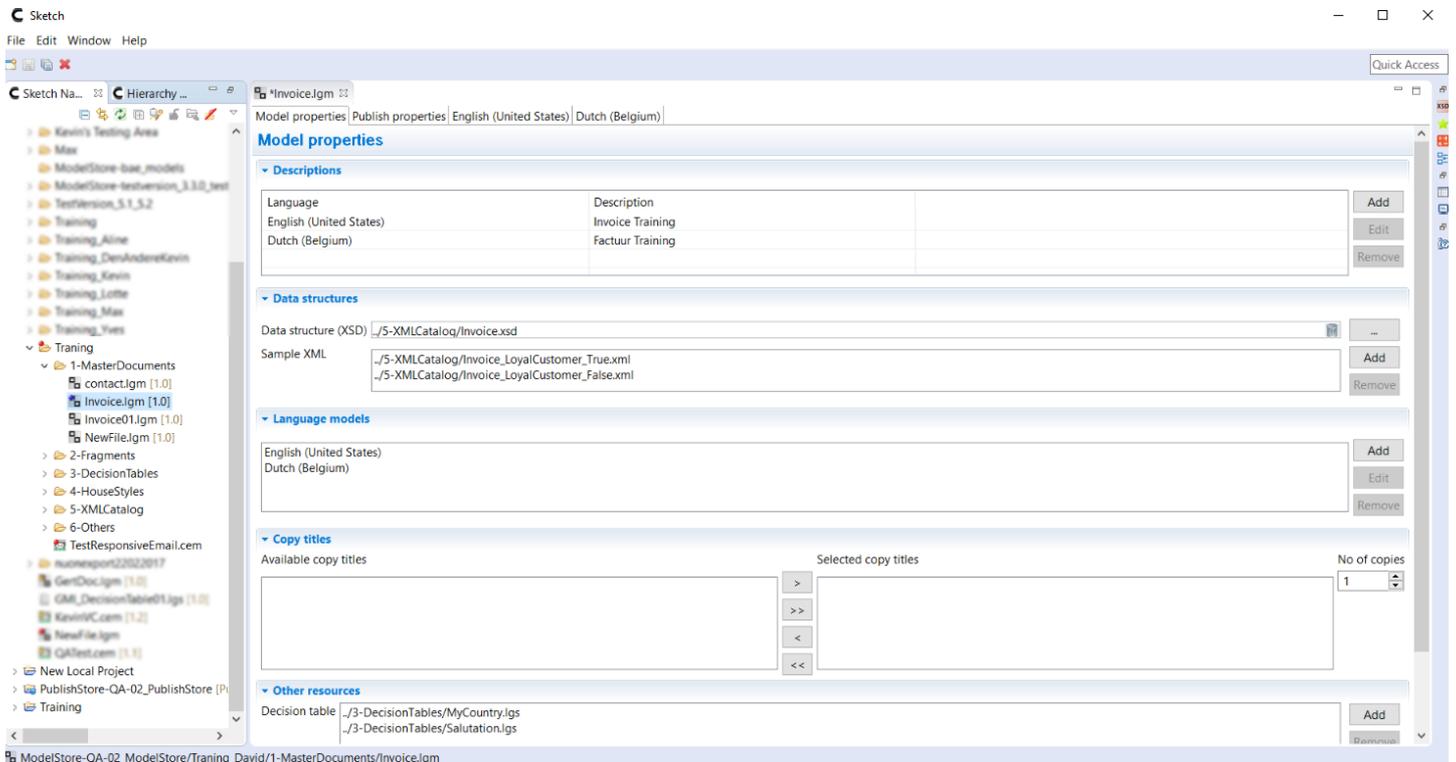
- Select the logical model.
- Click the **Check out** icon in the toolbar.



Or

- Select the logical model.
- Right-click and click **Model Actions > Check out**.

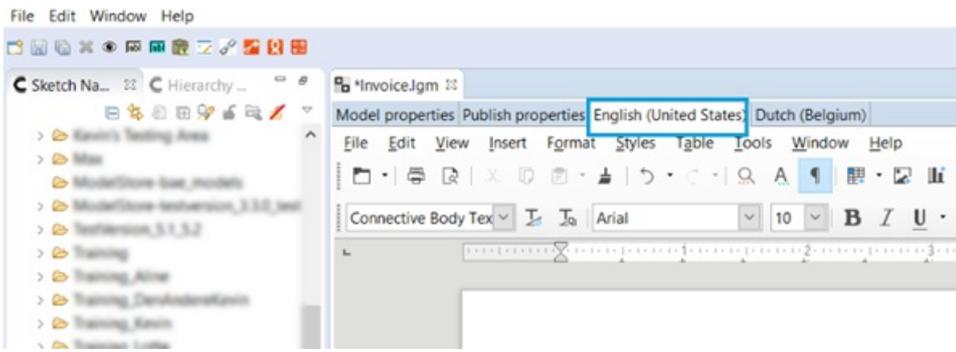
The logical model is checked out and opened, and you can start your modifications. The logical model is indicated by a blue symbol  in the ModelStore.



Tip: in case any elements are missing, e.g. an .xsd, .xml or decision table, a message is displayed showing which elements are missing.

Once the logical model is checked out, you only have to open the language model in case you want to modify the language model.

- Click the appropriate **language** tab in the Model Editor.

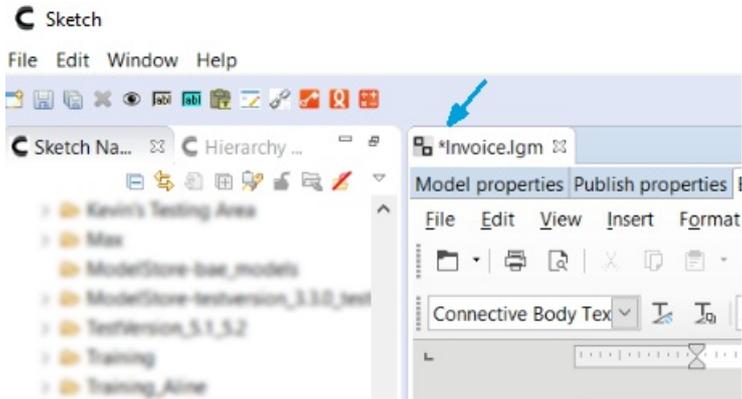


The language model is checked out and opened and you can start your modifications. The logical model is indicated by a blue symbol  in the ModelStore.

3.5 How to save a logical model locally

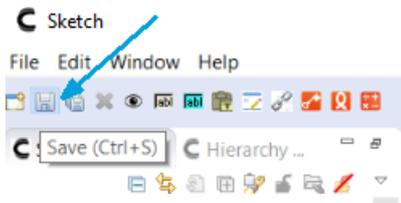
When you are creating a dynamic document, it is recommended to **save** your document on a regular basis. This Save action will save your model locally, the logical model will not be checked in.

A model that has been modified (and can be saved) is indicated by an **asterisk (*)** next to the model name:



To save locally, there are 3 options:

- Click the **Save** icon in the toolbar.



- Click **File > Save**.
- Press **[Ctrl+S]** on the keyboard.

When the model has been successfully saved, the asterisk (*) next to the model name disappears and the logical model is indicated by a red symbol  in the ModelStore. The model is still opened and you can continue your modifications.

3.6 How to check in a logical model

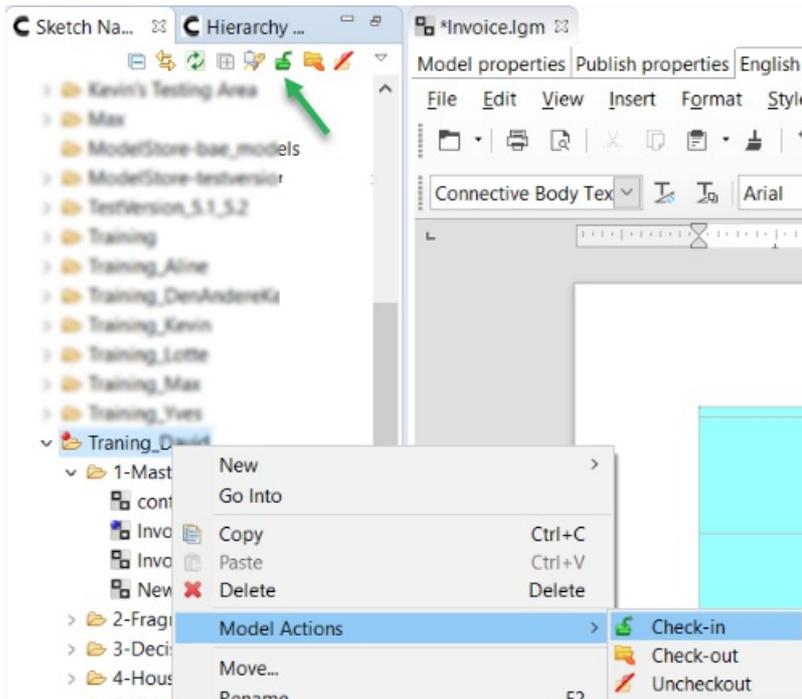
When your changes are done, you need to check in the model. When you do this, the locked copy in the database is replaced by the updated version. Once this is done, the model is available to the other users.

To check in a logical model, there are 2 options:

- Select the logical model.
- Click the **Check in** icon in the toolbar.

Or

- Select the logical model.
- Right-click and click **Model Actions > Check in**.



- Enter a **Comment** (optional).
- Click **OK**.

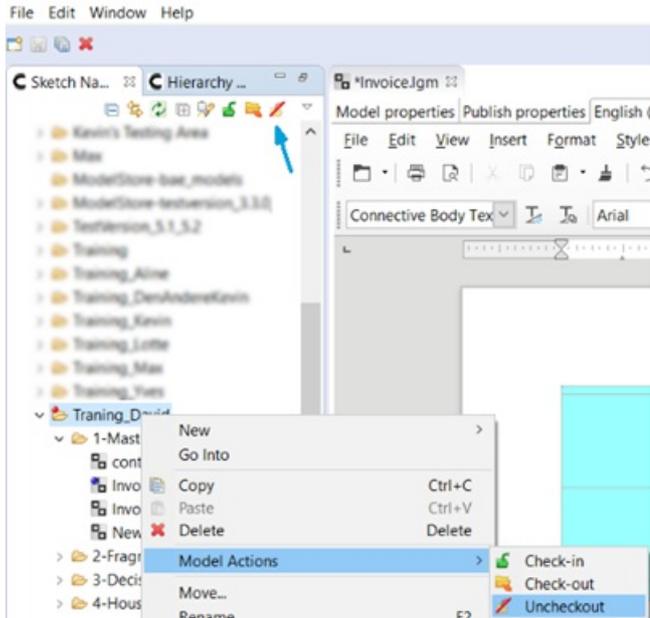
The logical model is checked in and indicated by a grey symbol  in the ModelStore.

3.7 How to cancel modifications to a logical model

In case you don't want to save the changes you made, you can **uncheckout** the model. This action cancels the checkout of the file and unlocks the locked copy in the database without replacing it with an updated version. All changes made to the model are canceled.

To cancel the modifications to a logical model:

- Select the logical model.
- Click the **Uncheckout** icon in the toolbar.
- Click **Yes** to confirm.



Or

- Select the logical model.
- Right-click and click **Model Actions > Uncheckout**.
- Click **Yes** to confirm.

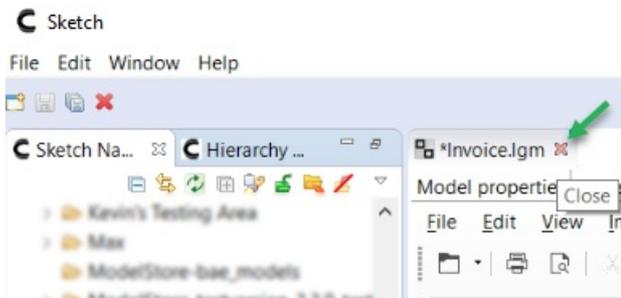
The logical model is closed and indicated by a grey symbol  in the ModelStore.

3.8 How to close a logical model

When you close a logical model, the model will be removed from the Model Editor.

To close a logical model, there are 3 options:

- Click the **Close** icon on the appropriate tab. **Tip:** do not click the close icon in the main toolbar. This deletes the selected element.



- Right-click the opened model tab and click **Close**.
- Press **[Ctrl+F4]** on the keyboard.

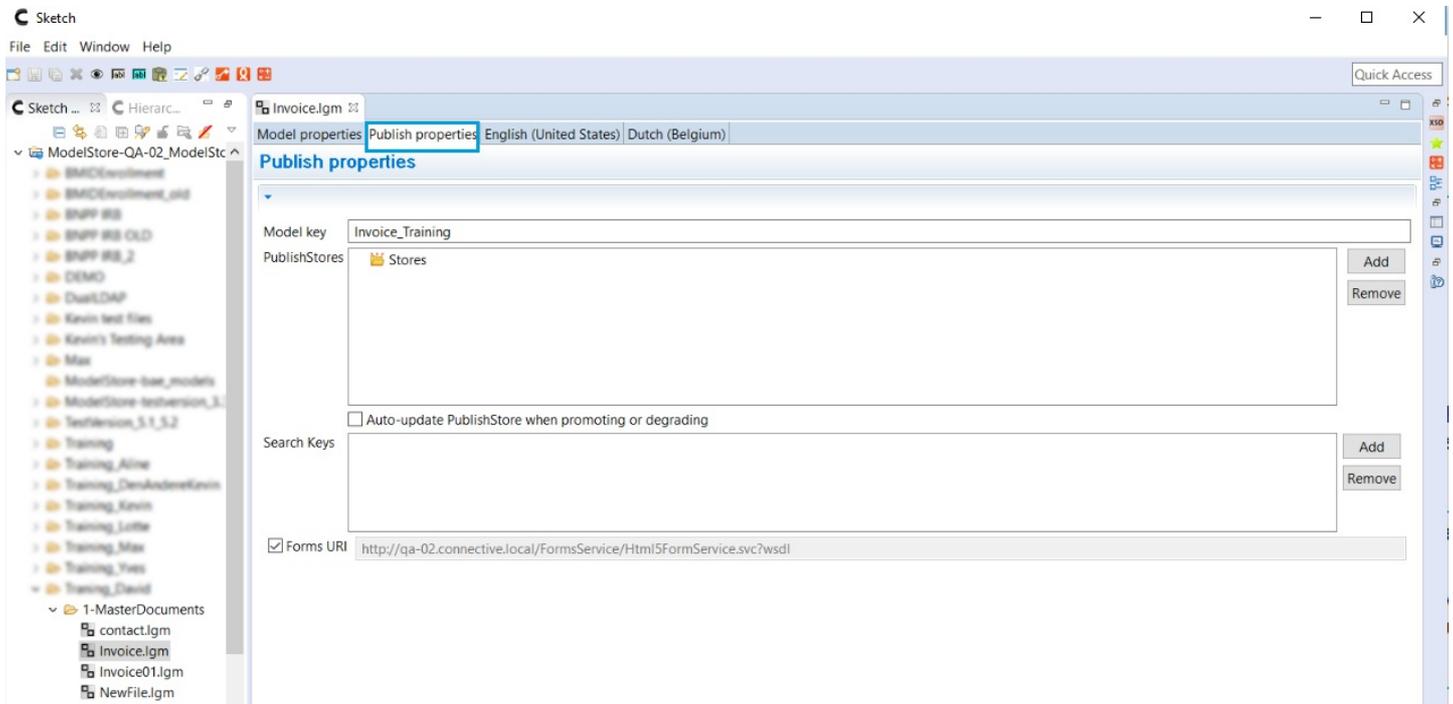
If the model has been modified but hasn't been saved locally yet, you will be prompted to save it.

3.9 How to publish a logical model

When your model has been created in the ModelStore, you have to make this template available for generation on the Generator server. This process of copying your logical model from a ModelStore to a PublishStore is called publishing.

3.9.1. Publish Properties

Before a model can be published, you need to make sure that the publish properties are correctly set. You can find these properties in the **Publish properties** tab of the model:



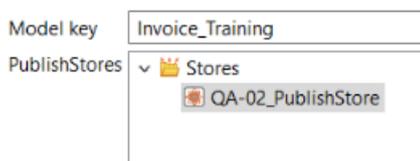
To add a Model key:

- Enter the name in the **Model key** text box.
- The model key must be **unique**.

To add a PublishStore:

- Select **Stores** under **PublishStores**.
- Click **Add**.
- Select the **Publish store(s)**.
- Click **OK**.

The selected publish store is now added under **Stores**.



To add a Unit:

- Select the added PublishStore.
- Click **Add**.
- Select the **Unit**.
- Select **OK**.

The selected unit is now added under the PublishStore. You can add one or more units.

To add a Category:

- Select the added unit.
- Click **Add**.
- Select the **Category**.
- Click **OK**.

The selected category is now added under the PublishStore. You can add one or more categories.

To add a Search key:

- Click **Add** under **Search Keys**.
- Enter the search key.
- Click **OK**.

The search key is now added to the list of search keys. You can add one or more search keys.

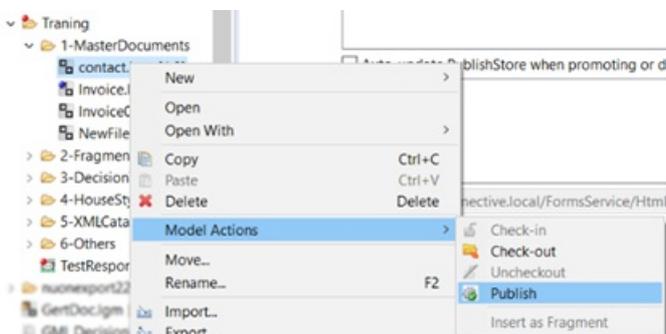
Notes:

- When the option **Auto-update PublishStore when promoting or degrading** is selected, the value of the PublishStore(s) will be replaced by the values that are stored as default value for the new PublishStore(s).
- When the option **Forms URI** is selected, the corresponding Forms will also be published and generated.

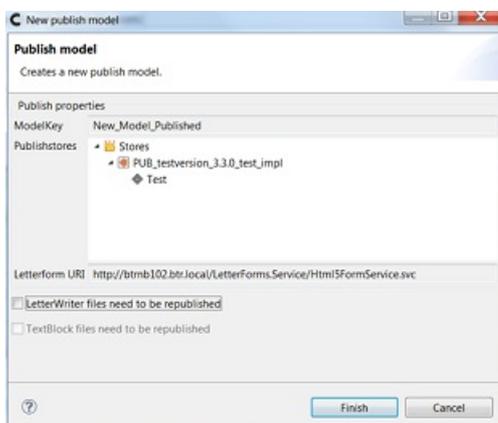
3.9.2. Publish a logical model

To publish a logical model:

- Select the model you want to publish.
- Right-click and click **Model Actions > Publish**.



- Select a Unit/Category.
- Select the option **LetterWriter files need to be republished** if necessary.



- Click **Finish**.
- Add a comment (Optional).

When the model has been successfully published, you can find the model key in the PublishStore.

You can verify when a model has last been published in the PublishStore. To do this, select one of the files in the published folder and check the date in the **last modified** field of the **Properties** tab at the bottom part of the Editor region.



The screenshot shows the 'Properties' tab of an IDE. At the top, there are two tabs: 'Properties' (selected) and 'Console'. Below the tabs is a table with two columns: 'Property' and 'Value'. The table contains three rows of data.

Property	Value
last modified	July 14, 2017 at 2:20:44 PM
Last modified by	qauser002
Last modified comment	

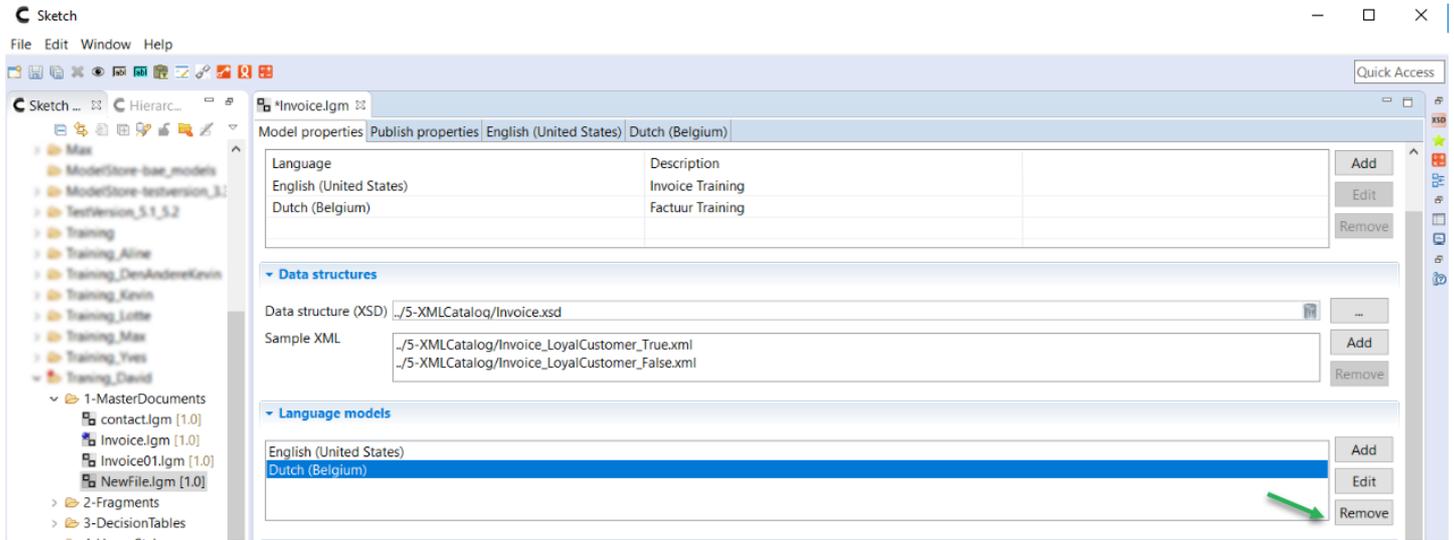
3.10 How to delete a model

3.10.1. Delete a language model

You can delete one of the language models associated with a logical model when you no longer need the language model. Note that you need sufficient rights to perform this action.

To delete a language model:

- Open the **Model Properties** tab of the logical model.
- Under **Language models**, select the language you want to remove.
- Click **Remove**.



The tab of the language model is removed.

3.10.2. Delete a logical model

It is possible to delete a logical model from the ModelStore when you no longer need the model. Note that you need sufficient rights to perform this action.

To delete a logical model:

- Select the logical model.
- Click the **Delete** icon in the toolbar.

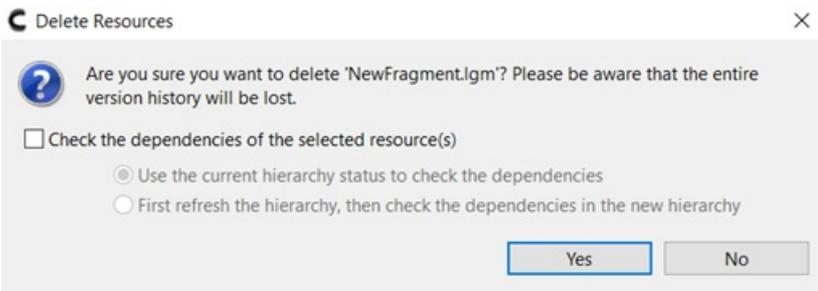
Or

- Select the logical model.
- Right-click and click **Delete**.

Or

- Select the logical model.
- Press **Delete** on the keyboard.

When you try to delete a model, a message will be displayed warning you that the entire version history will be lost.



Sketch can now check whether the resource you are going to delete has dependencies on other Sketch elements. To do so, click **Check the dependencies of the selected resource(s)**.

- To check the dependencies of the current hierarchy status, select the first option.

Attention: if other users are adding elements that are not yet reflected in the Hierarchy Viewer, these elements will not be taken into account.

- To refresh the entire ModelStore first, before checking the hierarchy, select the second option.

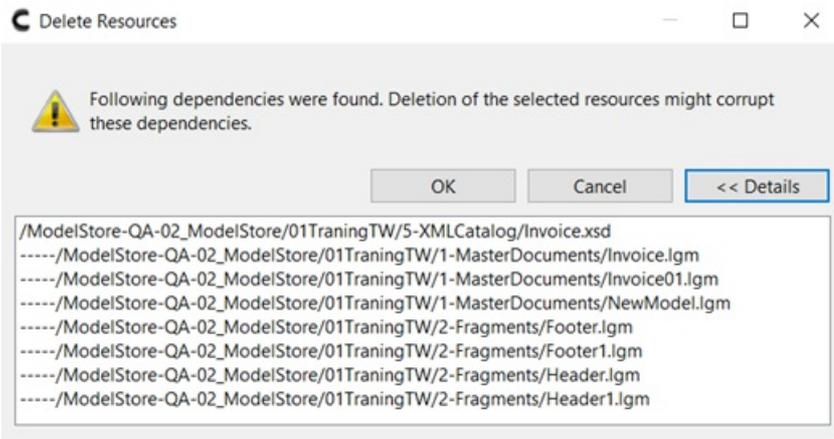
Attention: refreshing the entire ModelStore may take a long time depending on the size of the ModelStore.

Click **Yes** to continue.

If no dependencies are found, the resource(s) will be deleted. They will no longer be present in the ModelStore.

If dependencies are found, a warning message is displayed. Click **Details** to view them.

To delete the resource, click **OK**.



3.11 How to stage a logical model

Typically the development of a document is done in the test stage. When the document is ready or gets approval, it may be promoted to a higher level.

Depending on customer needs, it is possible to have two or more stages. E.g. test and production. A typical setting could be **DTAP** where documents have to follow the stages **Development, Test, Acceptance** and **Production**. This way, different versions of the same language model can exist in several stages at the same time.

By default, the destination of the promotion is set to the next stage, but depending on the settings, it is possible to select the desired stage from a selection list.

The opposite of 'promote' is 'degrade', copying language models from 'production' back to 'test'.

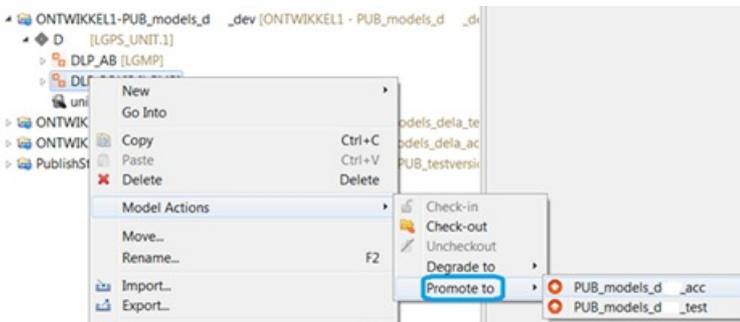
Note: whether the actions **Promote to** or **Degrade to** are available, depends on your user rights. Also, if your environment does not use different stages, these actions are not available.

3.11.1. Promoting

When promoting a language model, only the model itself is copied to another stage. The fragments used in the model are not promoted. However, the fragments must exist on a higher stage. If not, it will be impossible to promote the language model. An error message appears.

To promote a language model:

- Right-click the logical model.
- Click **Model Actions > Promote to**.
- Select the stage you want to promote the model to.



3.11.2. Degrading

Degrading will copy a language model back from a higher to a lower stage.

To degrade a language model:

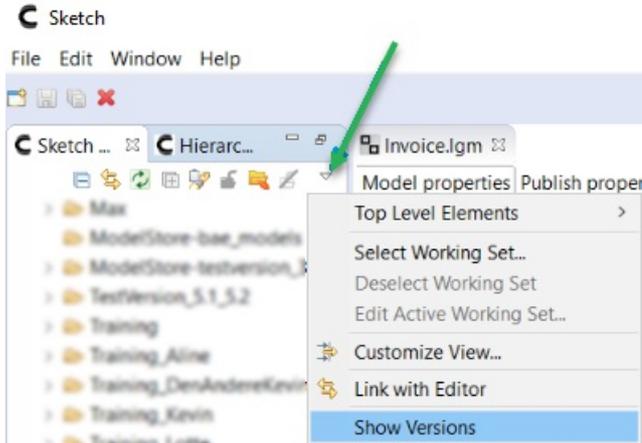
- Right-click the logical model.
- Click **Model Actions > Degrade to**.
- Select the stage you want to degrade the model to.

3.12 How to restore a logical model

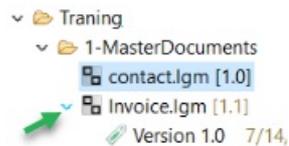
If you want to return to a previous version of a logical model, you can restore an earlier version.

To restore a logical model:

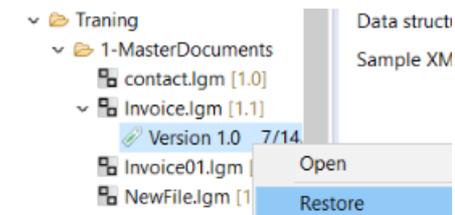
- Click the triangle icon to open the **View** menu.
- Click **Show Versions**.



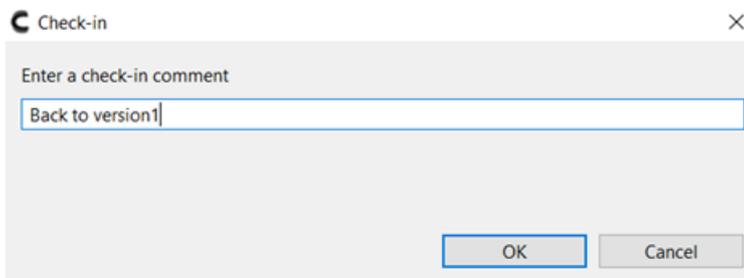
- Select the logical model you want to restore.
- Click the **triangle** in front of the model name.
- The list of different versions is opened.



- Right-click the version you want to restore and click **Restore**.



- You are now prompted to enter a check-in comment.
- Enter a comment and click **OK**.



The logical model is checked in, and the check-in comment is displayed in the **Model properties**.

Property	Value
last modified	February 16, 2018 at 12:18:10 PM
Last modified by	qauser002
Last modified comment	Back to version1
location	modelstore://qa-02.connective.local:15220/
name	Model 5.4.lgm

4. The Decision Table

Decision Tables are sets of available values for variables. They translate the variable content – a system's internal key – into language-dependent (descriptive) values. Example: the value of the variable Gender "F" will become "Miss" and "M" will become "Mister."

In other words, a decision table contains a set of data pairs: a key and the respective value. Every key in the list corresponds to one - and only one - value per language. If a decision table exists for a given variable, the key delivered by the XML or interactively selected by the user is automatically translated into the corresponding value. It is that value that will be displayed on the generated document. The key is the same for all languages, while the corresponding value is language dependent.

If the variable content is not a key in the decision table, the variable content itself will be displayed on the generated document.

When you insert a variable from the data structure (XSD) in your dynamic document, you can link a decision table to this variable. When a dynamic document is generated, the data delivered contains the key of the selection list, but it is the value that is displayed in the document. The key can be delivered in XML or entered manually via DataManager.

By doing this, the key from the sample data (XML) will be replaced with the value from the decision table at generation time.

Example of decision table

Key	Value (English)
BE	Belgium
NL	The Netherlands
IN	India
FR	France
BR	Brazil
...	...

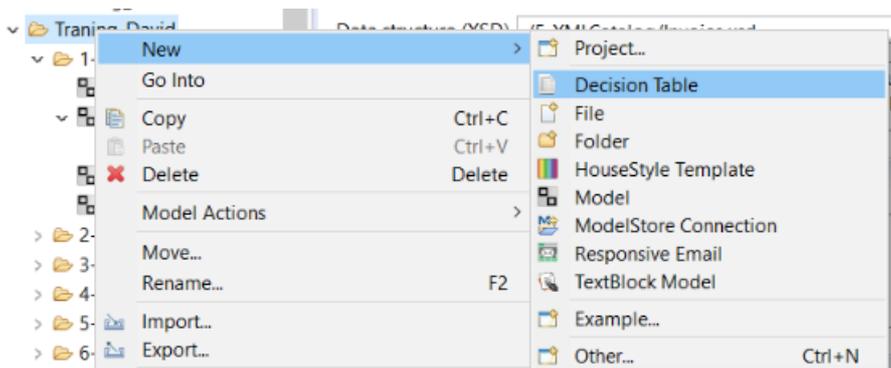
4.1 How to create a decision table

When you want to add a new decision table, you first have to decide where you want the model to be placed. This is very important. Once a decision table is created in a specific folder in the tree view and linked to a logical model, you cannot move the decision table afterwards without impacting the link to the logical model.

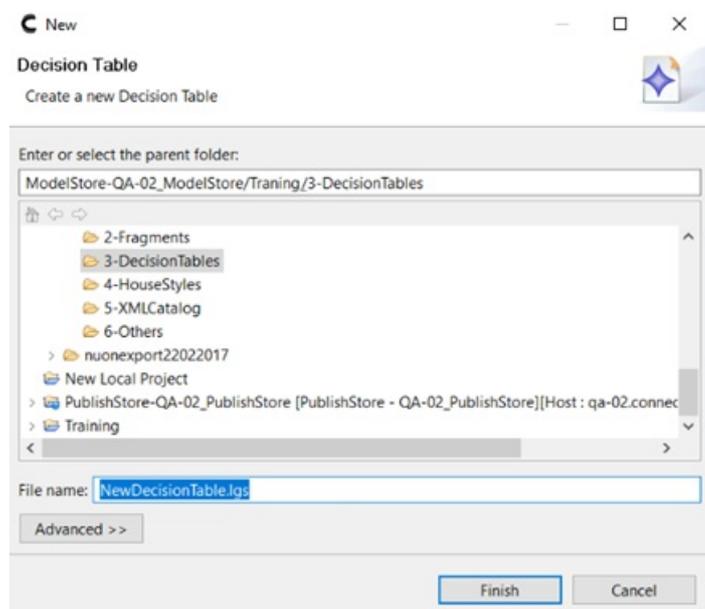
4.1.1. Add a decision table

To create a new decision table:

- Select the folder where you want the decision table to be located.
- Right-click and click **New > Decision Table**.



- The **New Decision Table** wizard appears.
- The folder you selected is by default indicated as **parent folder**. To change the location of the decision table, click the nodes to expand/collapse the different folders and then select the folder where you want to save the model.
- Enter the decision table name in the **File name** text box. A decision table needs to have the extension .lgs.
- Click **Finish**.



Your new decision table is created in the selected folder. The decision table is checked out and opened in the Editor region.

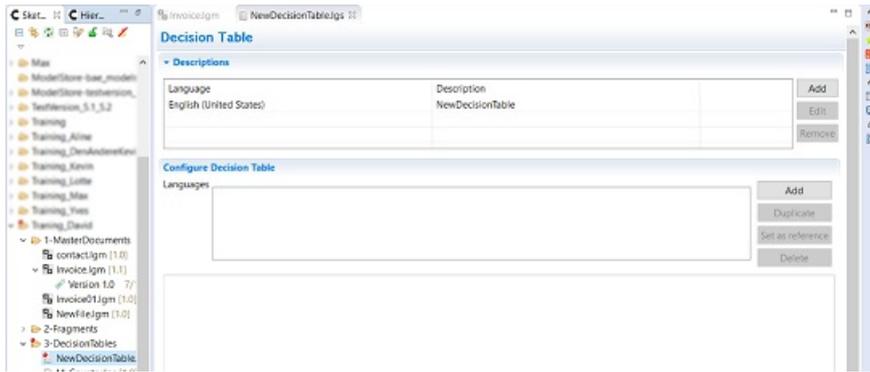
4.1.2. Add basic information to a decision table

Once the decision table has been created in the ModelStore, you can add some basic information about the decision table. This is not mandatory.

To add basic information:

- Under **Descriptions**, you can add a description in a new language (optional).
- Click **Add**.

- Select the **Language**.
- Add a **Description**.



4.1.3. Add the reference language

Once the decision table has been created in the ModelStore, you need to configure the decision table and add the keys with the corresponding values in one or more languages. The first language you add will be the reference language.

To add the reference language:

- Under **Languages**, click **Add**.
- Select the desired **language** from the selection list.
- Click **Finish**.
- A new **language** tab has been added at the bottom of the screen.

4.1.4. Add a row

Once the reference language has been set, you can add the values and their corresponding values.

To add a row:

- Click in the first row under **Key**.
- Enter the key.



- To add a corresponding **Value**, press Tab to switch from the **Key** column to the **Value** column.
- Now enter the value.

To add more rows:

- Click in the next available row under **Key** or press Tab to go to the next row.
- Enter the corresponding key.
- Press Tab once to switch to the **Value** column.
- Enter the corresponding value.

To use the Copy/Paste function:

- Select the text to copy.
- Press **Ctrl+C**.

- Click inside the cell where you want to paste the text.
- Press **Ctrl+V** or right-click and click **Paste** in the contextual menu.

4.1.5. Delete a row

If you need to remove a key and the corresponding value, you can easily delete a row from the decision table.

To delete a row:

- Position your cursor in front of the key in the row you want to delete.
- Right-click and select **Delete**.
- The selected row is now removed from the table.

4.1.6. Add another language

Once your reference language has been created and the keys and their corresponding values have been entered, you can add more languages.

To add another language:

- Under **Languages**, click **Add**.



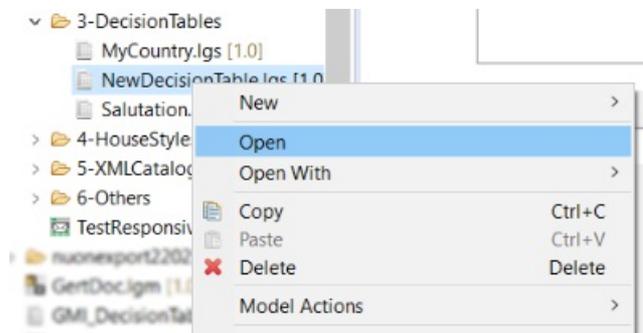
- Select the desired **language** from the selection list.
- Click **Finish**.
- A new **language** tab has been added at the bottom of the screen, next to the reference language.
- The Key has already been set, so you only have to fill in the **Value**.

4.2 How to view a decision table (read-only)

It is possible to view a decision table without checking out the file. In this case, a read-only copy of the decision table will be opened, so it is not possible to make any modifications.

To view a Decision Table:

- Select the decision table in the Sketch Navigator.
- Right-click and click **Open**.



Or

- Double-click on the decision table.

A read-only copy of the decision table is opened. The decision table is indicated by a grey symbol in the ModelStore.

4.3 How to check out a decision table

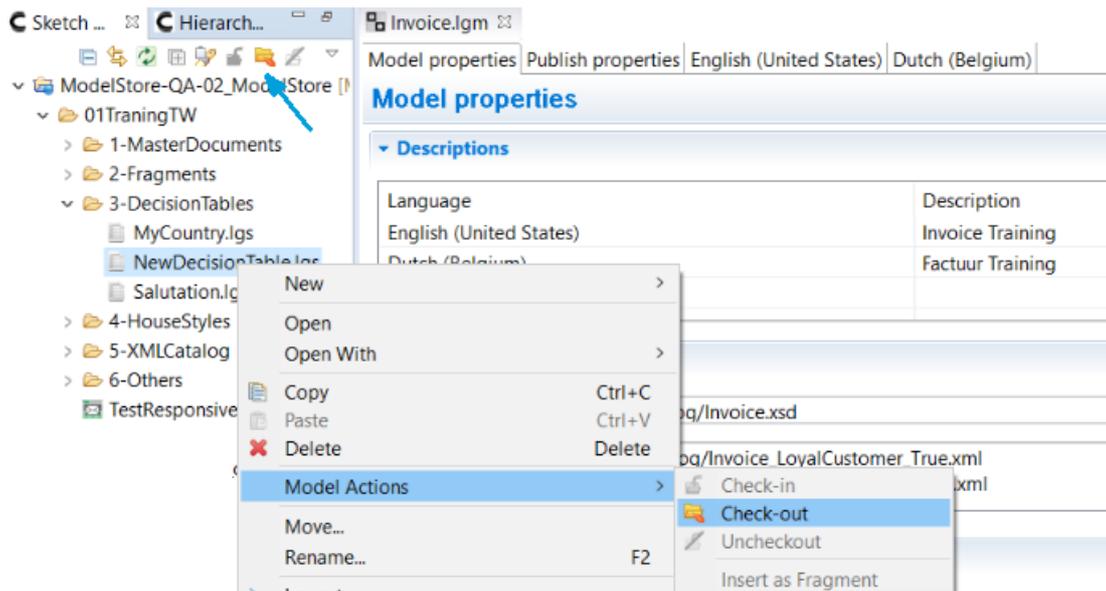
If you want to make changes to a decision table, you need to check out the decision table. This way, the latest version of the decision table will be available to you for modification and the file will be locked for other users.

To check out a decision table:

- Select the decision table.
- Click the **Check out** icon in the toolbar.

Or

- Select the decision table.
- Right-click and select **Model Actions > Check out**.

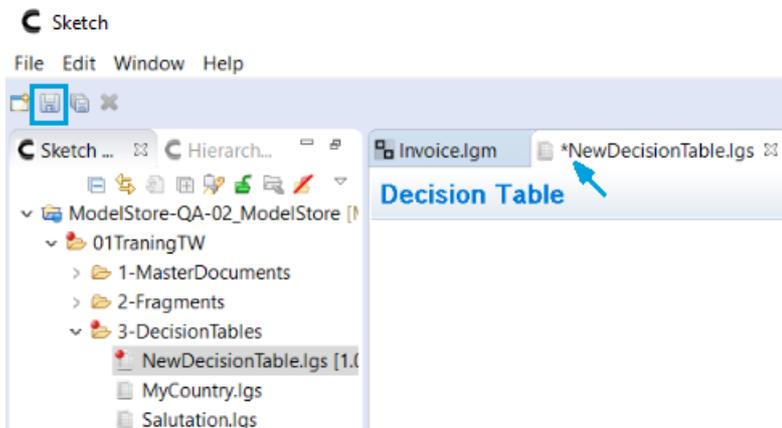


The decision table is checked out and opened and you can start your modifications. The decision table is indicated by a blue symbol in the ModelStore.

4.4 How to save a decision table

When you are creating a decision table, it is recommended to **save** your file on a regular basis. This Save action will save your decision table locally. The file will not be checked in.

A decision table that has been modified (and can be saved) is indicated by an asterisk (*) next to the name:



To save a decision table locally, there are 3 options:

- Click the **Save** icon in the toolbar.
- Click **File > Save**.
- Press **[Ctrl+N]** on the keyboard.

When the decision table has been successfully saved, the asterisk (*) next to the name disappears and the decision table is indicated with a red symbol in the ModelStore. The decision table is still opened and you can continue your modifications.

4.5 How to check in a decision table

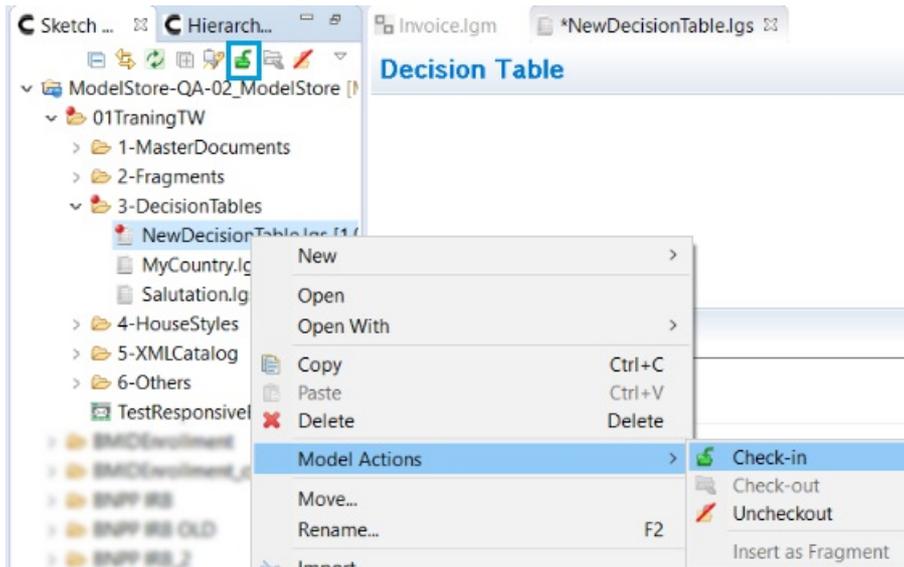
When your changes are done, you need to **check in** the decision table. When you do this, the locked copy in the database is replaced by the updated version. Once this is done, the file is available to the other users.

To check in a decision table, there are 2 options:

- Select the decision table.
- Click the **Check in** icon in the toolbar.

Or

- Select the decision table.
- Right-click and click **Model Actions > Check in**.



- Enter a **Comment** (optional)
- Click **OK**.

The decision table is checked in and indicated by a grey symbol in the ModelStore.

4.6 How to cancel the modifications to a decision table

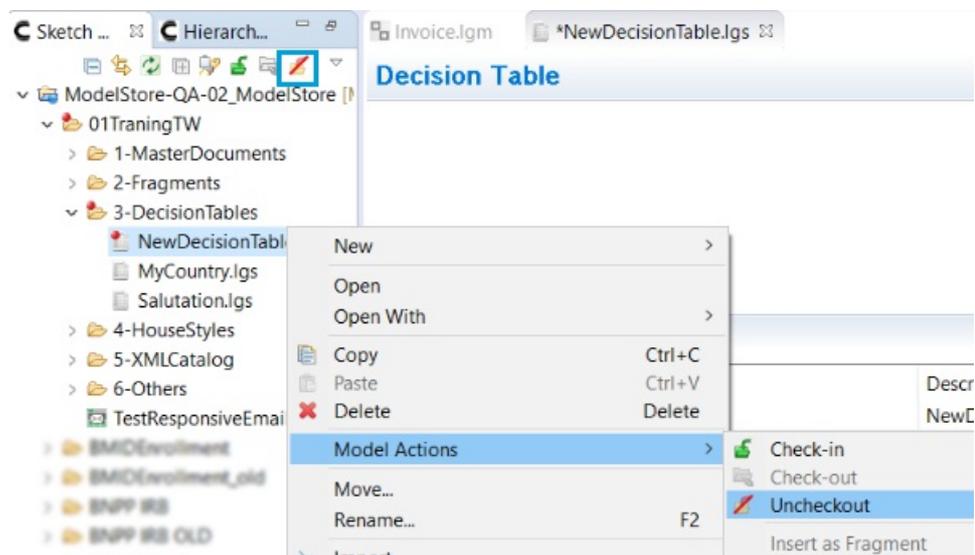
In case you don't want to save the changes you have made, you can **uncheckout** the decision table. This action cancels the checkout of the file and unlocks the locked copy in the database without replacing it with an updated version. All changes made to the decision table are canceled.

To cancel the modifications to a decision table:

- Select the decision table.
- Click the **Uncheckout** icon in the toolbar.
- Click **Yes** to confirm.

Or

- Select the decision table.
- Right-click and click **Model Actions > Uncheckout**.
- Click **Yes** to confirm.



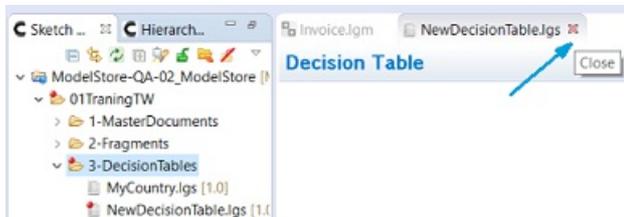
The decision table is closed and indicated by a grey symbol in the ModelStore.

4.7 How to close a decision table

When you close a decision table, the model will be removed from the Model Editor.

To close a decision table, there are 3 options:

- Click the **Close** icon in the toolbar. **Tip:** do not click the X icon in the main toolbar. This deletes the selected element.
- Right-click on the opened model tab and click **Close**.
- Press **[Ctrl+F4]** on the keyboard.



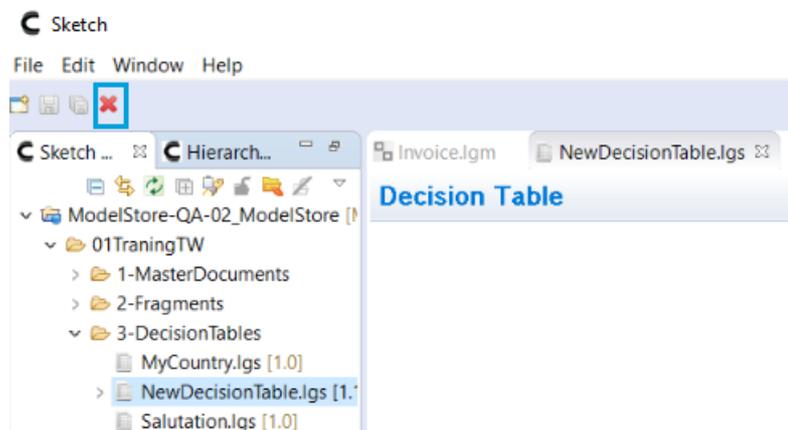
If the decision table has been modified, but hasn't been saved locally yet, you will be prompted to save it.

4.8 How to delete a decision table

You can delete a decision table from the ModelStore when you no longer need the file. Note that you need sufficient rights to perform this action.

To delete a decision table:

- Select the decision table.
- Click the **Delete** icon in the toolbar.



Or

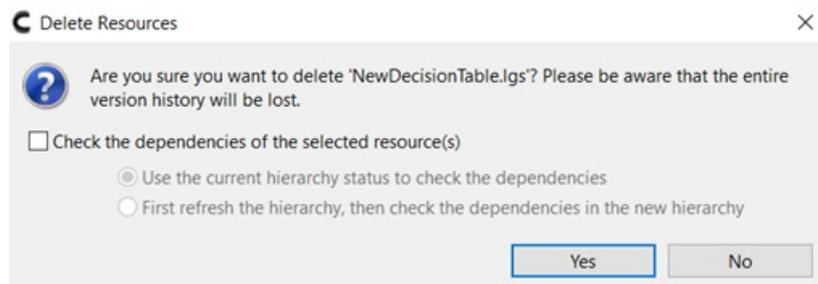
- Right-click the decision table and click **Delete**.

Or

- Select the decision table.
- Press **Delete** on the keyboard.

The deleted decision table is no longer present in the ModelStore.

When you try to delete a model, a message will be displayed warning you that the entire version history will be lost.



Sketch can now check whether the resource you are going to delete has dependencies on other Sketch elements. To do so, click **Check the dependencies of the selected resource(s)**.

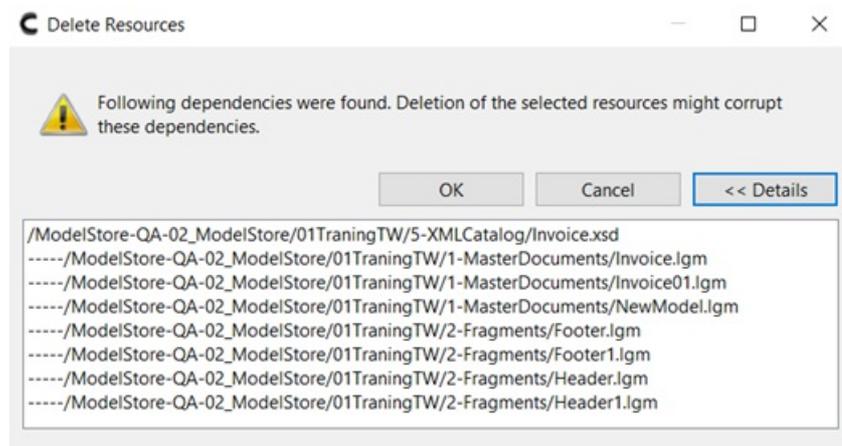
- To check the dependencies of the current hierarchy status, select the first option. **Attention:** if other users are adding elements that are not yet reflected in the Hierarchy Viewer, these elements will not be taken into account.
- To refresh the entire ModelStore first, before checking the hierarchy, select the second option. **Attention:** refreshing the entire ModelStore may take a long time depending on the size of the ModelStore.

Click **Yes** to continue.

If no dependencies are found, the resource(s) will be deleted. They will no longer be present in the ModelStore.

If dependencies are found, a warning message is displayed. Click **Details** to view them.

To delete the resource, click **OK**.

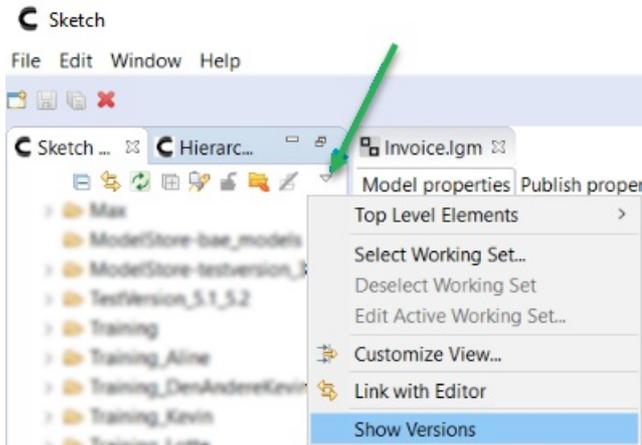


4.9 How to restore a decision table

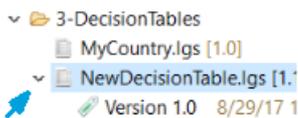
If you want to return to a previous version of a decision table, you can restore an earlier version.

To restore a decision table:

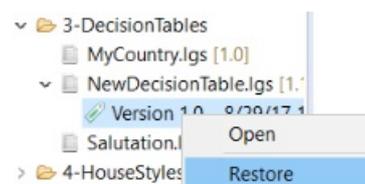
- Click the triangle icon to open the **View** menu.
- Click **Show Versions**.



- Select the decision table you want to restore.
- Click the **triangle** in front of the model name.
- The list of different versions is opened.



- Select the version you want to restore.
- Right-click and click **Restore**.



A new version of the decision table is created, which is the restored version.

4.10 How to use a decision table in a dynamic document

Once your decision table has been created, you can add it to your dynamic document. By doing this, a value from the sample data will be replaced by the value defined in the decision table at generation time. If the variable contains a key that is not defined in the decision table, the variable content will be printed as-is.

4.10.1. Link the decision table to the dynamic document

To link the decision table to the dynamic document:

- [Check out](#) and open the logical model.
- In **Model Properties**, under **Other resources > Decision table**, click **Add**.



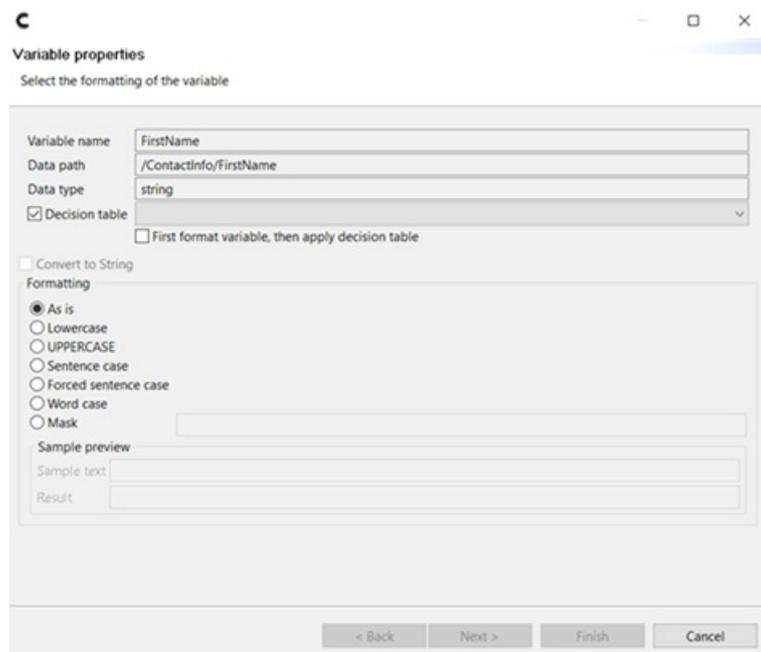
- Select the decision table from the tree view.
- Click **OK**.
- The decision table is now added to the **Model Properties**.

4.10.2. Link the decision table to a variable

In order to replace a value from the XML with a value from the decision table, you have to link the decision table to a variable used in the dynamic document.

To link the decision table to a variable:

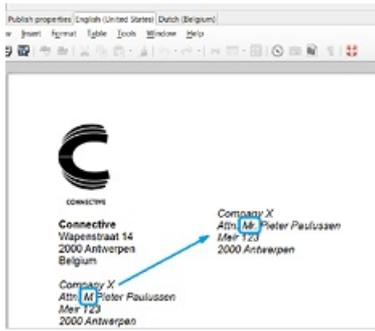
- Open the language model.
- Select the variable.
- The **Variable properties** wizard is opened.



- Check the **Decision table** check box.
- Select the decision table from the selection list.
- Click **Finish**.

4.10.3. Example of linked decision table

When **no decision table** is linked to the variable, the value 'M' from the sample data is printed. When there is a decision table linked to the variable, the value 'M' from the sample data is replaced with the value 'Mr.' that has been defined for the key 'M':



5. House Style Templates

As companies often want to send out documents with a similar look and feel, they need to streamline the layout of their documents. In Sketch, you can link a house style template to your dynamic documents in order to create templates with a similar layout.

In a House Style template you can define a variety of styles, like paragraph styles, list styles, page styles, etc. These styles can be used in your document once the house style template has been linked to the logical model.

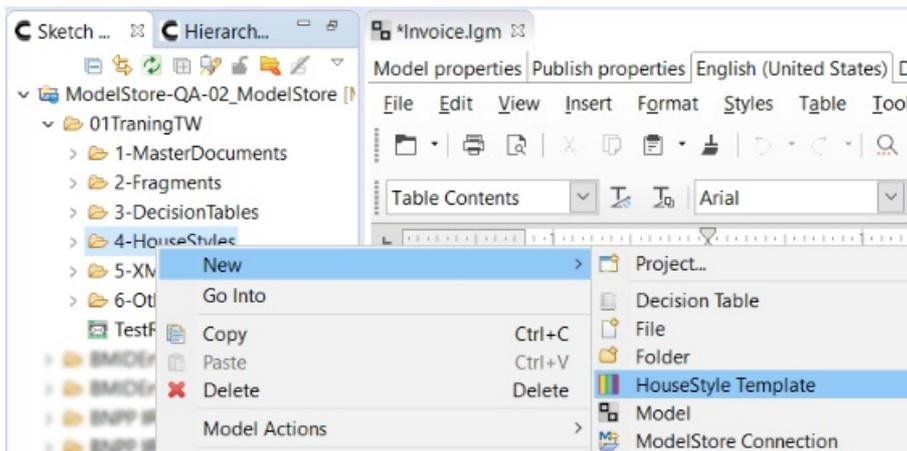
5.1 How to create a house style template

When you want to add a new house style template, you first have to decide where you want the model to be placed. This is very important. Once a house style template is created in a specific folder in the tree view and linked to a logical model, you cannot move the house style template afterwards without impacting the link with the logical model.

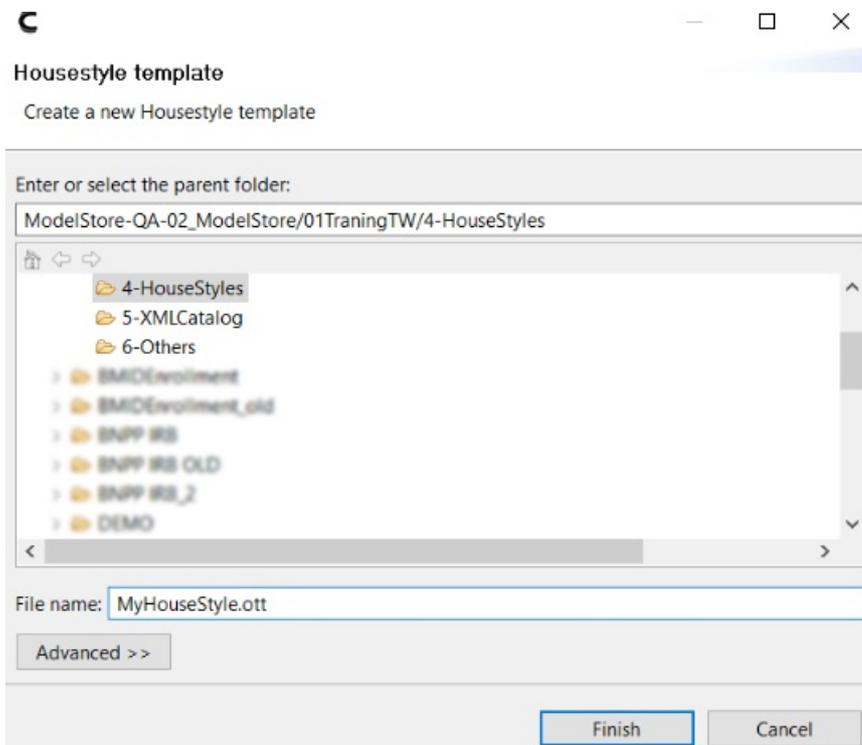
5.1.1. Add a house style template

To add a new house style template:

- Select the folder where you want the house style template to be located.
- Right-click and click **New > HouseStyle Template**.



- The **Housestyle template** wizard appears.
- The folder you've selected is by default indicated as **parent folder**. To change the location of the house style template, click on the nodes to expand/collapse the different folders and then select the folder where you want to save the model.
- Enter the **model name** in the **File name** text box. A decision table needs to have the extension .ott.
- Click **Finish**.

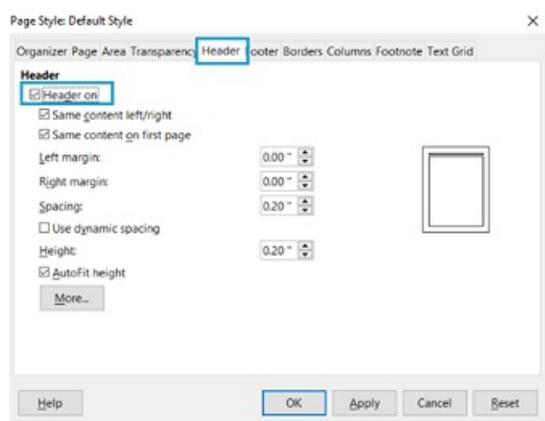


Your new house style template is created in the selected folder. The model is checked out and opened in the Editor region.

5.1.2. Add a header

To insert a header:

- In the house style template, click **Format > Page**.
- In the **Page Style** window, click the **Header** tab.



- Check the **Header on** checkbox.
- Change the properties for the header if necessary.
- Click **OK**.

An empty header is inserted in the house style template. You can create a personalized header by inserting fixed text, a logo, page numbers, a date, etc.

5.1.3. Add a footer

To insert a footer:

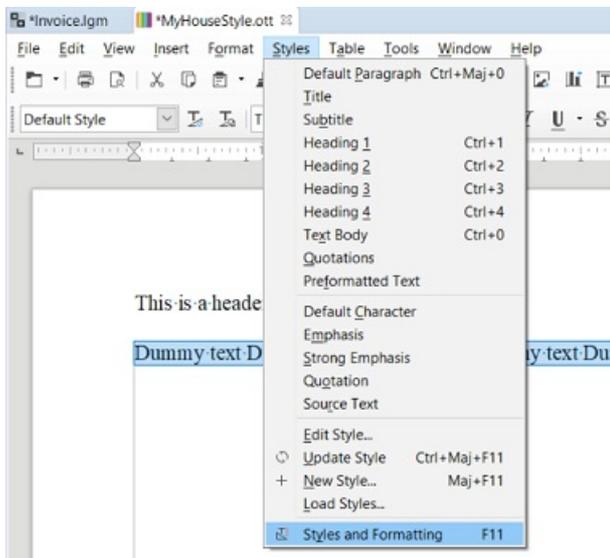
- In the house style template, click **Format > Page**.
- In the **Page Style** properties, click the **Footer** tab.
- Check the **Footer on** checkbox.
- Change the properties for the footer if necessary.
- Click **OK**.

An empty footer is inserted in the house style template. You can create a personalized footer by inserting fixed text, a logo, page numbers, a date, etc.

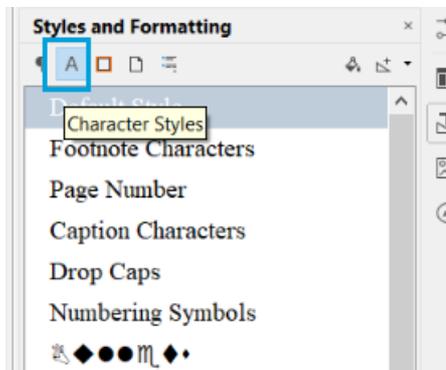
5.1.4. Add a new paragraph style

To create a new paragraph style:

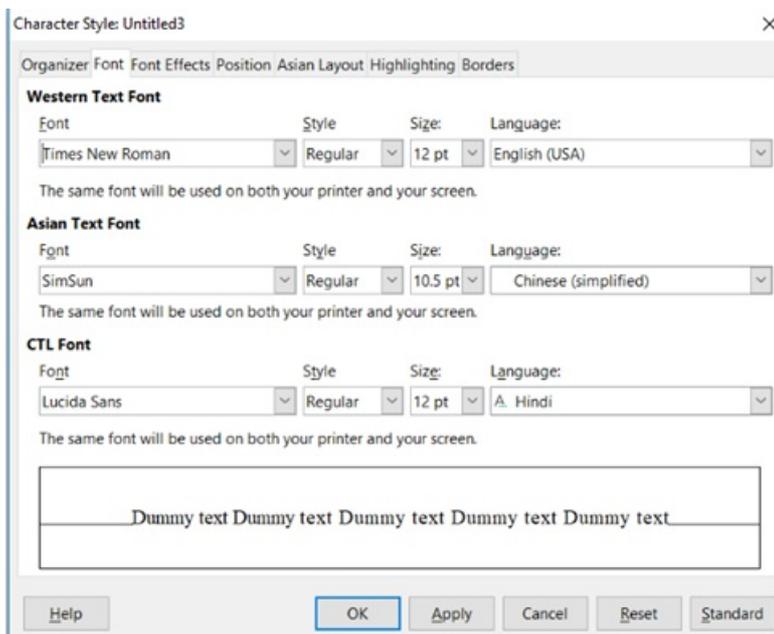
- Insert some **dummy text** in the body of the house style template.
- Select the dummy text and click **Styles > Styles and Formatting**.



- In the **Styles and Formatting** pane, click the **Character Styles** icon.
- Right-click inside the pane and click **New**.



- Click the **Font** tab.
- Specify the layout properties and click **OK** when you are done.

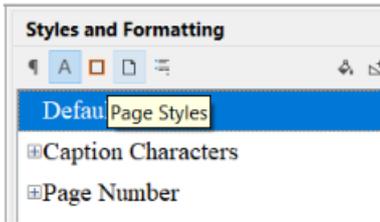


- The new paragraph style is added to the **Styles and Formatting** window.

5.1.5. Add a new page style

To add a new page style:

- Open the house style template.
- Click **Styles > Styles and Formatting**.
- In the **Styles and Formatting** window, click the **Page Styles** icon.



- Right-click and click **New...**
- Specify the **layout properties** for the new page style.
- Click **OK**.
- The **new page style** is added to the **Styles and Formatting** window.
- In the house style template, right-click on **Default Style** at the bottom of the page.
- Select the new page style.
- The new page style is now applied to the house style template.

5.1.6. Modify a style

To modify an existing style:

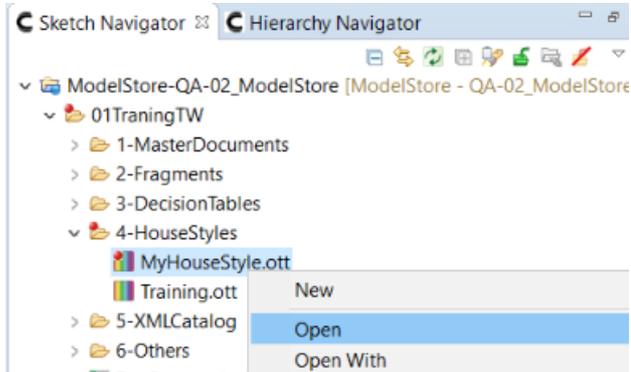
- Open the house style template.
- Click **Styles > Styles and Formatting**.
- In the **Styles and Formatting** pane, select the style you want to modify.
- Right-click and click **Modify**.
- Modify the layout properties of the selected style.
- Click **OK**.
- The modifications are reflected in the house style template.

5.2 How to view a house style template (read-only)

It is possible to view a house style template without checking out the file. In this case, a read-only copy of the house style template will be opened, so it is not possible to make any modifications.

To view a house style template:

- Select the house style template in the Sketch Navigator.
- Right-click and click **Open**.



Or

- Double-click the house style template.

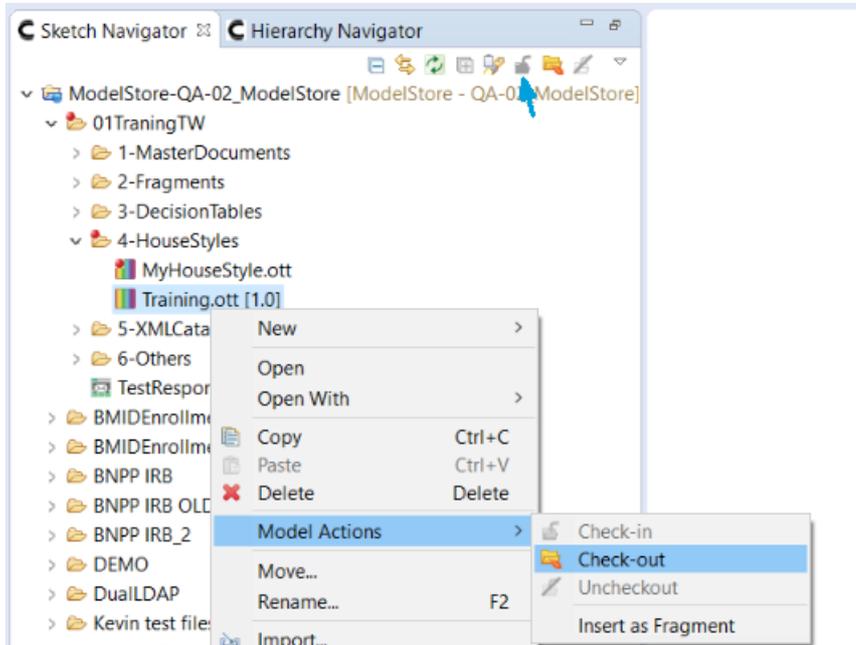
A read-only copy of the house style template is opened. The house style template is indicated with a grey symbol in the ModelStore.

5.3 How to check out a house style template

If you want to make changes to a house style template, you need to **check out** the file. This way, the latest version of the house style template will be available to you for modification and the file will be locked for other users.

To check out a house style template:

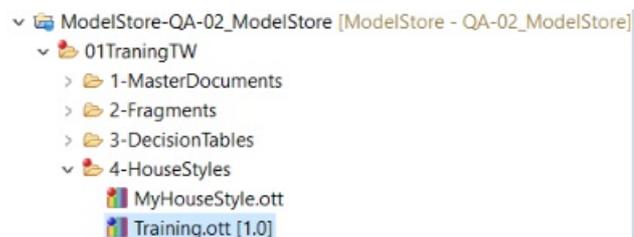
- Select the house style template.
- Click the **Check out** icon in the toolbar.



Or

- Select the house style template.
- Right-click and click **Model Actions > Check out**.

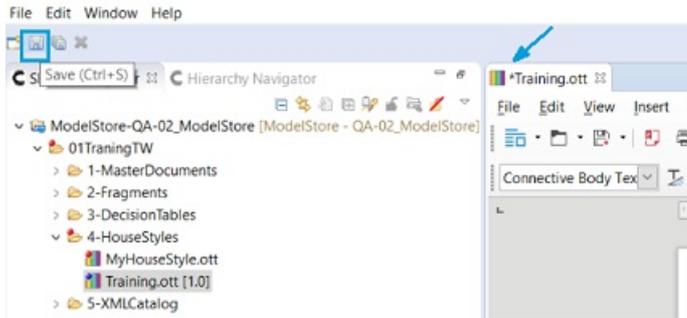
The house style template is checked out and opened and you can start your modifications. The house style template is indicated with a blue symbol in the ModelStore.



5.4 How to save a house style template

When you are creating a house style template, it is recommended to save your file on a regular basis. This Save action will save your house style template locally. The file will not be checked in.

A house style template that has been modified (and can be saved) is indicated with an asterisk (*) next to the name.



To save a house style template locally, there are 3 options:

- Click the **Save** icon in the toolbar.
- Click **File > Save**.
- Press **[Ctrl+N]** on the keyboard.

When the house style template has been successfully saved, the asterisk (*) next to the name disappears and the house style template is indicated with a red symbol in the ModelStore. The model is still opened and you can continue your modifications.

5.5 How to check in a house style template

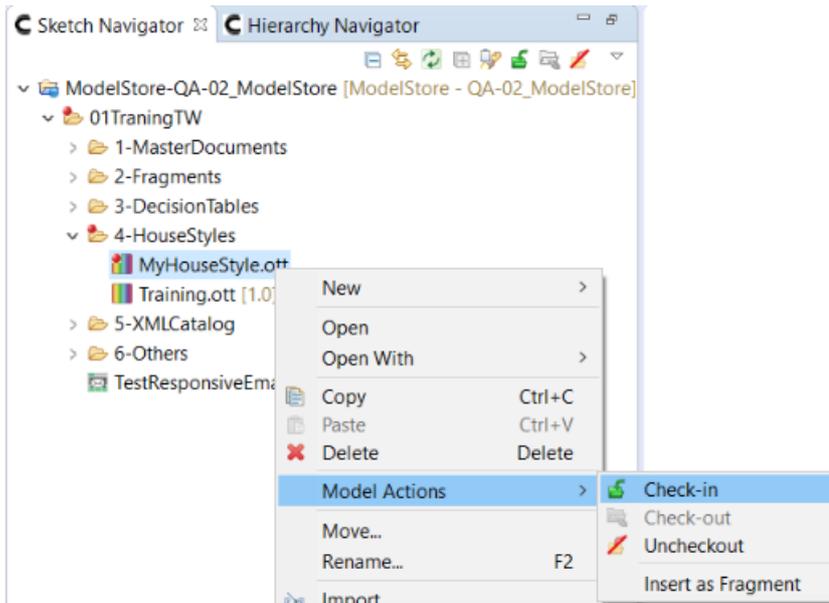
When your changes are done, you need to **check in** the house style template. When you do this, the locked copy in the database is replaced by the updated version. Once this is done, the file is available to the other users.

To check in a house style template, there are 2 options:

- Select the house style template.
- Click the **Check in** icon in the toolbar.

Or

- Select the house style template.
- Right-click and click **Model Actions > Check in**.



- Enter a **Comment** (optional).
- Click **OK**.

The house style template is checked in and indicated with a grey symbol in the ModelStore.

5.6 How to cancel the modifications to a house style template

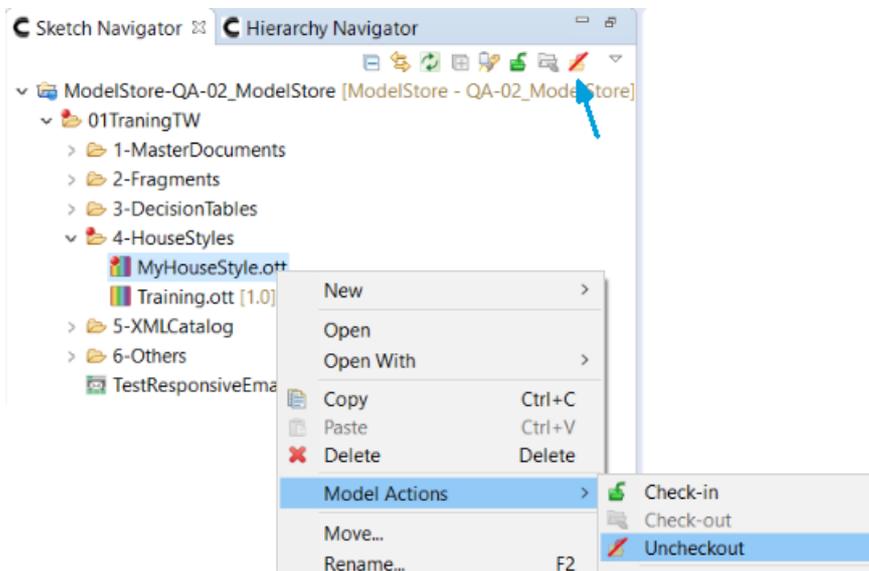
In case you don't want to save the changes you made, you can uncheckout the house style template. This action cancels the checkout of the file and unlocks the locked copy in the database without replacing it with an updated version. All changes made to the house style template are canceled.

To cancel the modifications to a house style template:

- Select the house style template.
- Click the **Uncheckout** icon in the toolbar.
- Click **Yes** to confirm.

Or

- Select the house style template.
- Right-click and click **Model Actions > Uncheckout**.
- Click **Yes** to confirm.



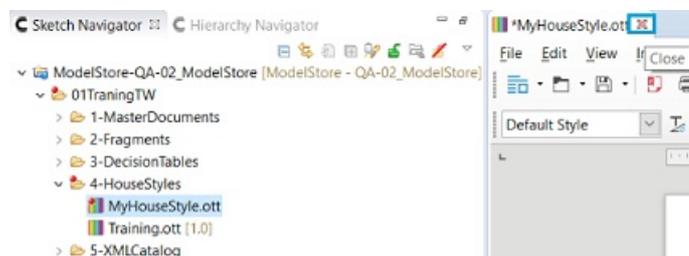
The house style template is closed and indicated by a grey symbol in the ModelStore.

5.7 How to close a house style template

When you close a house style template, the model will be removed from the Model Editor.

To close a house style template, there are 3 options:

- Click the **Close** icon on the appropriate tab. **Tip:** do not click the X icon in the main toolbar. This deletes the selected element.



- Right-click the opened model tab and click **Close**.
- Press **[Ctrl+F4]** on the keyboard.

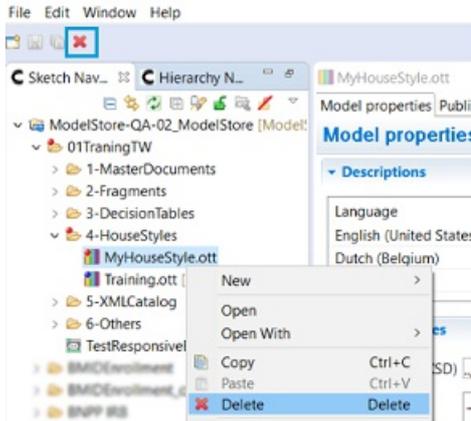
If the house style has been **modified**, but hasn't been saved locally yet, you will be prompted to save it.

5.8 How to delete a house style template

You can delete a house style template from the ModelStore when you no longer need it. Note that you need sufficient rights to perform this action.

To delete a house style template:

- Select the house style template.
- Click the **Delete** icon in the toolbar.



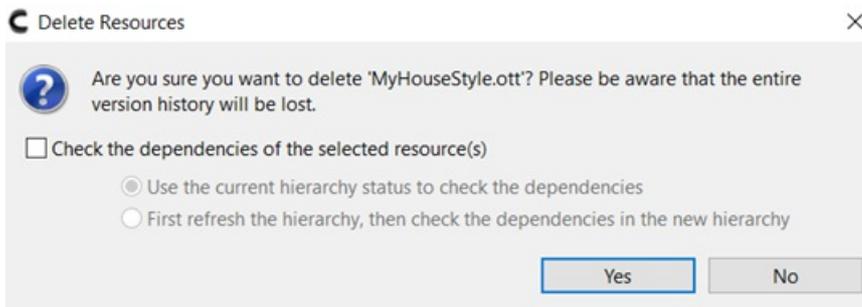
Or

- Right-click the house style template and click **Delete**.

Or

- Select the house style template.
- Press **Delete** on the keyboard.

When you try to delete a house style, a message will be displayed warning you that the entire version history will be lost.



Sketch can now check whether the resource you are going to delete has dependencies on other Sketch elements. To do so, click Check the dependencies of the selected resource(s).

- To check the dependencies of the current hierarchy status, select the first option. **Attention:** if other users are adding elements that are not yet reflected in the Hierarchy Viewer, these elements will not be taken into account.
- To refresh the entire ModelStore first, before checking the hierarchy, select the second option. **Attention:** refreshing the entire ModelStore may take a long time depending on the size of the ModelStore.

Click **Yes** to continue.

If no dependencies are found, the resource(s) will be deleted. They will no longer be present in the ModelStore.

If dependencies are found, a warning message is displayed. Click **Details** to view them.

To delete the resource, click **OK**.



Following dependencies were found. Deletion of the selected resources might corrupt these dependencies.

OK

Cancel

<< Details

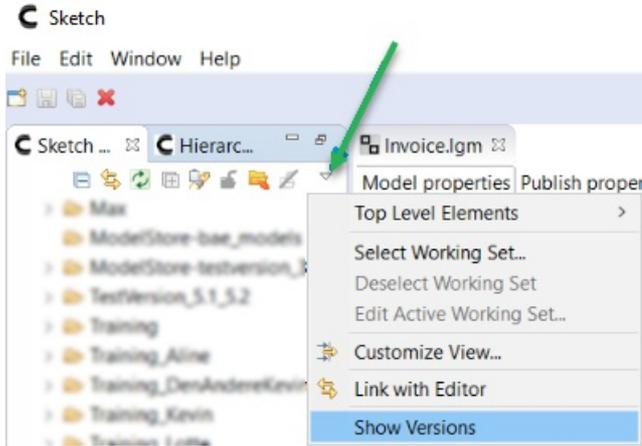
```
/ModelStore-QA-02_ModelStore/01TraningTW/5-XMLCatalog/Invoice.xsd  
-----/ModelStore-QA-02_ModelStore/01TraningTW/1-MasterDocuments/Invoice.lgm  
-----/ModelStore-QA-02_ModelStore/01TraningTW/1-MasterDocuments/Invoice01.lgm  
-----/ModelStore-QA-02_ModelStore/01TraningTW/1-MasterDocuments/NewModel.lgm  
-----/ModelStore-QA-02_ModelStore/01TraningTW/2-Fragments/Footer.lgm  
-----/ModelStore-QA-02_ModelStore/01TraningTW/2-Fragments/Footer1.lgm  
-----/ModelStore-QA-02_ModelStore/01TraningTW/2-Fragments/Header.lgm  
-----/ModelStore-QA-02_ModelStore/01TraningTW/2-Fragments/Header1.lgm
```

5.9 How to restore a house style template

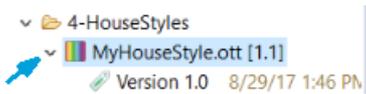
If you want to return to a previous version of a house style template, you can restore an earlier version.

To restore a house style template:

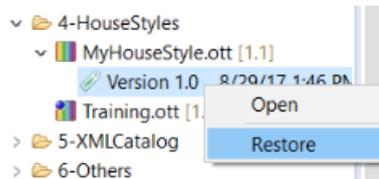
- Click the triangle icon to open the **View** menu.
- Click **Show Versions**.



- Select the house style you want to restore.
- Click the **triangle** in front of the house style name.
- The list of different versions is opened.



- Select the version you want to restore.
- Right-click and select **Restore**.



A new version of the logical model is created, which is the restored version.

5.10 How to use a house style template in a dynamic document

Once your decision table has been created, you can add it to your dynamic document. By doing this, a value from the sample data will be replaced by the value defined in the decision table at generation time. If the variable contains a key that is not defined in the decision table, the variable content will be printed as-is.

5.10.1. Link the house style template to the dynamic document

To link the house style template to a new language model:

- [Check out](#) and open the logical model.
- Under **Language models**, click **Add**.



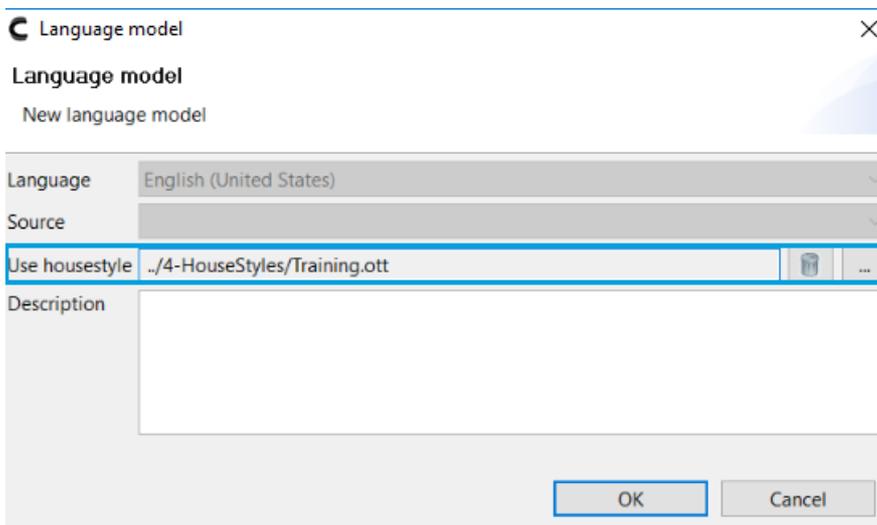
- Select the language from the **Language** list.
- Next to **Use housestyle**, click the browse button (...).
- Select the house style template you want to link to your language model.
- Click **OK**.
- The house style template is now linked to your language model.
- Click **OK**.

A new language model is created. It's a predefined template, representing the company house style and contains all paragraph styles, page styles etc. which had been associated with the house style template.

5.10.2. Check if an existing language model has a house style linked

To check whether a house style has been linked to an existing language model:

- [Check out](#) and open the logical model.
- Under **Language models**, select the language model for which you want to verify if a house style has been linked.
- Click **Edit**.
- If a house style has been linked, you see the house style .ott file in the **Use housestyle** field. If no house style has been linked this field is empty.



5.10.3. Change the house style in an existing language model

Changing the house style in an existing language model is not possible as such.

If needed, there is a workaround:

- Create a new language model in another language, based on the original language model, and link the new house style to this language model.
- Delete the original language model.
- Duplicate the newly created language model and recreate a language model in the original language.

5.10.4. Use the defined styles

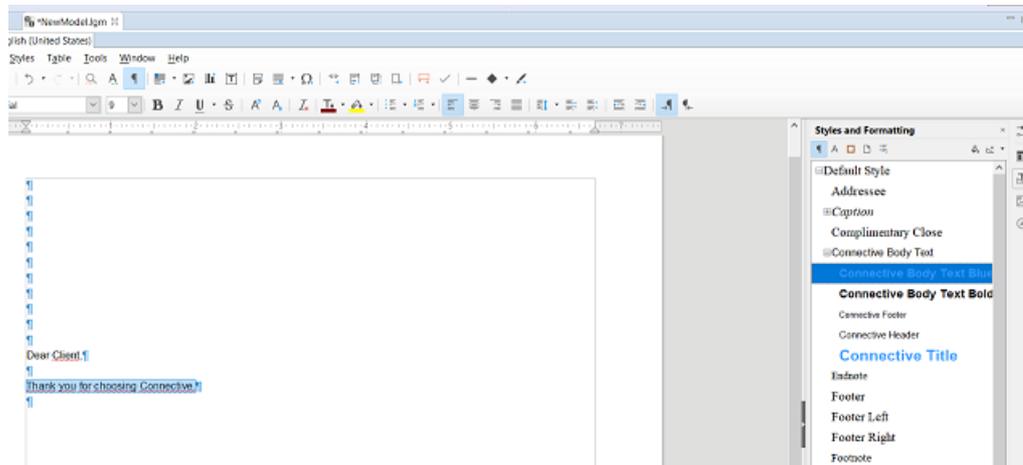
To use the styles that have been defined in the house style template in your dynamic document:

- [Check out](#) the logical model.
- Open the required language model.
- Open the styles that have been defined in the house style template:
 - Click **Styles > Styles and Formatting**.
 - Or press F11.

The **Styles and Formatting** pane is opened. It contains a list of all available styles.

To apply a specific paragraph style:

- Select the paragraph(s) in the Editor.
- Double-click on the style in the **Styles and Formatting** pane. The style will be applied to the paragraph.



The different icons available in the **Styles and Formatting** palette are:

ICON	DESCRIPTION
	Paragraph styles
	Character styles
	Frame styles
	Page styles
	List styles
	Apply paragraph style , to apply the selected style to the selection
	New style

6. Data Structure

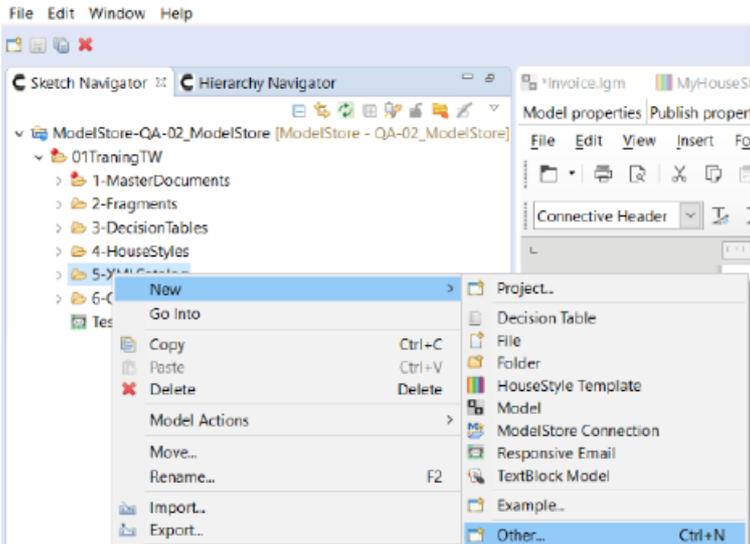
When you create a dynamic document, an XSD file needs to be linked to your logical model. This way, you can insert the variables in your dynamic document, add logic, etc.

For the basic information on XSD/XML, see [1.3.13 XML and XSD \(Data structure\) documents](#).

6.1 How to create an XSD

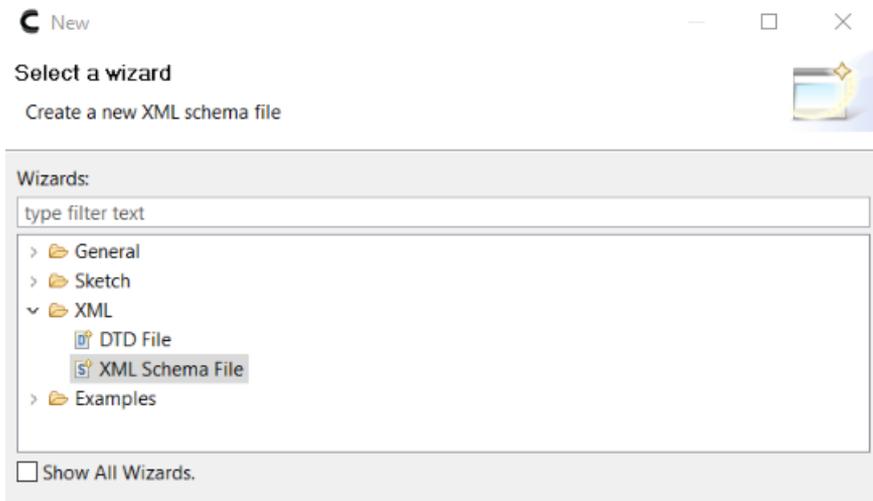
To create a new data structure file:

- Select the folder where you want the XSD to be located.
- Right-click and click **New > Other**.



The New wizard appears:

- Click **XML > XML Schema File**.
- Click **Next**.
- Enter the **file name**.
- Click **Finish**.



Your new data structure file is created in the selected folder. The XSD is checked out. Double-click the file to open the XSD in the Editor region.

6.2 How to edit an XSD

Once your new XSD has been created, you can modify the file in order to build your own data structure that you can use in your dynamic document.

Sketch offers you two different editors to modify your XSD: the **Sketch XSD Editor** and the **XML Schema Editor**.

Changing a data structure (XSD) will have an impact on existing variables, conditions and loops used in an existing language model.

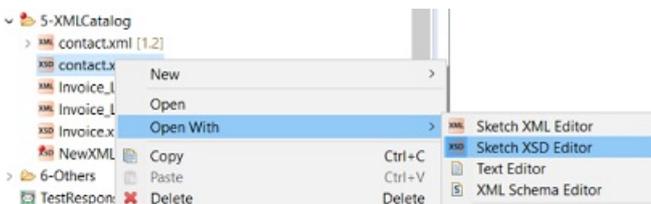
6.2.1. Sketch XSD Editor

The Sketch XSD Editor allows you to easily build or edit a data structure. You can add data blocks and variables in a very visual way, create calculated variables and add business rules on data blocks and/or variables.

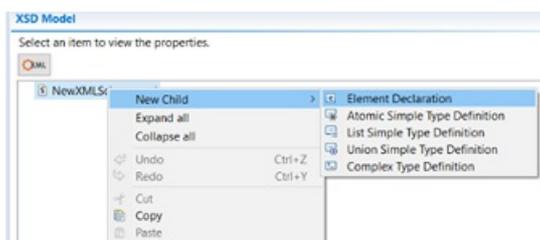
Advanced XSD Editor

The **Advanced XSD Editor** tab can be used to create and modify a data structure:

- In the Sketch Navigator check out the XSD file you want to edit.
- Right-click it and click **Open With > Sketch XSD Editor**.



- The XSD is opened in the Editor region.
- Make sure the **Advanced XSD Editor** tab (at the bottom of the window) is selected.
- Right-click the **root element**.
- A **context menu** is opened, where you can select the elements you want to add and edit them once added.



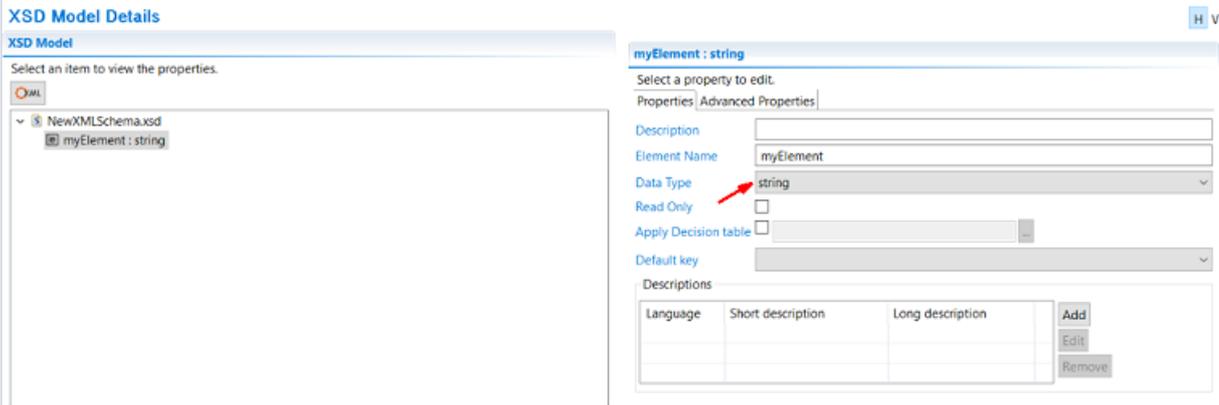
Important notes about editing an XSD:

When building an XSD from scratch, a certain structure needs to be maintained. First, you need to create a data block that will contain all other data blocks. Once the main data block is created you can create other data blocks to which you can add variables.

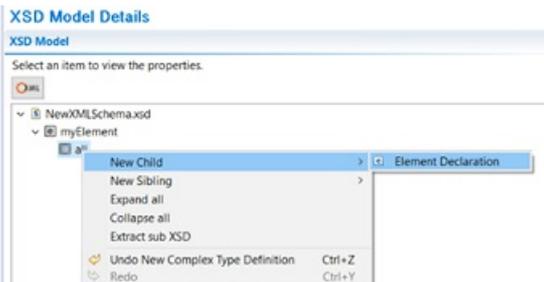
A data block is a collection of data blocks and/or variables. A variable is one piece of data.

To add a data block:

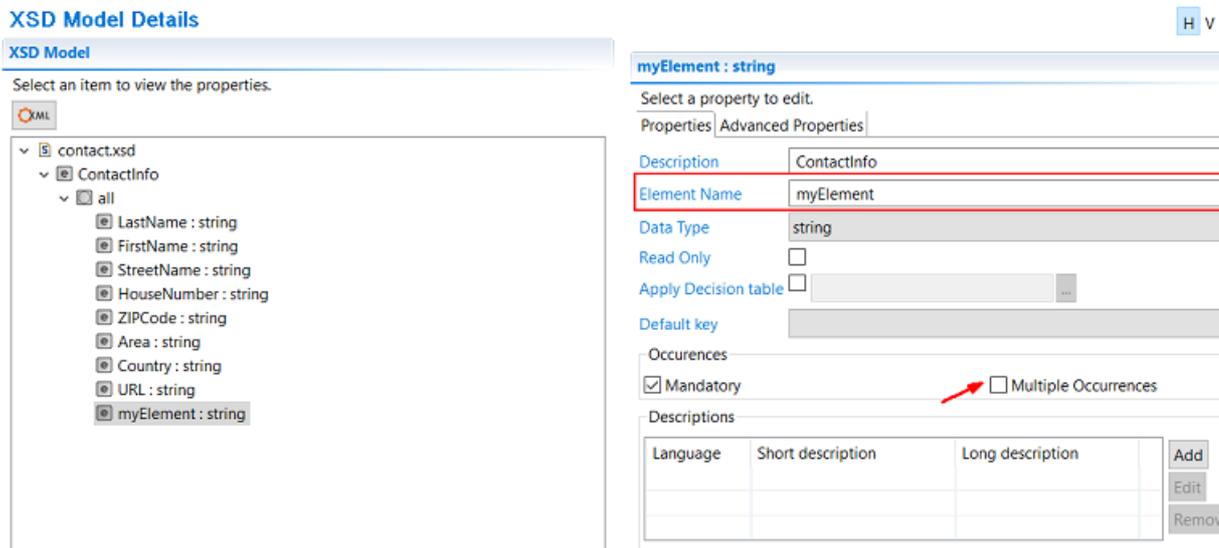
- Right-click on the root element, and click **New Child > Element Declaration**.
- Click the element you created.
- In the **Element Name** field, type a name for the variable.
- Select a variable type from the **Data Type** drop-down list (or **Browse** if not listed).



- Right-click the element you created, and click **New Child > Complex Type Definition**.
- The "all" block is now added beneath the element.
- Right-click the "all" block and click **New Child > Element Declaration**.



- In the **Element Name** field, type a name for the variable.
- Select a variable type from the **Data Type** drop-down list (or **Browse** if not listed).
- Check the **Mandatory** check box if the variable needs to be mandatory.
- Check the **Multiple Occurrences** check box if the variable needs to be repeated.



- Repeat the steps above to continue other variables.

To add a **new data block** at the **same level** of an existing data block:

- Right-click on a data block.
- Select **New Sibling > Element Declaration**.
- A new element is added.
- Right-click on this new element.
- Select **New Child > Complex Type Definition**.

To add a **new data block nested** in an existing data block:

- Right-click on the **ALL** node of a data block.
- Select **New Child > Element Declaration**.
- A new element is added.
- Right-click on this new element.
- Select **New Child > Complex Type Definition**.

Now you can fill in the appropriate fields in the Properties Section. Save by doing a check-in of the file.

XSD Model Details

The screenshot shows the 'XSD Model Details' window. On the left, a tree view shows the XSD model structure: 'NewXMLSchema.xsd' contains 'NewVariable : string' and 'myElement'. 'myElement' contains 'all', which contains 'myElement', which in turn contains 'all'. The right pane is titled 'myElement' and has a 'Select a property to edit' dropdown. Below it are two tabs: 'Properties' and 'Advanced Properties'. The 'Advanced Properties' tab is active, showing fields for 'Description', 'Element Name' (myElement), 'Data Type' (myElement_type), 'Read Only' (checkbox), 'Apply Decision table' (checkbox), and 'Default key'. Below these are 'Occurrences' settings: 'Mandatory' (checked) and 'Multiple Occurrences' (checkbox). At the bottom is a 'Descriptions' table with columns 'Language', 'Short description', and 'Long description', and buttons 'Add', 'Edit', and 'Remove'.

Business rule XSD Editor

With the Business rule XSD Editor, you can add business rules on data blocks and/or variables. These business rules will be used to show or hide the data blocks and/or variables in the Forms application.

To add or edit business rules:

- Check out the XSD.
- Right-click and click **Open With > Sketch XSD Editor**.

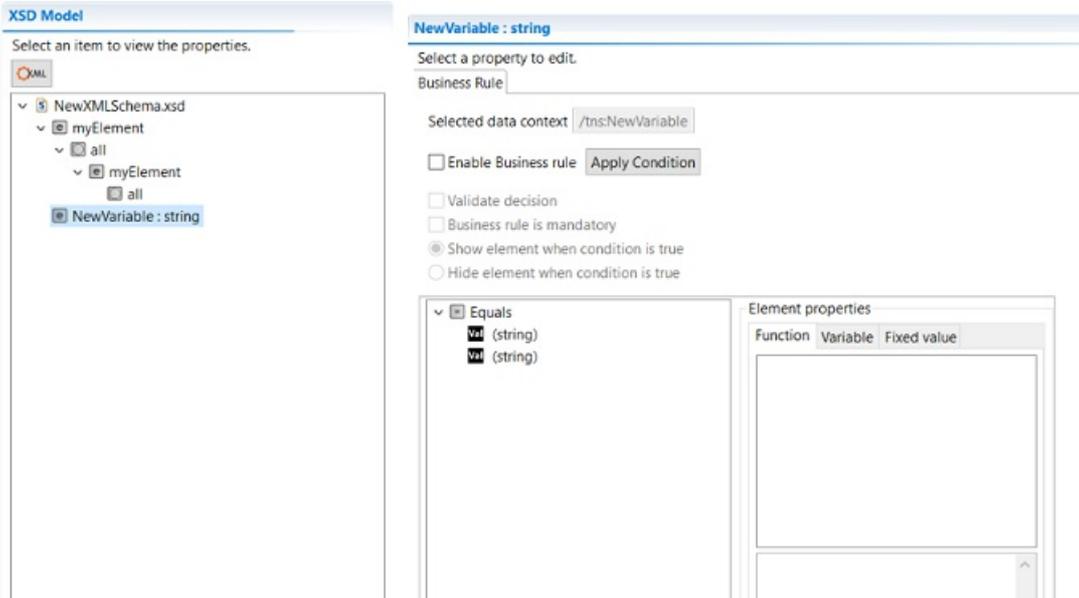
The screenshot shows a file explorer window with a context menu open over a file named '5-XMLCatalog'. The menu items are: 'New', 'Open', 'Open With', 'Copy', 'Paste', and 'Delete'. The 'Open With' option is selected, and a submenu is open showing: 'Sketch XML Editor', 'Sketch XSD Editor', 'Text Editor', and 'XML Schema Editor'.

- The XSD is opened in the Editor region.
- Click the **Business rule Editor** tab at the bottom of the screen.

The screenshot shows the bottom of the editor window with three tabs: 'Advanced XSD Editor', 'Business rule Editor', and 'Calculated Variable'. The 'Business rule Editor' tab is selected and highlighted.

- Select a data block or a variable.
- In the right pane, the **Business rule editor** is opened.

XSD Model Details



To add a business rule:

- Check the **Enable business rule** checkbox.
- [Create a condition.](#)
- Select **Apply Condition**.

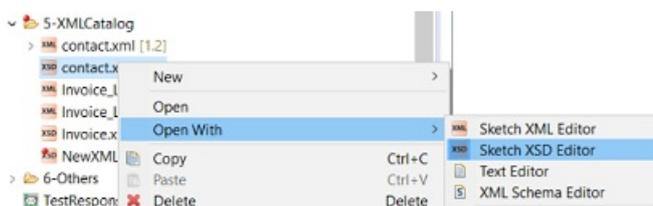
The business rule has now been created. Save by doing a check-in of the XSD file.

Calculated

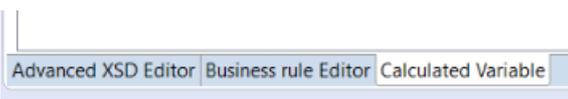
Sketch allows you to create calculated variables. A calculated variable executes a function on an existing variable that has been defined in the XSD. Example: based on the variable Cost Price per year, the calculated variable Cost Price per month can be calculated, by dividing the price per year by 12.

To add a calculated variable:

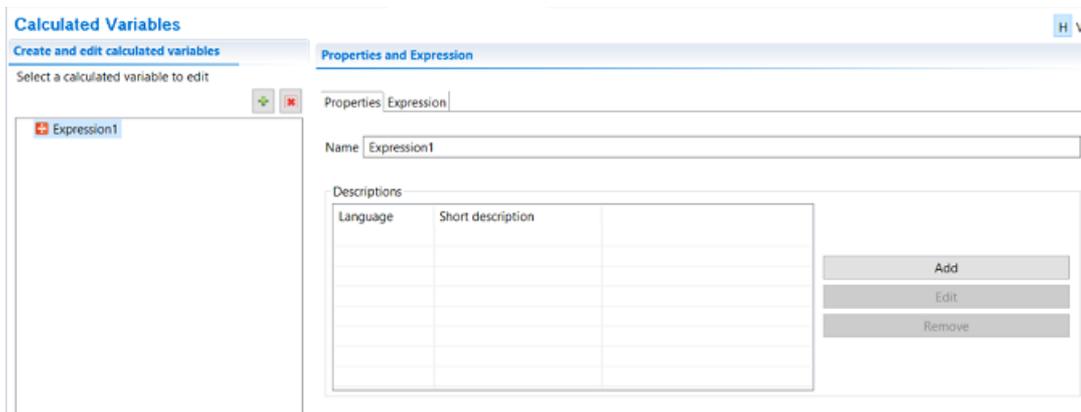
- Check out the XSD.
- Right-click and select **Open With > Sketch XSD Editor**.



- The XSD is opened in the Editor region.
- Open the **Calculated Variable** tab at the bottom of the window.

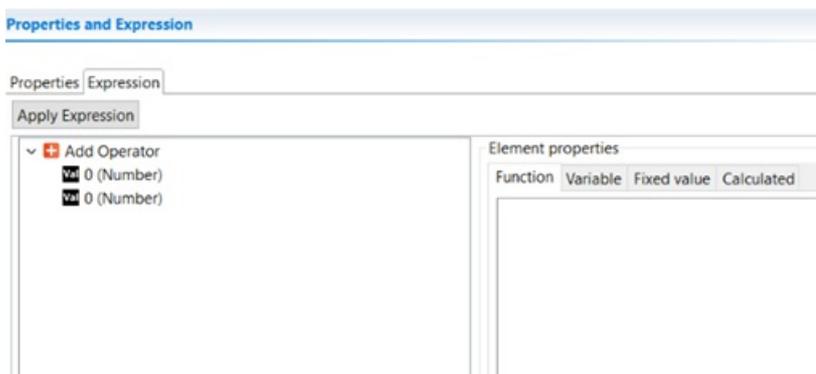


- Select the **green '+'** button.
- A new variable is added, named "Expression1".
- Select this variable.
- In the right pane, the **Property Section** tab is opened where you can give a name to the calculated variable and add descriptions in different languages.



- When you open the **Expression** tab, you can add the expression for the calculated variable.
- Select **Apply Expression** to finish.

The calculated variable has now been created. Save by doing a check-in of the XSD file.

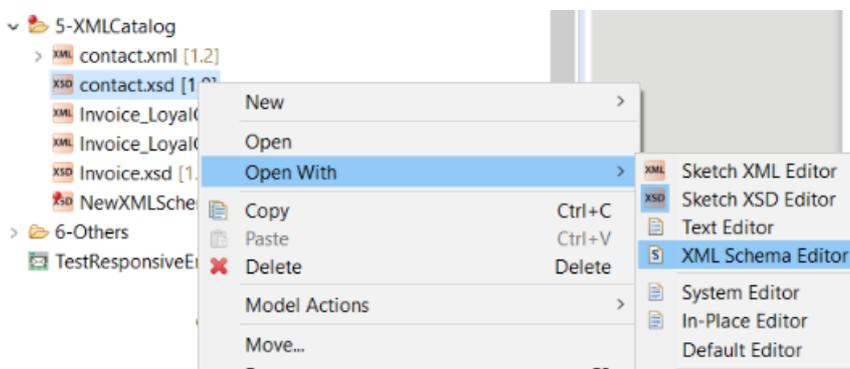


6.2.2. XML Schema Editor

The XML Schema Editor allows you to create or edit a data structure in a short time. You can copy/paste data blocks and/or variables, and edit them by changing names, data types etc. like you would in Notepad or any other text editor. This makes it easy to create or modify large data structures.

To open an XSD in the XML Schema Editor:

- Check out the XSD.
- Right-click and click **Open With > XML Schema Editor**.



- The **Design** tab (at the bottom of the window) gives you a visual overview of the different data blocks and elements.
- In the **Source** tab (at the bottom of the window), you can find a textual representation of the data structure.

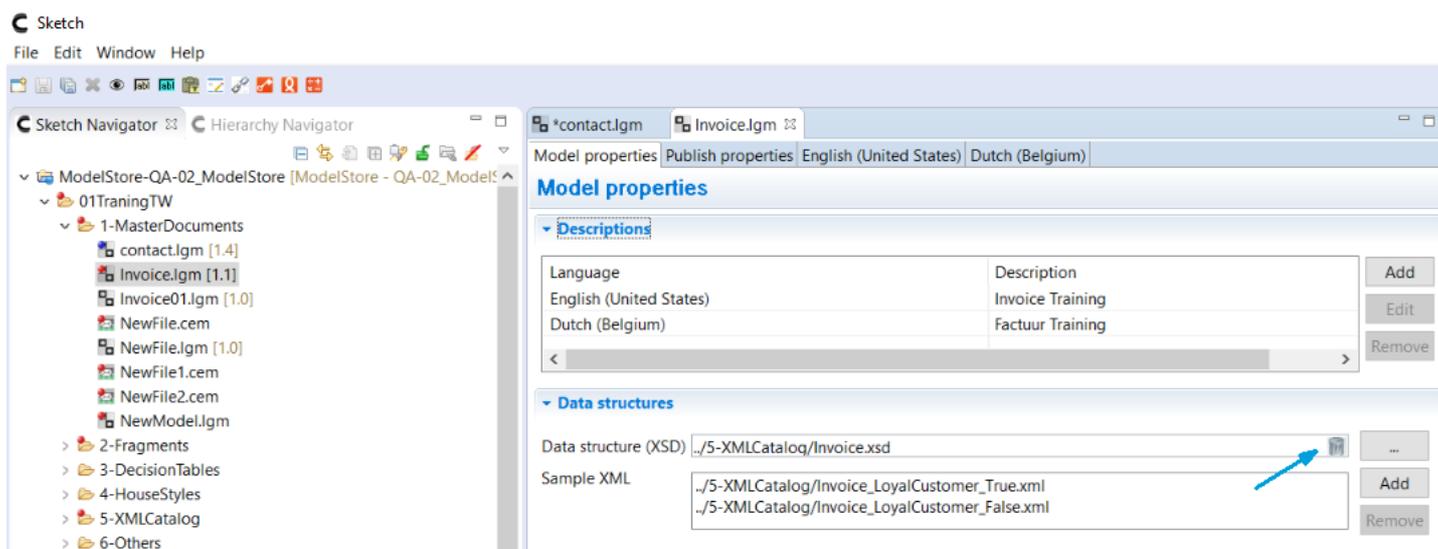
6.2.3. Associate an XSD with a logical model

For more information on how to add an XSD/XML to a logical model, see [3.2.1 Create a logical model](#).

6.2.4. Unlink an XSD from a logical model

In case you need to unlink an XSD file - and the corresponding XML files - from a logical model:

- Check out the logical model.
- Click the **Model Properties** tab.



- Click the unlink icon next to **Data structure (XSD)**.

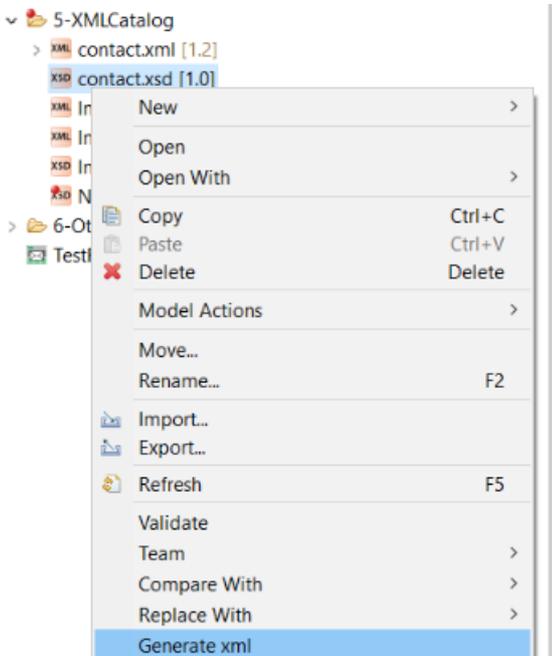
When you unlink the XSD, keep in mind that this affects the already inserted variables and constructed logic in the document and they will need to be reviewed.

6.3 How to create an XML

6.3.1 Create an XML based on an XSD

To automatically create an XML based on an existing XSD file:

- Select the **XSD** in the Sketch Navigator. **Important:** make sure the XSD is checked in.
- Right-click and click **Generate XML**.



- The **New XML File** wizard is opened.
- Select a different parent folder if needed.
- Enter a **File name**.
- Click **Next**, and then click **Finish**.

The XML is automatically created and opened in the Editor region.

You can modify the XML if needed and check in the file afterwards. Now you can use the sample data file in a logical model.



6.3.2. Associate an XML with a logical model

Sample data files can be removed from the logical model or replaced by another sample data. When you remove or replace an XML, keep in mind that the new XML must be valid against the associated XSD.

6.4 Restrictions for XSDs

You are recommended to use the built-in **XSD Editor** in Sketch to create XSDs. The XSD Editor allows you to create XSD structures that are supported in Sketch.

If you are using other business applications to build XSDs or already have existing XSD files, make sure they meet the following restrictions.

- Only use **xs:complexType** with **all** as child. Other attributes are not validated by Sketch and are therefore not guaranteed to work.

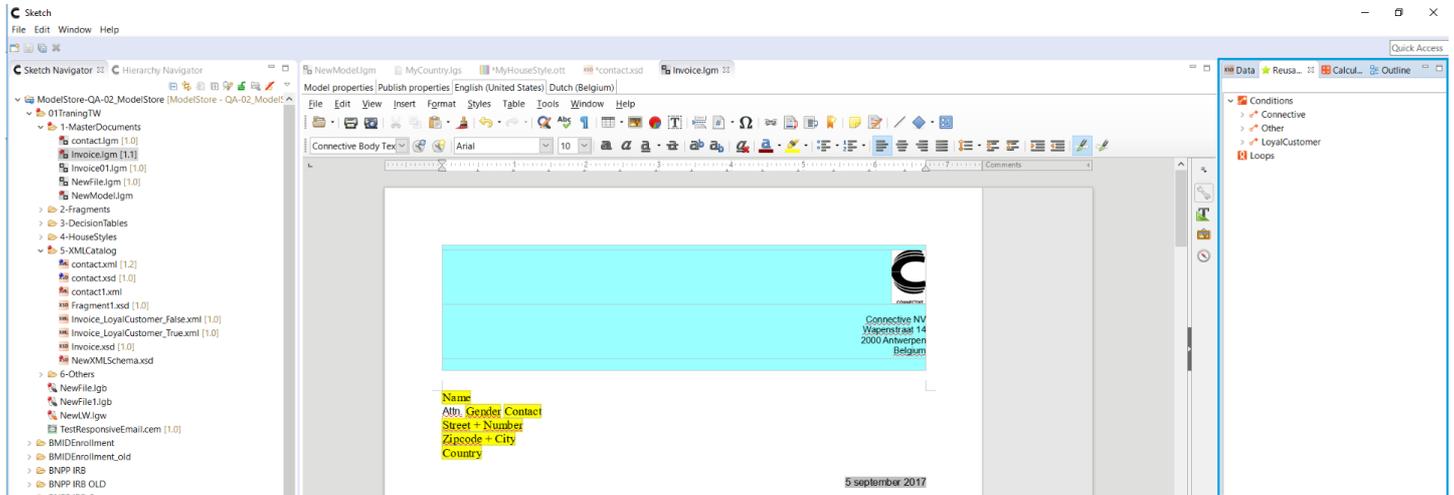


- Only use the following attributes for **<xs:element>**:
 - minOccurs
 - maxOccurs
 - Name
 - Type:
 - STRING
 - INTEGER
 - INT
 - LONG
 - SHORT
 - UNSIGNED_INT
 - UNSIGNED_LONG
 - UNSIGNED_SHORT
 - UNSIGNED_BYTE
 - DECIMAL
 - FLOAT
 - DOUBLE
 - DATETIME
 - DATE
 - TIME
 - gYEAR_MONTH
 - gYEAR
 - gMONTH_DAY
 - gDAY
 - gMONTH
 - BOOLEAN

7. Sketch-specific Building Blocks

Sketch allows you to use specific building blocks in order to make your document 'dynamic'. All these Sketch-specific building blocks are available in the **Objects Palette**.

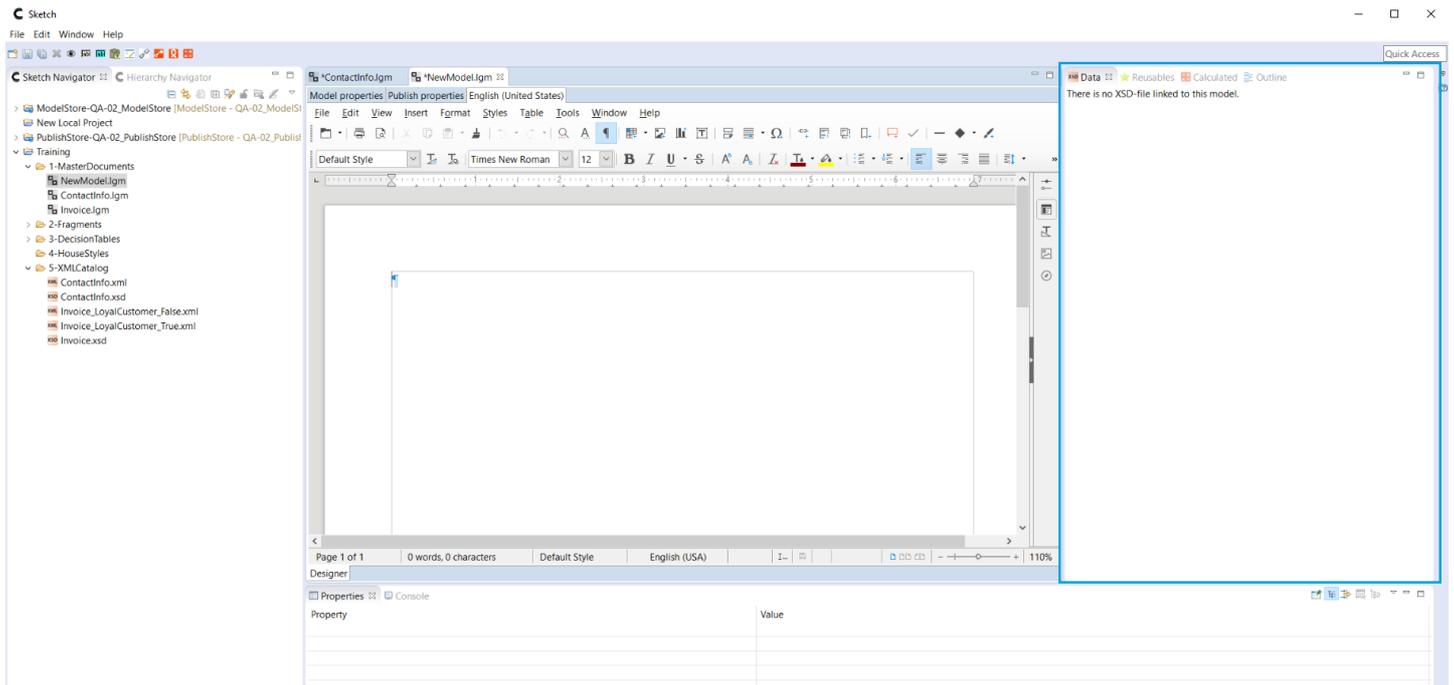
Once your language model is opened, you will find the Objects Palette on the right side of the model editor:



The three most important objects are **Data**, **Reusables** and **Calculated**. Every object corresponds to a separate tab in the objects palette.

7.1 Data

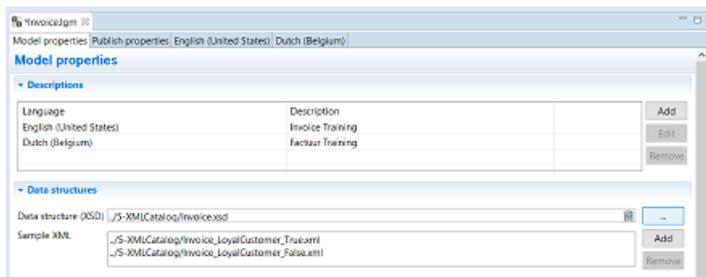
Before you can add data to your dynamic document, you have to make sure a data structure file is linked to the logical model. If no data structure is linked to your model, the Data tab in the objects palette will be empty.



7.1.1. Link XSD to a logical model

To link the data structure and a valid sample data to your model:

- [Check out](#) and open the logical model.
- Under **Data Structures**, click the browse button (...).
- Browse and select the **XSD** file you want to associate with the logical model.
- Click **OK**.

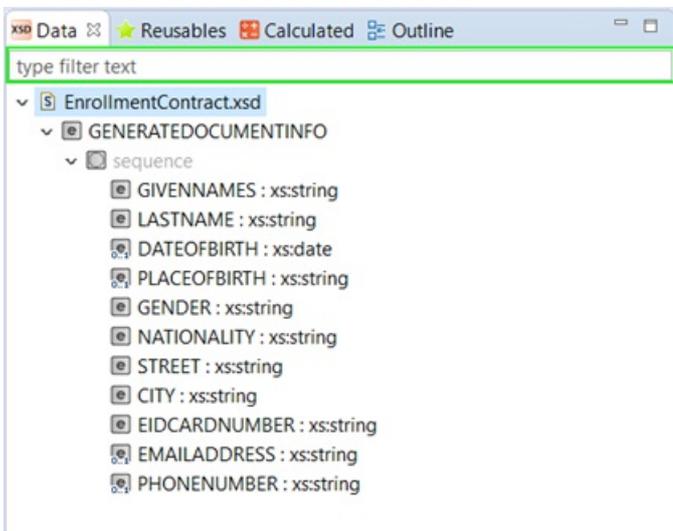


To add a sample XML:

- Under **Data Structures**, click **Add**.
- Browse and select one or more **XML** files you want to add to the logical model.
- Click **OK**.
- If the XML is not compatible with the XSD, an error message will appear.

Once the XSD is linked to the logical model, you will see the data structure reflected in the Data tab on the right side of the model editor when you open a language model. The search field allows you to search for elements inside the data. Enter the element you are searching for and press Enter on our keyboard.

The variables, defined in the data structure, can now be inserted in your document.



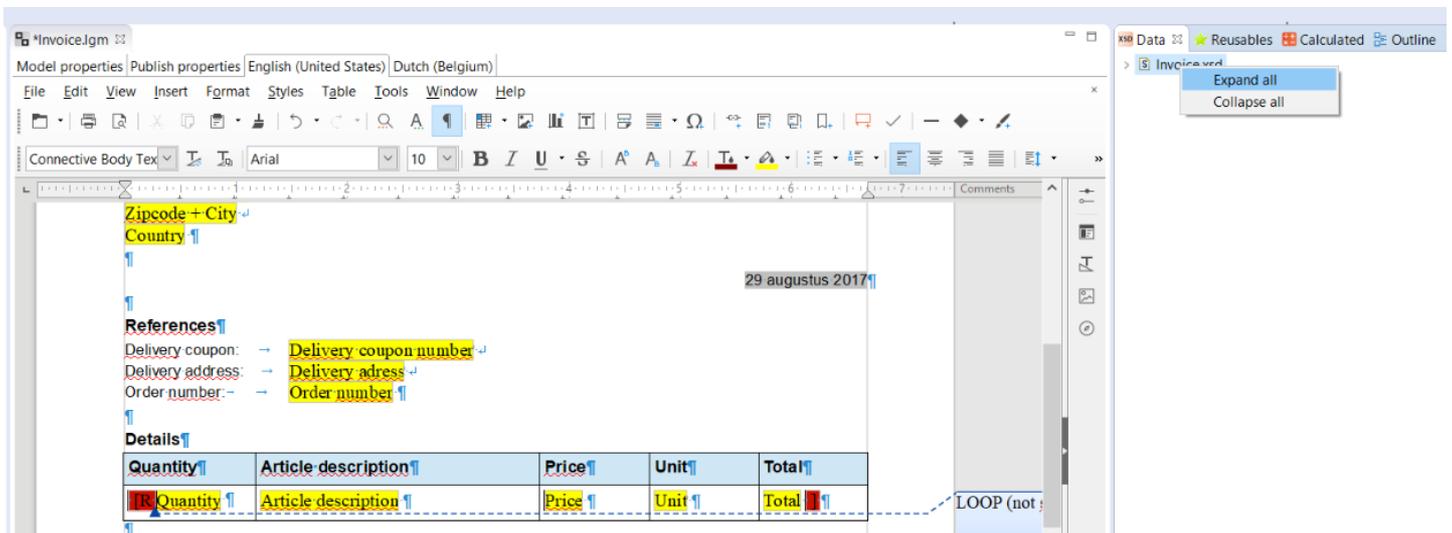
7.1.2. Insert a variable

In order to make your document dynamic, you can insert variables. These variables have been defined in the data structure (XSD) and will be replaced by their values as defined in the sample data file at generation time or after a preview.

A variable can be inserted in a language model or used in a condition or in a loop filter.

To insert a variable in your document:

- Check out the logical model.
- Open the language model.
- Unfold the Data Structure:
 - Right-click the node and click **Expand all**.



- To add a variable:
 - Double-click the variable you want to insert in the **Data** tab.
 - The variable is now inserted in the document and marked with a yellow frame.

7.1.3. Insert a dynamic image

A dynamic image is in fact a variable image. The size of the image is fixed at design time when you insert the image, but the content of the image is variable and is determined at generation time. The variable image corresponds to a variable in the data structure. This variable should contain the full path to the image that needs to be inserted at generation time.

Example of a path: <SIGNATURE>file:///d:/DynImages/WWilson.jpg</SIGNATURE>

To insert a dynamic image:

- Place the **cursor** in the document where you want to insert the image.
- Right-click the **variable** from the data structure.
- Select **Insert as dynamic image**.
- Click **Finish**.

A placeholder appears where the cursor was positioned.



If needed, you can modify most of the image settings, like size, position, add a border etc. This placeholder will be replaced by the actual image at generation time.

7.1.4. Representation in the Model Editor

A variable is represented as a yellow frame object and contains the variable's description:

Name

It is possible to change the size of the yellow frame object. This is only to help you fine-tune the appearance of variables, conditions and loops it will not affect the result after generation.

To change the size of the yellow frame object:

- Select the variable. **Note:** don't double-click the variable, or it will be opened twice.
- The **Variable properties** window appears.
- Click **Next**.
- At the bottom of the window, you can change the **Variable Visualisation** size.

- Click **Finish**.

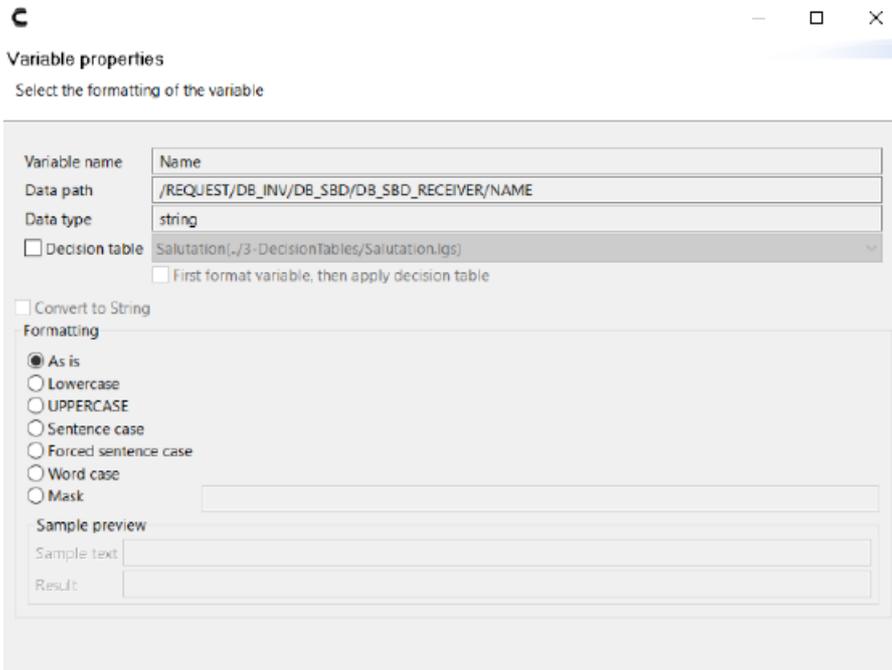
7.1.5. Format a variable

When you insert a variable, the default format is selected. It is possible, however, to apply a specific format on your variable.

To modify a variable's format:

- In your document, click on the yellow text box which represents the variable.

- The **Variable properties** wizard appears where you can select a different formatting.



The available formats depend on the type of the variable (string, numeric field, etc.). We will now give an overview of the available formats for every variable type.

String formats

The different formats for a string are:

STRING FORMAT	DESCRIPTION
As is	The variable is printed as is, no formatting is applied.
Lower case	The variable is printed in lower case.
Upper case	The variable is printed in upper case.
Sentence case	he first word of the variable is printed with sentence case. The other words are printed as is.
Forced sentence case	The first word of the variable is printed with sentence case. The other words are printed without sentence case.
Word case	The first letter of every word is printed in upper case.
Mask	When you select this type of formatting, the Mask text box becomes available where you can define yourself how the variable should be formatted.

Numeric formats

The different formats for a numeric field are:

NUMERIC FORMAT	DESCRIPTION
Convert to String	When you check this checkbox, the variable will be converted to a string and all String-related formatting options will become available instead of the formats related to a numeric field.

NUMERIC FORMAT	DESCRIPTION
Full formatting	The variable is formatted depending on the country settings. For example: in a Dutch language model, ',' is used as thousand separator and '.' is used as decimal separator. In an English language model, '.' is used as thousand separator and ',' is used as decimal separator.
As is	The variable is printed as is, no formatting is applied.
Predefined formatting	You can select a predefined formatting from the selection list.
As text	The numeric field is printed as text. For this format, you need to select Advanced mode and you can customize the format when you click Configure...
Mask	When you select this type of formatting, the Mask text box becomes available where you can define yourself how the variable should be formatted. Example: mask '#G##0D00', output '1.235,10'

Date formats

The different formats for a Date field are:

DATE FORMAT	DESCRIPTION
Convert to String	When you check this check box, the variable will be converted to a string and all String-related formatting options will become available instead of the formats related to a numeric field.
As is	The variable is printed as is, no formatting is applied.
Predefined formatting	You can select a predefined formatting from the selection list.
Mask	When you select this type of formatting, the Mask text box becomes available where you can define yourself how the variable should be formatted. See the examples below

Example: mask '[D] [MN,*-3] [Y,*-2]', output '31 DEC 01'

Example: mask '[FNn,*-3][D]/[M]/[Y,*-2]', output 'Mon 31/12/01'

Boolean formats

The different formats for a Boolean field are:

BOOLEAN FORMAT	DESCRIPTION
Convert to String	When you check this check box, the variable will be converted to a string and all String-related formatting options will become available instead of the formats related to a numeric field.
Zero or one	The Boolean is printed as zero (when false) or one (when true).

BOOLEAN FORMAT	DESCRIPTION
As checkbox	The Boolean is printed as an empty check box (when false) or a checked checkbox (when true).

Decision table

When the **Decision table** checkbox is checked, a drop-down list becomes available where you can select one of the decision tables that has been linked to the logical model. At generation time, the value from the XML will be replaced by the value that has been defined in the decision table.

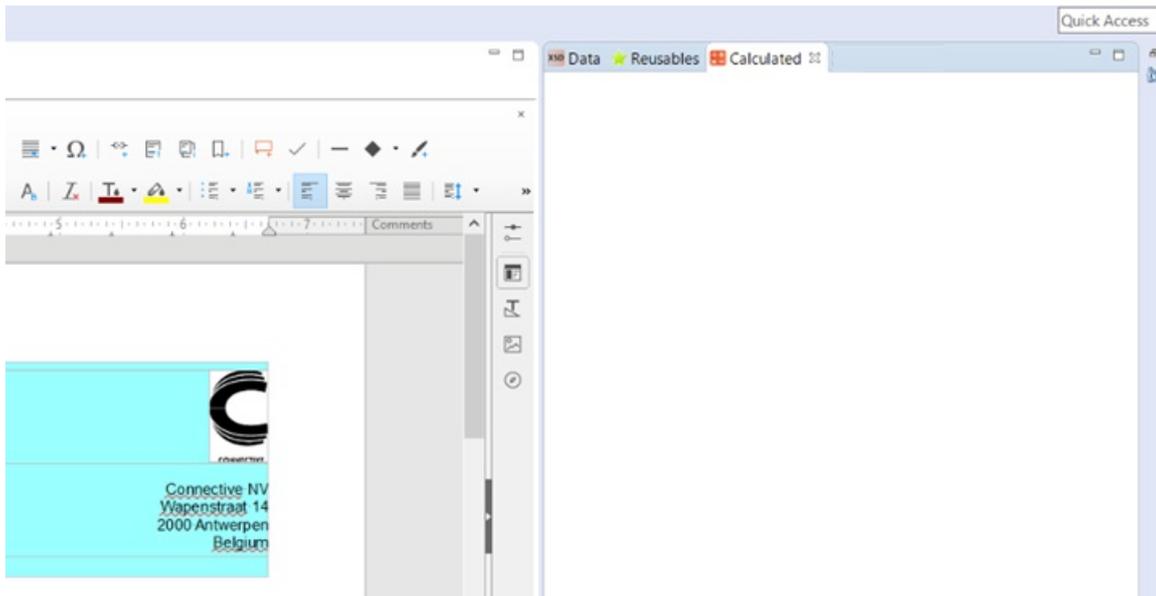
For more information on decision tables, see [chapter 4. The Decision Table](#).

7.2 Calculated

Sketch allows you to create calculated variables. A calculated variable executes a function on an existing variable that has been defined in the XSD. Example: based on the variable Cost Price per year, the calculated variable Cost Price per month can be calculated, by dividing the price per year by 12.

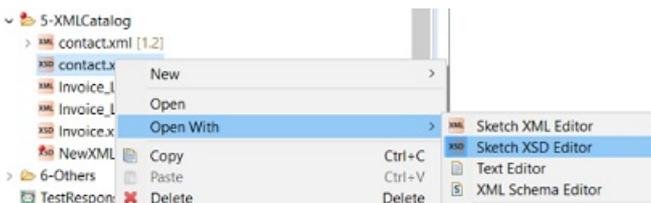
To add a calculated variable to your dynamic document you must take the following steps:

- Create a calculated variable in the XSD.
- Insert the calculated variable in your dynamic document. To do this the XSD data structure that contains the calculated variables to the logical model. If no data structure with calculated variables is linked to your model, the Calculated tab in the objects palette will be empty.



7.2.1 Create a calculated variable in the XSD

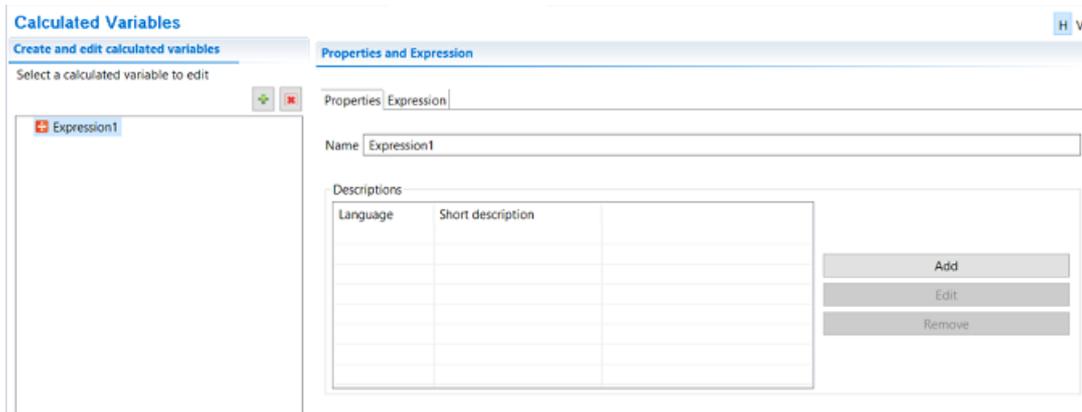
- [Check out](#) the XSD.
- Right-click and select **Open With > Sketch XSD Editor**.



- The XSD is opened in the Editor region.
- Open the **Calculated Variable** tab at the bottom of the window.

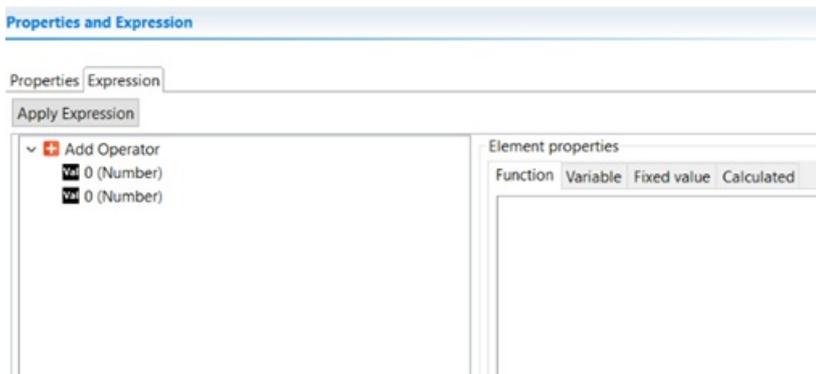


- Select the **green '+'** button.
- A new variable is added, named "Expression1".
- Select this variable.
- In the right pane, the **Property Section** tab is opened where you can give a name to the calculated variable and add descriptions in different languages.



- When you open the **Expression** tab, you can add the expression for the calculated variable. Select **Apply Expression** to finish.

The calculated variable has now been created. Save by doing a check-in of the XSD file.



7.2.2. Insert a calculated variable in a dynamic document

In order to make your document dynamic, a calculated variable must be inserted. As explained above, calculated variables are defined in the data structure (XSD) and will be replaced by a calculated value at generation time or after a preview.

A calculated variable can be inserted in a language model or used in a condition or in a loop filter:

To insert a calculated variable in your document:

- Check out the logical model.
- **Open** the language model.
- Click the **Calculated** tab.
- Double-click the variable you want to insert in the **Data** tab.
- The variable is now inserted in the document and marked with an orange frame.

7.2.3. Representation in the Model Editor

A calculated variable is displayed as an orange frame object and contains the variable's description:

New Calculated Variable

7.3 Reusables

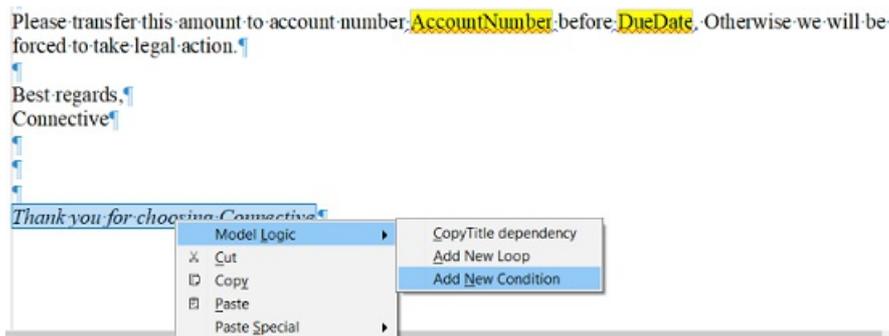
The **Reusables** tab contains an overview of the saved logic. When designing your dynamic document, you can save the conditions and/or loops you create in order to reuse them afterwards.

7.3.1. Saved condition

Save a condition

To save a condition:

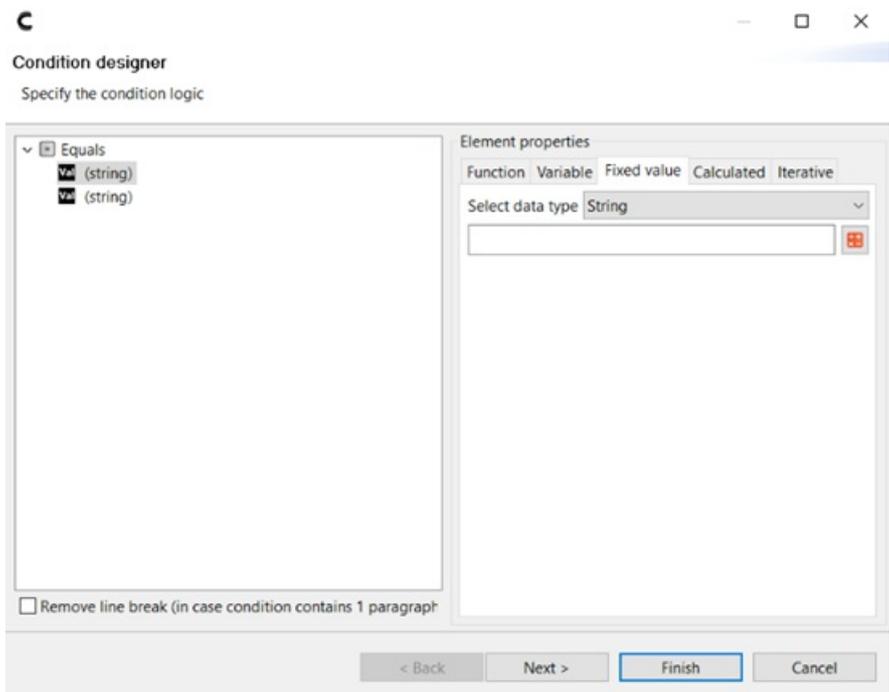
- [Check out](#) the logical model.
- Open the language model.
- Select the text you want to make conditional.
- Right-click and click **Model Logic > Add New Condition**.



- [Create the condition](#) in the Condition designer.

Tip: click the link for more information on how to create a condition.

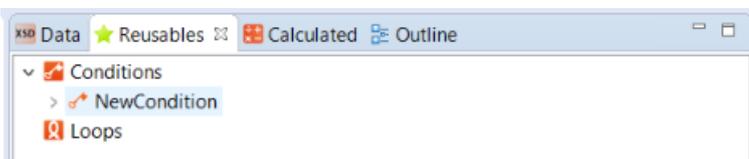
- Click **Next** (3 times).



- When you reach the final step, select **Save condition** under **Reusability**.
- The **Name** text box becomes available.
- Enter a name for the condition.



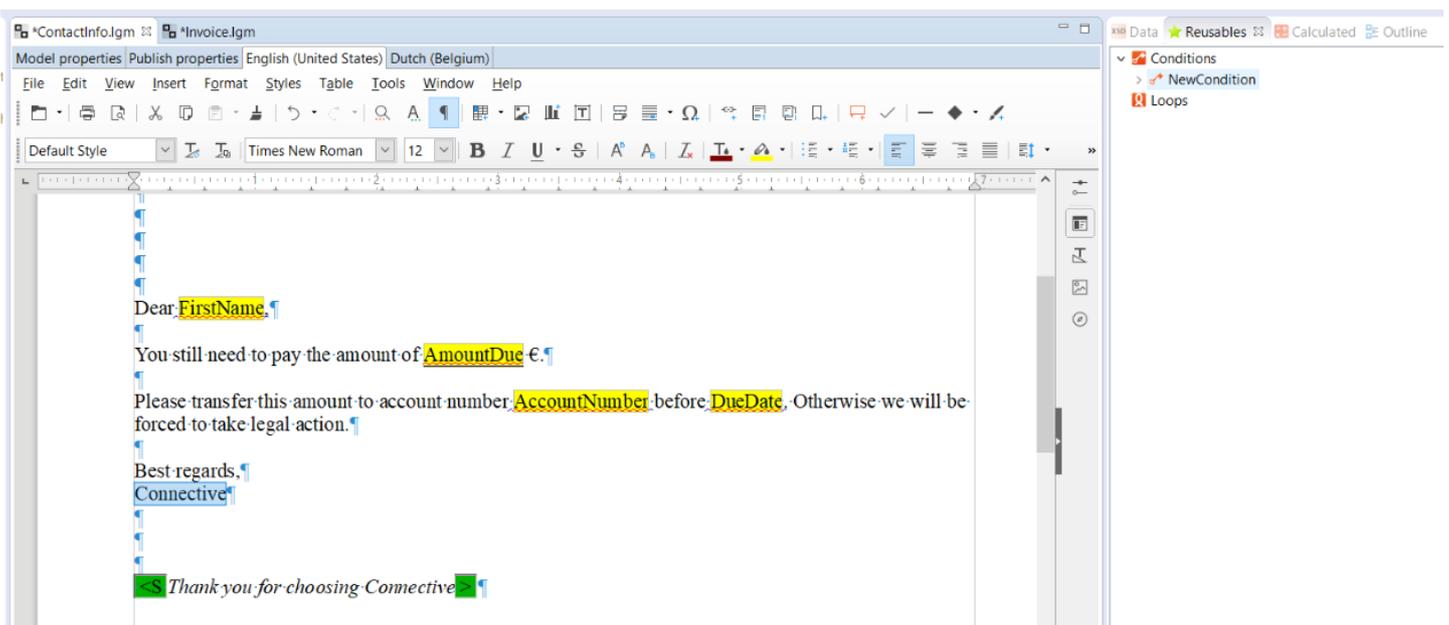
- Click **Finish**.
- The saved condition is added to the **Reusables** tab.



Reuse a saved condition

To reuse a saved condition:

- [Check out](#) the logical model.
- Open the language model.
- Select the text you want to make conditional.
- Double-click on the saved condition in the **Reusables** tab.
- The **Condition designer** is opened and the condition that has been saved is already present.
- Click **Finish**.



- The saved condition is added to the dynamic document.

Best regards,

<S Connective >

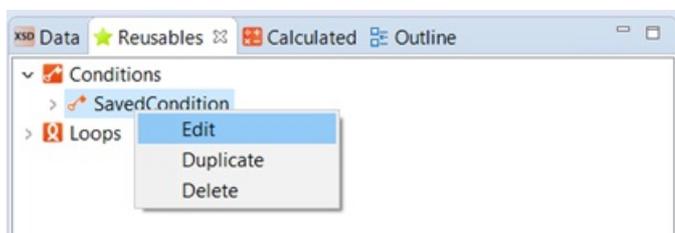
<S Thank you for choosing Connective >

Edit a saved condition

If you already created a saved condition, it is possible to edit this condition. It is important to keep in mind, however, that when you modify a saved condition, all instances of this condition will be updated!

To edit a saved condition:

- Select the saved condition in the **Reusables** tab.
- Right-click and click **Edit**.
- The **Condition designer** is opened.
- Update the condition.
- Click **Finish**.

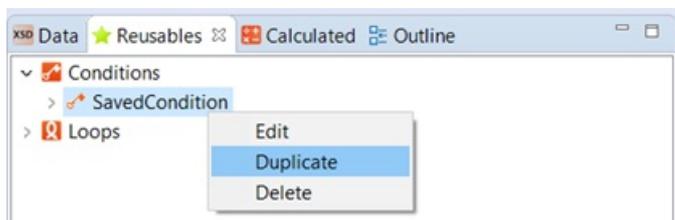


Duplicate a saved condition

When you want to add a complex condition that resembles a saved condition you have already created, it might be easy to duplicate this saved condition so that you only have to modify some small things instead of having to recreate the complex condition from scratch.

To duplicate a saved condition:

- Select the saved condition in the **Reusables** tab.
- Right-click and click **Duplicate**.



- Enter a new name for the condition.
- Click **OK**.
- The **duplicated condition** is now added in the **Reusables** tab.
- To edit the duplicated condition, see the section above, **Edit a saved condition**.

Representation in the Model Editor

A saved condition is displayed with green tags, just like a normal condition. Specific for the saved condition, is that there is the letter 'S' in the opening tag:

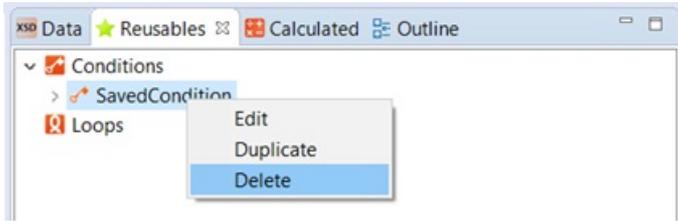
<S This is a saved condition. >

Delete a saved condition

When a saved condition is no longer used, you can choose to delete it.

To delete a saved condition:

- Select the saved condition in the **Reusables** tab.
- Right-click and click **Delete**.



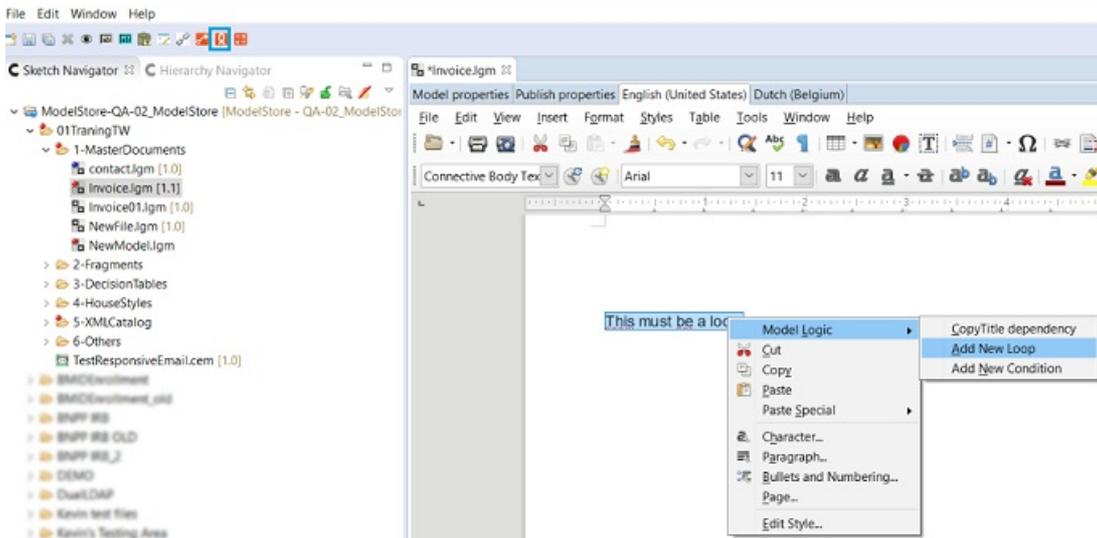
Note: if the saved condition is still used in a language model, a pop-up will appear informing you it is not possible to delete the condition. Make sure the condition is no longer used in any language model.

7.3.2. Saved loop

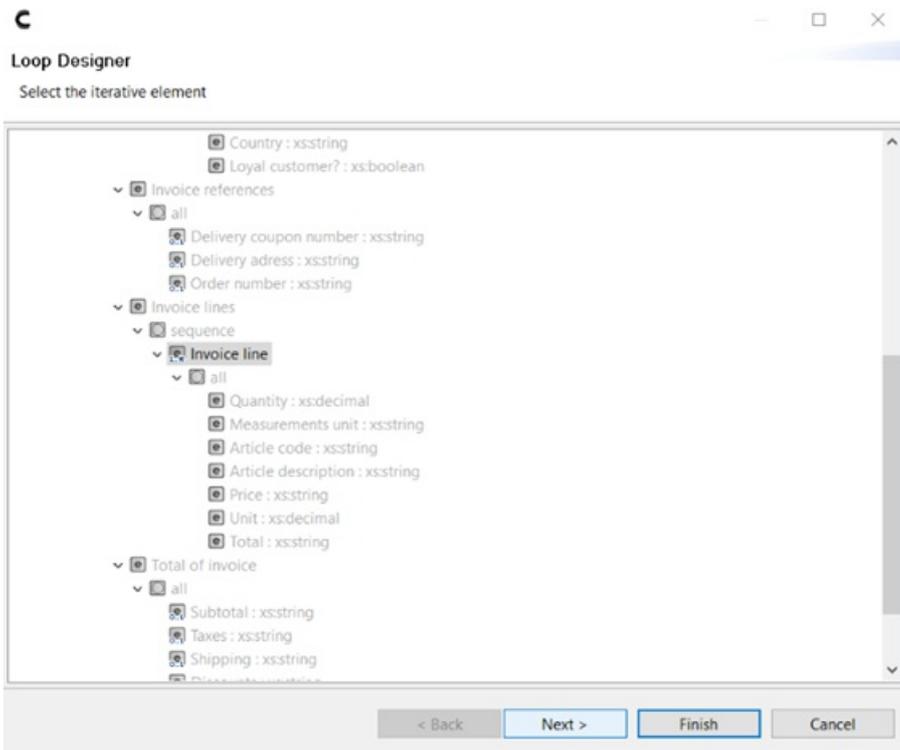
Save a loop

To save a loop:

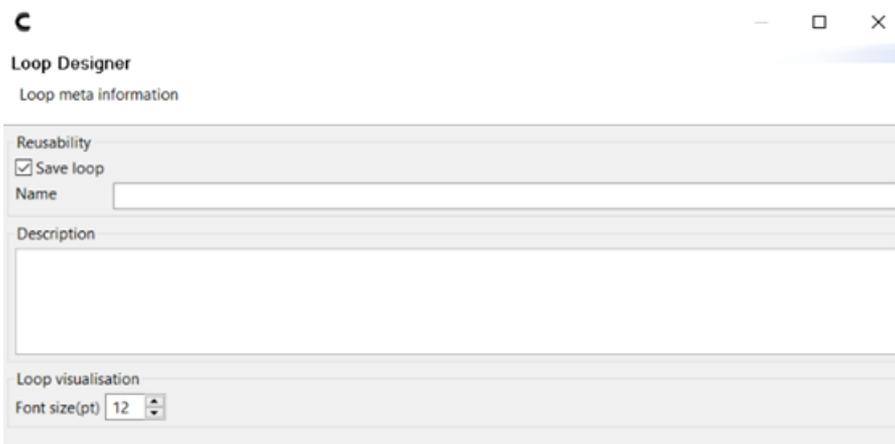
- [Check out](#) the logical model.
- Open the language model.
- Select the text on which you want to apply the loop.
- Right-click and click **Model Logic > Add New Loop**.



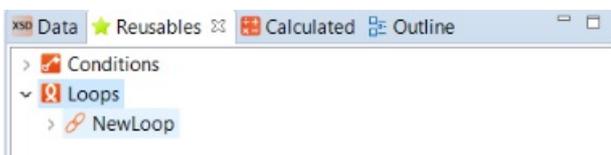
- In the **Loop Designer** that opens, select the iterative element.
Tip: you'll notice that only an iterative element is clickable.
- Click **Next** (4 times).



- When you reach the final step, select **Save loop** under **Reusability**.
- The **Name** text box becomes available.
- Enter a name for the loop.
- Click **Finish**.



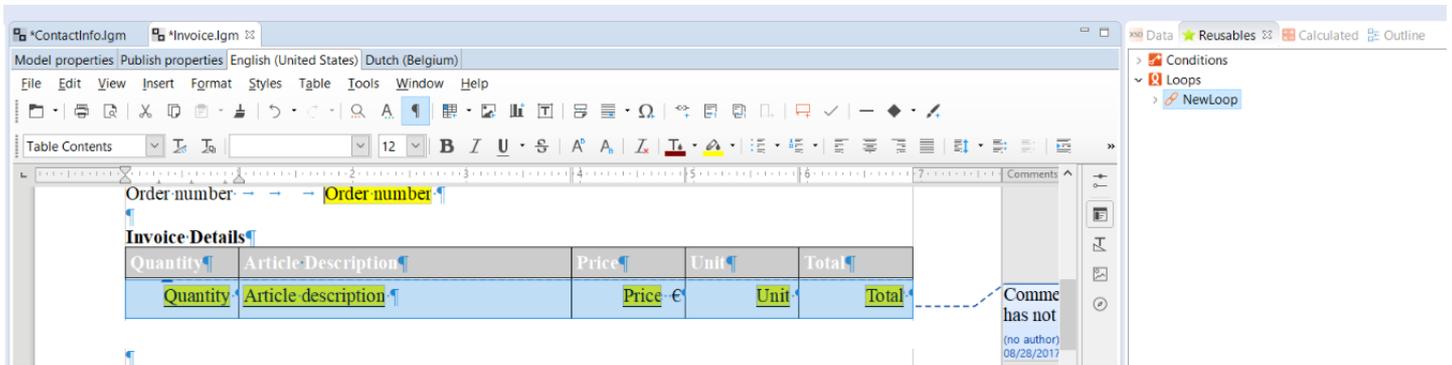
- The saved loop is added to the **Reusables** tab.



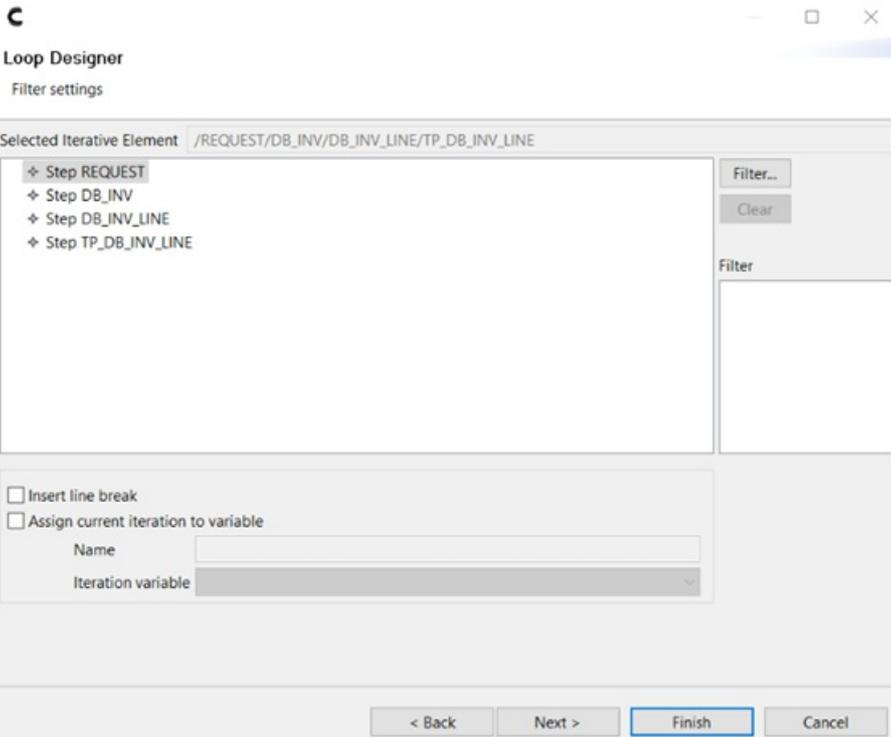
Reuse a saved loop

To reuse a saved loop:

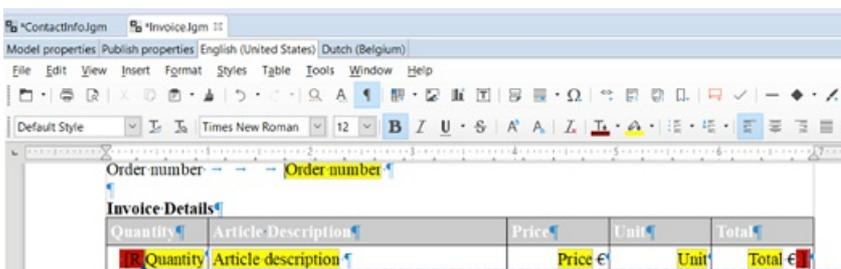
- [Check out](#) the logical model.
- Open the language model.
- Select the text to which you want to apply the loop.
- Double-click on the saved loop in the **Reusables** tab.



- The **Loop designer** is opened and the loop that has been saved is already present.
- Click **Finish**.



- The saved loop is added to the dynamic document.

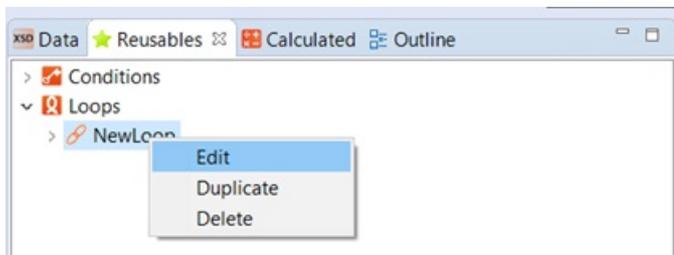


Edit a saved loop

If you already created a saved loop, it is possible to edit this loop. It is important to keep in mind, however, that when you modify a saved loop, all instances of this loop will be updated!

To edit a saved loop:

- Right-click the saved loop in the **Reusables** tab, and click **Edit**.



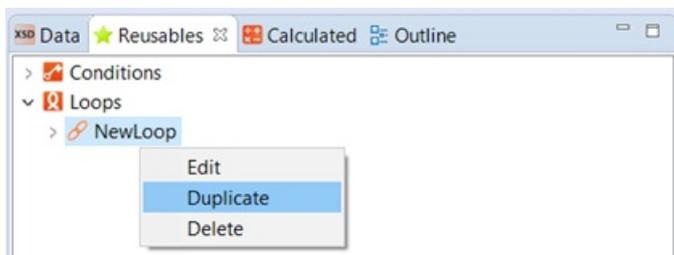
- The **Loop designer** is opened.
- Update the loop.
- Click **Finish**.

Duplicate a saved loop

If you already created a saved loop and you need to create a similar, but slightly different, loop, you don't need to recreate this loop from scratch in order to save it. It is possible to duplicate the saved loop, then make some small changes and save the loop under another name.

To duplicate a saved loop:

- Right-click the saved loop in the **Reusables** tab, and click **Duplicate**.



- Enter a new name for the loop.
- Click **OK**.
- The duplicated loop is now added to the **Reusables** tab.
- To edit the duplicated loop, see the section above, **Edit a saved loop**.

Representation in the Model Editor

A saved loop is displayed with red tags, just like a normal loop. Specific for the saved loop, is that there is the letter 'S' in the opening tag:

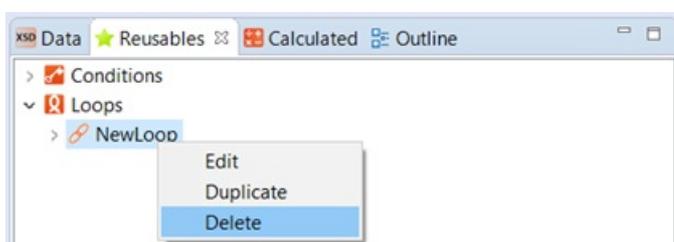
Delete a saved loop

When a saved loop is no longer used, you can choose to delete it.



To delete a saved loop:

- Select the saved loop in the **Reusables** tab.
- Right-click and click **Delete**.



Note: if the saved loop is still used in a language model, a pop-up will appear informing you it is not possible to delete the loop.

Make sure the loop is no longer used in any language model.



8. Fragments

A fragment is a **reusable** document that is inserted in one or more master documents. The advantage of working with fragments is that you only have to maintain the fragment in one place, instead of on every occurrence.

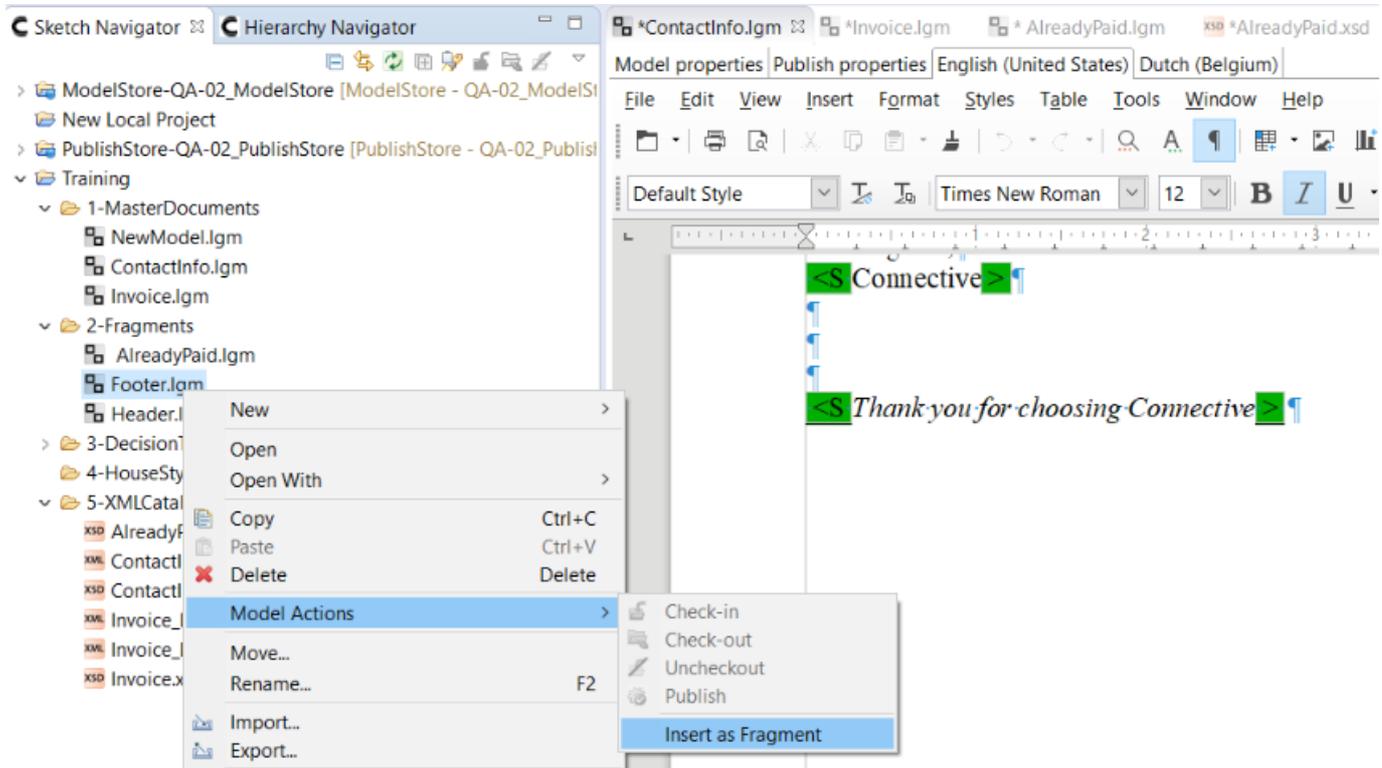
A fragment is **context related**, which means that it can only be inserted when the **data structure** of the fragment is the **same** as or a subset from the data structure of the main document. If this requirement is not met, an error message will appear during insertion.

Once a fragment is inserted in a language model, the fragment is **read-only**. Its content can only be **modified** in the fragment itself and not in the document where the fragment is inserted in.

8.1 How to insert a fragment?

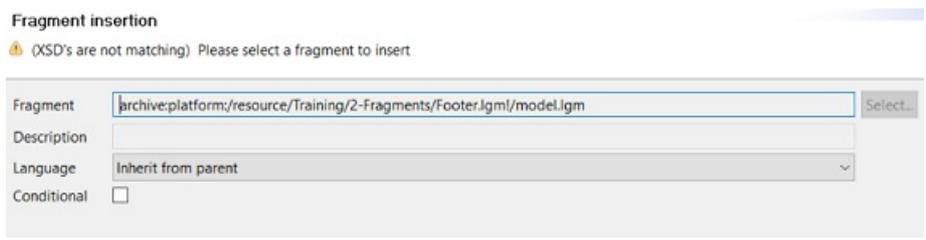
To insert a fragment:

- Place the **cursor** in the document where you want to insert the fragment.
- In the Sketch Navigator, right-click the .Igm file you want to insert as fragment.
- Click **Model Actions > Insert as fragment**.



- The **Fragment insertion** wizard opens.

If no XSD is linked to your fragment, you will see a message in the **Fragment insertion** wizard: **(XSDs are not matching) Please select a fragment to insert**. Note that this message is just a warning and not an error message. An XSD does not necessarily need to be linked to a fragment.

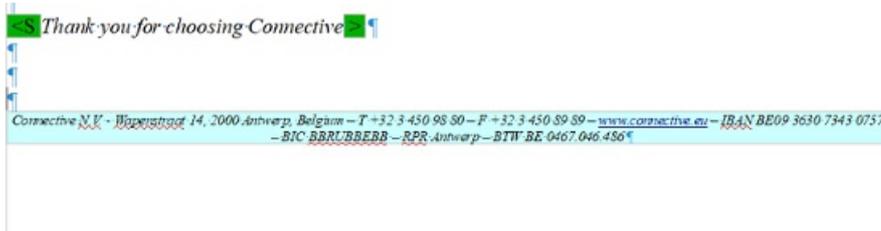


Note: it is now possible to use multiple subset XSDs within a fragment XSD. This feature facilitates the maintenance of shared XML elements among fragments/models.

- By default, **Inherit from parent** is selected as **Language**. This means that the fragment will be printed in the same language as the master document. In case the language of the master document is not available or if you would like the fragment to be printed in a specific language, you can select another language from the drop down-list.
- If you want to make the fragment conditional, check the **Conditional checkbox** (see **8.1.1 Conditional fragment** below).
- Select Next.
- If an **XSD** is linked to your fragment, you can specify the data context.
 - The **Matches found** drop-down list shows the subset data structures of the fragment. (For more information, see **8.1.2 Contextual fragment** below.)

- Select a **Context**.
- Click **Next**.
- The **Styles and Formatting** window is opened (see **8.1.2 Contextual fragment**).
 - Select if you want the fragment to keep its own formatting (**No merge**), adopt the formatting of the main document (**Merge**) or adopt the formatting of the previous paragraph (**Integrate with context style**).
 - Indicate whether you want to **remove the line-break** before or after the inserted fragment.
 - Click **Finish**.

The fragment is inserted where the cursor was positioned and represented with a blue background:



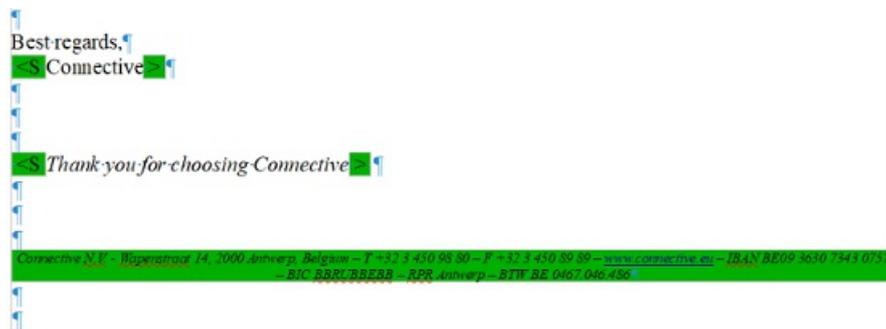
8.1.1. Conditional fragment

If you only want the fragment to be printed when a specific condition is met, you can make your fragment conditional. This means that the fragment will only be printed when the condition you entered when inserting the fragment, is met.

To insert a conditional fragment:

- Place the **cursor** in the document where you want to insert the fragment.
- In the Sketch Navigator, right-click the .lgm file you want to insert as fragment.
- Click **Model Actions > Insert as fragment**.
- The **Fragment insertion** wizard opens.
- Check the **Conditional** check box.
- Click **Next** (twice).
- [Create your condition](#) in the next window. When this condition is valid, the fragment will be printed. If not, the fragment will not be visible.

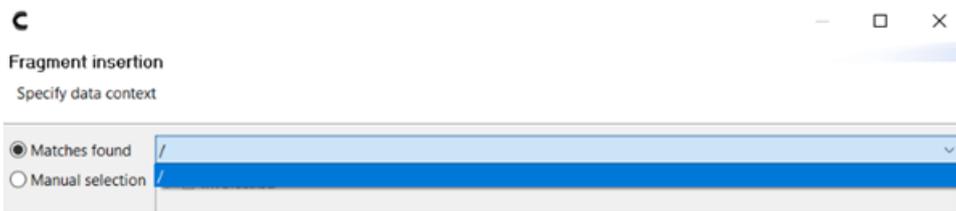
The conditional fragment is inserted where the cursor was positioned and represented with a **green background**.



8.1.2. Contextual fragment

If your fragment contains data that can occur in different contexts, you have to select the appropriate context in the drop down list of the **Specify data context** screen of the **Fragment insertion** wizard.

- Select the fragment's **context** from the drop-down list.
- If no context is applicable, the list is empty and only the root symbol ('/') is shown.



Example of a context: An address data block can exist in its identical form as an 'invoicing address', 'delivery address', 'sender address', 'receiver address', 'warehouse address' etc. To be reusable, all the data blocks 'Address' must be described by the same structure, for example 'name, street, number, zip code, city'.

8.1.3. Styles and formatting

When inserting a fragment, you can select different **Style and formatting** options.

As is (no merge)

The fragment **keeps its own paragraph styles and formatting**. The fragment will not adopt the paragraph styles and formatting of the master document. This is the **default** setting.

The background color of a fragment inserted 'As is' is **light blue**.

Example:

EXAMPLE	DESCRIPTION
<p><i>Before Fragment</i> Fragment 1 in Connective Body Text (Times New Roman 12). <i>After Fragment</i></p>	<p>Before generation:</p> <p>Master: font Arial</p> <p>Fragment: font Times New Roman</p>
<p><i>Before Fragment</i> Fragment 1 in Connective Body Text (Times New Roman 12). <i>After Fragment</i></p>	<p>After generation:</p> <p>Master: font Arial</p> <p>Fragment: font Times New Roman</p>

Inherit styles (merge) When you choose this option, the fragment adopts the paragraph styles and formatting of the master document. This option however, will only work if the paragraph style in your master document and your fragment have the **same name**.

The background color of a fragment inserted with option 'Merge' is **darker blue**.

Example:

EXAMPLE	DESCRIPTION
<p><i>Before Fragment</i> Fragment 1 in Connective Body Text (Times New Roman 12). <i>After Fragment</i></p>	<p>Before generation:</p> <p>Master: font Arial</p> <p>Fragment: font Times New Roman</p>
<p><i>Before Fragment</i> Fragment 1 in Connective Body Text (Times New Roman 12). <i>After Fragment</i></p>	<p>After generation:</p> <p>Master: font Arial</p> <p>Fragment: font Arial</p>

Integrate with context style

In this case, the fragment will adopt the paragraph style and formatting of the previous paragraph. The background color of a fragment which inserted to 'Integrate with context style' is also **darker blue**.

Example:

EXAMPLE	DESCRIPTION
<i>Before Fragment_</i> Fragment 1 in Connective Body Text (Times New Roman 12). <i>After Fragment_</i>	Before generation: Previous paragraph: font Arial italic Fragment: font Arial italic
<i>Before Fragment_</i> <i>Fragment 1 in Connective Body Text (Times New Roman 12).</i> <i>After Fragment_</i>	After generation: Previous paragraph: font Arial italic Fragment: font Arial italic

Remove line break before and/or behind fragment

When you check one or both check boxes, the corresponding line break will be removed at generation time. This way you can insert fragments in line with the text.

Remove line break before:

EXAMPLE	DESCRIPTION
<i>Before Fragment_</i> Fragment 1 in Connective Body Text (Times New Roman 12). <i>After Fragment_</i>	Before generation: Line break before fragment Line break behind fragment
<i>Before Fragment_</i> <i>Fragment 1 in Connective Body Text (Times New Roman 12).</i> <i>After Fragment_</i>	After generation: NO line break before fragment Line break behind fragment

Remove line break behind:

EXAMPLE	DESCRIPTION
<i>Before Fragment_</i> Fragment 1 in Connective Body Text (Times New Roman 12). <i>After Fragment_</i>	Before generation: Line break before fragment Line break behind fragment
<i>Before Fragment_</i> <i>Fragment 1 in Connective Body Text (Times New Roman 12).</i> <i>After Fragment_</i>	After generation: Line break before fragment NO line break behind fragment

Remove line break before and behind:

EXAMPLE	DESCRIPTION
<p><i>Before Fragment_</i> Fragment 1 in Connective Body Text (Times New Roman 12). <i>After Fragment_</i></p>	<p>Before generation: Line break before fragment Line break behind fragment</p>
<p><i>Before Fragment_</i> Fragment 1 in Connective Body Text (Times New Roman 12). <i>After Fragment_</i></p>	<p>After generation: NO line break before fragment NO line break behind fragment</p>

8.2 How to modify a fragment

Once a fragment is inserted in a language model, the fragment is read-only. Its content can only be modified in the fragment itself and not in the document where the fragment is inserted in.

To modify the content of a fragment, you have to make the changes in the model of the fragment itself. This means checking out the logical model and modifying the language model(s). For more detailed information and instructions, see [3.4 How to check out a logical model](#).

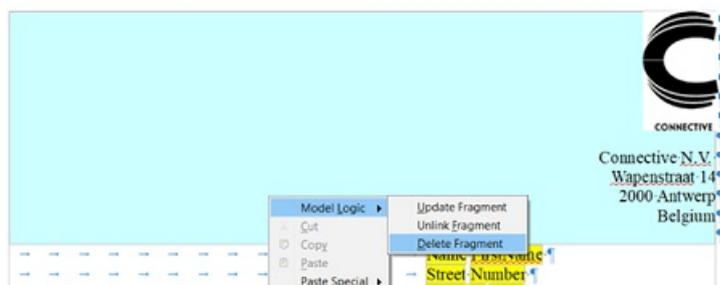
You have to keep in mind that when you update the fragment, the changes will be reflected in all instances of this fragment.

8.3 How to delete a fragment

Before you can delete a fragment, make sure the master document is checked out. See [3.4 How to check out a logical model](#) for more information and detailed instructions.

To delete a fragment:

- In the Model Editor, place the cursor in the document where you want to delete the fragment.
- Right-click and click **Model Logic > Delete Fragment**.



The fragment is no longer present in the master document.

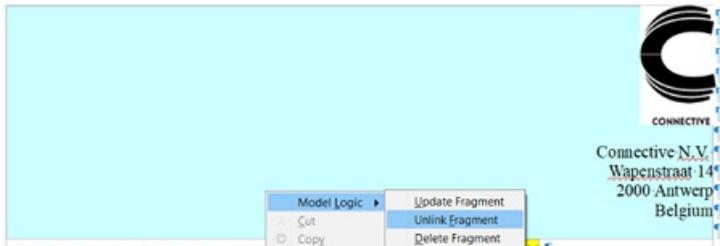
By doing this, the fragment has disappeared from the language model. The logical model of the fragment, however, still exists in the repository. The action of deleting a fragment will delete the instance of the fragment used in the master, but not the logical model of the fragment itself. To delete a fragment entirely, see [3.10 How to delete a model](#).

8.4 How to unlink a fragment

When you unlink a fragment, the link between the instance of the fragment in the master document and the logical model of the fragment will be **broken**: future changes in the fragment will not be reflected in the master document.

To unlink a fragment from the master document:

- In the Model Editor, place the cursor in the document where you want to delete the fragment.
- Right-click and click **Model Logic > Unlink Fragment**.



The content of the fragment is still present in the master document but the blue background has disappeared:



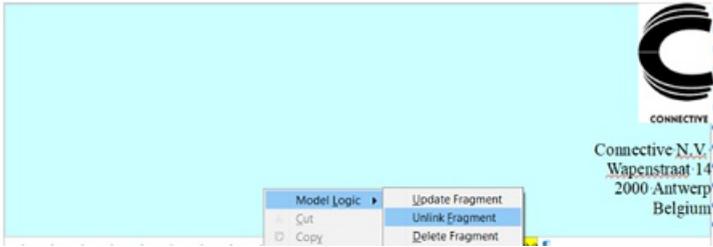
8.5 How to update a fragment

When a language model is opened, the fragments are not automatically updated. If a document contains more than 1 fragment, you can choose to either update all fragments in the document or a single fragment at a time.

8.5.1. Update a single fragment

To update a single fragment:

- Make sure the master document is **checked out**.
- In the Model Editor, place the cursor in the document where you want to update the fragment.
- Right-click and click **Model Logic > Update Fragment**.



The selected fragment is updated and the other fragments have not been modified.

8.5.2. Update all fragments

The **Update Fragments** button is available when the master document is checked in as well as when the master document is checked out.

Master is checked out

When you update all fragments and the model is checked out, the content of the fragments will be updated. When you save and check in the model afterwards, the updated versions of the fragments will be **saved**. So next time you open the document, you will see the updated content of the fragments.

To update all fragments:

- Make sure the master document is [checked out](#).
- In the upper left corner, click the **Update Fragments** icon.
- Click **Yes** to confirm.



All fragments are updated at once.

Master is checked in

When you update all fragments while the model is checked in, the content of all fragments will be updated so you can see the correct content of your document. However, this updated version of the fragments will not be saved. Next time you check out the master, you will still see the previous version of the fragments.

To update all fragments:

- Open the language model.
- In the upper left corner, click the **Update Fragments** icon.
- Click **Yes** to confirm.

Sketch

File Edit Window Help



Sketch Navigator Hide Update Fragments

All fragments are updated at once.

9. Adding logic: Conditions

In order to make your document dynamic, Sketch allows you to easily create conditions. These logical building blocks are added in a Wizard approach, which makes it possible for everyone to add logic in a simple way.

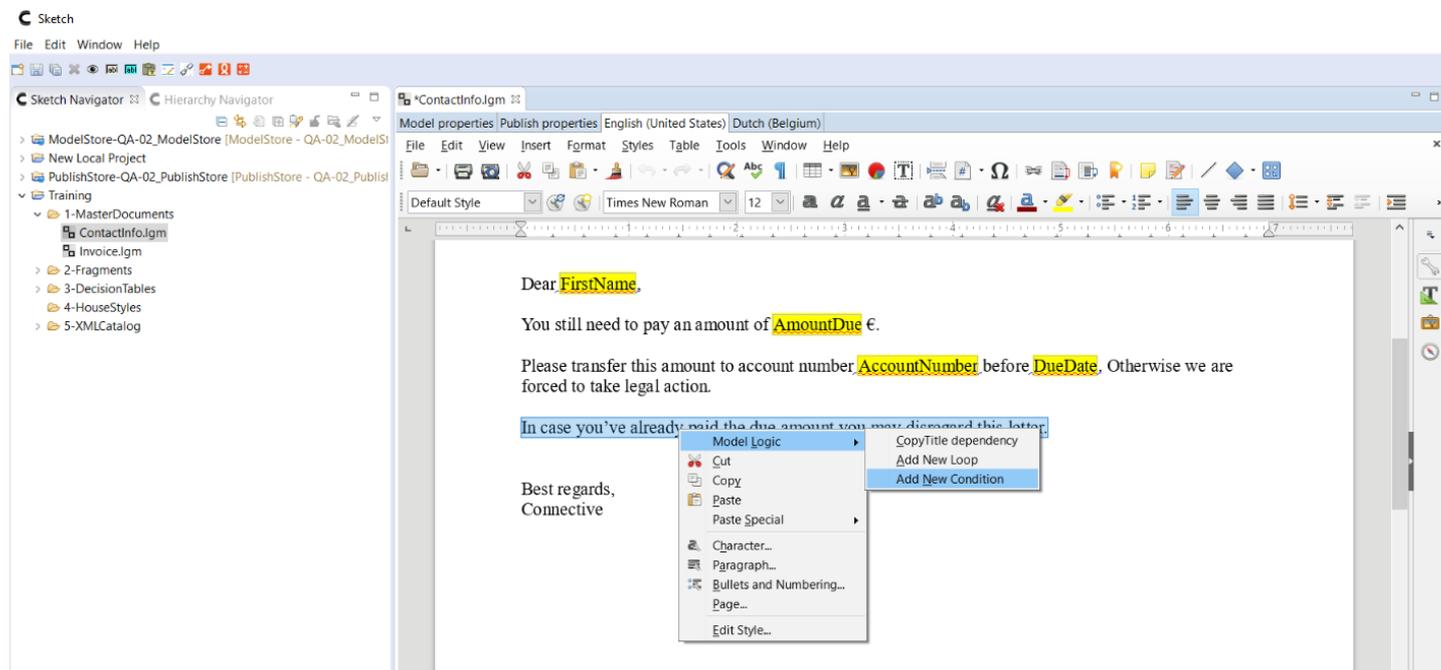
A condition is used to show or hide a word, a sentence, a fragment or even a complete document. Various operators allow you to set certain criteria to validate variables and fixed data elements.

Combining a loop with a condition inside is possible but should be done in an 'inside out' operation. Build the condition first and then the loop around it.

9.1 How to create a condition

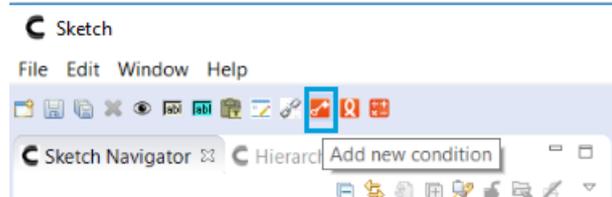
To create a condition:

- Make sure the logical model is checked out.
- Open the language model.
- Select the text you want to make conditional.
- Right-click and click **Model Logic > Add New Condition**.



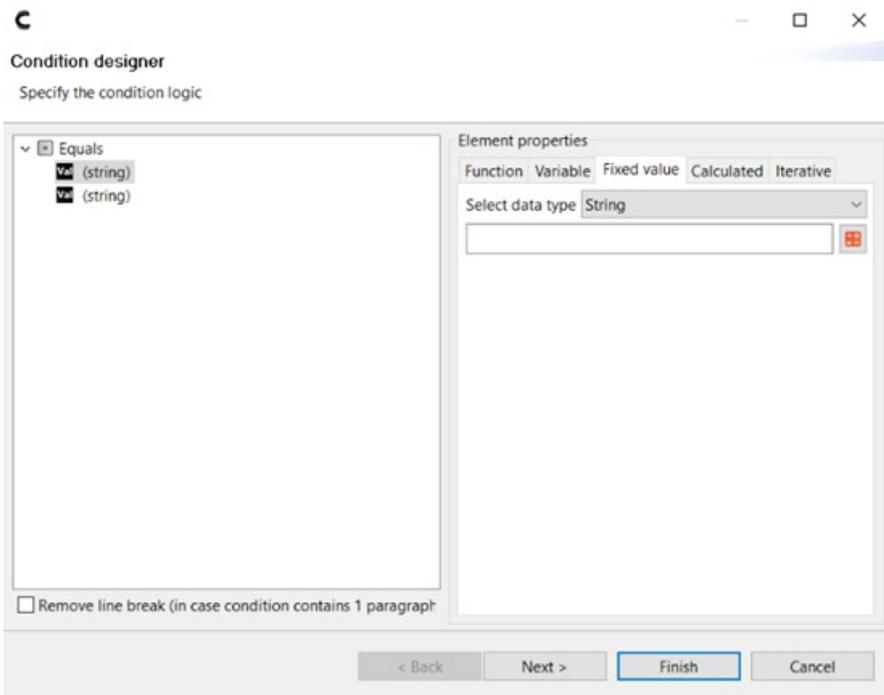
Or

- Click the **Condition** icon in the toolbar.

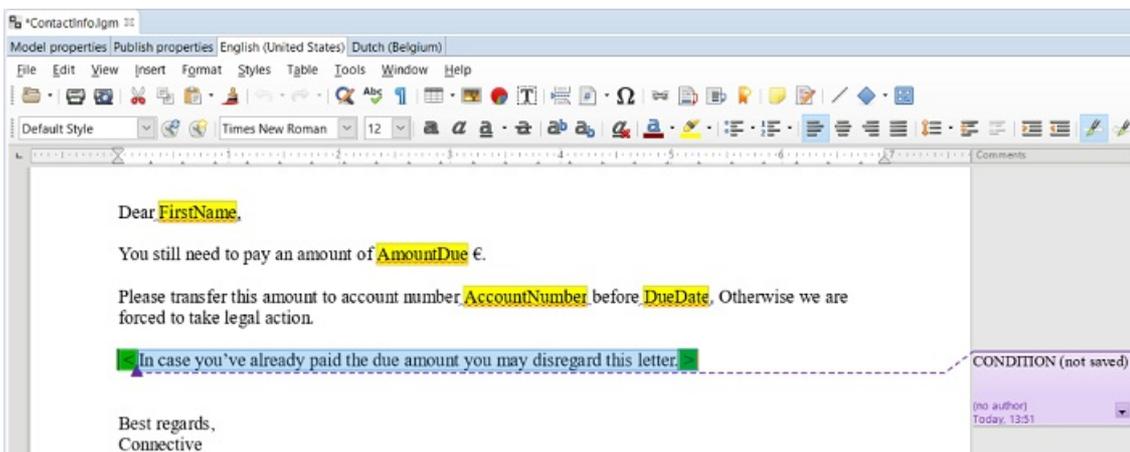


The **Condition designer** is opened where you create the condition.

- By default the **Equals** operator is selected.
- You can use **functions**, **(calculated) variables** and **fixed values** to create your condition.
- Click **Finish**.



- The condition tags are added around the selected text.



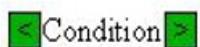
Tip: if you have some difficulties to select the text using the mouse, try using the keyboard: hold down [Shift] while you press the left/right arrow keys. If you still experience some trouble, try selecting from the end to the beginning.

Make sure there is text (or spaces) where you right-click. Otherwise **Add New Condition** might not appear.

9.2 Visualization of the condition

9.2.1 Representation in the Model Editor

A condition is displayed between a green start tag and a green end tag.

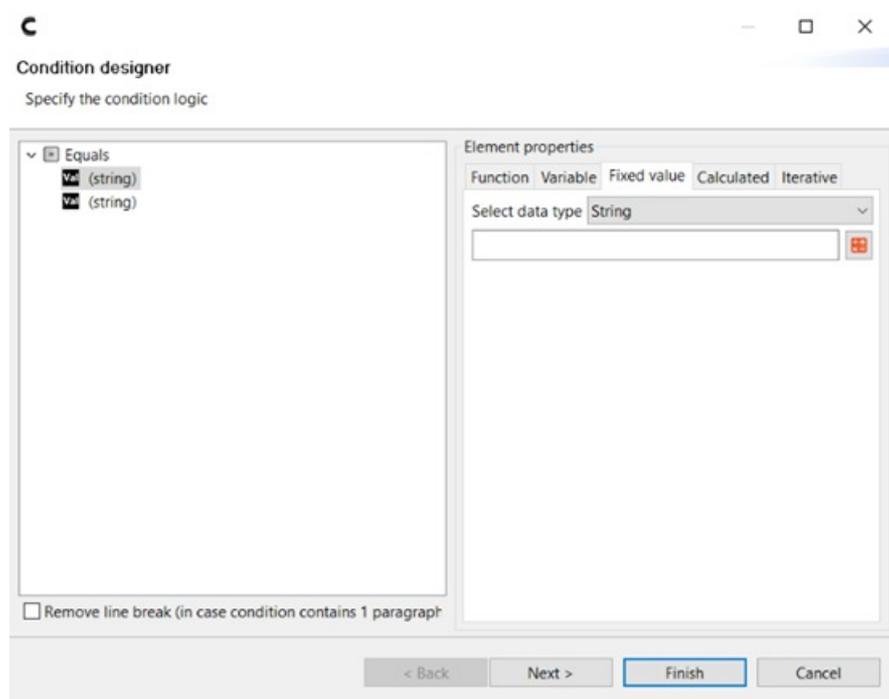


Everything in between these tags is made conditional. The conditional data can be variables, fragments, pictures, words, sentences, paragraphs or even entire documents containing multiple pages.

It is possible to change the size of the green frame objects. This is only to help you finetune the appearance of the conditions, it will not affect the result after generation.

To change the size of the green frame object:

- Open the condition by double-clicking on the start tag.



- The **Condition designer** is opened.
- Click **Next (3 times)**.
- At the bottom of the window, you can change the **Font size** under **Condition Visualization**.

9.2.2. Highlight a condition

To see where a condition starts and ends:

- Click the **start tag** of the condition .
- The text between frame tags will be **highlighted in green**. This is the conditional text.

Dear FirstName,

You still need to pay an amount of AmountDue €.

Please transfer this amount to account number AccountNumber before DueDate. Otherwise we are forced to take legal action.

In case you've already paid the due amount you may disregard this letter.

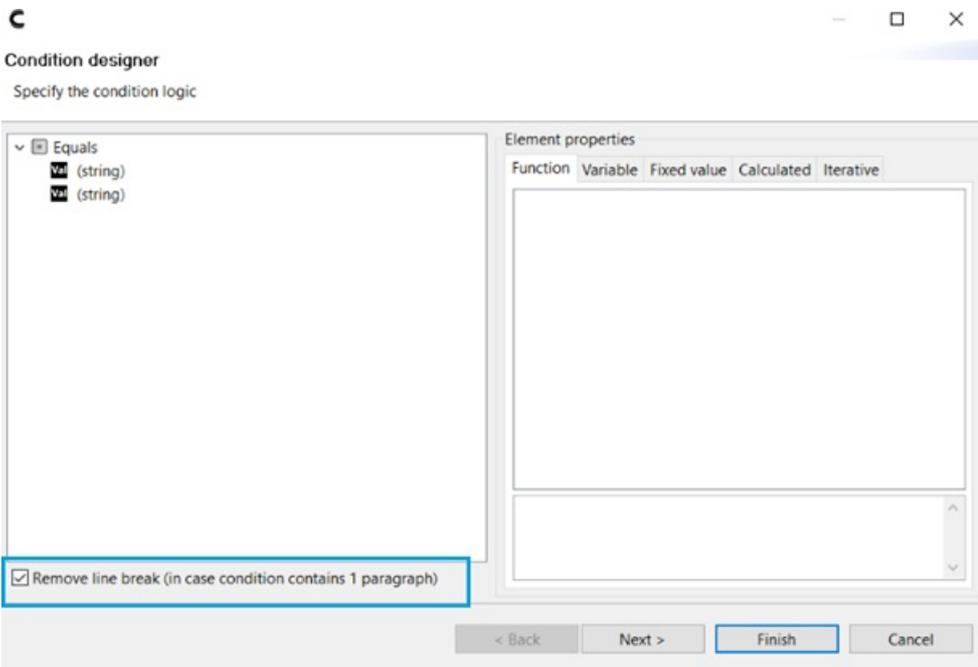
Best regards,
Connective

9.2.3. Remove line break when condition is invalid

If a condition has been applied to only 1 paragraph, there is an option available to easily remove the conditional paragraph in case the condition is false. This way, there will be no empty paragraph after preview or generation.

To remove the line break in case the condition is false:

- Double-click on the start tag of the condition.
- The **Condition designer** is opened.
- Check **Remove line break (in case condition contains 1 paragraph)** at the bottom of the window.



The result after preview when the checkbox is **not checked** looks as follows:

Before condition

After condition

The result after preview when the checkbox is **checked** looks as follows:

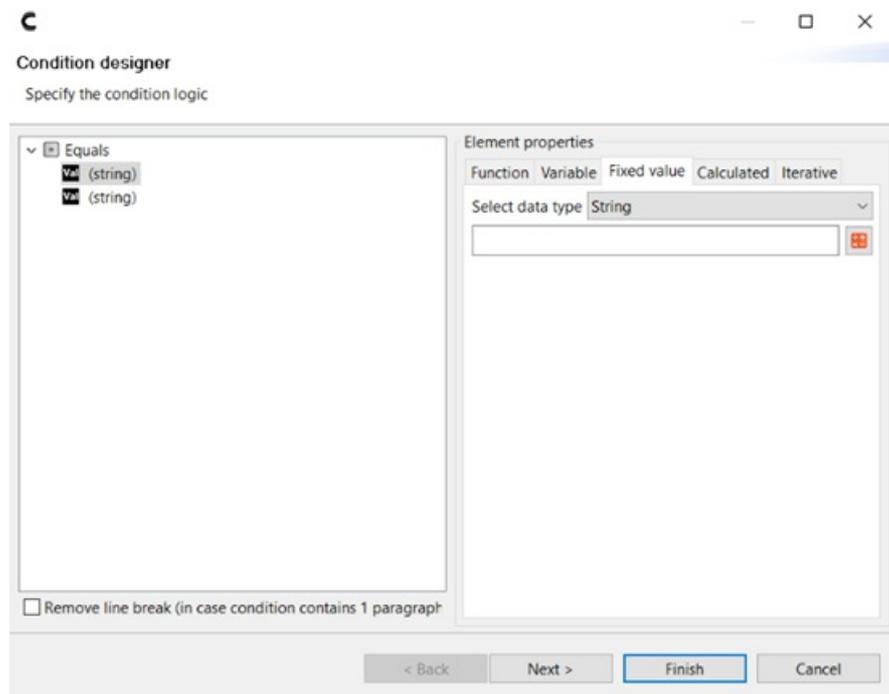
Before condition

After condition

9.3 Condition logic

With the Condition Designer wizard, you can make your conditions as simple or as complex as you want. When you create a condition, the Equals operator is selected by default and there are 2 arguments present which are empty string values.

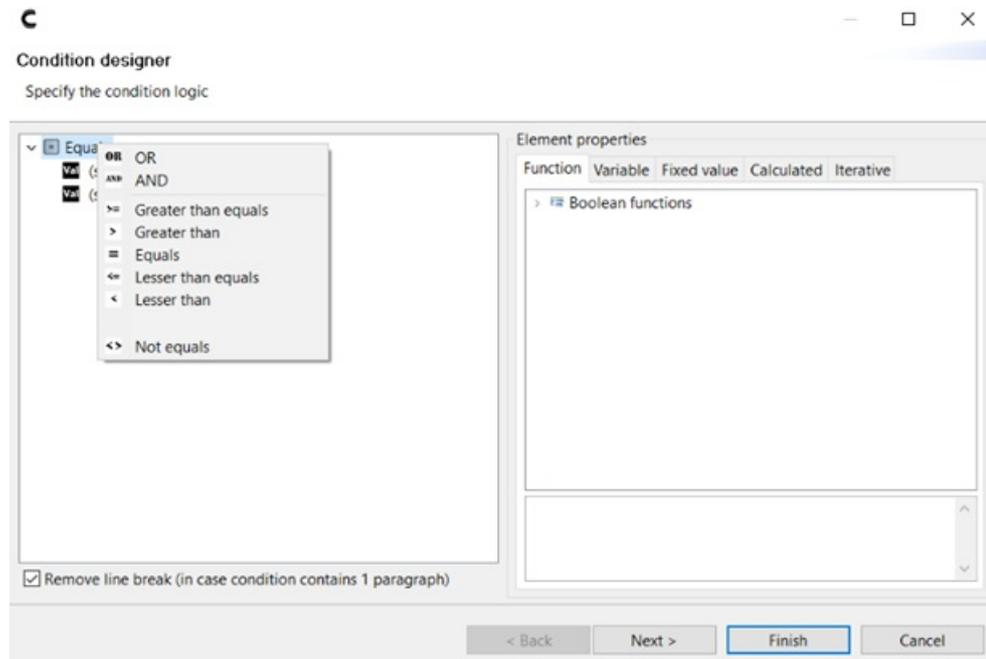
The default Condition when the Condition designer is opened looks as follows:



9.3.1. Operators

There are various operators available in the Condition designer. To see all operators:

- Right-click the **Equals** operator.
- A drop-down list of **all operators** is opened and you can select another operator.



The available operators are:

OPERATOR	DESCRIPTION
Greater than equals (>=)	Compares 2 values
Greater than (>)	Compares 2 values
Equals (=)	Compares 2 values
Lesser than equals (<=)	Compares 2 values
Lesser than (<)	Compares 2 values
Not equals (<>)	Compares 2 values
OR	Combines two or more expressions, where at least one expression must be met for the condition to be true.
AND	Combines two or more expressions, where all the expressions must be met for the condition to be true.

9.3.2. Arguments

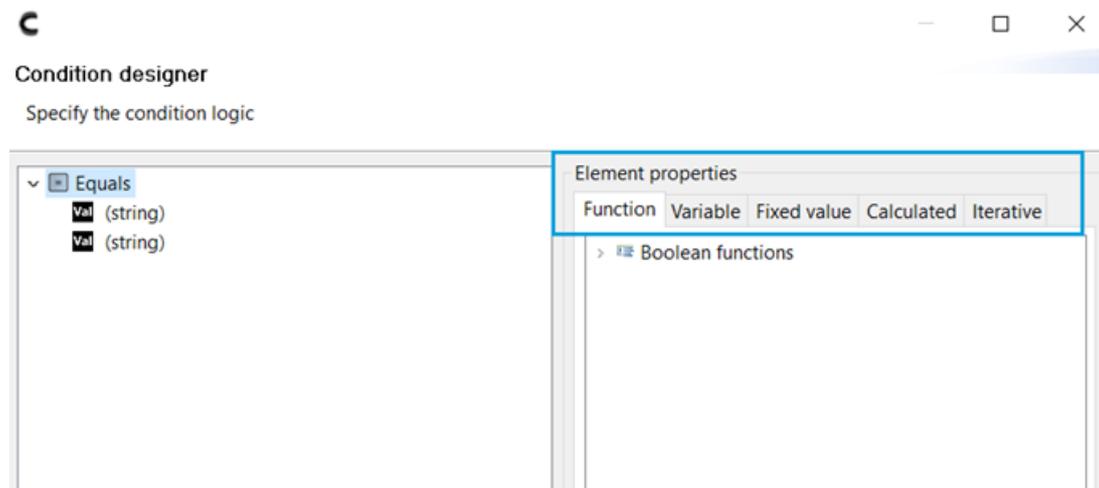
A condition consists of at least 2 arguments: 1 argument is compared to the other argument and checked against the operator. When you create a new condition, the **Fixed value** argument is selected by default.

Apart from the Fixed value argument, 3 other types are available:

In the Condition designer, select one of the arguments. Under **Element properties**, you will see the different argument types:

- Function
- Variable
- Fixed value
- Calculated

By default, the **Fixed value** argument is selected.

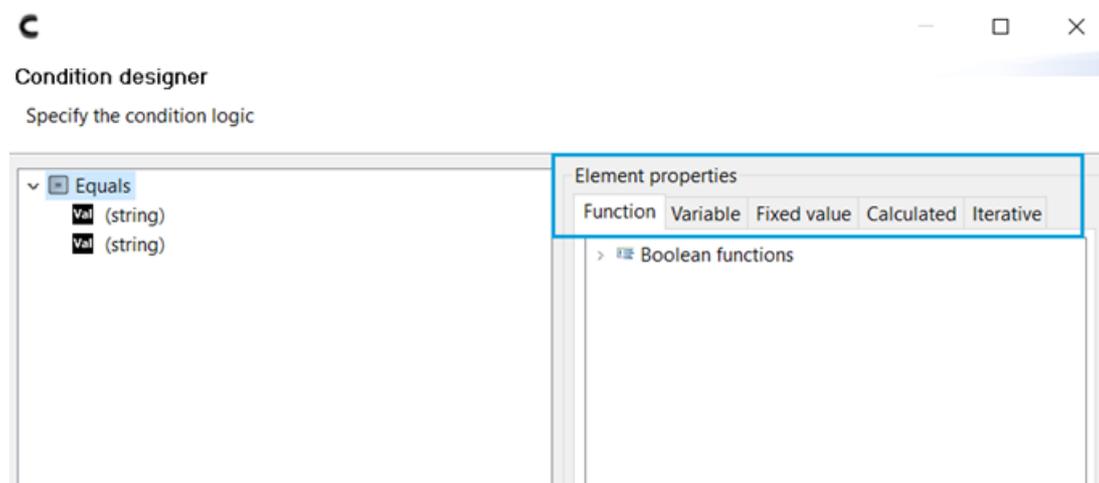


To select another argument type:

- Select the appropriate tab in **Element properties** in the Condition designer.

Or

- Right-click the argument and select the new argument type you want to use.
- The corresponding tab is automatically activated in the **Element properties** window.

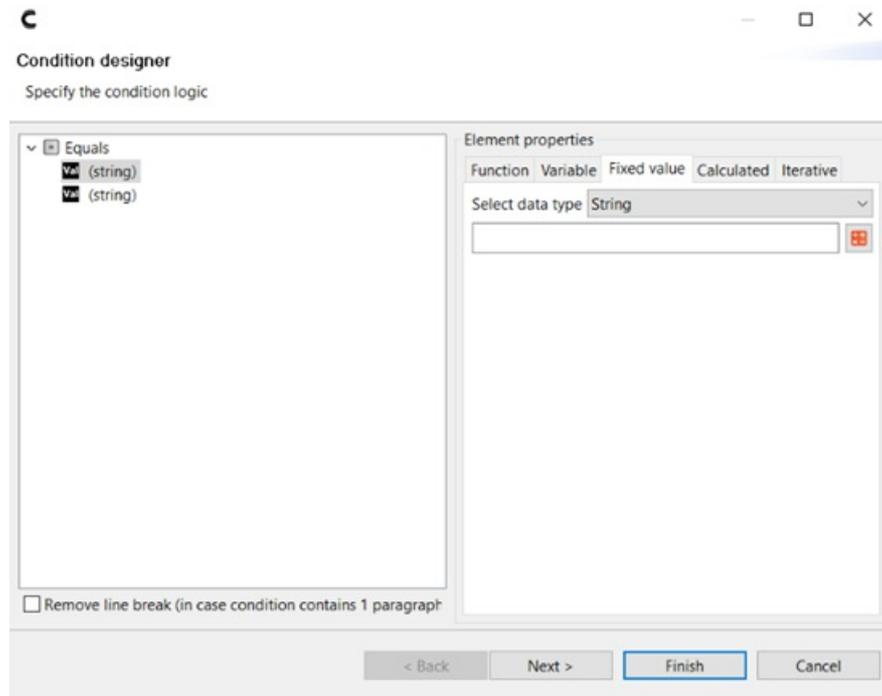


9.3.3. Argument type 1: FIXED VALUE

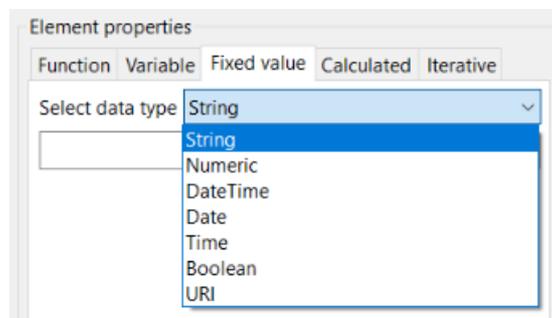
The user can directly fill in a value. Often in a condition, a variable/calculated variable/function is compared to a fixed value.

To use a fixed value in a condition:

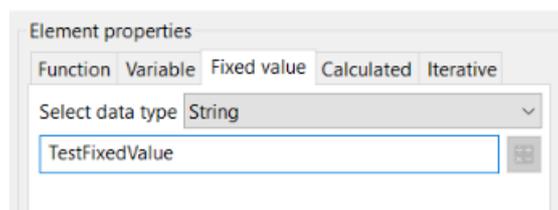
- Select the argument.
- In **Element properties**, the **Fixed value** tab is automatically selected.



- Select the appropriate data type from the list.



- In the text box, enter the value you want to compare.



- Select the other argument when finished.

Or

- Press **Enter**.

The fixed value is now added as argument.



Condition designer

Specify the condition logic

▼ Equals

- Val TestFixedValue (string)
- Val (string)

Element properties

Function Variable Fixed value Calculated Iterative

Select data type String ▼

TestFixedValue

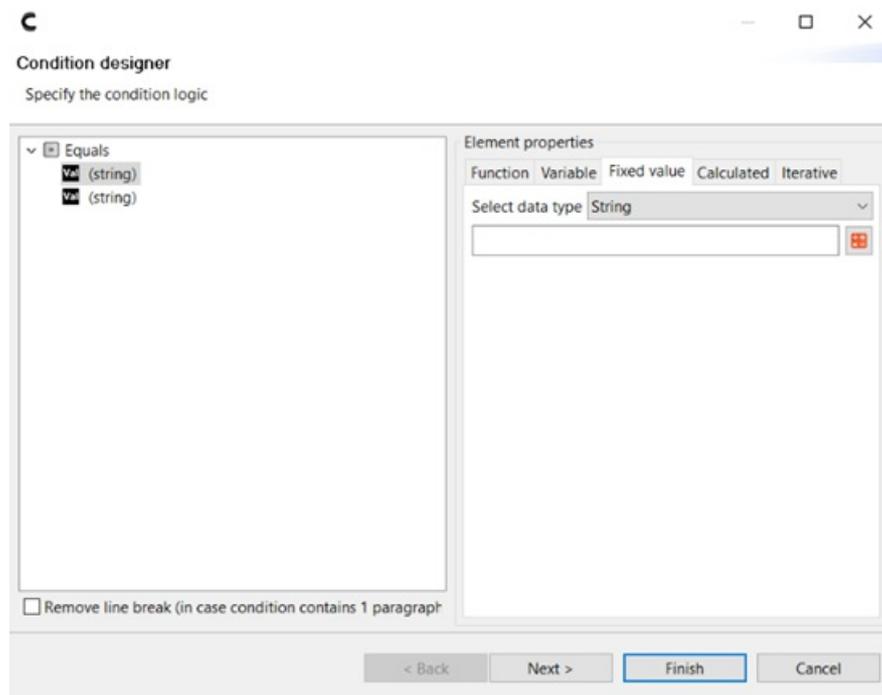
9.3.4. Argument type 2: VARIABLE

If you want to use a variable in a condition, a data structure needs to be attached to your logical model. See [3.2.1 Create a logical model](#) for more information and detailed instructions.

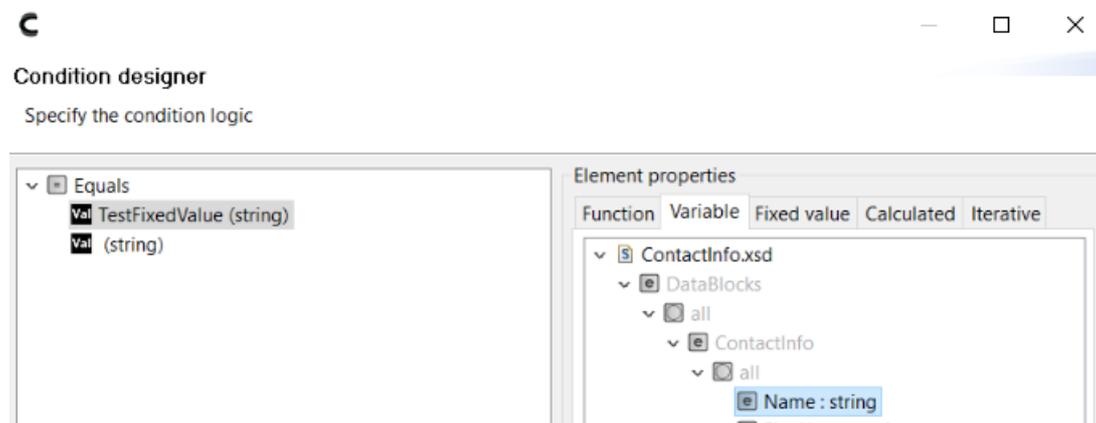
When the **Variable** tab is selected, the tree structure of the attached Data Structure (XSD) is shown. This tree structure can easily be collapsed/expanded by clicking on the triangles in front of the different data blocks. Only the leaf elements will be enabled and can be inserted. The other elements (nodes) will be disabled.

To use a variable in a condition:

- Select the argument.
- In **Element properties**, the **Fixed value** tab is automatically selected.



- Click the **Variable** tab in the **Element properties** area.
- In the data structure, double-click the variable you want to compare to the other argument.
- The variable is now added as argument.



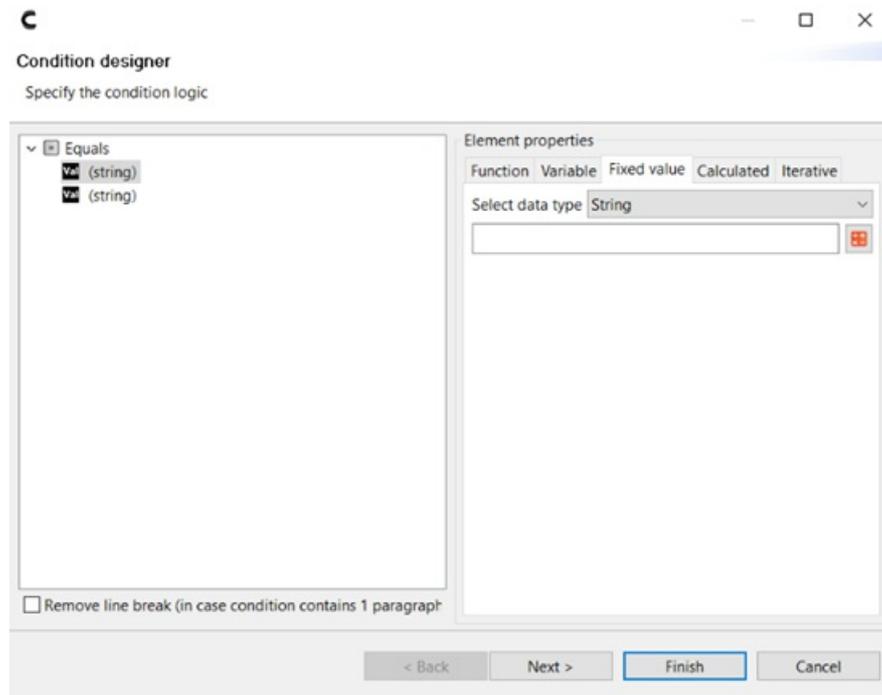
9.3.5. Argument type 3: CALCULATED VARIABLE

If you want to use a calculated variable in a condition, a data structure needs to be attached to your logical model. See [3.2.1 Create a logical model](#) for more information and detailed instructions. In this data structure, at least one calculated variable needs to be defined. See [7.2 Calculated](#) for more information on this topic.

When the **Calculated** tab is selected, the calculated variables that have been created in the XSD are listed.

To use one of these calculated variables in a condition:

- Select the argument.
- In **Element properties**, the **Fixed value** tab is automatically selected.



- Click the **Calculated** tab in the **Element properties** area.
- In the data structure, double-click the calculated variable you want to compare to the other argument.
- The calculated variable is now added as argument.



Condition designer

Specify the condition logic

Equals

- Calculated Number Expression AmountDue+Adminis
 - Add Operator
 - DataBlocks/Invoice/AmountDue
 - Val 16 (Number)
 - Val 0 (Number)

Element properties

Function Variable Fixed value Calculated Iterative

AmountDue+AdministrationFee

Remove line break (in case condition contains 1 paragraph)

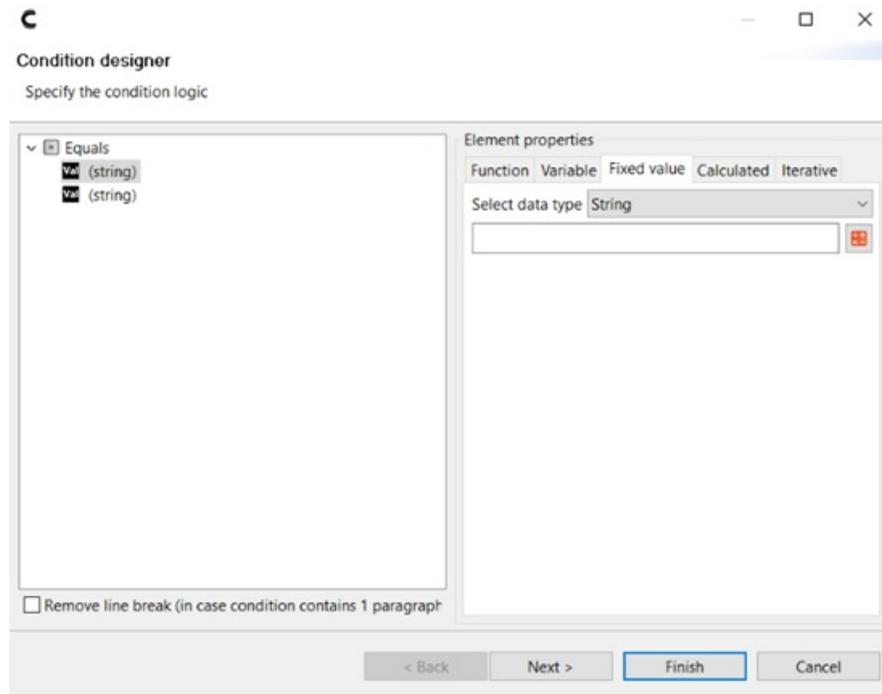
< Back Next > Finish Cancel

9.3.6. Argument type 4: FUNCTION

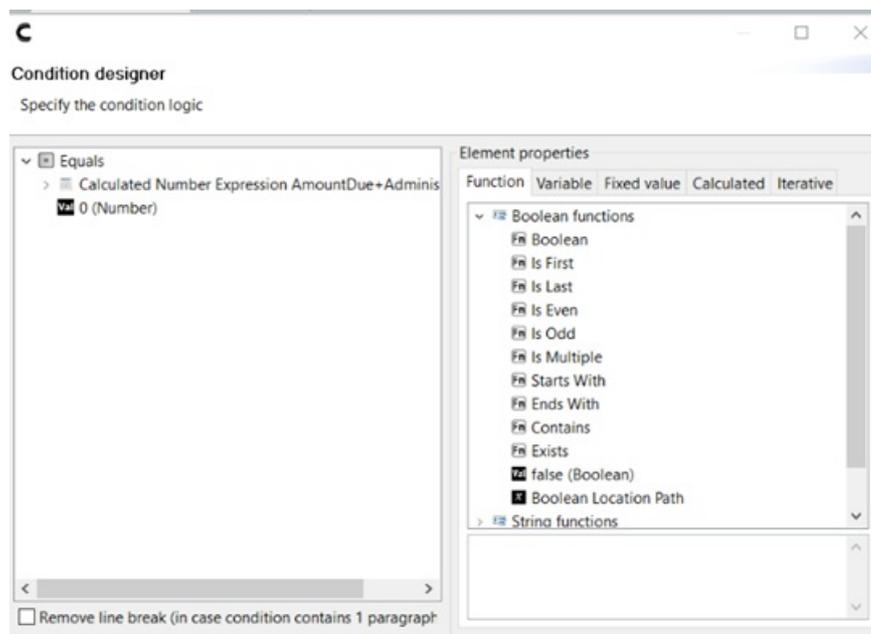
There are 3 main types of functions: Boolean functions, String functions and Number functions. In the next paragraphs, we will give an overview of all available functions.

To use a function in a condition:

- Select the **argument**.
- In **Element properties**, the **Fixed value** tab is automatically selected.



- Click the **Function** in the **Element properties** area.
- Click the triangle in front of the function type you want to use.
- The function is now added as argument.

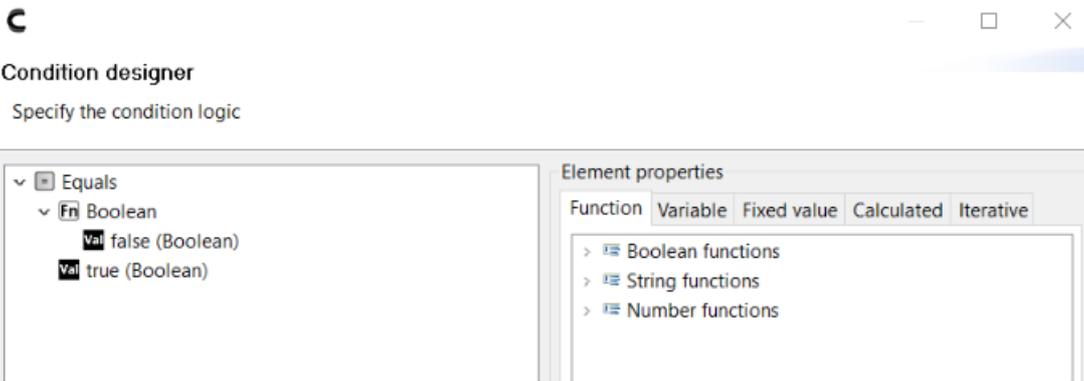


Boolean functions: BOOLEAN

This function returns a Boolean value for a number, string or node-set and indicates whether the element is present in the sample data or not.

This function doesn't need another parameter.

- If the variable is **present** in the XML, the condition is **true**.
- If the variable is **not present** in the XML, the condition is **false**.



Boolean functions: IS FIRST

This function is applied on a repeating element and checks the current position of the repeating data block in a loop. The function allows you to check if the specific occurrence is the first occurrence and returns a Boolean value (true/false).

This function doesn't need another parameter.

- If the current position is the **first** occurrence, the condition is **true**.
- If the current position is **not the first** occurrence, the condition is **false**.

Condition designer

Specify the condition logic



Boolean functions: IS LAST

This function is applied on a repeating element and checks the current position of the repeating data block in a loop. The function allows you to check if the specific occurrence is the last occurrence and returns a Boolean value (true/false).

This function doesn't need another parameter.

- If the current position is the **last** occurrence, the condition is **true**.
- If the current position is **not the last** occurrence, the condition is **false**.

Condition designer

Specify the condition logic



Boolean functions: IS EVEN

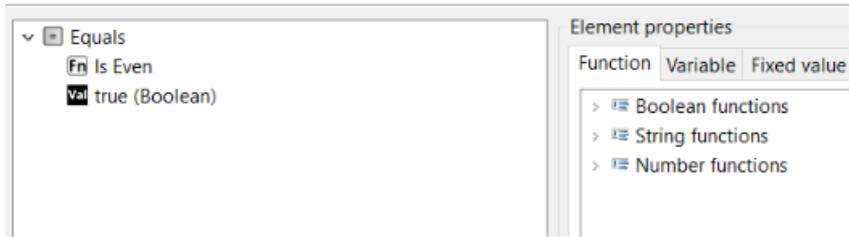
This function is applied on a repeating element and checks the current position of the repeating data block in a loop. The function allows you to check if the specific occurrence is even (so the 2nd, 4th, 6th, etc. occurrence) and returns a Boolean value (true/false).

This function doesn't need another parameter.

- If the current position is **even**, the condition is **true**.
- If the current position is **odd**, the condition is **false**.

Condition designer

Specify the condition logic



Boolean functions: IS ODD

This function is applied on a repeating element and checks the current position of the repeating data block in a loop. The function allows you to check if the specific occurrence is odd (so the 1st, 3rd, 5th, etc. occurrence) and returns a Boolean value (true/false).

This function doesn't need another parameter.

- If the current position is **odd**, the condition is **true**.
- If the current position is **even**, the condition is **false**.

Condition designer

Specify the condition logic



Boolean functions: IS MULTIPLE

This function is applied on a repeating element and checks the current position of the repeating data block in a loop. The function allows you to check if the specific occurrence is a multiple of a given number and returns a Boolean value (true/false).

This function needs a parameter. This parameter can be any fixed value, variable or other function that will return a numeric value.

- If the current position is a **multiple** of the number defined in the parameter, the condition is **true**.
- If the current position is **not a multiple** of the number defined in the parameter, the condition is **false**.

Condition designer

Specify the condition logic



Boolean functions: STARTS WITH

This function checks if a string variable starts with a specific string and returns a Boolean value (true/false).

This function needs 2 parameters. The first parameter is the source string. The second parameter is the string which is to be

matched. Both parameters can be a fixed value, a (calculated) variable or another function that will return a string value.

- If the source string **starts** with the string defined in the 2nd parameter, the condition is **true**.
- If the source string does **not start** with the string defined in the 2nd parameter, the condition is **false**.

Condition designer

Specify the condition logic



Boolean functions: ENDS WITH

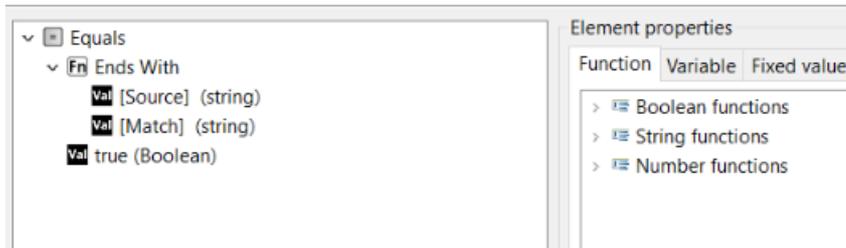
This function checks if a string variable ends with a specific string and returns a Boolean value (true/false).

This function needs 2 parameters. The first parameter is the source string. The second parameter is the string which is to be matched. Both parameters can be a fixed value, a (calculated) variable or another function that will return a string value.

- If the source string **ends** with the string defined in the 2nd parameter, the condition is **true**.
- If the source string does **not end** with the string defined in the 2nd parameter, the condition is **false**.

Condition designer

Specify the condition logic



Boolean functions: CONTAINS

This function checks if a string variable contains a specific string and returns a Boolean value (true/false).

This function needs 2 parameters. The first parameter is the source string. The second parameter is the string which is to be matched. Both parameters can be a fixed value, a (calculated) variable or another function that will return a string value.

- If the source string **contains** the string defined in the 2nd parameter, the condition is **true**.
- If the source string does **not contain** the string defined in the 2nd parameter, the condition is **false**.

Condition designer

Specify the condition logic



Boolean functions: EXISTS

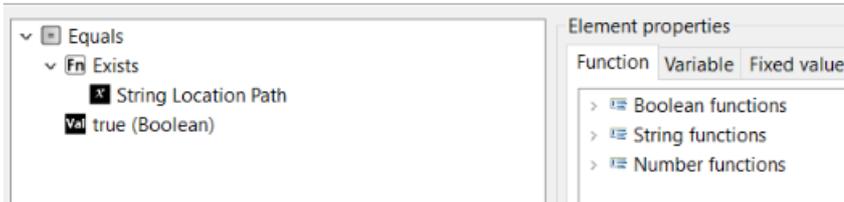
This function checks if a variable exists and returns a Boolean value (true/false).

This function needs a parameter, which existence will be checked. This parameter can be a variable or a calculated variable.

- If the variable **exists** (this means that the variable contains a value or is empty), the condition is **true**.
- If the variable does **not exist** (this means that the variable does not exist in the sample data, but is described in the data structure), the condition is **false**.

Condition designer

Specify the condition logic



String functions: STRING

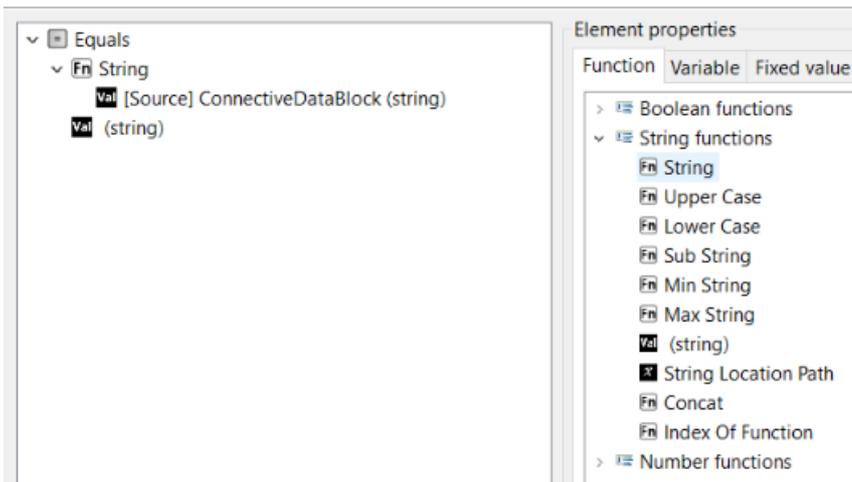
This function allows you to get the string representation of any value.

This function needs a parameter, which value will be represented as a string. This parameter can be a fixed value, a (calculated) variable or another function.

- If the string representation of the parameter **equals** the string value defined in the 2nd argument, the condition is **true**.
- If the string representation of the parameter does **not equal** the string value defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



String functions: UPPER CASE

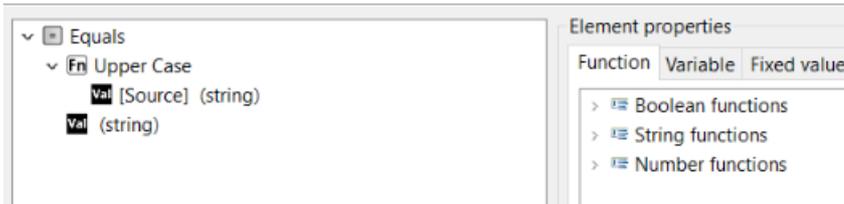
This function allows you to translate a string to upper case.

This function needs a string parameter. This parameter can be a fixed value, a (calculated) variable or another function.

- If the upper case representation of the parameter **equals** the string value defined in the 2nd argument, the condition is **true**.
- If the upper case representation of the parameter **does not equal** the string value defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



String functions: LOWER CASE

This function allows you to translate a string to lower case.

This function needs a string parameter. This parameter can be a fixed value, a (calculated) variable or another function.

- If the lower case representation of the parameter **equals** the string value defined in the 2nd argument, the condition is **true**.
- If the lower case representation of the parameter **does not equal** the string value defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



String functions: SUB STRING

This function allows you to extract a part of a string variable.

This function needs 3 parameters. The first parameter is the source string and can be a fixed value, a (calculated) variable or another function. The second parameter is an integer value indicating the start position for the extraction. The third (optional) parameter is also an integer value indicating the end position for the extraction.

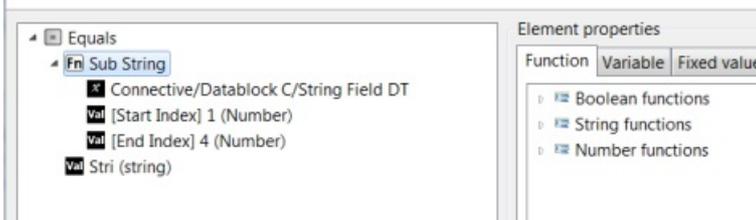
Example: substring('Beatles',1,4) -> The result is 'Beat'

Example: substring('Beatles',2) -> The result is 'eatles'

- If the extracted value **equals** the string value defined in the 2nd argument, the condition is **true**.
- If the extracted value **does not equal** the string value defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



String functions: MIN STRING

This function allows you to get the lowest value of a string variable in a repeating data block, without actually having to create a loop around this variable.

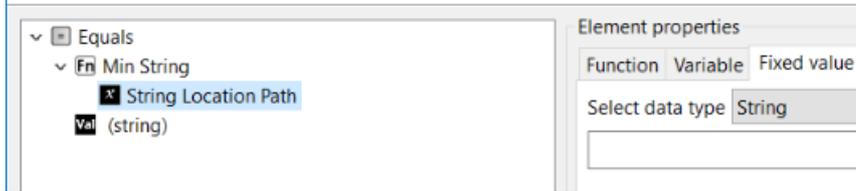
Example: STRING_1 is part of a repeating data block. There are 3 occurrences of this variable and the different values are 'm', 's' and 'a'. The lowest value of STRING_1 is then 'a'.

This function needs a string parameter. This parameter can be a fixed value, a (calculated) variable or another function.

- If the lowest value of the string variable **equals** the string value defined in the 2nd argument, the condition is **true**.
- If the lowest value of the string variable **does not equal** the string value defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



String functions: MAX STRING

This function allows you to get the highest value of a string variable in a repeating data block, without actually having to create a loop around this variable.

Example: STRING_1 is part of a repeating data block. There are 3 occurrences of this variable and the different values are 'm', 's' and 'a'. The highest value of STRING_1 is then 's'.

This function needs a string parameter. This parameter can be a fixed value, a (calculated) variable or another function.

- If the highest value of the string variable **equals** the string value defined in the 2nd argument, the condition is **true**.
- If the highest value of the string variable **does not equal** the string value defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



String functions: CONCAT

This function allows you to concatenate two string variables and compare the result of this concatenation.

This function needs 2 parameters. These parameters are the values to concatenate.

- If the concatenation of the parameters **equals** the string value defined in the 2nd argument, the condition is **true**.
- If the concatenation of the parameters **does not equal** the string value defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



String functions: INDEX OF FUNCTION

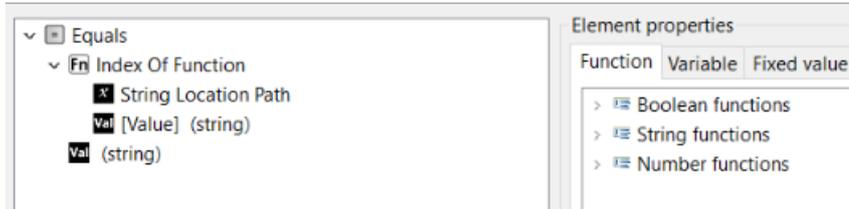
This function allows you to search for a value in a string variable.

This function needs 2 parameters. The first parameter is the search string, what you would like to search for. The second parameter is optional and defines the position where you want the search to begin.

- If the search string is **present** in the 1st parameter, the condition is **true**.
- If the search string is **not present** in the 1st parameter, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: MOD

This function returns the remainder of a division. The second argument of the condition is a number.

This function needs 2 parameters, both numerical values.

- If the remainder of the 1st parameter divided by the 2nd parameter **equals** the value of the 2nd argument, the condition is **true**.
- If the remainder of the 1st parameter divided by the 2nd parameter **does not equal** the value of the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: STRING LENGTH

This function allows you to get the length of a specific string field. The second argument of the condition is a number.

This function needs a parameter, which can be a fixed value, a (calculated) variable or another function.

- If the string length of the parameter **equals** the value of the 2nd argument, the condition is **true**.
- If the string length of the parameter **does not equal** the value of the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: COUNT

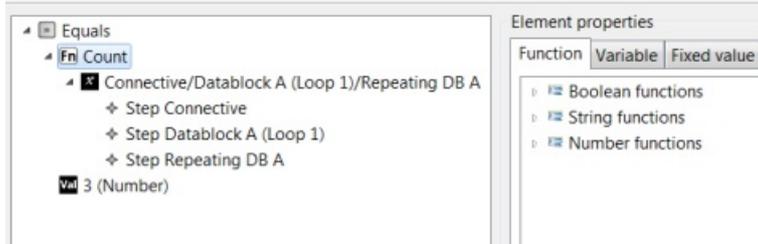
This function allows you to get the number of iterations in a loop. In other words, this function returns the number of occurrences for a repeating data block in the sample data. The second argument of the condition must be a numeric value.

This function needs a parameter, this is the repeating data block.

- If the number of occurrences for a repeating data block **equals** the number defined in the 2nd argument, the condition is **true**.
- If the number of occurrences for a repeating data block does **not equal** the number defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: AVERAGE

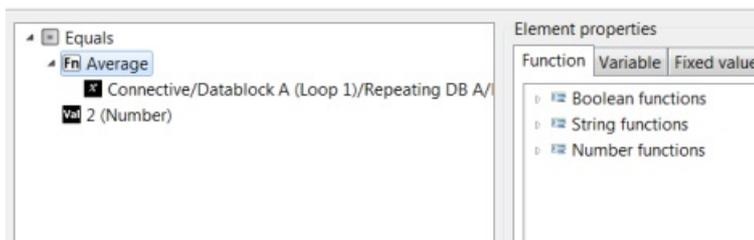
This function returns the average numeric value of the selected variable in an iterative block without actually creating a loop. The second argument of the condition must be a numeric value.

This function needs a parameter, this is the variable that belongs to a repeating data block.

- If the average of the selected variable in a repeating data block **equals** the number defined in the 2nd argument, the condition is **true**.
- If the average of the selected variable in a repeating data block does **not equal** the number defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: SUM

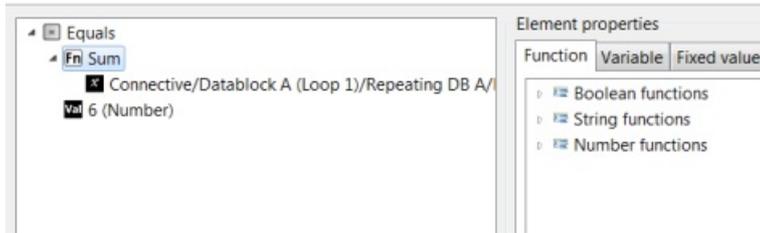
This function returns the total numeric value of the selected variable in an iterative block without actually creating a loop. The second argument of the condition must be a numeric value.

This function needs a parameter, this is the variable that belongs to a repeating data block.

- If the sum of the selected variable in a repeating data block **equals** the number defined in the 2nd argument, the condition is **true**.
- If the sum of the selected variable in a repeating data block does **not equal** the number defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: MIN NUMBER

This function allows you to get the lowest value of a numeric variable in a repeating data block, without actually having to create a loop around this variable. The second argument of the condition must be a numeric value.

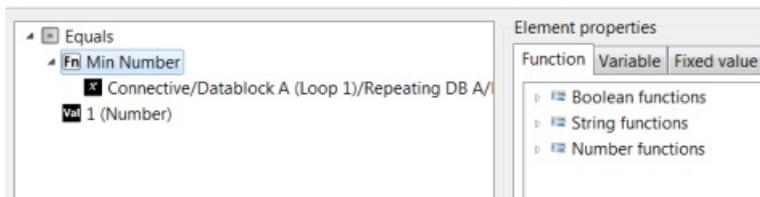
Example: NUM_1 is part of a repeating data block. There are 3 occurrences of this variable and the different values are '4', '7' and '3'. The lowest value of NUM_1 is then '3'.

This function needs a numeric parameter. This parameter can be a fixed value, a (calculated) variable or another function.

- If the lowest value of the numeric variable **equals** the numeric value defined in the 2nd argument, the condition is **true**.
- If the lowest value of the numeric variable does **not equal** the numeric value defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: MAX NUMBER

This function allows you to get the highest value of a numeric variable in a repeating data block, without actually having to create a loop around this variable. The second argument of the condition must be a numeric value.

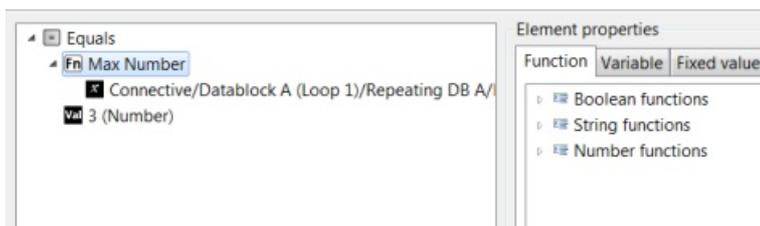
Example: NUM_1 is part of a repeating data block. There are 3 occurrences of this variable and the different values are '4', '7' and '3'. The highest value of NUM_1 is then '7'.

This function needs a numeric parameter. This parameter can be a fixed value, a (calculated) variable or another function.

- If the highest value of the numeric variable **equals** the numeric value defined in the 2nd argument, the condition is **true**.
- If the highest value of the numeric variable does **not equal** the numeric value defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: NUMBER

This function allows you to convert a variable to a number, as long as the variable to convert is valid. The second argument of the

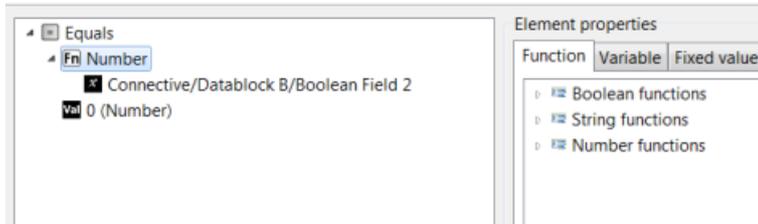
condition must be a numeric value.

This function needs a parameter, which value will be represented as a number. This parameter can be a fixed value, a (calculated) variable or another function.

- If the number representation of the parameter **equals** the numeric value defined in the 2nd argument, the condition is **true**.
- If the number representation of the parameter does **not equal** the numeric value defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: ABSOLUTE VALUE

This function allows you to get the absolute value, this is the positive value, of a numeric variable. The second argument of the condition must be a numeric value.

This function needs a numeric parameter. This parameter can be a fixed value, a (calculated) variable or another function.

- If the absolute value of the parameter **equals** the numeric value defined in the 2nd argument, the condition is **true**.
- If the absolute value of the parameter does **not equal** the numeric value defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: CEILING

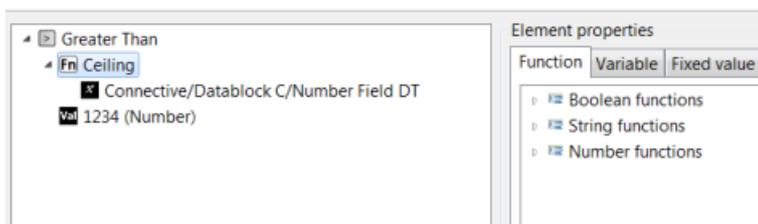
This function allows you to get the smallest integer that is greater than the selected decimal value. The second argument of the condition must be a numeric value.

This function needs a numeric parameter. This parameter can be a fixed value, a (calculated) variable or another function.

- If the ceiling value of the parameter **equals** the numeric value defined in the 2nd argument, the condition is **true**.
- If the ceiling value of the parameter does **not equal** the numeric value defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic

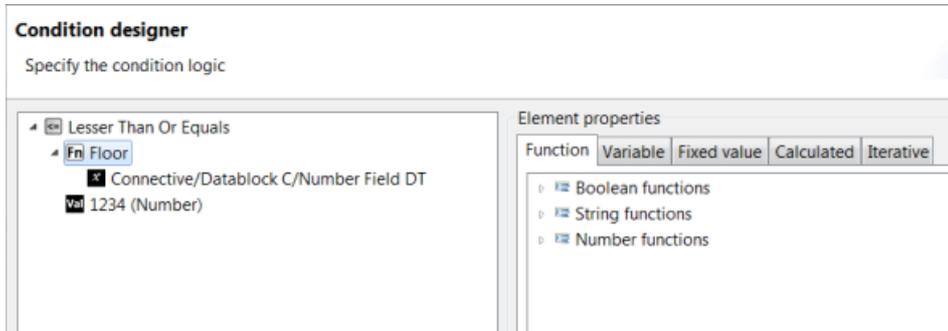


Number functions: FLOOR

This function returns the largest integer that is not greater than the selected decimal variable. The second argument of the condition must be a numeric value.

This function needs a numeric parameter. This parameter can be a fixed value, a (calculated) variable or another function.

- If the floor value of the parameter **equals** the numeric value defined in the 2nd argument, the condition is **true**.
- If the floor value of the parameter does **not equal** the numeric value defined in the 2nd argument, the condition is **false**.



Number functions: ROUND

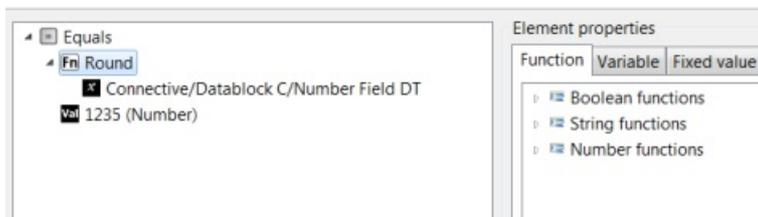
This function rounds to the nearest integer of the selected decimal variable. The second argument of the condition must be a numeric value.

This function needs a numeric parameter. This parameter can be a fixed value, a (calculated) variable or another function.

- If the rounded value of the parameter **equals** the numeric value defined in the 2nd argument, the condition is **true**.
- If the rounded value of the parameter does **not equal** the numeric value defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: POSITION

This function allows you to get the current iteration position in a loop. The second argument of the condition must be a numeric value.

This function doesn't need a parameter.

- If the position of the repeating data block **equals** the number defined in the 2nd argument, the condition is **true**.
- If the position of the repeating data block does **not equal** the number defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic

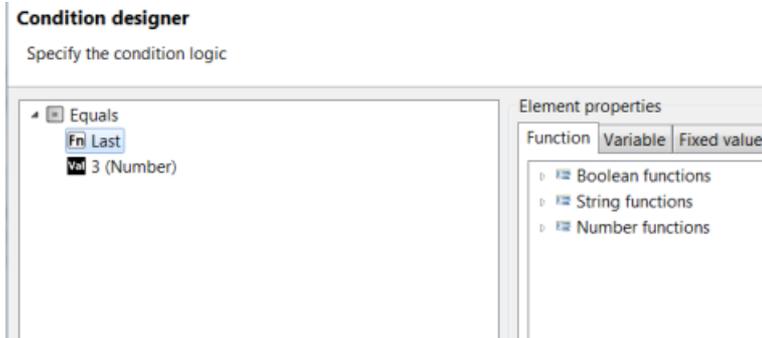


Number functions: LAST

This function allows you to get the number of occurrences of a repeating data block. The second argument of the condition must be a numeric value.

This function doesn't need a parameter.

- If the number of occurrences of the repeating data block **equals** the number defined in the 2nd argument, the condition is **true**.
- If the number of occurrences of the repeating data block does **not equal** the number defined in the 2nd argument, the condition is **false**.

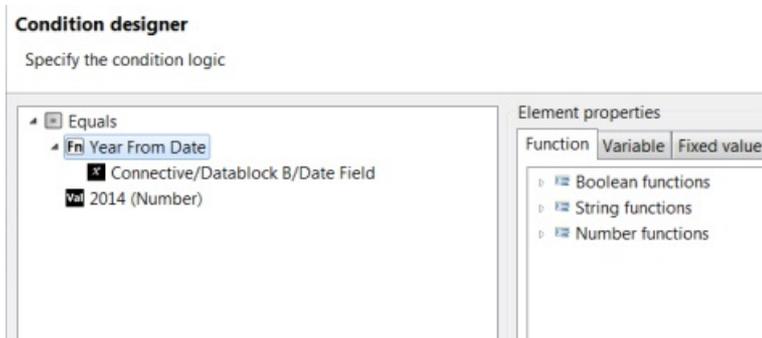


Number functions: YEAR FROM DATE

This function allows you to extract the year from a variable of the data type 'Date'. The second argument of the condition must be an integer.

This function needs a parameter of the data type 'Date'. This parameter can be a fixed value, a (calculated) variable or another function.

- If the extracted value **equals** the number defined in the 2nd argument, the condition is **true**.
- If the extracted value does **not equal** the number defined in the 2nd argument, the condition is **false**.



Number functions: YEAR FROM DATETIME

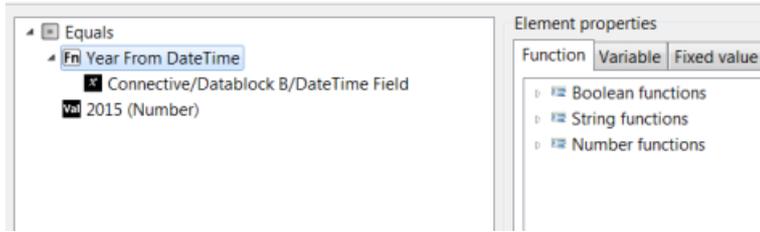
This function allows you to extract the year from a variable of the data type 'DateTime'. The second argument of the condition must be an integer.

This function needs a parameter of the data type 'DateTime'. This parameter can be a fixed value, a (calculated) variable or another function.

- If the extracted value **equals** the number defined in the 2nd argument, the condition is **true**.
- If the extracted value does **not equal** the number defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: MONTH FROM DATE

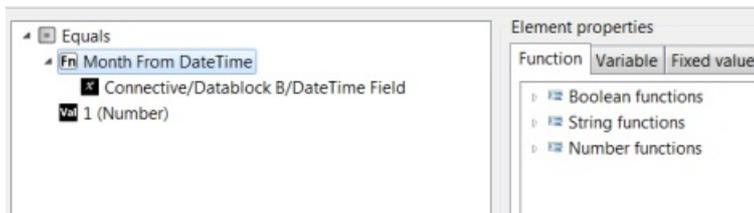
This function allows you to extract the month from a variable of the data type 'Date'. The second argument of the condition must be an integer.

This function needs a parameter of the data type 'Date'. This parameter can be a fixed value, a (calculated) variable or another function.

- If the extracted value **equals** the number defined in the 2nd argument, the condition is **true**.
- If the extracted value does **not equal** the number defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: MONTH FROM DATETIME

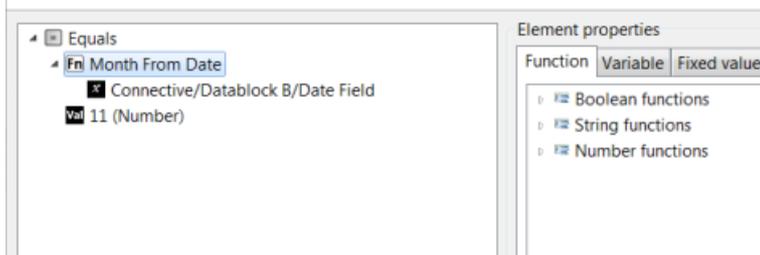
This function allows you to extract the month from a variable of the data type 'DateTime'. The second argument of the condition must be an integer.

This function needs a parameter of the data type 'DateTime'. This parameter can be a fixed value, a (calculated) variable or another function.

- If the extracted value **equals** the number defined in the 2nd argument, the condition is **true**.
- If the extracted value does **not equal** the number defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: DAY FROM DATE

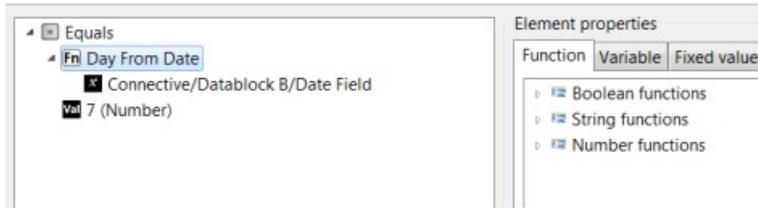
This function allows you to extract the day from a variable of the data type 'Date'. The second argument of the condition must be an integer.

This function needs a parameter of the data type 'Date'. This parameter can be a fixed value, a (calculated) variable or another function.

- If the extracted value **equals** the number defined in the 2nd argument, the condition is **true**.
- If the extracted value does **not equal** the number defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: DAY FROM DATETIME

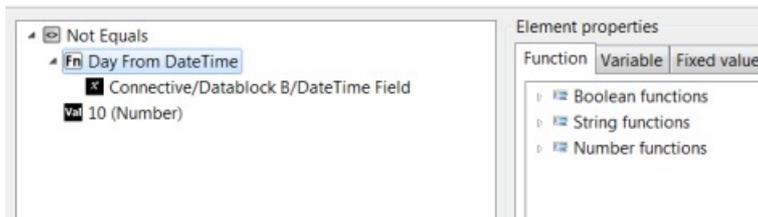
This function allows you to extract the day from a variable of the data type 'DateTime'. The second argument of the condition must be an integer.

This function needs a parameter of the data type 'DateTime'. This parameter can be a fixed value, a (calculated) variable or another function.

- If the extracted value **equals** the number defined in the 2nd argument, the condition is **true**.
- If the extracted value does **not equal** the number defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



Number functions: HOURS FROM DATETIME

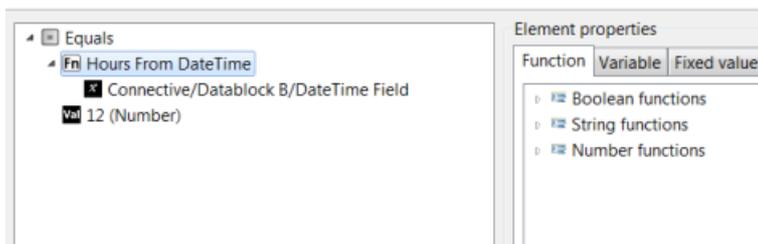
This function allows you to extract the hours from a variable of the data type 'DateTime'. The second argument of the condition must be an integer.

This function needs a parameter of the data type 'DateTime'. This parameter can be a fixed value, a (calculated) variable or another function.

- If the extracted value **equals** the number defined in the 2nd argument, the condition is **true**.
- If the extracted value does **not equal** the number defined in the 2nd argument, the condition is **false**.

Condition designer

Specify the condition logic



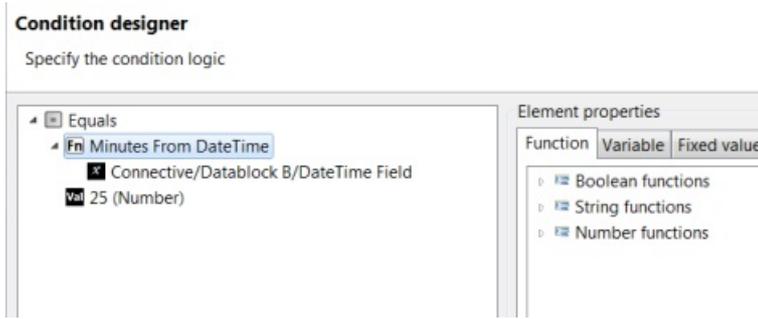
Number functions: MINUTES FROM DATETIME

This function allows you to extract the minutes from a variable of the data type 'DateTime'. The second argument of the condition must be an integer.

This function needs a parameter of the data type 'DateTime'. This parameter can be a fixed value, a (calculated) variable or another

function.

- If the extracted value **equals** the number defined in the 2nd argument, the condition is **true**.
- If the extracted value does **not equal** the number defined in the 2nd argument, the condition is **false**.

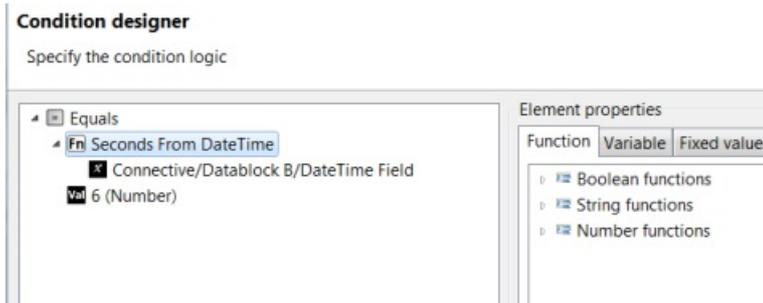


Number functions: SECONDS FROM DATETIME

This function allows you to extract the seconds from a variable of the data type 'DateTime'. The second argument of the condition must be an integer.

This function needs a parameter of the data type 'DateTime'. This parameter can be a fixed value, a (calculated) variable or another function.

- If the extracted value **equals** the number defined in the 2nd argument, the condition is **true**.
- If the extracted value does **not equal** the number defined in the 2nd argument, the condition is **false**.



Number functions: INTEGRAL PART

This function allows you to extract the integral part from a numeric variable. The second argument of the condition must be an integer.

This function needs a numeric parameter. This parameter can be a fixed value, a (calculated) variable or another function.

- If the extracted value **equals** the number defined in the 2nd argument, the condition is **true**.
- If the extracted value does **not equal** the number defined in the 2nd argument, the condition is **false**.



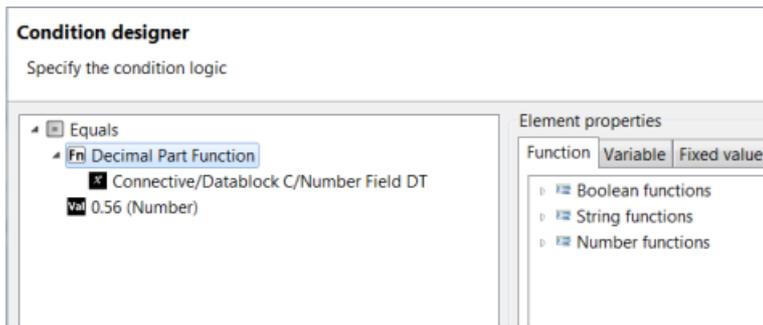
Number functions: DECIMAL PART

This function allows you to extract the decimal part from a numeric variable. The second argument of the condition must be an integer.

This function needs a numeric parameter. This parameter can be a fixed value, a (calculated) variable or another function.

- If the extracted value **equals** the number defined in the 2nd argument, the condition is **true**.
- If the extracted value does **not equal** the number defined in the 2nd argument, the condition is **false**.

Condition designer
Specify the condition logic



The screenshot shows the 'Condition designer' interface. On the left, a tree view shows a hierarchy: 'Equals' (selected), 'Fn Decimal Part Function', 'Connective/Datablock C/Number Field DT', and 'Val 0.56 (Number)'. On the right, the 'Element properties' panel has three tabs: 'Function', 'Variable', and 'Fixed value'. The 'Function' tab is active, showing a list of function categories: 'Boolean functions', 'String functions', and 'Number functions'.

9.3.7. Modify how the conditional text will be displayed

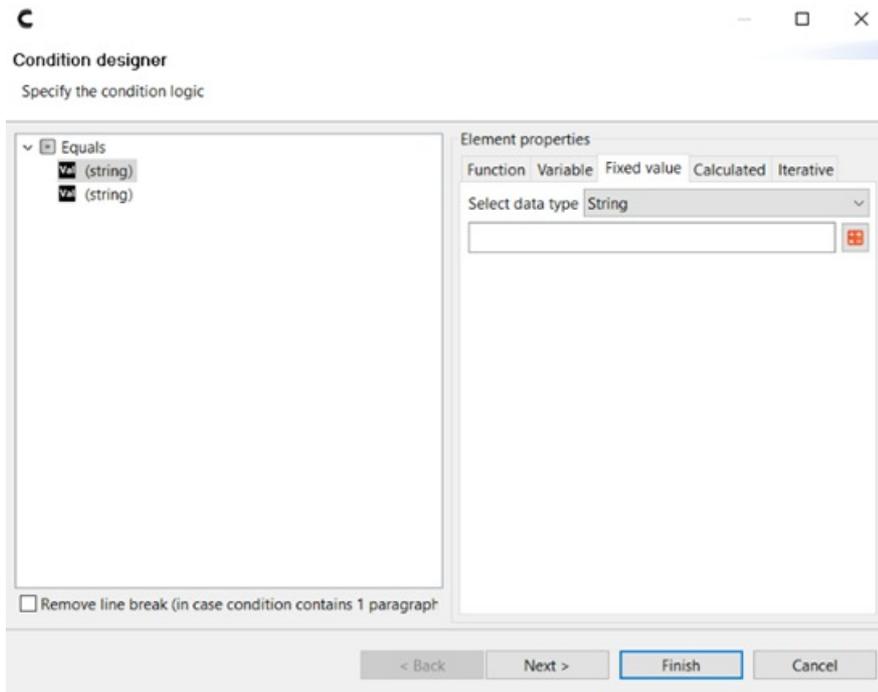
By default, conditions are used to display a selection of text when the condition is valid. It is possible, however, to modify how the conditional text will be displayed. These changes can be done in the Condition designer.

Structure of the Condition designer

The structure of the Condition designer is **IF** (Expression), **THEN** (Action), **ELSE** (Other Action):

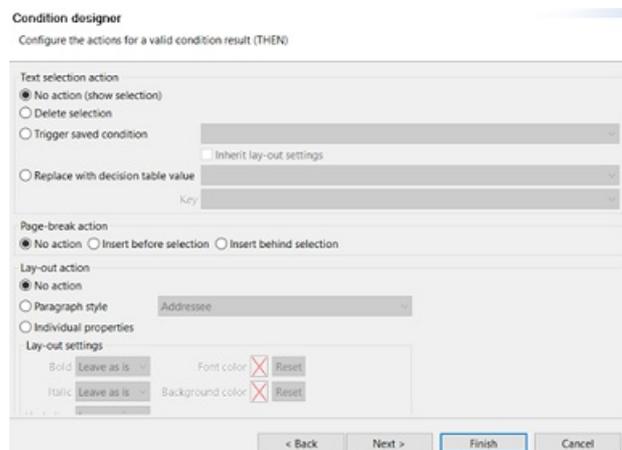
IF

- By default, the Condition designer shows the window where you have to specify the **condition logic**.
- Click **Next**.



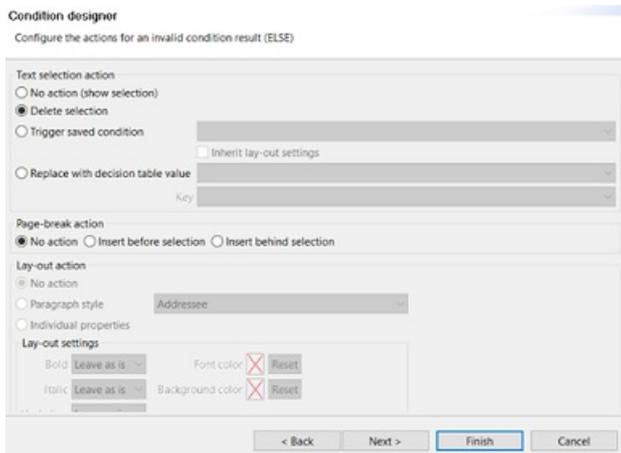
THEN

- Actions for a **valid** condition result.
- Click **Next**.



ELSE

- Actions for an **invalid** condition result.



Configure the actions for a condition

The options are the same for both the THEN and the ELSE clause, so the same options are available for both a valid condition and an invalid condition.

Available configurations for the conditional text are:

- **Text selection**
 - No action: the selection will be visible (by default for THEN clause).
 - Delete selection: the selected text will be deleted (by default for IF clause).
 - Trigger subcondition: you can select a saved condition from the drop-down list. This additional condition will then be applied to the selection.
 - Replace with decision table value: you can select a decision table from the drop-down list. The value of the selected key will then be printed.

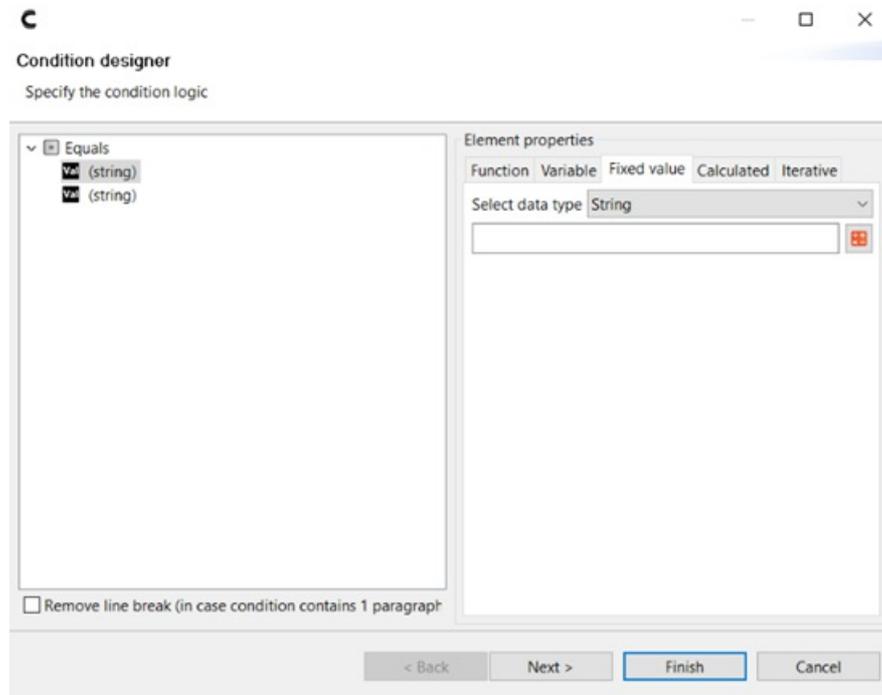
- **Page break action**
 - No action: no page break will be inserted (by default).
 - Insert before selection: a page break will be inserted before the selected text.
 - Insert behind selection: a page break will be inserted after the selected text.

- **Layout action**
 - No action: the layout will not be modified (by default).
 - Paragraph style: apply a different paragraph style to the selected text.
 - Individual properties: modify the characteristics of the font (bold, italic, underline, font color, background color).

9.4 How to modify a condition

To modify an existing condition:

- Double-click on the start tag of the condition .
- The **Condition designer** wizard is opened.
- You can now modify the condition.



9.5 How to delete a condition

To delete a condition:

- Delete the end tag of the condition .
- Click **OK** to confirm.
- The condition is removed from the document.

9.6 Reusability of a condition

Conditions can be saved in a logical model, so that you can easily reuse these conditions in other places in the document. For more information and detailed instructions, see [7.3.1 Saved condition](#).

9.7 Conditional Fragments

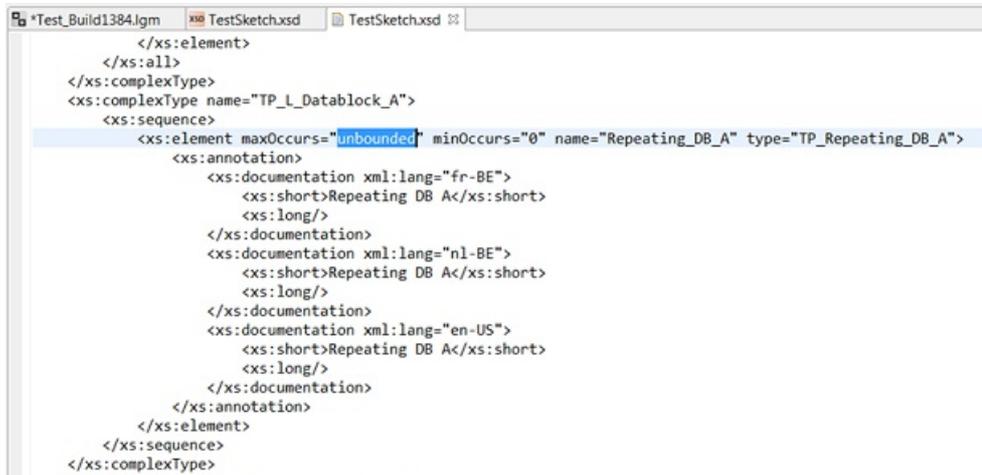
Fragments can be made conditional when you insert them into your document. For more information and detailed instructions, see [8.1.1 Conditional fragment](#).

10. Adding logic: Loop

In order to make your document dynamic, Sketch allows you to easily create loops. These logical building blocks are added in a Wizard approach, which makes it possible for everyone to add logic in a simple way.

A loop is used to represent the repeating data that is present in the data structure attached to the logical model.

In the XSD, you can recognize a repeating element when the parameter `maxOccurs > 1`:



```
</xs:element>
</xs:all>
</xs:complexType>
<xs:complexType name="TP_L_Datablock_A">
  <xs:sequence>
    <xs:element maxOccurs="unbounded" minOccurs="0" name="Repeating_DB_A" type="TP_Repeating_DB_A">
      <xs:annotation>
        <xs:documentation xml:lang="fr-BE">
          <xs:short>Repeating DB A</xs:short>
          <xs:long/>
        </xs:documentation>
        <xs:documentation xml:lang="nl-BE">
          <xs:short>Repeating DB A</xs:short>
          <xs:long/>
        </xs:documentation>
        <xs:documentation xml:lang="en-US">
          <xs:short>Repeating DB A</xs:short>
          <xs:long/>
        </xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
</xs:complexType>
```

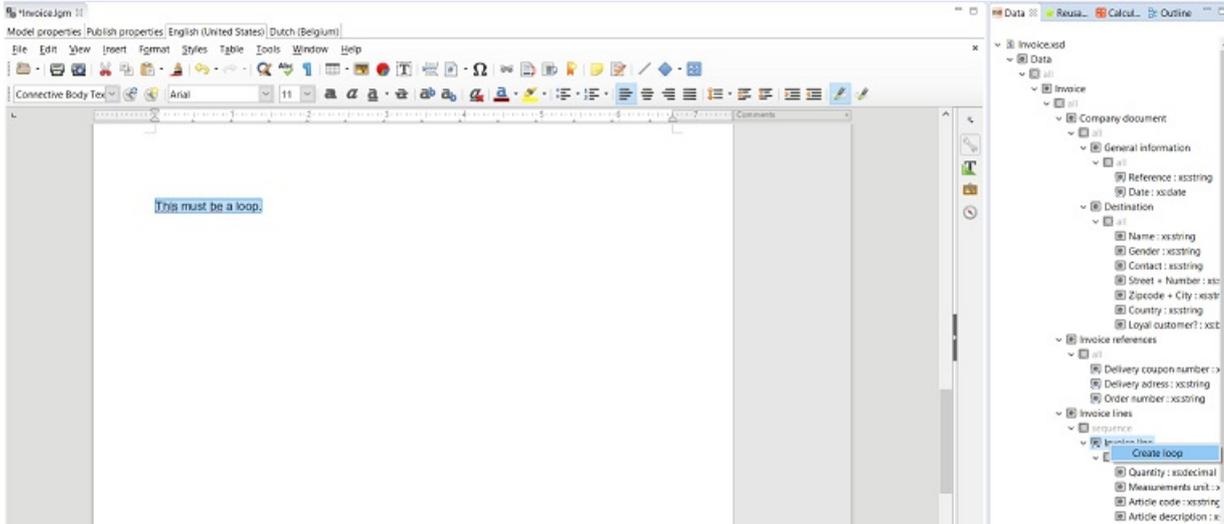
Combining a loop with a condition inside is possible but should be done in an 'inside out' operation. Build the condition first and then the loop around it.

10.1 How to create a loop

10.1.1. Create a loop via Data Navigator

To create a loop via the Data Navigator:

- Make sure the logical model is [checked out](#).
- Open the language model.
- Select the text you want to apply the loop on.
- Unfold the data structure in the Data Navigator at the right.
- Right-click the iterative element and click **Create loop**.



- A default loop is added around the selected text.
- The **Loop Designer** is *not* shown.

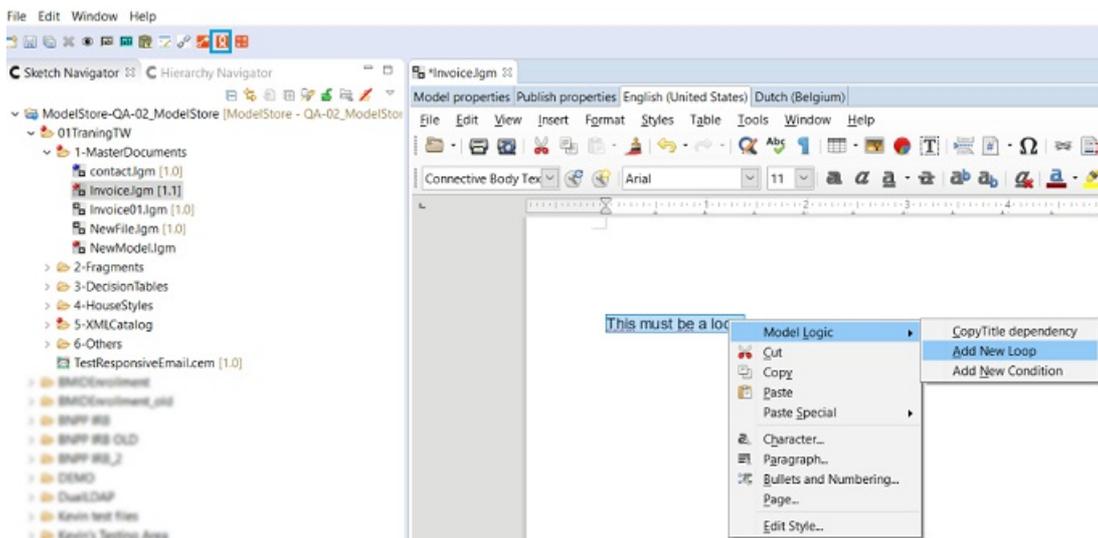
10.1.2. Create a loop via Loop Designer

To create a loop via the Loop Designer:

- Make sure the logical model is [checked out](#).
- Open the language model.
- Select the text you want to apply the loop on.
- Right-click and click **Model Logic > Add New Loop**.

Or

- Click the **Loop** icon  in the toolbar.

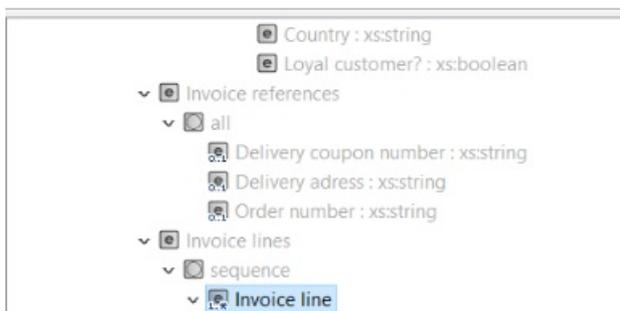


- The **Loop Designer** is opened.
- Right-click the Data structure and click **Expand all**.
- Select the iterative element. You'll notice that only elements available for iteration are clickable.

C

Loop Designer

Select the iterative element



- Click **Finish**.
- The loop tags are added around the selected text.

Tip: if you have some difficulties to select the text using the mouse, try using the keyboard: hold down the [Shift] key while you move around using the arrow keys. If you still experience some trouble, try selecting from the end to the beginning.

Make sure there is text (or spaces) where you right-click. Otherwise **New Loop** might not appear.

10.2 Visualization of the loop

10.2.1. Representation in the Model Editor

Default loop

A loop is displayed between a red start tag and a red end tag.

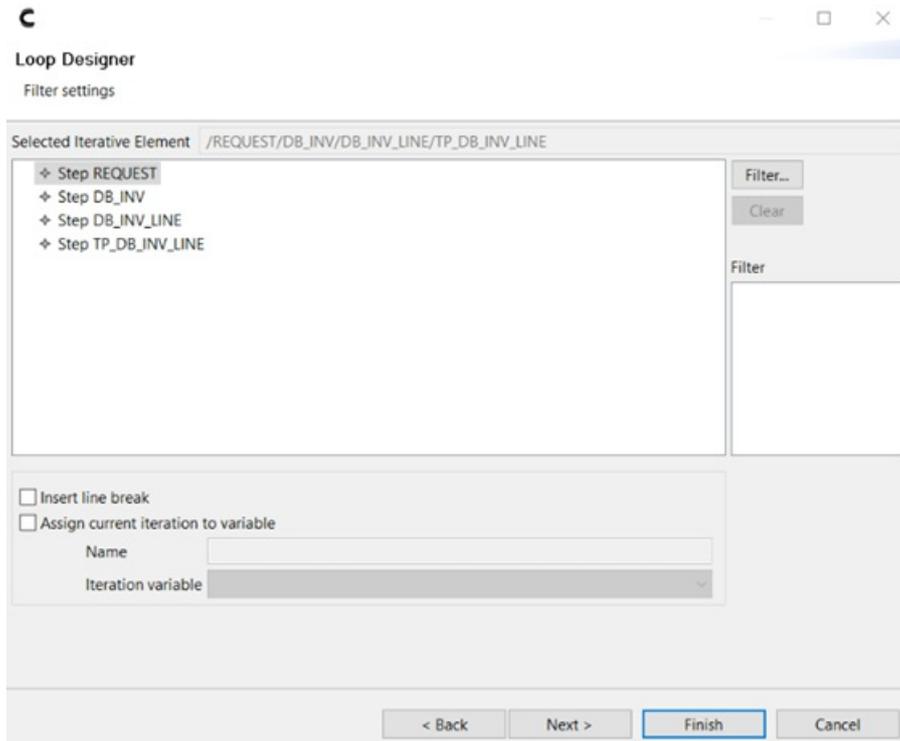


Everything in between these tags is part of the loop. The data can be variables, fragments, pictures, words, sentences, paragraphs or even entire documents containing multiple pages.

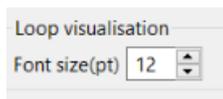
It is possible to change the size of the red frame objects. This is only to help you finetune the appearance of the loops. It will not affect the result after generation.

To change the size of the red frame object:

- Open the loop by double-clicking on the start tag.
- The Loop Designer is opened.
- Click **Next** (3 times).



- At the bottom of the window, you can change the **Font size** under **Loop visualization**.



Loop on table (row)

When using a loop in a table, the number of table rows will grow depending on the occurrences. A loop in a table is displayed between red tags, with an R (rows) added to the opening tag.

Details

Quantity	Article description	Price	Unit	Total
[R]Quantity	Article description	Price	Unit	Total

Tip: Due to layout issues, it is considered a good practice to create loops in a table structure, with header/ titles in row 1 and dynamic content in row 2. This way, the layout of the loop is easily manageable.

Select the entire second row to create the loop as explained above. If however, the loop is set on a single element and a single column table structure is used, you should only select the variable and not the entire row.

10.2.2. Highlight a loop

To see where a loop starts and ends:

- Select the **start tag** of the loop.
- The text in between the frame tags will be **highlighted in red**, this is the repeating data.

|| This must be a loop. ||

10.2.3. Insert line break after every occurrence

If a loop has been applied to only 1 paragraph, there is an option available to easily insert a line break after every occurrence. This way, you don't have to manually insert a line break after your repeating paragraph.

To insert a line break after every occurrence of 1 paragraph:

- Double-click on the start tag of the loop.
- The Loop Designer is opened.
- Check the **Insert line break after every occurrence** check box at the bottom of the window.

EXAMPLE	DESCRIPTION
This is repeating dataThis is repeating dataThis is repeating dataThis is repeating data	Result after preview when checkbox is not checked .
This is repeating data. This is repeating data. This is repeating data. This is repeating data.	Result after preview when checkbox is checked .

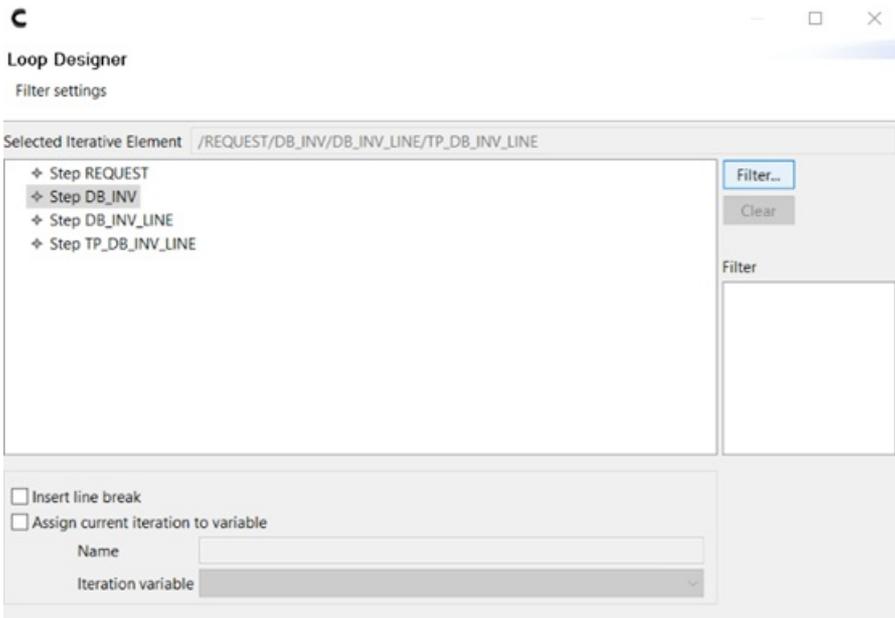
10.3 Loop Logic

10.3.1 Filter

A loop can be combined with a filter. This way, only a selection of the repeating elements will be printed. The filter is defined as a condition (with an operator and arguments defined as a function, (calculated) variable or a fixed value).

To add a filter to the loop:

- Make sure the logical model is checked out.
- Open the language model.
- Double-click on the start tag of the loop to **open the Loop Designer**.
- Select the data block you want to apply the filter on, and then click the **Filter** button.



- The **Loop filter builder** is opened. This is basically the same as the Condition Designer.
- Create the condition. For more information, see [9.1 How to create a condition](#) and [9.3 Condition logic](#).
- Click **Finish**. The filter is now visible under the selected data block.



- Click **Finish**.

10.3.2. Sort

A loop can be sorted. This way, the repeating data will not be displayed as in the sample data, but sorted based on an element you selected. The sort order can be ascending or descending.

To sort a loop:

- Make sure the logical model is checked out.

- Open the language model.
- Double-click on the start tag of the loop to open the **Loop Designer**.
- Click **Next**.
- Expand the data structure and select the element to sort on.



- Select the **sort order** (ascending or descending). By default, ascending is selected.
- Click **Add**. The element is added in the box below the **Add** button and the sort order is indicated.



- Click **Finish**.

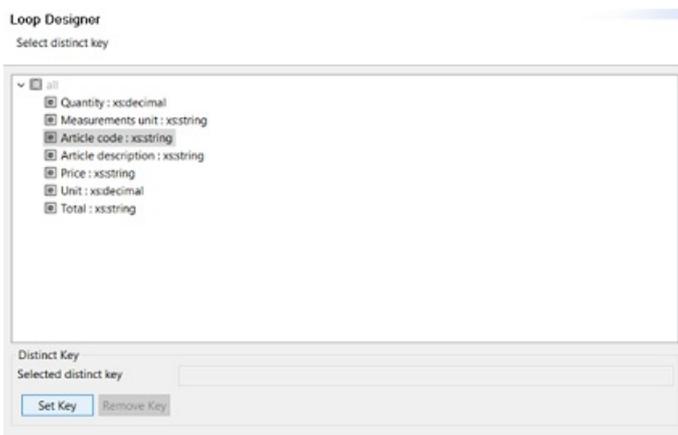
10.3.3. Distinct Key

A loop can also be filtered with a Distinct Key. In this case, only repeating data with a unique variable value are considered. This means that not the entire list of repeating elements is returned but only the unique records. Double records will be removed.

Example: `distinct-values((1, 2, 3, 1, 2))` > Result: (1, 2, 3)

To filter a loop with a distinct key:

- Make sure the logical model is checked out.
- Open the language model.
- Double-click on the start tag of the loop to **open the Loop Designer**.
- Click **Next** (2 times).
- Expand the data structure.
- Select the distinct key you want to use, and click **Set Key**.

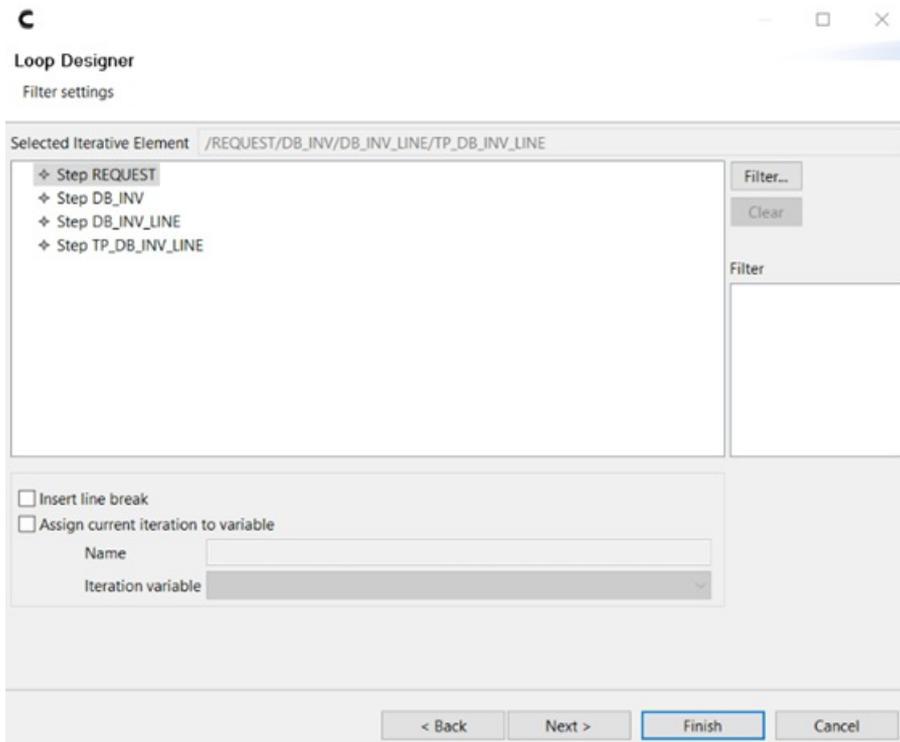


- Click **Finish**.

10.4 Modify a loop

To modify an existing loop:

- Double-click on the start tag of the loop.
- The **Loop Designer** is opened.
- Now you can modify the loop.



10.5 How to delete a loop

To delete a loop:

- Delete the end tag of the loop.

This must be a loop.

- Click **Ok** to confirm.
- The loop is no longer present.

10.6 Reusability of a loop

Loops can be saved in a logical model, so that you can easily reuse these loops in other places in the document. For more information and detailed instructions, see [7.3.2 Saved loop](#).

11. Adding complexity: nested conditions, loops and fragments

In the previous chapters, you have learned about the basic functionalities in Sketch. But in order to create more complex dynamic documents, it is possible to combine these functionalities and have a combination of loops, conditions, variables and fragments.

[11.1 Nested conditions](#)

[11.2 Nested loops](#)

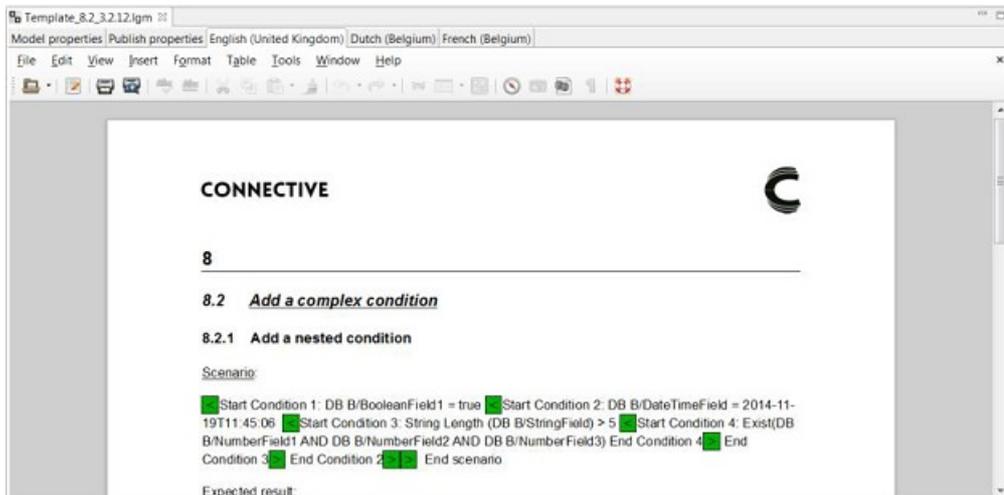
[11.3 Nested fragments](#)

11.1 Nested conditions

A nested condition is the use of a condition in a condition. This way you can create a cascade of conditions: a certain paragraph will be displayed only if a certain condition is true. Within that conditional paragraph, there can be a subset of text that depends on another condition. Only if the first condition is true, the second gets validated.

Building nested conditions is done from the inside out. Add a condition around the inner paragraph first. Afterwards, you add the condition around the outer paragraph, which also includes the inner paragraph.

Example:

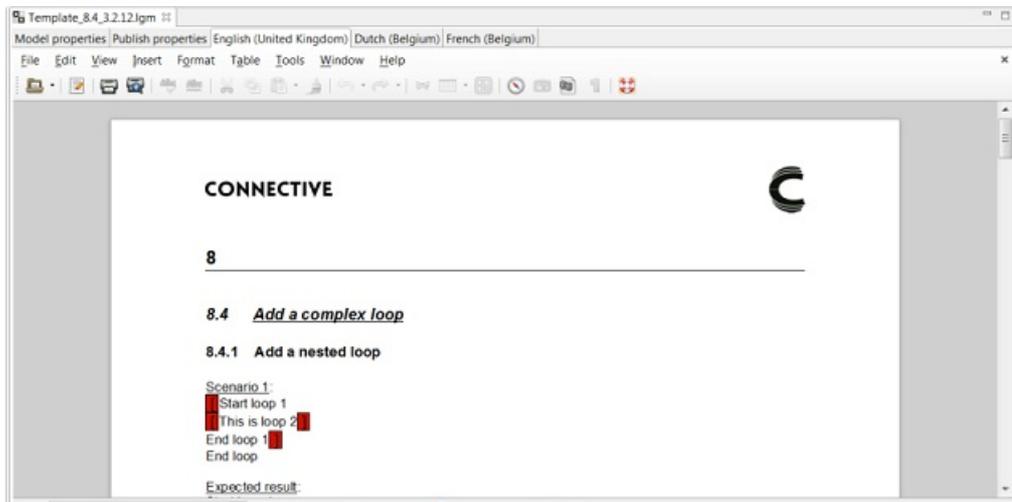


11.2 Nested loops

A nested loop is the use of a loop in a loop in a loop.

The logic is entirely the same as for conditions: always build loops from the inside out.

Example:

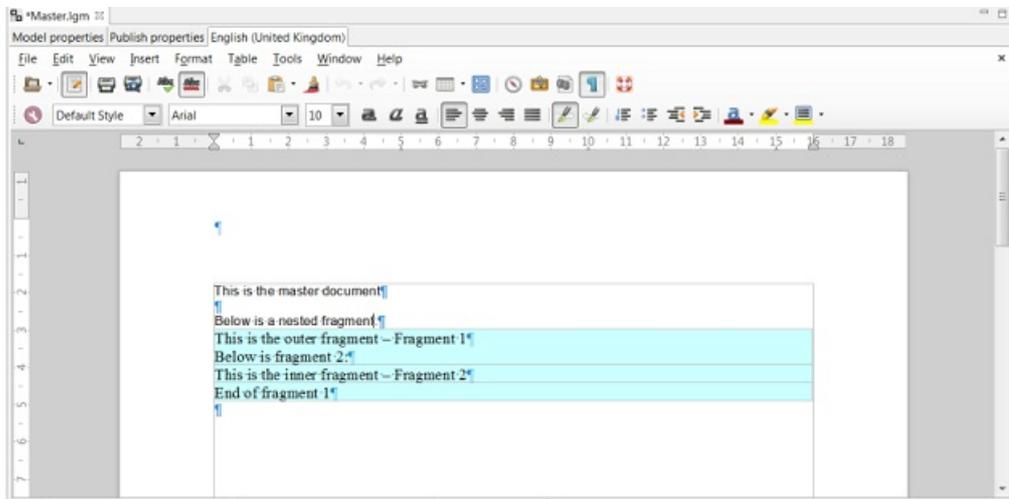


11.3 Nested fragments

A nested fragment is the use of a fragment inside a fragment.

When you use nested fragments, you should be aware of the consequences when you want to change, delete and/or unlink nested fragments. Changes in nested fragments will have a cascade effect on all depending fragments. Changing the content in the inner fragment will change the content of every fragment and/or model that uses the fragment.

Example:



11.3.1 Changing content in linked nested fragments

The content of linked fragments can only be changed in the fragment itself. Changes in that fragment will have their effects on all the other documents using that same fragment.

Changes in fragments will have their effects in other fragments, always from the inside out: the changes in the inner fragment change other fragments who call upon the inner fragment, etc.

If you want to change only the content of the fragment for one document, you have to unlink the fragment first.

11.3.2 Changing content in unlinked nested fragments

After a fragment has been unlinked, that fragment is no longer part of the main document. Only the content of the fragment is still present in the main document, the link between the fragment and the main document has been broken.

When you change the content of what used to be the fragment, this will only impact the main document. There will be no consequences for other documents using that fragment.

12. Copy Titles

The Copy Titles function is designed to create several - customized - copies of one and the same document. In many use cases, an almost identical document must be distributed among different parties. For instance, a copy must be sent to the customer, to the bank and to the bank's archiving department. Each copy, however, contains specific elements that are not required in the other copies.

To allow you to easily generate these different copies, you can add Copy Title-dependent text blocks to your dynamic documents in Sketch. These conditional building blocks will be printed only for specific Copy Title copies. For instance, a copy meant for archiving may contain a barcode or QR code for easy storage and retrieval. This QR or barcode will only be visible in the copy meant for the archiving department.

12.1 How to manage Copy Titles

12.1.1 Define the Copy Titles

Important: Copy Titles must be defined on *Connective Server level*. Contact your system administrator to make sure you have access to the Connective server level.

To define Copy Titles:

- Open the installation folder of the Connective Server.
- Right-click the **configuration.xml** file and click **Edit**.
- In the configuration.xml file the list of Copy Titles for a specific ModelStore will be defined. Every Copy Title has a language-dependent description.

Below you find an example of how Copy Titles may look like:

Example:

```
<CopyTitles>
<CopyTitle Code="kopie cliënt">
<Description Code="nl">kopie cliënt</Description>
<Description Code="fr">copie client</Description>
<Description Code="en">copy client</Description>
<Description Code="de">Kopie Kunde</Description>
</CopyTitle>
<CopyTitle Code="kopie kantoor">
<Description Code="nl">kopie kantoor</Description>
<Description Code="fr">copie agence</Description>
<Description Code="en">copy office</Description>
<Description Code="de">Kopie Zweigstelle</Description>
</CopyTitle>
<CopyTitle Code="kopie dossier">
<Description Code="nl">kopie dossier</Description>
<Description Code="fr">copie dossier</Description>
<Description Code="en">copy file</Description>
<Description Code="de">Kopie Dossier</Description>
</CopyTitle>
</CopyTitles>
```

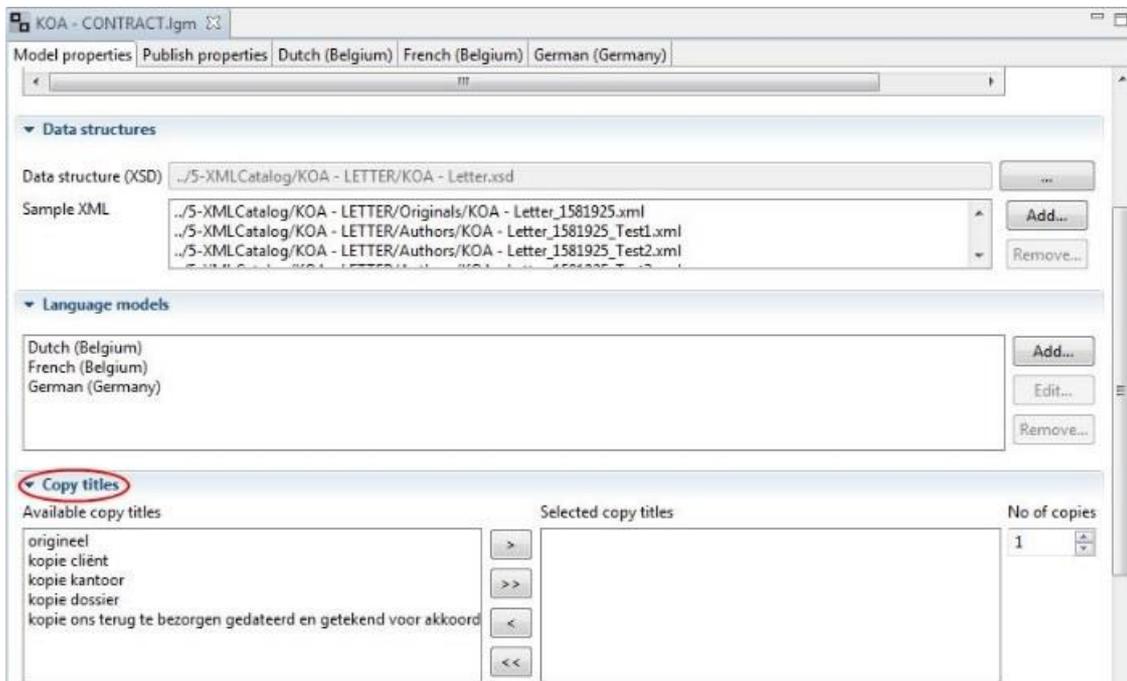
12.1.2. Assign Copy Titles to a logical model

Once the Copy Titles have been defined on server level, (a selection of) those Copy Titles have to be added to the logical model.

To add Copy Titles to a logical model:

- Check out the logical model.
- In the **Model Properties** tab under **Copy Titles** the available copy titles that have been defined in the configuration.xml are listed.

Note: if no Copy Titles are listed here, this means they have not been correctly defined in the .xml file.



To add a **specific Copy Title** to your logical model:

- Select the copy title under **Available Copy Titles**.
- Click the > button.

To add **multiple Copy Titles** to your logical model:

- Select the Copy Titles under **Available copy titles**.
- Click the >> button.

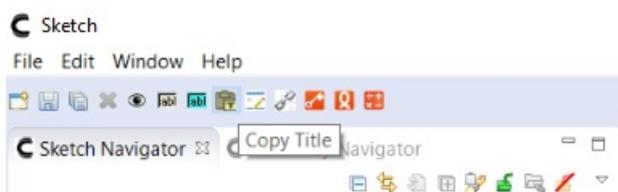
The selected Copy Titles will now be available for the logical model and all associated language models.

12.1.3. Insert the Copy Title variable into the language model

Once the Copy Titles have been added to the logical model, a Copy Title variable needs to be inserted in the document, on the location where you want the Copy Title-description to appear after generation.

To insert a Copy Title variable into the language model:

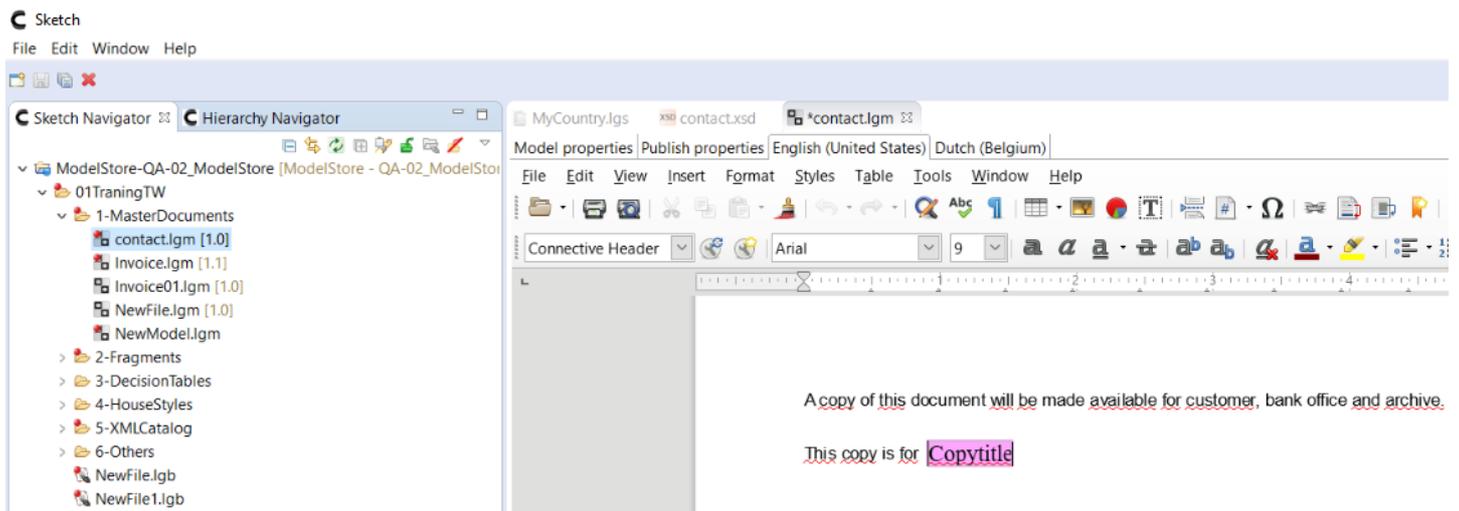
- Make sure the logical model is checked out.
- Open the language model.
- Place the cursor where you want the Copy Title variable to be located.
- Click the **Copy Title** icon in the toolbar.



A dialog box appears where you can select the formatting of the Copy Title variable.

- Select the formatting.
- Click **OK**.
- The **Copy Title variable** is inserted in your dynamic document.

- A Copy Title variable is represented as a **purple frame**.



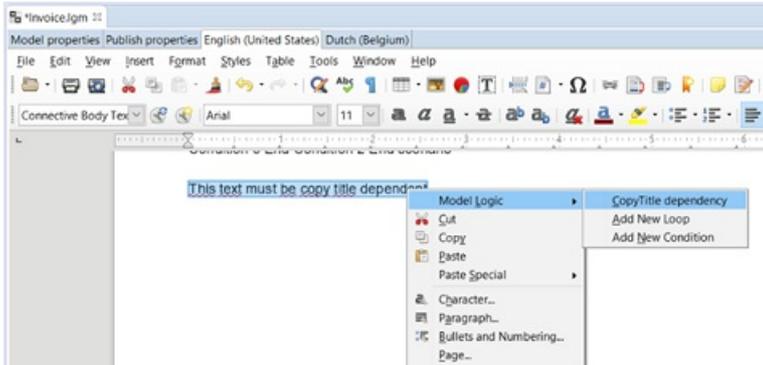
- After **preview**, the Copy Title variable is still displayed as a purple frame.
- After **generation**, the Copy Title variable will be replaced by the (language-dependent) description, as defined in the configuration.xml on server level.

12.2 How to insert a Copy Title-dependent text block

Apart from the Copy Title description (defined in the configuration.xml on Connective server level), you can add Copy Title-dependent text blocks to your dynamic document. This text block will then only be printed in a specific copy version.

To insert Copy Title-dependent text:

- Make sure the logical model is checked out.
- Open the language model.
- Select the text you want to make copy title-dependent.
- Right-click and click **Model Logic** > **CopyTitle dependency**.



- The **Copy Title dependency** wizard is displayed.
 - Select the copy titles for which the text block will be shown.
 - Add a description in the **Description** text box.
 - Click **OK**.

The Copy Title dependencies will be executed after the generation. In other words: first the variables and the logic will be generated, second the Copy Title dependencies will be handled.

13. Preview your dynamic document

When you create a dynamic document by adding logic to the template, it can be very useful to preview your document from time to time. This way, you can check the result and make modifications if needed.

Before you can do a preview of your document, make sure one or more sample data files have been added to the Model Properties tab of your logical model. For more information, see [3.2.1 Create a logical model](#).

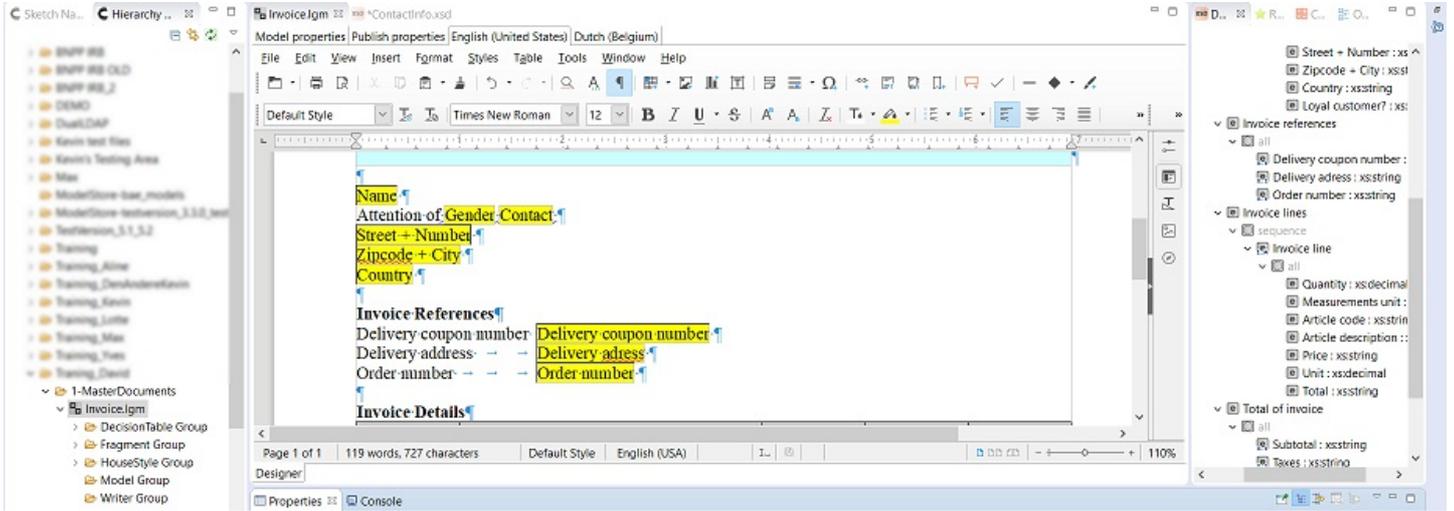
The **Page Preview** feature found in the **File** menu (or the **Page Preview** icon shown in the toolbar) in Design View is not the same feature as the one explained here. That feature displays how the unprocessed language model would print in Design View.

13.1 How to preview a language model

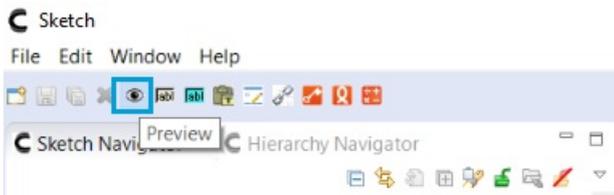
13.1.1. Preview when 1 XML is linked

To preview a language model when only 1 sample data file is linked to the logical model:

- Open a language model.



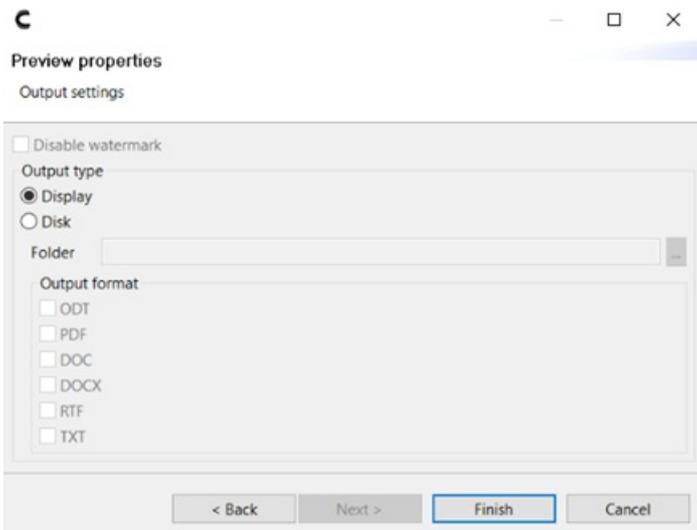
- Click the preview icon in the toolbar.



- If the logical model is checked out and has been modified, you are prompted to save the document before preview. Click **Yes** to save the model locally.
- Click **Next**.
- Keep **Display** selected to preview on screen.

Or

- Select **Disk** to save the preview to disk.
 - Click the browse button to select the folder in which the preview must be saved.
 - Select the output format of the preview.



- Click **Finish**.

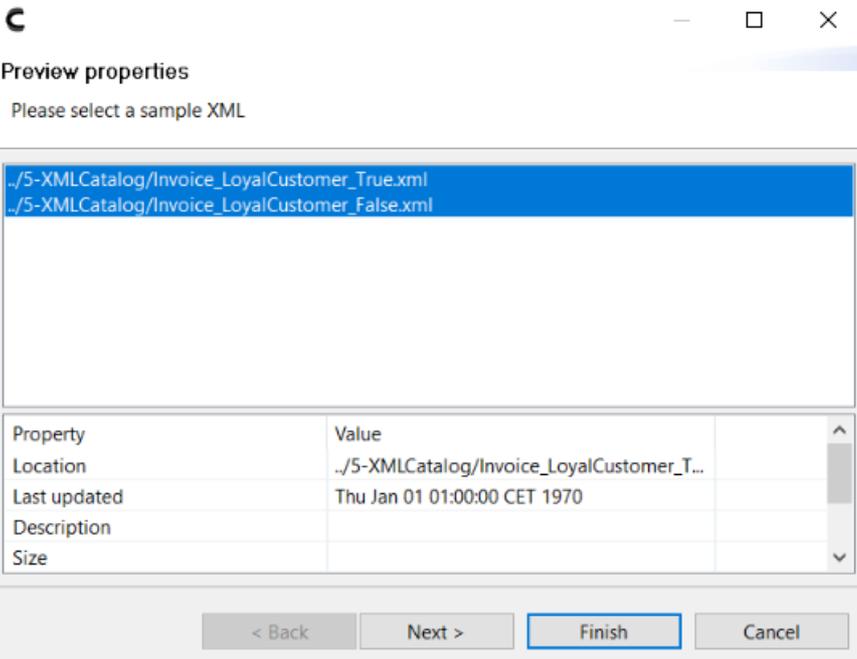
The preview is shown in read-only mode. No editing can be done.



13.1.2. Preview when multiple XMLs are linked

To preview a language model when multiple sample data files are linked to the logical model:

- Open a language model.
- Click the preview icon in the toolbar.
- If the logical model is checked out and has been modified, you are prompted to save the document before preview. Click Yes to save the model locally.
- In the **Preview properties** screen select one or more sample XMLs and click **Next**.



- Keep **Display** selected to preview on screen.

Or

- Select **Disk** to save the preview to disk.
 - Click the browse button to select the folder in which the preview must be saved.
 - Select the output format of the preview.
- Click **Finish**. The preview is shown in read-only mode. No editing can be done.

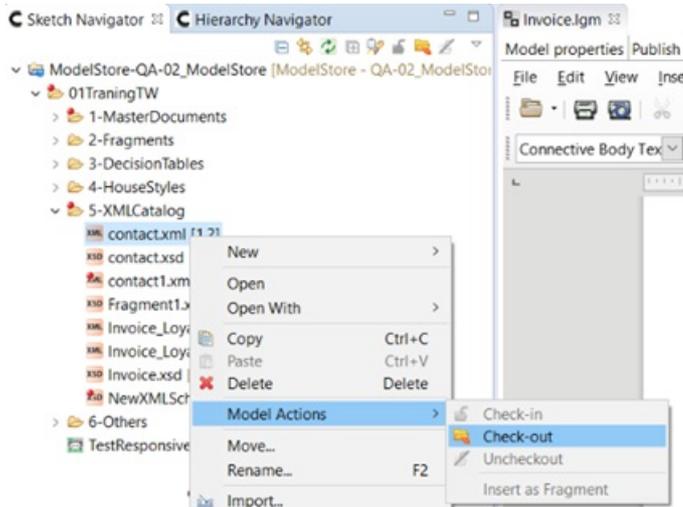
13.2 How to modify the sample data

In a dynamic document there will be conditional content, loops will be created, variables inserted etc. From time to time, you might want to check if the logic you added is correct for all different scenarios.

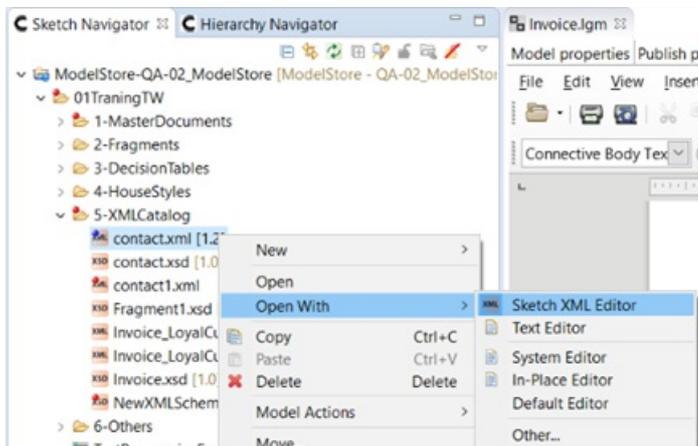
You can easily modify the sample data which is attached to your logical model in order to check these different scenarios.

To modify the sample data:

- Select the XML file in the Sketch Navigator.
- Right-click and click **Model Actions > Check-out**.



- Now right-click the checked out XML file in the Sketch Navigator.
- Click **Open with > Sketch XML Editor**.



- Click the **Design** tab.
- Click on the nodes to expand/collapse the data structure.
- Edit the value according to your expected scenario.
- Save the sample data.

Node	Content
?-? xml	version="1.0" encoding="UTF-8" standalone="no"
⌵	
⌵ tns:ContactInfo	all(LastName, FirstName, StreetName, HouseNumb
Ⓞ xml:ns:tns	http://www.example.org/contact
Ⓞ xml:ns:xsi	http://www.w3.org/2001/XMLSchema-instance
Ⓞ xsi:schemaLocation	http://www.example.org/contact modelstore://qa-l
Ⓞ tns:LastName	
Ⓞ tns:FirstName	
Ⓞ tns:StreetName	Wapenstraat
Ⓞ tns:HouseNumber	14
Ⓞ tns:ZIPCode	2000
Ⓞ tns:Area	Antwerpen
Ⓞ tns:Country	België

Make sure to enter the correct data format depending on the type of variable.

14. Import and export

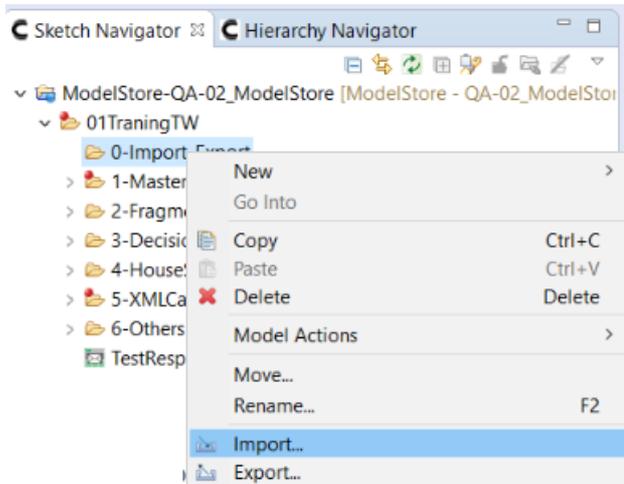
14.1 Import

14.2 Export

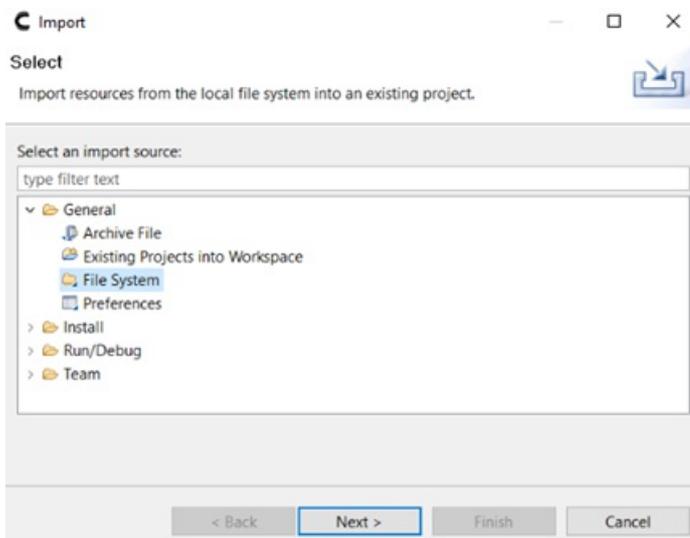
14.1 Import

To import one or more files, or a whole file structure:

- Select the folder where you want to do the import.
- Right-click and click **Import**.



- The **Import** wizard is opened.
- Select **General > File System**.
- Click **Next**.
- Browse and select the files and/or folders you want to import.
- Click **Finish**.



The imported folder/files are now visible in the Repository and are checked out.

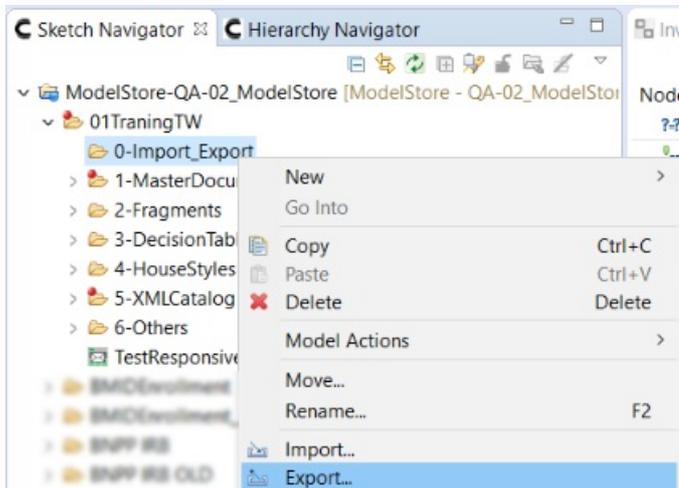
Instead of using the **Import** wizard, files can also be copy-pasted directly from a Windows Explorer window to the Sketch ModelStore.

14.2 Export

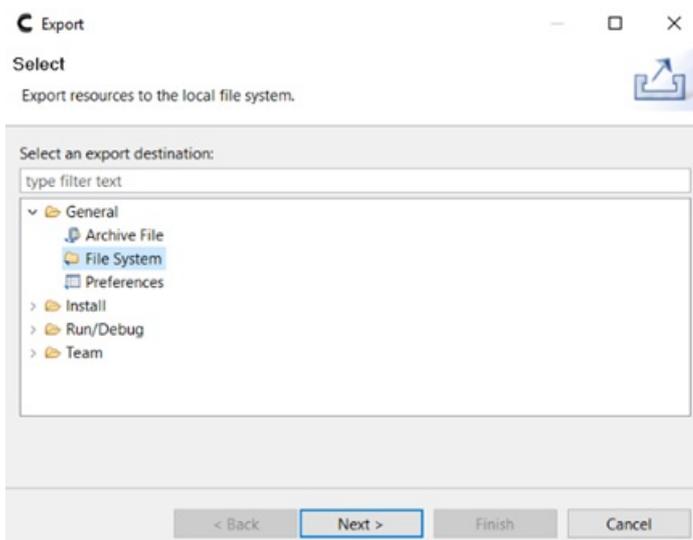
An easy way to make a backup in Sketch, is doing an export. All files in the ModelStore can be exported: logical models, sample XMLs, XSD files, house styles, decision tables etc.

To do an export:

- Select the file or folder you want to export.
- Right-click and click **Export**.



- The **Export** wizard is opened.
- Click **General > File System** (or **Archive File** if you want to back up a large folder)
- Click **Next**.
- Select the files and/or folders you want to export.
- Select the directory to which the files must be exported.
- Click **Finish**.



15. Using variables in charts

With Sketch you can use variables in all kinds of ways. You can even create charts using the available xml data.

To do this, create a chart as you would do it in a LibreOffice Writer document, but use the variables from the data structure instead of typing the labels and the data in the cells.

First, create a table that will contain the labels and the data. The general idea is to place the labels in the top row and in the leftmost column, while filling the other cells with the data. You might want to try this with some figures first to get used to the chart feature.

Example of column chart and Example of X-Y diagram

	North	Center	South
Product 1	345	165	625
Product 2	243	255	437
Product 3	412	279	846

Quantity	Result
10	234
20	795
30	2093

Instead of typing the labels and the figures, insert them using the variables of the data structure as explained in [7.1 Data](#).

Example of column chart and Example of X-Y diagram

	VarLabelX	VarLabelY	VarLabelZ
VarLabel1	VarX1	VarY1	VarZ1
VarLabel2	VarX2	VarY2	VarZ2
VarLabel3	VarX3	VarY3	VarZ3

VarLabelA	VarLabelB
VarA1	VarB1
VarA2	VarB2
VarA3	VarB3

When the placement of the variables is done, you can create the chart:

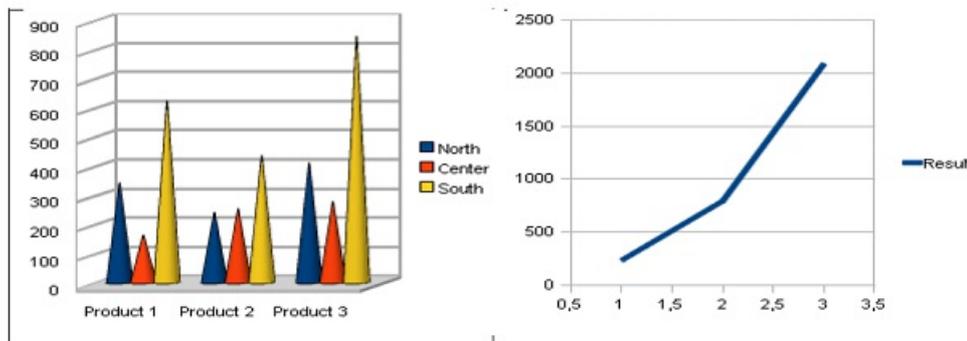
Select the part of the table you want to use in the chart. Most of the times you will want to exclude the rows containing the totals when available.

- Click **Object > Chart ...** from the **Insert** menu.
- Use the wizard to define the chart type and other settings.

You can choose from Column, Bar, Pie, Doughnut, Area, Line, X-Y (Scatter), Bubble, Net Stock, as well as Column and Line combination charts. Every part of the chart can be configured specifically.

As the chart figures depend on XML data, you will only see the result when previewing or generating the document.

Example of a 3D column chart - example of an X-Y diagram



When you are done, you can change the place and layout settings of the chart. If you do not want the table to appear in the generated document, select the table and hide it using the 'hidden' character property.

Some things however are important to succeed in creating a chart:

- The thousand separator and decimal separator should be in line with the locale setting language of LibreOffice Writer.

- Do not hard code the thousand separator in the data. Use **Table > Number Format...** instead.
-

16. Writer

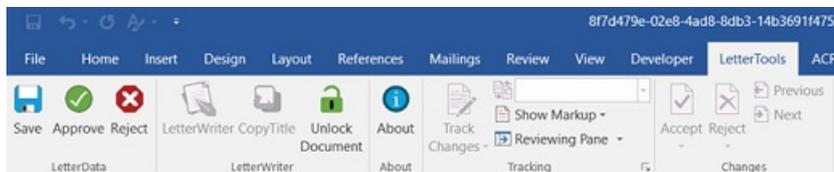
Writer is a separate application, independent of Sketch. It is installed as a Microsoft Word Add-in and allows you to modify dynamic documents created in Sketch.

To create a dynamic document that is compatible with writer, you need to create a Writer Model (.lgw) in Sketch. A Writer Model may contain several TextBlock Models (.lgb). TextBlock Models are to Writer Models what Fragments are to Logical Models, i.e. reusable building blocks.

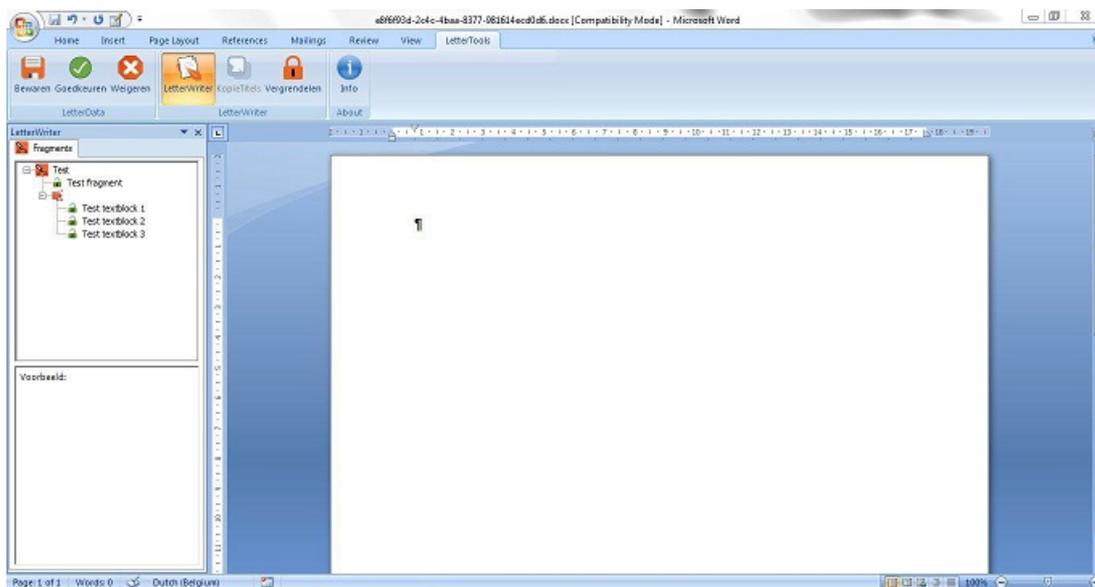
Once a Writer Model has been created, it must be published, and then generated in a Writer flow (for instance through Data Manager). Once this is done, an .lde document is created. This .lde document can be opened and modified in Word via the Writer Add-in.

To open a document in Writer:

- Double-click the .lde document you created. It is by default opened in Word.
- When the Writer is correctly installed, the **LetterTools** tab is available in top toolbar in Word.
- Click the **LetterTools** tab to display the Writer functionalities.



- If your documents contains TextBlocks Models (created in Sketch), they are displayed in the pane on the left.



- To start working on your document, click **Unlock Document**.
- You can now add or remove predefined TextBlocks, edit free text like you would do in a regular Word document. The advantage of Writer is that your document never leaves its document flow. Once you have made your modifications, you can send the document to its next processing step, for instance approval by your manager.

For detailed information about how to actually use the Writer, consult the separate **Writer User Guide**.

In the next sections we will explain the following actions:

- [Create TextBlock Models](#)
- [Create Writer Models](#)
- [Link TextBlock Models to a Writer Model](#)
- [Publish the Writer Model](#)

16.1 Sketch TextBlock Models

TextBlock models are to Writer Models what Fragments are to Logical Models: they are reusable building blocks.

Like regular Fragments, TextBlocks are created in the Sketch Model Editor and use the word processing functionalities of LibreOffice Writer. A TextBlock document can contain one or more paragraphs, sentences or even characters, tables, logos, signatures or any other element.

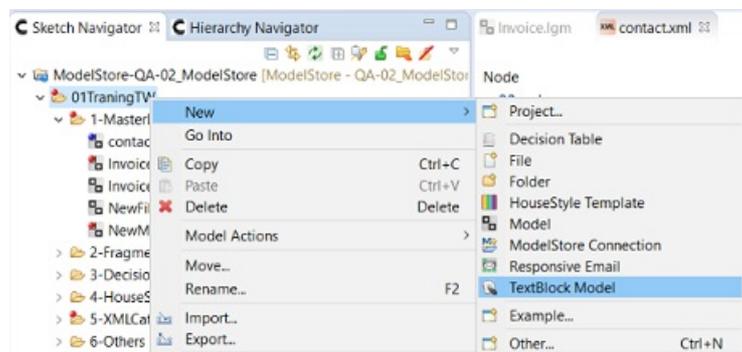
Once a TextBlock Model is created it can be used inside a Writer Model.

16.1.1. Creating a TextBlock Model

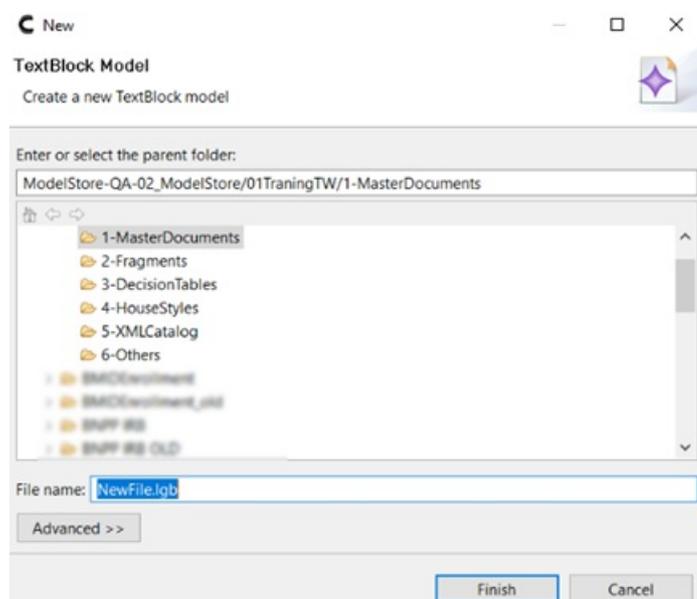
Like a logical model, a TextBlock model consists of one or more language documents. These language documents known as Language text blocks, are the physical documents in a specific language.

To create a new TextBlock model:

- Right-click in the Sketch Navigator and click **New > TextBlock Model**.



- Enter a file name. Make sure to keep the extension .lgb.
- Click **Finish**.

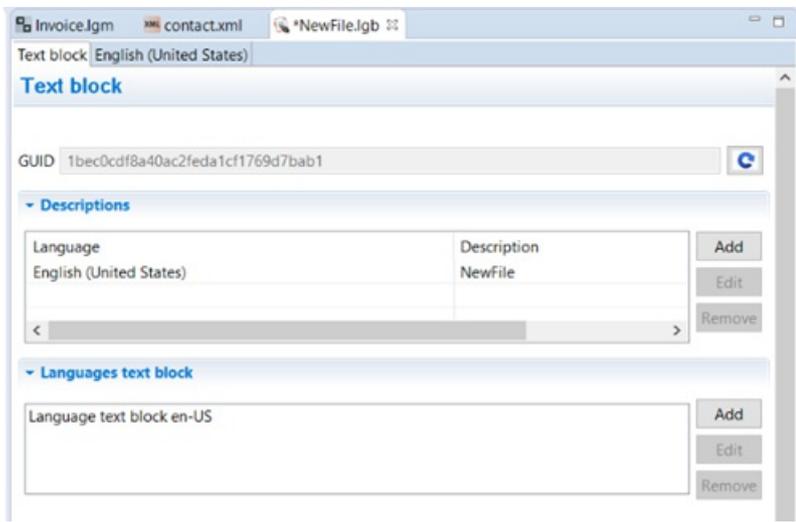


To add a language:

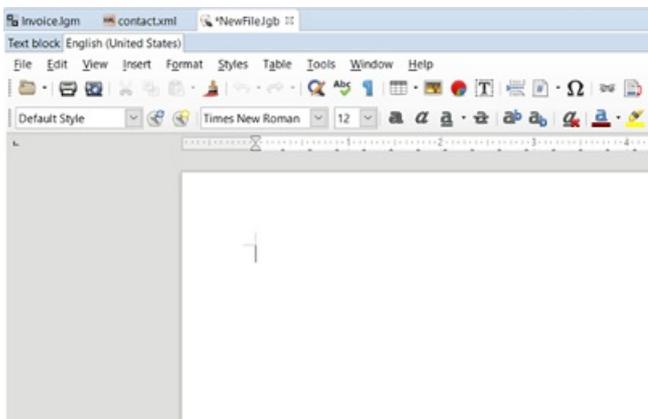
- Under **Descriptions**, click **Add to add a language**.
- Select the required language from the list.
- Enter a description.
- Click **OK**.

To create a language text block:

- Click **Add** under **Languages text block**.
- Select the required language from the **Language** list. **Note:** the **Source** list is only available if another language text block was already created.
- You can choose to add a [House style](#) (optional).
- Click **OK**.



- The language text block is checked out for modifications and is opened in the Editor.
- In the Editor you can add the content of your document. All functionalities of LibreOffice are supported.



Once you have defined the content, save and check in the TextBlock Model.

16.1.2. Saving a TextBlock Model

Make sure to save your changes on a regular basis. When you save, the save will only be done locally and the document will not be checked in. So the latest version will still not be available for other users.

To save locally, there are 3 options:

- Click the **Save** icon in the toolbar.



- Click **File > Save**.
- Press **[Ctrl+S]** on the keyboard.

After saving the model it will remain open in the Editor region, and you can continue your modifications.

16.1.3. Checking in a TextBlock Model

- Select the TextBlock Model.
- Click the **Check in** icon in the toolbar.

Or

- Select the TextBlock Model.
- Right-click and click **Model Actions > Check in**.
- Enter a **Comment** (optional).
- Click **OK**.

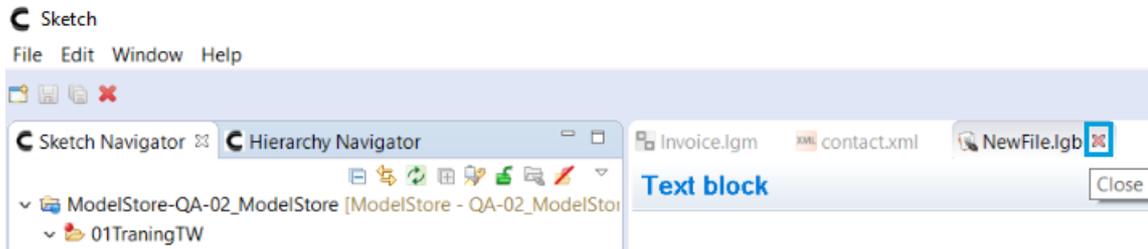
When the TextBlock Model is checked in it can be linked to a Writer Model.

16.1.4. Closing a TextBlock Model

Whenever you saved or checked in an open TextBlock Model, it will stay visible in the Editor region. This is to let you continue your work. If you didn't make any changes you can select Uncheckout on an open model and it will close in the Editor region.

To close a TextBlock Model, there are 3 options:

- Click the **Close** icon in the toolbar.



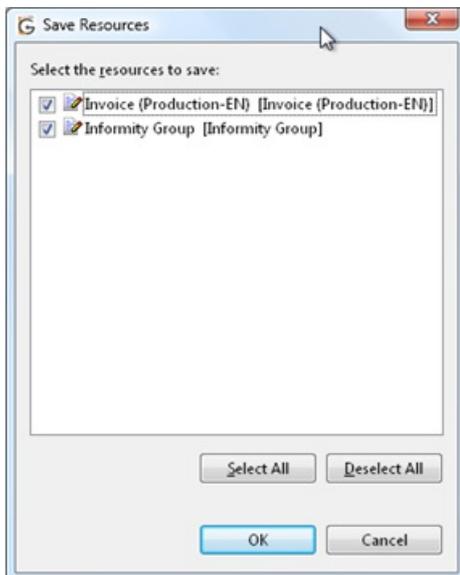
- Right-click the opened model tab and click **Close**.
- Press [**Ctrl+F4**] on the keyboard.

If the TextBlock Model has been modified, but hasn't been saved locally yet, you will be prompted to save it.

Just like models, house style templates and decision tables share the same title bar. The following tips applies to all of them:

- You can close all of them at once: right-click in the document title bar and select **Close All**.
- If you want to close all but one, activate the document to keep and select **Close Others**.

If more than one document was modified but not saved, you get the following dialog box:



You are then prompted to select the documents you want to save before closing them.

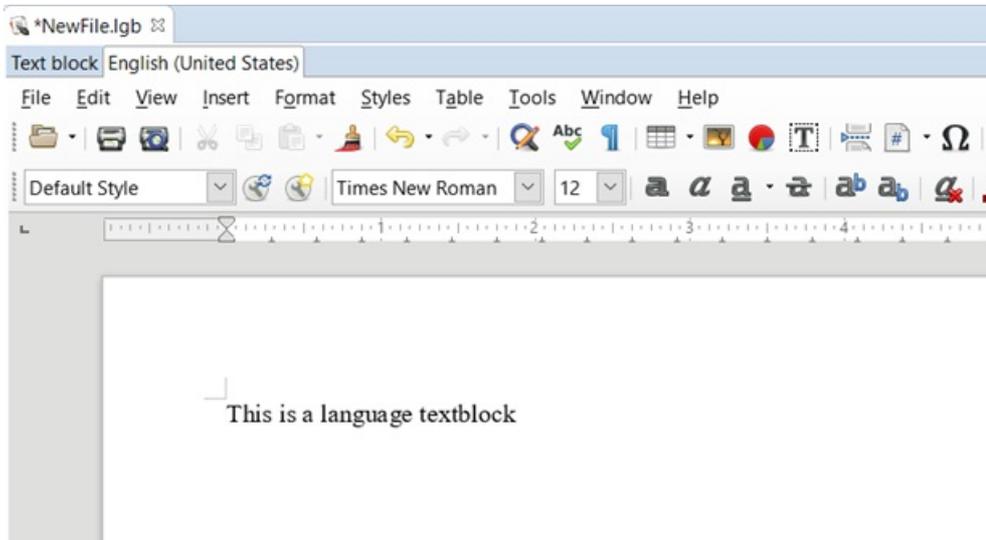
16.1.5. Viewing a text block model

A TextBlock Model can be viewed without checking it out from the ModelStore server. The TextBlock Model and its language models are 'read-only' and cannot be modified.

To view a TextBlock Model:

- Right-click on the text block model in the Sketch Navigator and click **Open**. You can also double-click on the model.
- Open the required language model.

A language text block is by default opened in read-only mode. No changes can be made to the model. You can however perform some tasks like print, modify or preview.

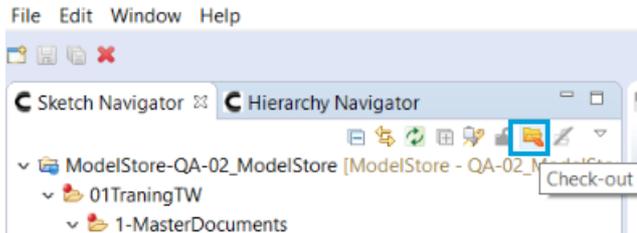


16.1.6. Modifying a Language text block

To modify a language text block the author has to check out the TextBlock Model. By doing so he claims the sole authority to change the content and the formatting in the language text block.

To check out a TextBlock Model:

- Select the text block model.
- Click the Check out icon in the toolbar.



Or

- Select the text block model.
- Right-click and click **Model Actions > Check out**.

Another way to check out the text block model:

- Select the text block model.
- Right-click and click **Open**.

Or

- Double-click on the text block model.

16.1.7. Opening a TextBlock Model

If you closed a TextBlock Model without checking it in you can open it next time you want to resume your work.

There are two ways to open a TextBlock Model:

- In the Sketch Navigator right-click the required TextBlock Model and click Open.

Or

- Double-click the required TextBlock Model in the Sketch Navigator.

16.1.8. Applying Modifications to a TextBlock Model

All modifications will be saved and the TextBlock Model will be checked in. The latest version becomes available again for other users.

To check in a TextBlock Model:

- Select the TextBlock Model.
- Click the **Check in** icon in the toolbar.

Or

- Select the TextBlock Model.
- Right-click and click **Model Actions > Check in**.

After the TextBlock Model has been checked in, it will remain viewable in the Editor region. This state is the same as what you get with View Model. (See 16.1.2 Viewing a Language text block above for more information).

16.1.9. Canceling Modifications to a TextBlock Model

The check-out will be undone and all changes made to the language text block will be canceled. Everything will be restored to the previous version of the document.

To cancel the modifications to a TextBlock Model:

- Select the TextBlock Model.
- Click the **Uncheckout** icon in the toolbar.
- Click **Yes** to confirm.

Or

- Select the text block.
- Right-click and click **Model Actions > Uncheckout**.
- Click **Yes** to confirm.

16.1.10. Deleting a TextBlock Model

This action deletes a TextBlock Model.

To delete a TextBlock Model:

- Select the TextBlock Model.
- Click the **Delete** icon in the toolbar.

Or

- Select the TextBlock Model.
- Right-click and click **Delete**.

Or

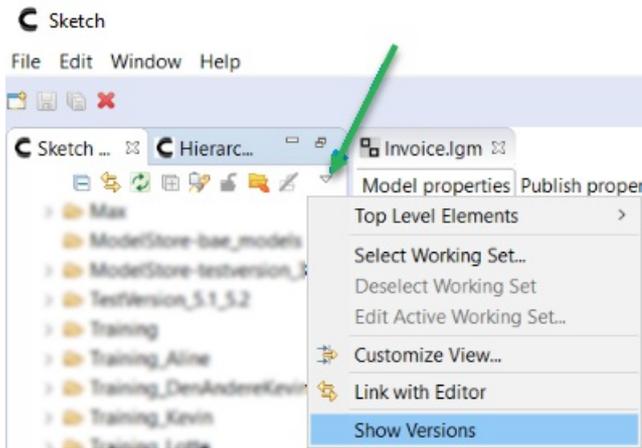
- Select the TextBlock Model.
- Press **Delete** on the keyboard
- Click **Yes** or **Yes To All** to confirm.

16.1.11. Restoring a TextBlock Model

The author can return to a previous version of a TextBlock Model (or Decision table, house style template etc). The Navigator/ModelStore region is like a versioned file system: you have directories and files inside those directories. Files are versioned in a linear way. On every check-in, the version of the file is increased.

To restore a TextBlock Model:

- Click the triangle icon to open the **View** menu.
- Click **Show Versions**.



- Select the TextBlock Model you want to restore.
- Click the **triangle** in front of the model name.
- The list of different versions is opened.
- Right-click the version you want to restore and click **Restore**.

16.2 How to create a Writer Model

In order to generate a document that can be opened in the Writer application, you need to create a Writer Model (.lgw). Once you've created a Writer Model you can link TextBlock models and their embedded language models to the Writer model. Then, the Writer Model must be linked to a Logical Model.

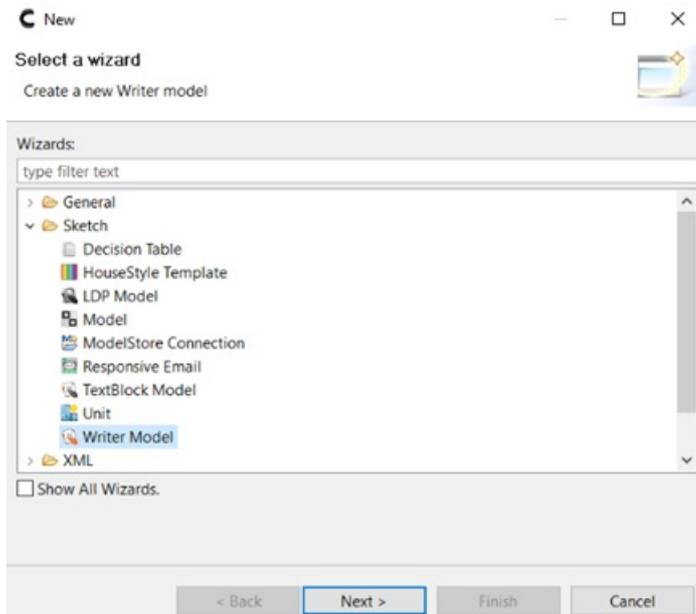
16.2.1. Creating a Writer Model

A Writer model may consist of one or more TextBlock Models. As we've explained in the previous section, TextBlock Models are like Fragments; they are reusable building blocks inside a Writer Model.

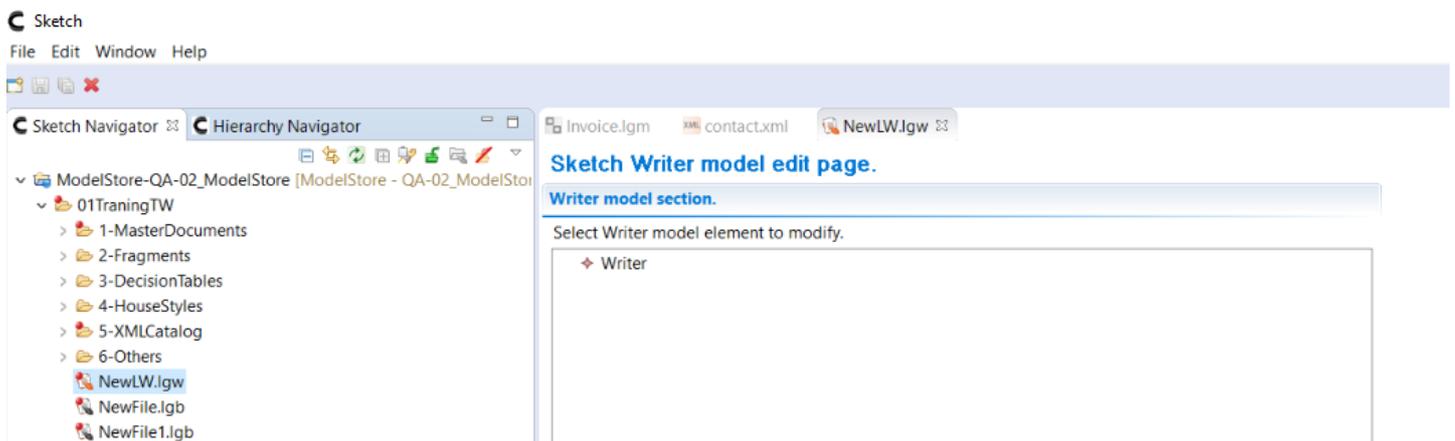
Different TextBlock Models can be linked in a structured way by making a **Group**.

To create a new Writer model:

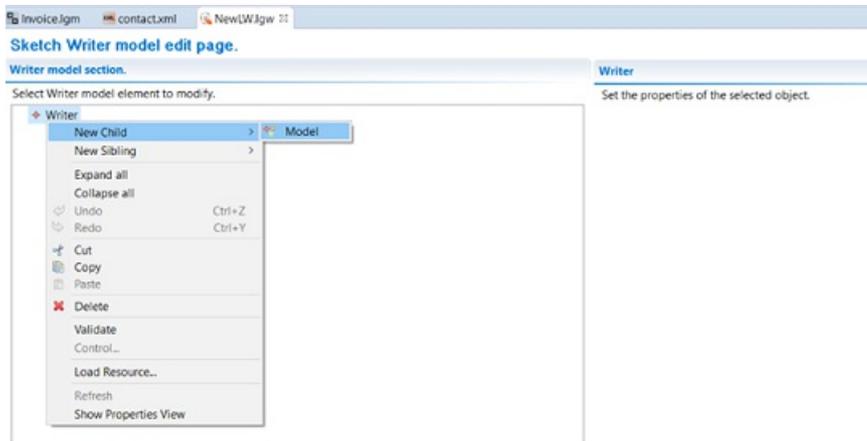
- Right-click in the Sketch Navigator and click **New > Other**.
- In the Wizard, click **Sketch > Writer Model**.



- Click **Next**.
- Enter a name for the Writer Model. Make sure to keep the .lgw extension.
- Click **Finish**. A new Writer Model is checked out and displayed in the Editor region:

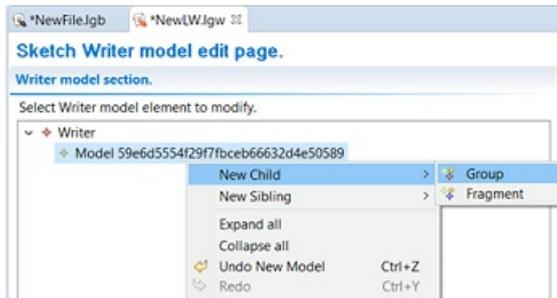


- Right-click in the Editor region and click **New Child > Model**:



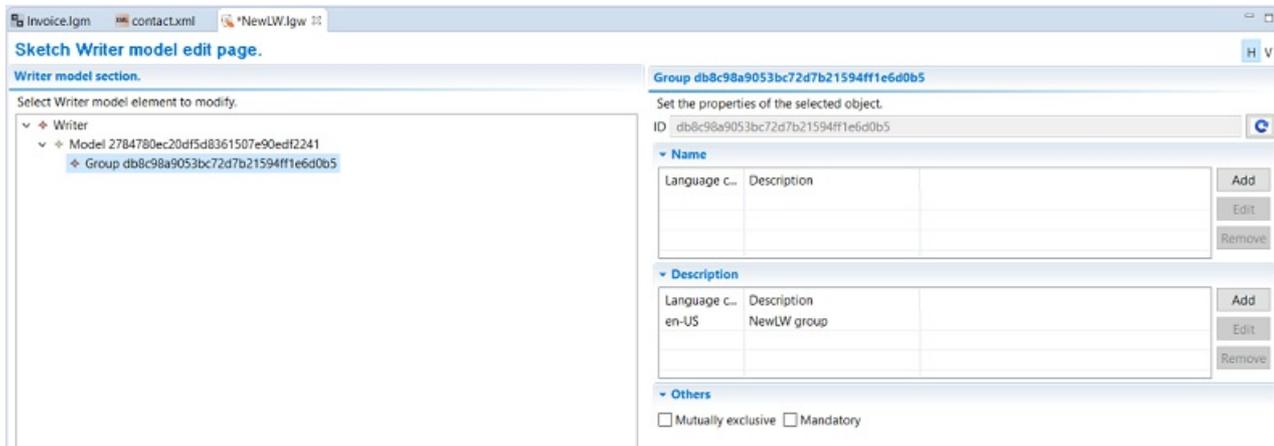
To add a Group for structure:

- Expand the root element.
- Right-click the Model node and click **New Child > Group**.



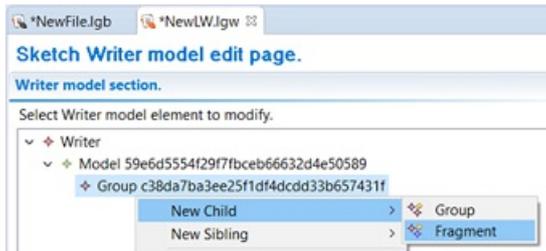
Now you can fill in the appropriate fields in the Group section:

- Under **Name**, click **Add**.
- Select the language.
- Add a **Description**.
- Click **OK**.
- Under **Description**, click **Add**.
- Select the language.
- Add a **Description**.
- Click **OK**.
- Check the **Mutually exclusive** check box if you want only one of the TextBlocks belonging to the group to exist at a time in the document.
- Check the **Mandatory** check box if one of the TextBlocks belonging to the group must be added to the document.



16.2.2. Linking a TextBlock Model to a Writer Model:

- Make sure you followed the steps in 16.2.1 Creating a Writer Model above.
- Right-click the Group node in the Editor region and click **New Child > Fragment**.

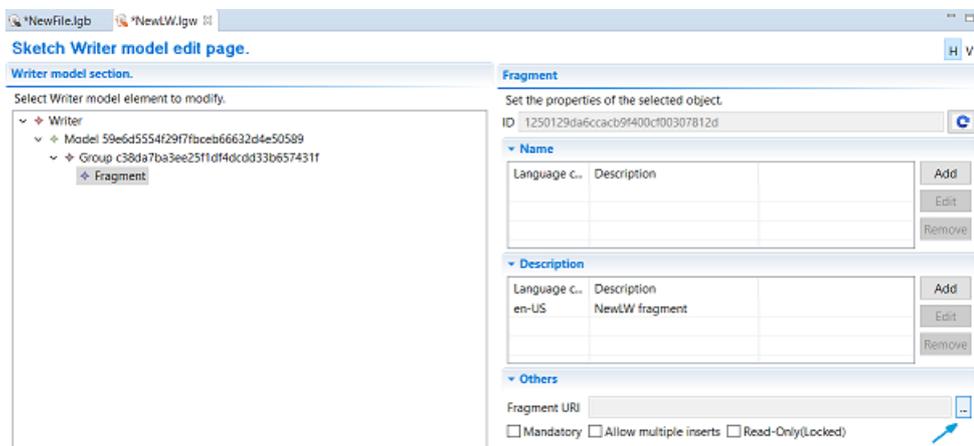


Now you can fill in the appropriate fields in the **Fragment** section:

- Under **Name**, click **Add**.
- Select the language.
- Add a description.
- Click **OK**.
- Under **Description**, click **Add**.
- Select the language.
- Add a description.
- Click **OK**.

To link the fragment, to the TextBlock Model you created:

- Click on the browse button (...) next to **Fragment URI** button under **Others**.
- Browse for the .lgb file you created earlier.



- Click **OK**. The file is now linked.
- Check the **Mandatory** check box if one of the TextBlocks belonging to the group must be added to the document.
- Check the **Allow multiple inserts** check box if more than one occurrence of the TextBlock may exist in the document.
- Check the **Read-Only(Locked)** check box to prevent the text block fragment from being modified in the document.

To add multiple fragments, repeat the steps above.

16.2.3. Linking a Writer Model to a Logical Model

In order to use a Writer Model, you must link the Writer model (.lgw) to a Logical Model.

To link a Writer model to a Logical Model:

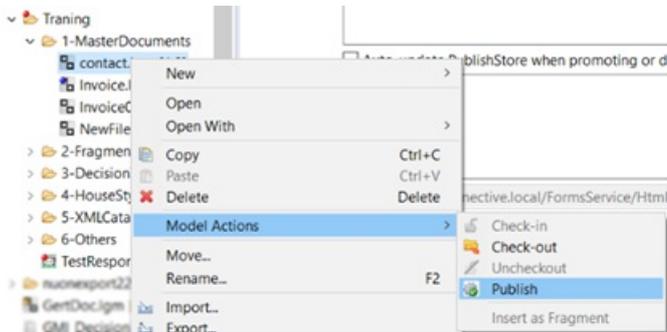
- Check out the logical model.
- Open the **Model properties** tab.
- Under **Other resources > Writer**, click **Add**.
- Select the required .lgw file, and click **OK**. The Writer file is now attached.

16.2.4. Publishing a Logical Model with a Writer Model

Before you can use Writer in a generated Word document, at least one Writer model must be published.

To define the publish properties:

- [Check out](#) the logical model.
- Determine the **Publish properties**. For more information, see [3.9 How to publish a logical model](#).
- Check the **Letterform URI** check box to generate and publish the corresponding web form.
- [Check in](#) the logical model.
- Right-click the logical model in the Sketch Navigator, and click **Model Actions > Publish**.



- Select a Unit/Category.
- Select the option **Writer files need to be republished** if necessary.
- Select the option **Text blocks need to be republished** to have modifications republished.
- Click **Finish**.
- Add a comment (Optional).
- Click **OK**.

You can verify when a model has last been published in the PublishStore. To do this, select one of the files in the published folder and check the date in the last modified field of the Properties tab at the bottom part of the Editor region.



Property	Value
last modified	July 14, 2017 at 2:20:44 PM
Last modified by	qauser002
Last modified comment	

Once the model has been published it can be generated in a Writer flow (for instance through Data Manager). Once this is done, an .Ide document is created. This .Ide document can be opened and modified in Word via the Writer Add-in.

17. Responsive Email

In Sketch you can create responsive emails. A responsive email is an email that adapts its content to the size of the screen on which it is displayed.

Responsive emails in Sketch are created in a similar way as Logical Models. This means you can also use the Sketch-specific building blocks, logic, conditions, etc. inside an email to make it dynamic.

[17.1 How to create a responsive email](#)

[17.2 How to publish a responsive email model](#)

17.1 How to create a responsive email

When you want to create a new responsive email, you first have to decide where you want the email model to be placed. This is very important. Once an email model is created in a specific folder in the Sketch Navigator, links will be established to other elements that are associated with that model (for example an .xsd data structure, decision tables, fragments, etc.). It is recommended not to change the location of models. Moving the model impacts the links that point towards the model.

[17.1.1 Create a responsive email](#)

[17.1.2 Create the content of a responsive email](#)

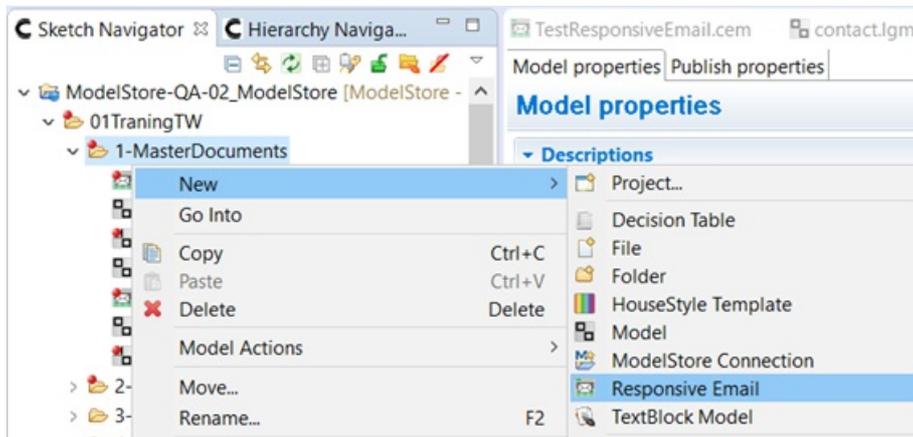
[17.1.3 Examples of variables](#)

17.1.1 Create a responsive email

When you want to create a new responsive email, you first have to decide where you want the email model to be placed. This is very important. Once an email model is created in a specific folder in the Sketch Navigator, links will be established to other elements that are associated with that model (for example an .xsd data structure, decision tables, fragments, etc.). It is recommended not to change the location of models. Moving the model impacts the links that point towards the model.

To create a new responsive email:

- Select the folder where you want the responsive email to be located.
- Right-click and click **New > Responsive Email**.



- The **Responsive Email** wizard opens.
- The folder you selected is by default indicated as **parent folder**. To change the location of the email model, click on the nodes to expand/collapse the different folders and then select the folder where you want to save the email model.
- Enter the email name in the File name text box. An email model needs to have the extension .cem.
- Click **Finish**.

Your new email model is created in the selected folder. The model is checked out and opened in the Editor region.

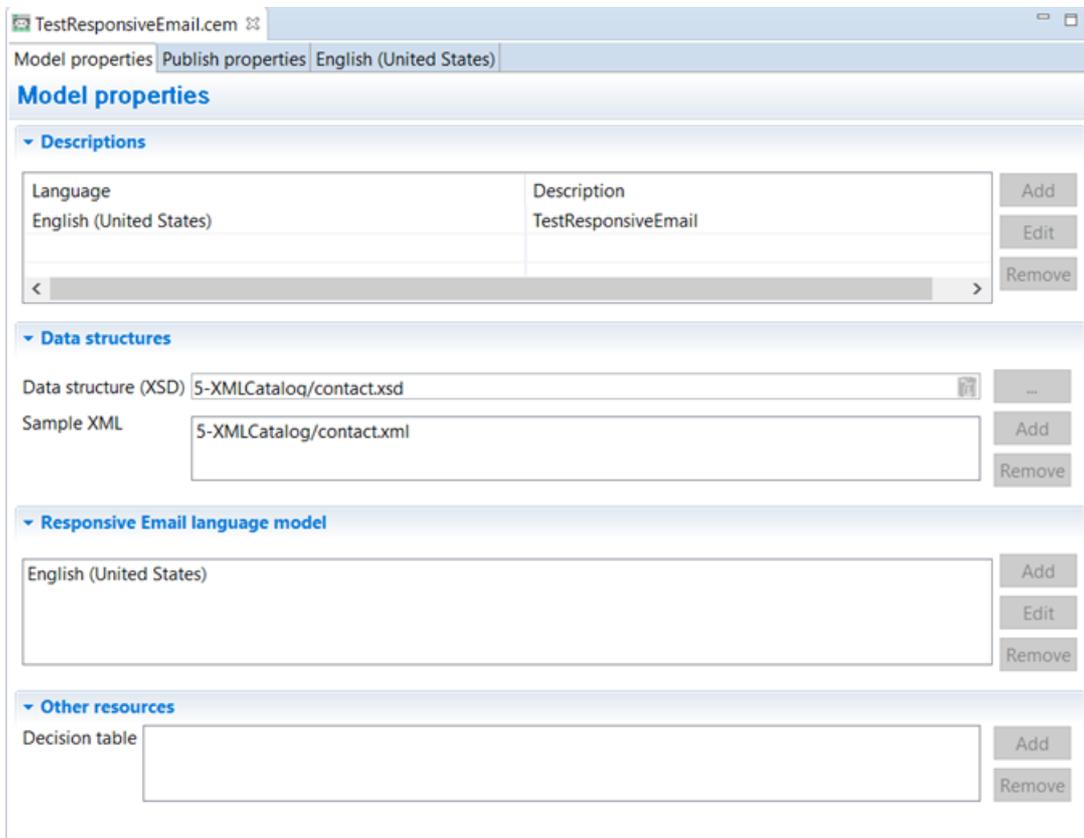
Now you can add the **Model Properties**.

Model Properties

Just like any other Sketch Model, the **Model Properties** must be configured in order to create the responsive email model.

To add a Description:

- Under **Descriptions**, click **Add**.
- Select the **Language**.
- Add a **Description**.
- Click **OK**.



XSD file

To add an XSD file:

- Under **Data Structures**, click the browse button (...)
- Browse and select the **XSD file** you want to associate with the logical model.
- Click **OK**.

XML file

To add a sample XML:

- Under **Data Structures**, click **Add**.
- Browse and select one or more **XML file(s)** you want to add to the email model.
- Click **OK**.
- If the XML is not compatible with the XSD, an error message will appear.

Responsive Email Language model

An email model can contain several language models. A language model represents the physical document in a specific language.

Contrary to logical models and fragments - which are opened in the Libre Office Text Editor - responsive email models are opened in an Internet Explorer environment. This is to edit the HTML-based structure of these models. As a result, the communication between Internet Explorer and Sketch is limited to JavaScript features.

Create a new language model from scratch

- Under **Responsive Email language model**, click **Add...**
- Select the required language from the **Language** list.
- Add a **description** (optional).
- Click **OK**.

The new language model is opened in a new tab and you can start working in the language model. To learn how to do so, see

17.1.2 Create the content of a responsive email.

Create a new language model based on a source language

If you already created a dynamic document in 1 language, you can create a new language model based on this language. The advantage is that you don't have to rebuild all the logic a second time. You can start working with the exact copy of the source language, so all you have to do is translate the fixed text in order to get a working dynamic document in a second language.

To add a language model based on a source language:

- Check out and open the logical model.
- Under **Responsive Email language model**, click **Add**.
- Select the required language from the **Language** list.
- Select the **Source** language. **Note:** the **Source** language list is only available if another language has already been selected.
- Add a **description** (optional).
- Click **OK**.

The new language model is opened in a new tab and the content is a copy of the source language. To learn how to create the content, see [17.1.2 Create the content of a responsive email](#).



Decision table

To add a Decision table (optional):

- Under **Other resources > Decision table**, click **Add**.
- Browse and select one or more **Decision table(s)** you want to add to the email model.
- Click **OK**.

Publish Properties

Just like any other Sketch Model, the **Publish Properties** must be configured before you can publish the responsive email model.

See [17.2 How to publish a responsive email](#) for more information.

17.1.2 Create the content of a responsive email

To create the content of a responsive email:

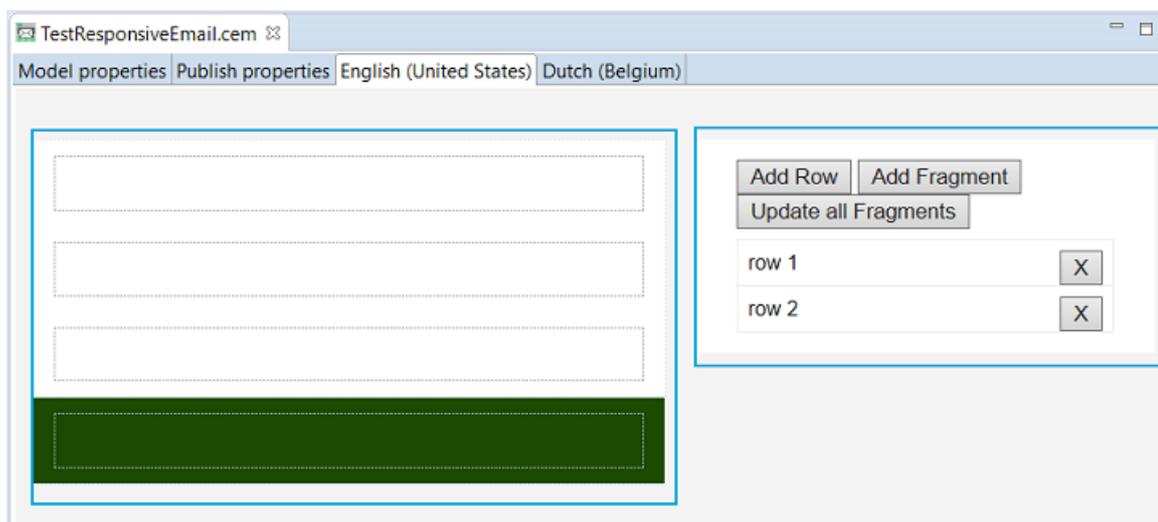
- Check out the responsive email model.
- Open the Responsive Email Language model you created.

Note: if you haven't created a Language model yet, see [17.1.1 Create a responsive email](#) > Responsive Email Language model.



The language model consists of two parts in an Internet Explorer (HTML) environment:

- The Content Editor on the left hand side.
- The Design Editor on the right hand side.



Design Editor

By default, the rows that are available in the Content Editor are displayed in the Design Editor. Modifying the rows in the Content Editor will automatically modify the rows in the Design Editor, and vice versa.

Add a row

To add a row, click **Add Row**. A row is always added beneath the last row. It is not possible to add a row between two existing rows. It is also not possible to drag a row to a different position.

Add a column

To add a column inside a row, click **Add Column**.

A column is always added to the currently selected row.

A column can be resized by means of the slider.

Note that a column cannot be removed separately. You need to remove the entire row.

Add a fragment

To add a Sketch Email fragment, click **Add Fragment**.

- Click **Select** to browse for a fragment.

Important: only responsive email models are compatible with other responsive email models as fragments. Logical models, which have the extension .lgm, cannot be added as fragments. Only responsive email models (.cem) can be used as fragments inside other responsive email models.

- Select the **Language**.
- If you want to make the fragment conditional, check the Conditional checkbox (see [8.1.1 Conditional fragments](#)).

Note that a fragment is always added at the end of the main email model.

Adding fragments to a responsive email works in a similar way as adding fragments to a logical model. See the general 8. Fragments chapter for more information.

Update a fragment

To update all fragments, click **Update all Fragments**. This action updates all inserted Responsive Email fragments.

Delete a row

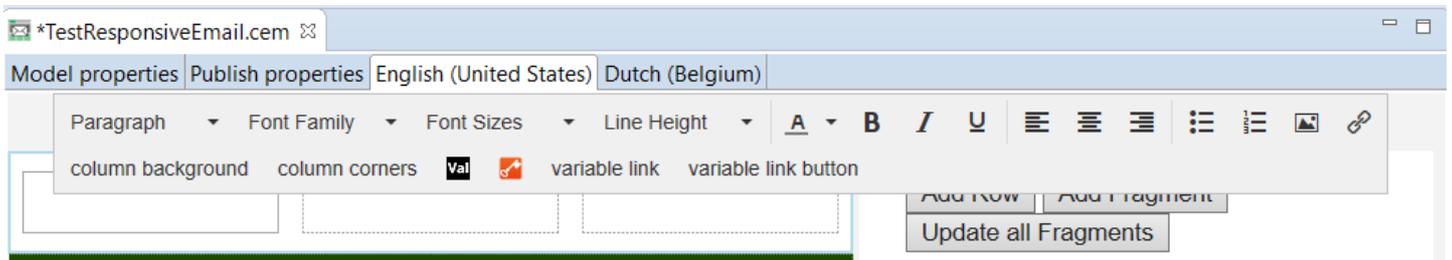
To delete a row, click the **X** button next to the row you want to delete.

Content Editor

By default, a single row is displayed in the Content Editor.

To add content to a row:

- Click inside the row.
- Enter the content of your choice.
- When you click inside a row, a toolbar appears. In this toolbar you can select the font and font size of your text, the paragraph styles, line height, alignment, etc.

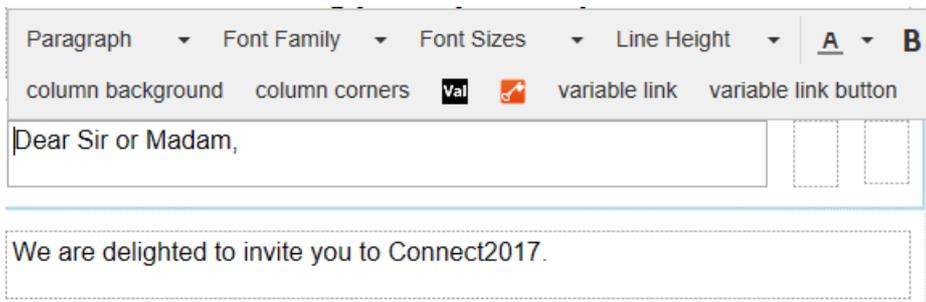


- In the toolbar you can also do the following actions:
 - Insert images
 - Create hyperlinks
 - Insert variables
 - Insert conditions
 - Create a variable link
 - Create a variable link button

Attention: do not copy-paste HTML content. This will result in loss of style and responsive email property as it will copy the properties from the source as well.

Adding text

- Click inside the row to add content.
- Type in the text. Use the toolbar to format the text to your liking.



Adding an image

- Click inside a row.
- Click the image icon  in the toolbar.
- Enter the Source of the image. Note that the image must be inserted from a public location, for instance a URL. E.g. http://www.connective.eu/wp-content/uploads/2016/05/Connective_logo_FULL_MONO_BLACK.jpg
- Enter an Image description (optional).
- The Dimensions (in pixels) are filled in automatically when you enter a valid source.
 - To change the size of the image you can enter the **Dimensions** manually.
 - The option **Constrain proportions** is selected by default. This ensures that the image is always in proportion when you resize it.

Insert/edit image ✕

Source

Image description

Dimensions x Constrain proportions

- Click **OK**.

Tip: to change the properties of an image afterwards, select it and click the image icon again.

Adding a hyperlink

- Click inside a row.
- Position the cursor where you want to insert the hyperlink.

Or

- Select the text or image to which you want to add a hyperlink.
- Click the hyperlink icon  in the toolbar.

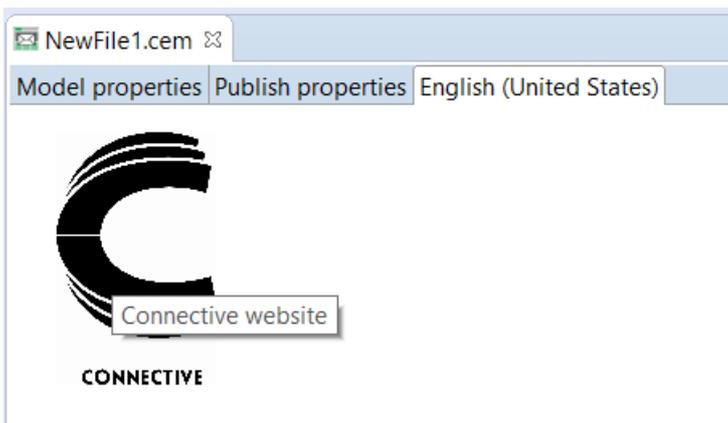
Insert link



Url	<input type="text" value="http://www.connective.eu"/>	
Title	<input type="text" value="Connective website"/>	
Target	<input type="text" value="None"/>	

Ok Cancel

- Enter a valid **URL**. This is the link to the website you want to reference to.
- The **Text to display** is filled in automatically as you enter the URL. To display a different text, type in the required text. This is the text that will be displayed in the responsive email.
- Enter a **Title** for the hyperlink (optional). When you enter a title, this title will appear when you hover the mouse over the hyperlink in the responsive email.
- Select the Target: **none** or **New window**.
 - **None**: the link opens inside the email when you click it.
 - **New window**: the link opens in a new window when you click it.



Example of hyperlinked image

In this example when you click the Connective logo the Connective website opens inside the responsive email.

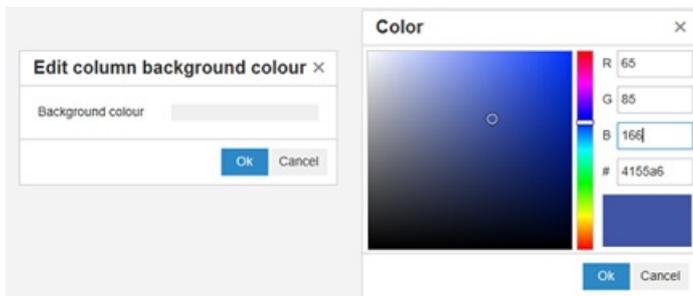
Tip: you can also add a variable link. When adding a variable link, you don't manually enter the URL that must be accessed, but you link it to a variable from the .XSD file. This way, the address of the link is determined by the content of the data structure.

Adding a column background color

- Click inside the column to which you want to add a background color.
- In the toolbar click **column background**.
- Click the **Background color** bar.
- Enter the RGB values.

Or

- Click inside the color palette to select the appropriate color.

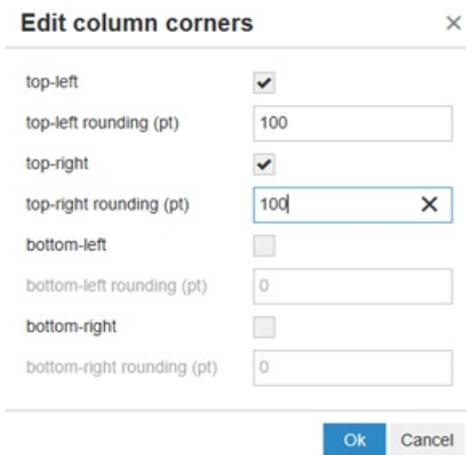


- When you are done, click **OK**.

Editing column corners

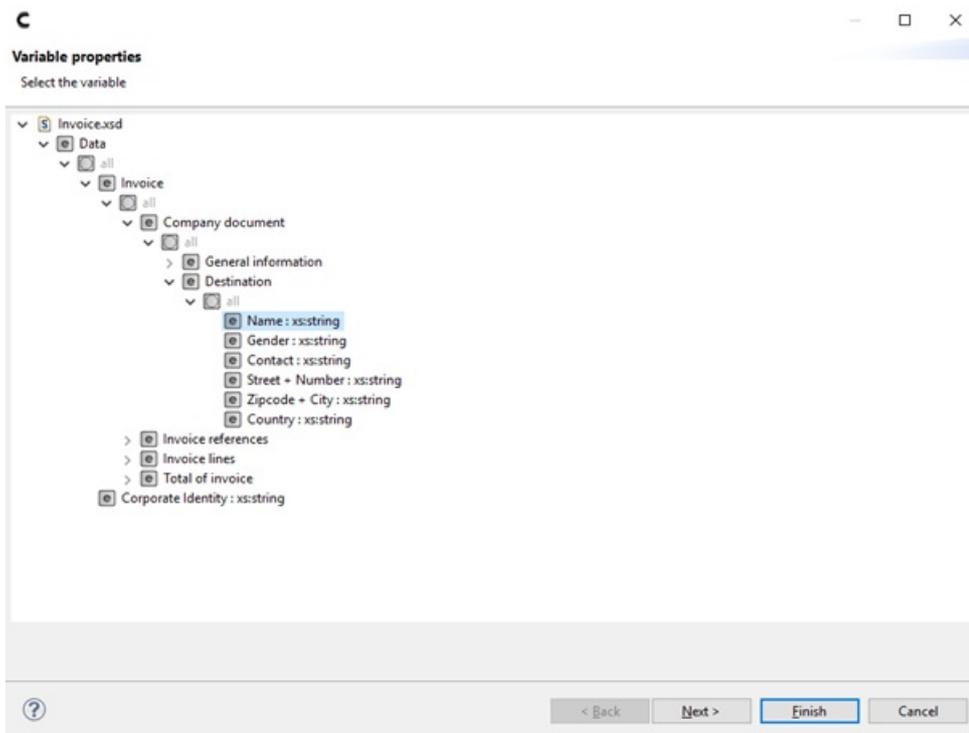
The corners of a column can be rounded.

- Click inside a column.
- Click column corners in the toolbar.
- Select which corners must be rounded: top left, top right, bottom left and bottom right.
- Set a rounding value (in pt) for each corner you selected. The higher the value the bigger the rounding will be.



Adding a variable

- Click inside a row.
- Click the variable icon **Val** in the toolbar.
- The **Variable properties** window now opens.
 - Right-click the root element and click **Expand all**. All available variables from the XSD file that is linked to the Responsive Email model are shown.



- Select the variable you want to insert.
 - Click **Finish** to instantly insert the variable.
 - Click **Next** to configure the variable further. You can do the following actions:
 - Add a decision table.
 - Select the **Formatting**.
 - Configure the **Input fields**.
 - Add a **Description**.

The features and functionalities available in this properties window are exactly the same as the default variable properties windows in the Sketch Logical Model (.lgm).

Adding a variable link

Important: variable links can now adopt the style of the sentence you add them to.

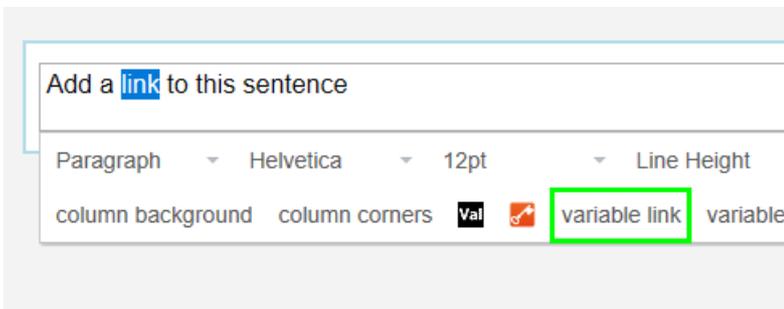
To make sure the variable link will take the style of the sentence:

- Customize the style of the sentence to your liking (font, font style, color, size, etc.)
- Then add the link. The link will now take the same style as your sentence.

Note that adding a link to a sentence, and then changing the style of the sentence is not supported; in this case the variable link will keep its original style.

To add a variable link:

- Select the text you want to link, and click **variable link**.



- In the **Link text** field, enter the text the link must display.

Insert/edit variable link ×

Link text

- Then click **OK**.
- The **Variable link properties** window now opens. **Note:** if no .XSD file is linked, an error message will be displayed.
- Right-click the root element and click **Expand all**.
- Now select the variable you want to link to, and click **Finish**. The content of the variable will be used as link.

For a concrete example, see [17.1.3 Examples of variables](#)

Adding a variable link button

This feature adds a button containing a variable link to the email.

- Click inside a row.
- Click **variable link button**.
- Enter the **Link text** you want to be displayed inside the button.

Insert/edit variable link button ×

Link text

Button text colour

Button background colour

Rounded corners

Corner rounding %

- Click the **Button text color** bar to select the text color of the button.
- Enter the RGB values, or
- Click inside the color palette to select the appropriate color.
- Click the **Button background color** bar to select the background color of the button.
- The button has **Rounded corners** by default. Set the **Corner rounding** percentage.
- To deactivate rounded corners, clear the Rounded corners check box.
- Click **OK**. **Note:** if no .XSD file is linked, an error message will be displayed.

- Right-click the root element and click **Expand all**.
- Now select the variable you want to link to, and click **Finish**. The content of the variable will be used as link.

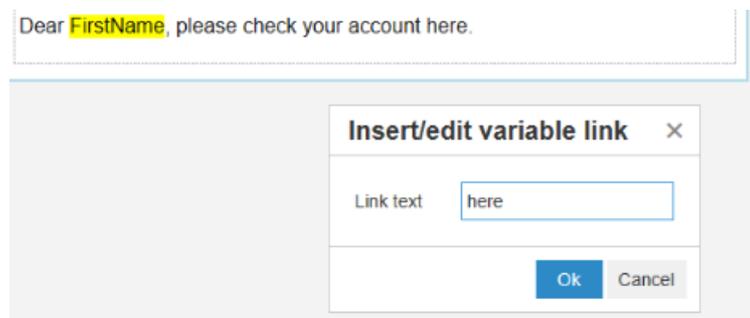
For a concrete example, see [17.1.3 Examples of variables](#).

17.1.3 Examples of variables

In this section we provide some examples of the features described in [17.1.2. Create the content of a responsive email](#).

Example of a variable link

- Click inside a row and type "Dear FirstName, please check your account here." Note that "FirstName" is a variable.
- Select "here" and click on **variable link** in the toolbar.
- The link window opens and "here" is entered in the **Link text** field.

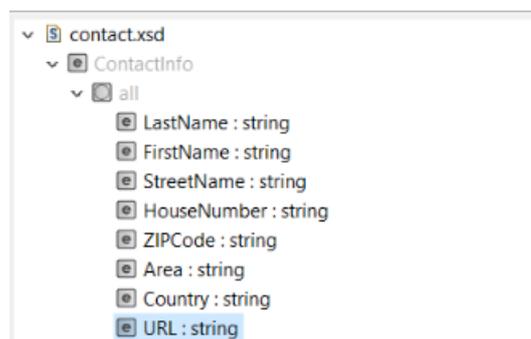


- Click **OK**.
- Browse for the variable URL.

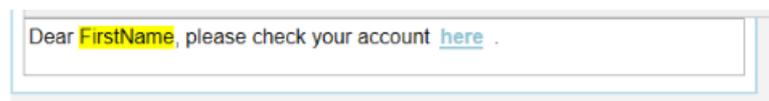
C

Variable link properties

Select the element to be inserted as Variable link



- Click **Finish**.
- The variable link looks as follows. Clicking the link "here" will open FirstName's account, which is stored in the URL variable in the XML file.



Example of variable link button

Click inside a row and type "Dear FirstName, please click the button below to download your invoice."

Click **variable link** button on the toolbar.

Enter "Download" in the **Link text** field.

Choose the **Button text color** and **Button background color**.

Click **OK**.

Browse for the variable URL.

Click **Finish**.

The variable link button looks as follows. Clicking the **Download** button will open FirstName's invoice, which is stored in the URL variable in the XML file.



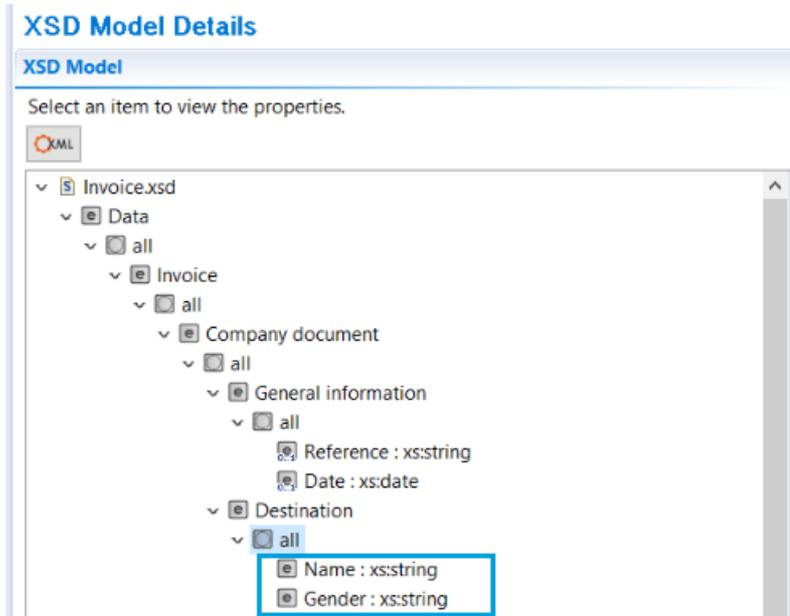
Tip: to resize the button, add a second column, add the "Download" button to that column and then use the slider to resize the columns.



Example of variable with conditions

For this example to work, there are a few preliminary steps to take:

- Create an **.xsd** file and add the variables **Name** and **Gender**.



- Create an **.xml** file based on the **.xsd** file you created.
- Create a decision table **.lgs** file that contains the keys "M" and "F" and the values "Mr." and "Mrs."

Configure Decision Table

Languages

English (United States)

Key	Value
M	Mr.
F	Mrs.

- Link the three files above to your responsive email model.
- Open the Language model of your responsive email.



- Click inside a row and type "Mr. Mrs."
- Click the variable icon **Val**.
- Right-click the root element and click **Expand all**.
- Select the **Name** variable and click **Finish**.

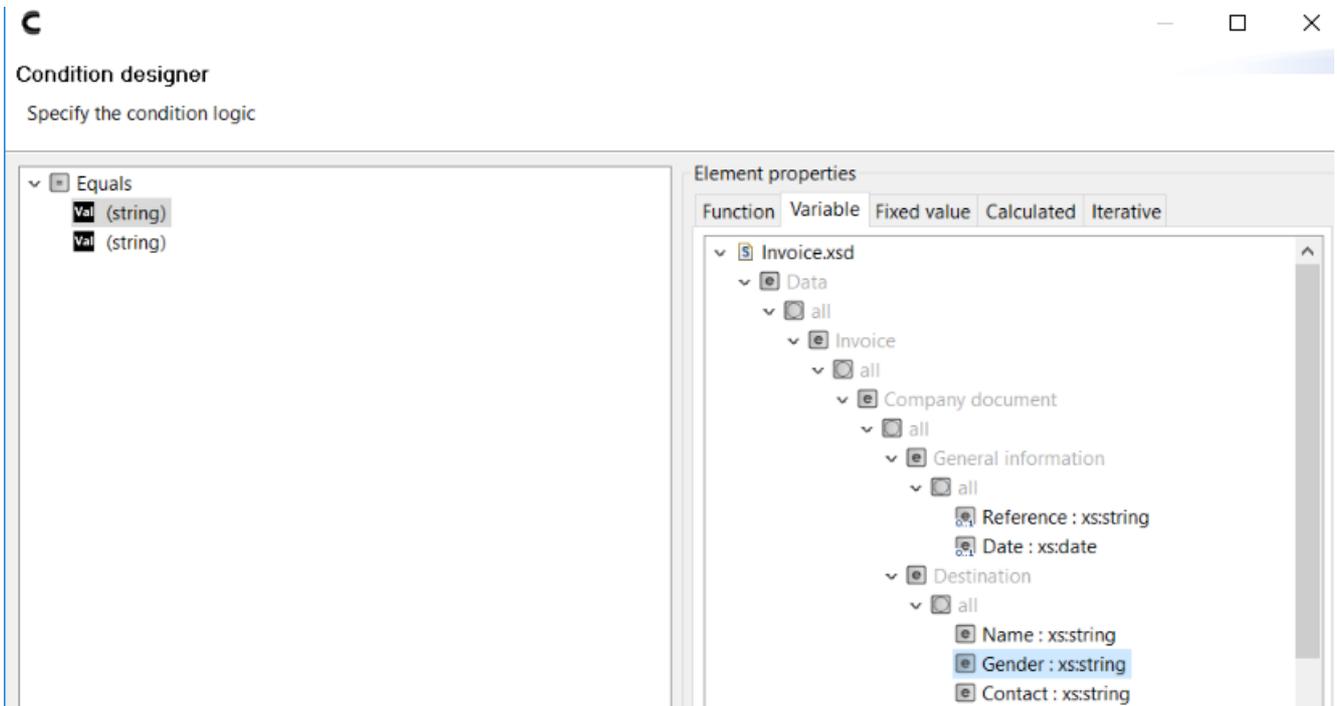
C

Variable properties

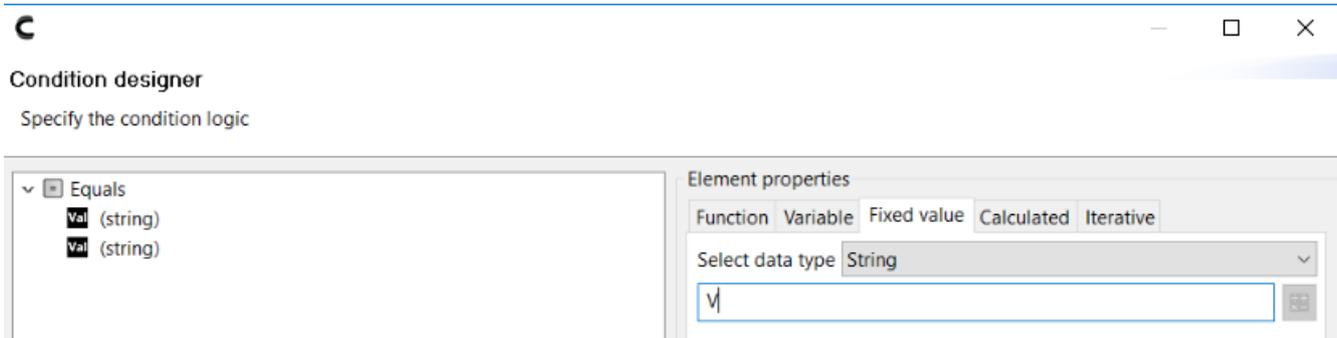
Select the variable

- ▼ Invoice.xsd
 - ▼ Data
 - ▼ all
 - ▼ Invoice
 - ▼ all
 - ▼ Company document
 - ▼ all
 - ▼ General information
 - ▼ all
 - Reference : xs:string
 - Date : xs:date
 - ▼ Destination
 - ▼ all
 - Name : xs:string**
 - Gender : xs:string

- Select "Mr." and click on the conditions icon
- Keep the default operator **Equals**.
 - Click the first **Val** (string), and then click the **Variable** tab.
 - Expand all and double-click the **Gender** variable.



- o Click the **Fixed value** tab and enter "V" as value.



- Click **Next**.
- Select **Delete selection**.

If the gender equals "V", "Mr." will be deleted.

- Click **Next**.
- Select **no action (show selection)**.

If the gender equals "M", "Mr." will be shown.

- Click **Finish**.
- Now select "Mrs." and click on conditions icon .
- Keep the default operator **Equals**.
 - o Click the first **Val (string)**, and then click the **Variable** tab.
 - o Expand all and double-click the **Gender** variable.
 - o Click the **Fixed value** tab and enter "M" as value.

- Select **delete section**.

If the gender equals "M", "Mrs." will be deleted.

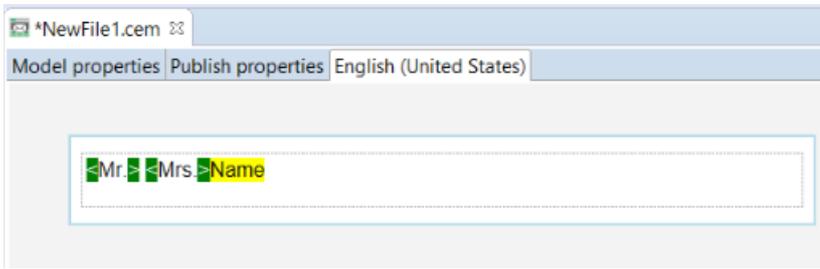
- Click **Next**.

- Select **no action**.

If the gender equals "V", "Mrs." will be shown.

- Click **Finish**. The result now looks as follows:

Just like in Logical Models, a condition has distinctive green brackets around the data to be conditioned.



17.2 How to publish a responsive email model

Publishing a responsive email model is done in the same way as publishing a logical model. See [3.9 How to publish a logical model](#) for detailed information.

18. Tips and Tricks

18.1 LibreOffice

18.1 LibreOffice

What to do in case of connection loss?

It might happen that the LibreOffice connection is lost unexpectedly. If this happens, do not try to save the model changes; saving the changes might corrupt the model.

To avoid corrupting the model, a warning message is displayed informing you that the LibreOffice connection is lost. Click OK to close and reopen all the opened editors without saving the unsaved changes.



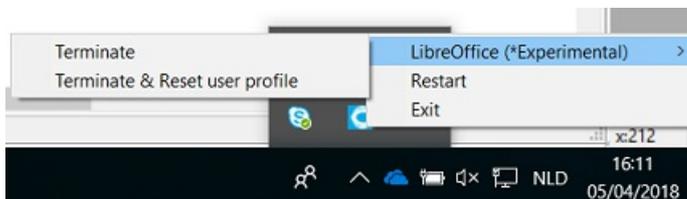
What to do when LibreOffice is not responding, the LibreOffice bridge is disposed or the Sketch interface becomes unresponsive?

Do not use the Windows Task Manager to end LibreOffice or Sketch. Instead, use the new Sketch system tray icon.



Right-click the Sketch icon and choose one of the available options:

- **LibreOffice > Terminate**: this restarts LibreOffice. Use this option in case your model is not responding.
- **LibreOffice > Terminate & Reset user profile**: this restarts LibreOffice and restores the default settings. Note that personal settings in LibreOffice need to be re-entered (such as personalized units, colors, etc.)
- **Restart**: restarts the full Sketch application. Use this option when the entire Sketch interface becomes unresponsive.



Note:

The Sketch icon is by default added to the system tray when Sketch is launched. In some cases it may be advised to disable this feature, for instance in a Citrix environment.

To disable the Sketch system tray icon:

- Open the **Sketch.ini** file from the Sketch installation folder.
- Add "-Deu.connective.sketch.trayicon=false" to the configuration file.

```
Sketch - Notepad
File Edit Format View Help
-Xms512m
-Xmx1024m
-server
-Dbellettergen.product=Sketch
-Dlettergen.user.home=C:\Users\YourUser\Sketch
-Dosgi.configuration.area=C:\Users\YourUser\Sketch\workspace5.5/configuration
-DOOO_HOME=C:\Program Files\Sketch v5.5.0\libreoffice
-DOOO_PROGRAM_PATH=C:\Program Files\Sketch v5.5.0\libreoffice/program
-DOOO_COMANDLINE_ARGS=#nologo #nodefault #noreset #invisible
-Djava.library.path=C:\Program Files\Sketch v5.5.0\libreoffice/program
-Dbellettergen.configuration.user=configuration.xml
-Dfile.encoding=UTF-8
-Dos.version=6.2
-Deu.connective.sketch.trayicon=false
```

- Save the **Sketch.ini** file in the Sketch installation folder.

19. FAQ

In the FAQ section we handle the most frequently asked questions about Sketch. The questions are bundled per topic where possible.

19.1 Sketch

19.1.1 What if Sketch won't open?

19.1.2 Where can I find the Sketch logs?

19.1.3 What if Sketch is not responding?

19.1.1 What if Sketch won't open?

Possible cause

A possible cause could be that Sketch isn't closed correctly and because of this a logical model or folder is blocked.

Solution

- Close Sketch.
- Empty your workspace. The workspace can be found in C:\Users\USERNAME\Skyetch\workspaceX.X.
- If this doesn't work, delete the complete folder. **Note:** changes on checked-out items will be lost.
- Restart Sketch.
- Re-establish your connection to the ModelStore.

19.1.2 Where can I find the Sketch logs?

The Sketch log files folder is configured during installation. The default location is C:\Users\USERNAME\Sketch\logs.

The logs folder contains all the log files. Sketch.log is the active log file. Other files are archived logs.

19.1.3 What if Sketch is not responding?

When Sketch is not responding try the following steps:

- Verify that your internet connection is working.
- Right-click the ModelStore and click the refresh icon . A progress bar now appears.
- Force Sketch to close:
 - Start Windows Task Manager and end the Sketch task.
 - Verify that the LibreOffice task is ended too. (This is the soffice.bin process in the Task Manager.)
 - Reopen Sketch.
- Right-click the ModelStore and click **Manage Account**. Then confirm your login and password.
- Right-click the ModelStore and click **Delete**. Then reconnect to your ModelStore.
 - Close Sketch.
 - Rename the Workspace folder. For example by adding a date: C:\Users"USER PROFILE"\Sketch\workspaceX.X_DATE.
 - Restart Sketch. Your ModelStore and PublishStore connection is now removed.

19.2 ModelStore

19.2.1 How to make a ModelStore connection?

19.2.2 What if error messages occur while making a connection to the ModelStore?

19.2.3 What if the ModelStore is disconnected?

19.2.4 What if an error message appears while publishing?

19.2.5 What if the user doesn't see models in the ModelStore?

19.2.6 How are files sorted?

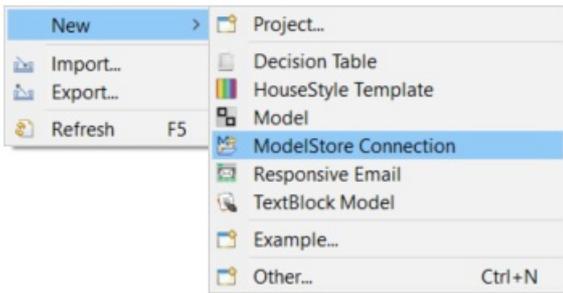
19.2.7 Why are not all models visible in the ModelStore?

19.2.8 Why are newly created models not visible in the ModelStore?

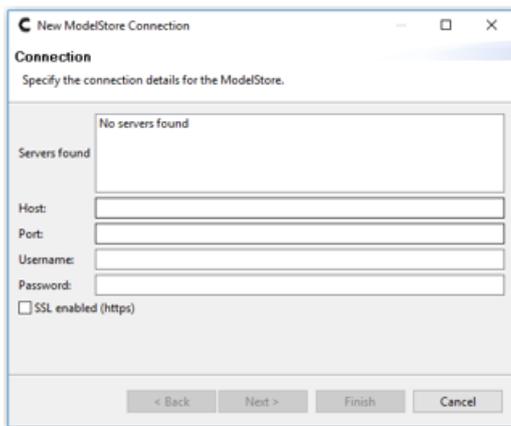
19.2.9 Why are changes made by a colleague not visible in the ModelStore?

19.2.1 How to make a ModelStore connection?

- Right-click in the Sketch navigator (i.e. the left pane of the screen).
- Click **New > ModelStore Connection**.



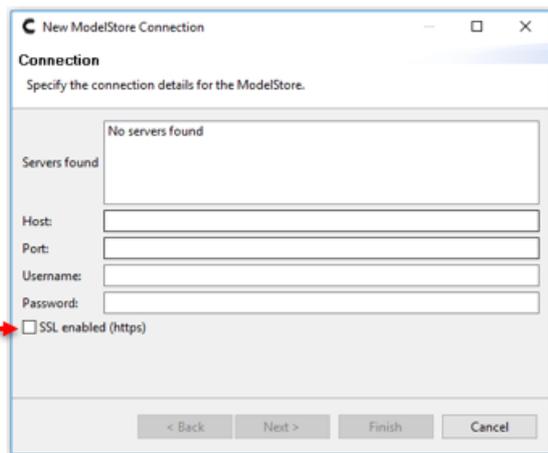
- Enter your credentials and click **Next**.



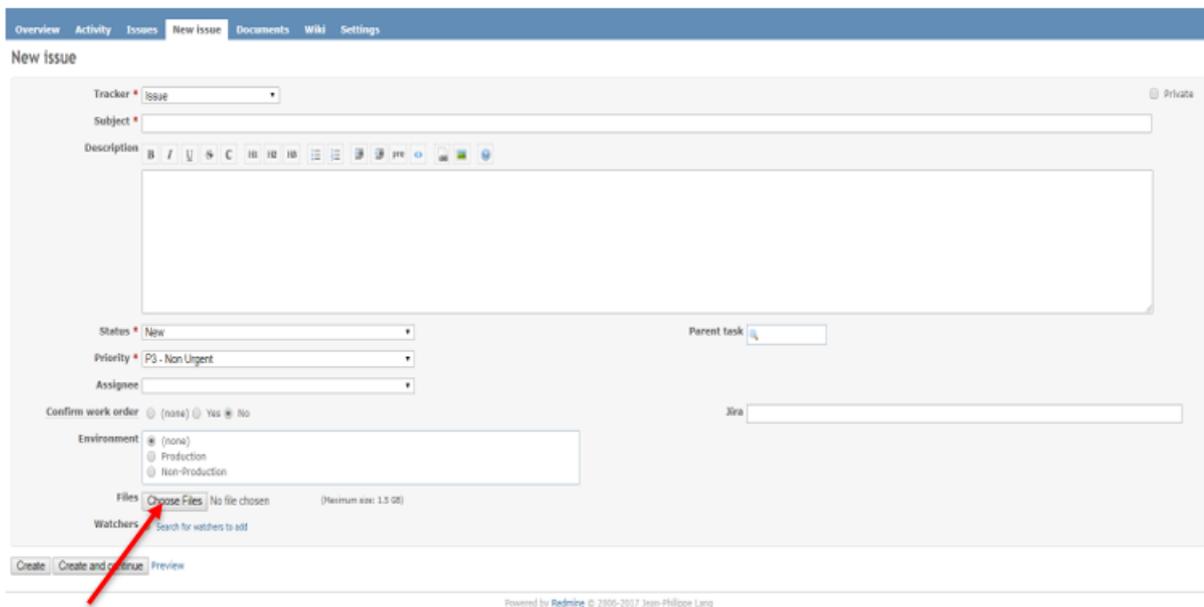
- In the next screen select the required ModelStore(s) and PublishStore(s).
- Click **Finish**.

19.2.2 What if error messages occur while making a connection to the ModelStore?

- Verify that all fields and configuration details are correct. Pay extra attention to the SSL checkbox.



- If you keep getting error messages, check the log files.
- Create a support ticket and add the log files.



19.2.3 What if the ModelStore is disconnected?

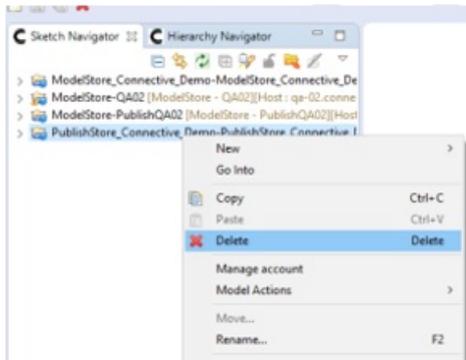
- Check your network/internet connection.
- Check the server connection.

19.2.4 What if an error message appears while publishing?

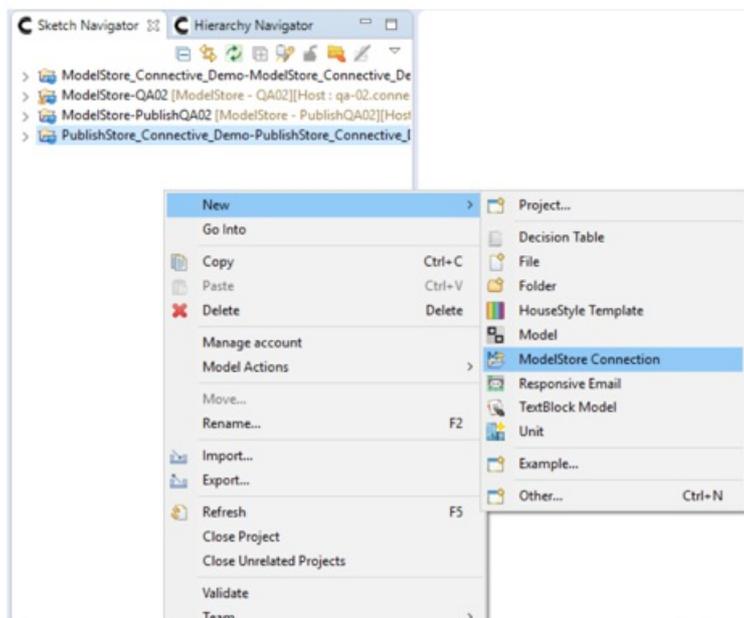
The error message **ModelStoreEvent – Disconnected** might appear while publishing. This error message is displayed when the PublishStore is not connected when you try to publish the models.

To solve this issue:

- Delete the PublishStore connection.



- Re-establish the ModelStore connection.



- Enter the required credentials.

19.2.5 What if the user doesn't see models in the ModelStore?

Possible cause

This issue might be caused by a caching problem. The login credentials are cached and should refresh every ~20 minutes.

Solution

- Refresh the cache.
- Check your ModelStore connection.

19.2.6 How are files sorted?

Files are sorted alphanumerically: first in numerical order and then in alphabetical order.

19.2.7 Why are not all models visible in the ModelStore?

To solve this issue:

- Refresh the Workspace.
- Select the ModelStore or folder.
- Press F5 or click the refresh icon  in the Sketch navigator toolbar.

19.2.8 Why are newly created models not visible in the ModelStore?

To solve this issue:

- Refresh the Workspace.
- Select the ModelStore or folder.
- Press F5 or click the refresh icon  in the Sketch navigator toolbar.

19.2.9 Why are changes made by a colleague not visible in the ModelStore?

To solve this issue:

- Refresh the Workspace.
- Select the ModelStore or folder.
- Press F5 or click the refresh icon  in the Sketch navigator toolbar.

19.3 Logical Model

19.3.1 [How to rollback to previous versions of models?](#)

19.3.2 [How can I check historic version comments?](#)

19.3.3 [Is it possible to change the default language of the description field in the Model Properties tab?](#)

19.3.4 [How can I see which models are linked to a fragment?](#)

19.3.5 [Why is the wrong language of a variable shown in a preview?](#)

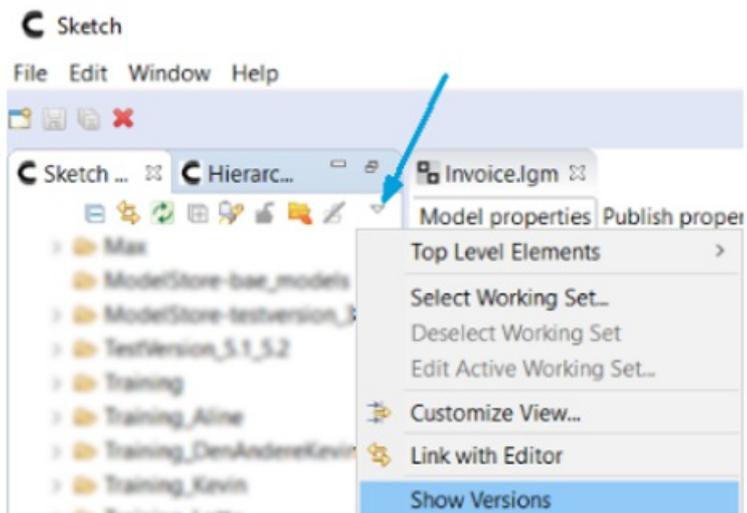
19.3.6 [What to do when a Model Assembling Failed?](#)

19.3.7 [Why does my moved/copied model show an error message?](#)

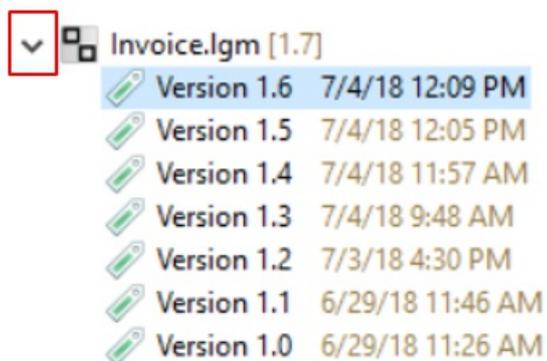
19.3.8 [Why is copy-pasted logic not working?](#)

19.3.1 How to rollback to previous versions of models?

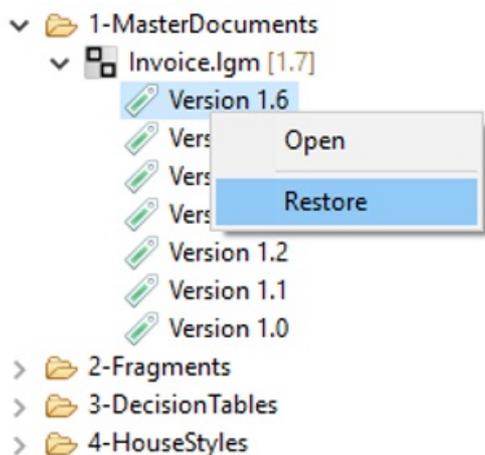
- In the Sketch Navigator toolbar click the triangle icon, and then click **Show Versions**.



- Select the logical model you want to restore.
- Click the triangle icon in front of the model name. The list of different versions is displayed.



- Right-click the version you want to restore and click **Restore**.

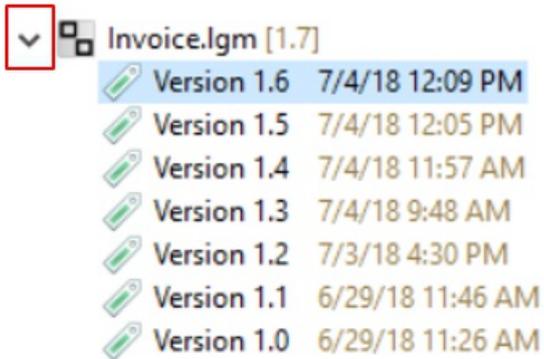


19.3.2 How can I check historic version comments?

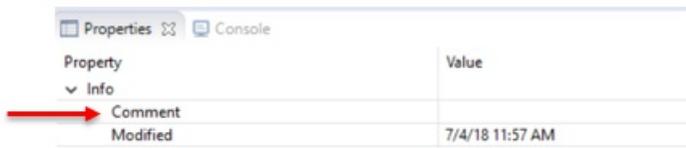
The most recent comment is visible in the properties.

To display older comments:

- In the Sketch Navigator toolbar click the triangle icon, and then click Show Versions.
- Select the logical model of which you want to check older comments.
- Click the triangle icon in front of the model name. The list of different versions is displayed.



- Select a version from the list.
- The version information is shown in the **Properties** tab of the selected version (in the bottom right corner in default view).



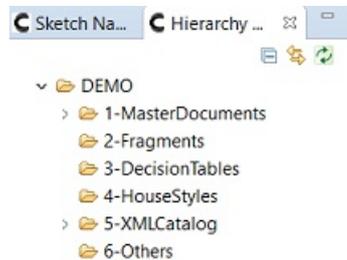
19.3.3 Is it possible to change the default language of the description field in the Model Properties tab?

No. The default language is English (United States) and cannot be changed.

19.3.4 How can I see which models are linked to a fragment?

Via the Hierarchy Navigator a different look on the ModelStore is provided. You can easily see which files are used in which model.

The Hierarchy Navigator is displayed as a tree with each block representing a node in the hierarchy. The parental node is the ModelStore with the list of model templates. Each model template can be expanded, visualizing the underlying blocks to the user. By default, only the first node **ModelStore** will be expanded. The view represents bidirectional references of each element in a group, meaning that also the inverted reference is presented in this view.



A block represents the following items:

A group of models (.lgm)

A group of used fragments (.lgm) with a group of models in which they are used

A group of decision tables (.lgs) with a group of models in which they are used

A group of house styles (.ott) with a group of models in which they are used

A group of Writer files (.lgw) with a group of models in which they are used and with a group of Textblocks used in LetterWriter Files (.lgb)

A group of XSDs (.xsd) with a group of models in which they are used

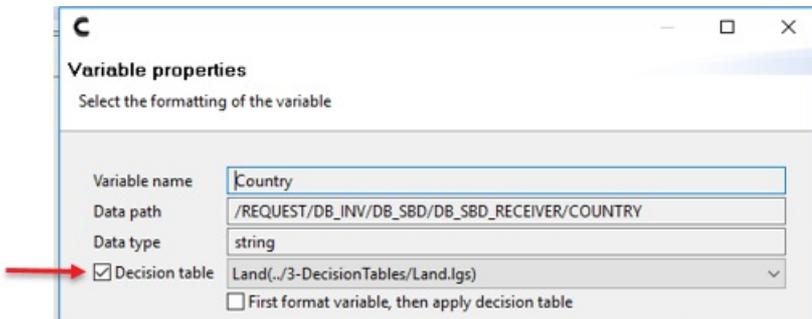
A group of XMLs (.xml) with a group of models in which they are used

A group of responsive emails (.cem) with a group of models in which they are used

19.3.5 Why is the wrong language of a variable shown in a preview?

Check the following points:

- Check in the XML if the input data is in the right language.
- Check if a Decision Table is linked to the variable.



- Check if everything in the Decision Table is translated correctly.

| Key | Value | Reference - Dutch (Belgium) |
|-----|----------|-----------------------------|
| M | monsieur | de heer |
| V | madame | mevrouw |

- Check if the variable content in the XML is the same as the 'keys' in your decision table. If not, the value will be shown as defined in the XML and therefore the linked decision table will not work.

| Key | Value | DB_SBD_RECEIVER |
|-----|----------|-----------------|
| M | monsieur | NAME Royer BVBA |
| V | madame | GENDER V |

19.3.6 What to do when a Model Assembling Failed?

- Verify that the model and its related files (XML/XSD/Fragments/HouseStyle) are available.
- Check the paths to the related files. Pay attention to changes in capital letters. Sketch is case sensitive; a change in capital letters can cause an error. Change the paths of the related files.

19.3.7 Why does my moved/copied model show an error message?

Error messages may be displayed when models are moved or copied in the folder structure.

The links related to the files (XML, XSD, fragments, housestyles) might be broken. They are 'encoded' in the model.

When moving models / copying models to another folder, you also should ensure that the linked files (*.xsd, *.xml, fragments, HouseStyle) are reinstated in your new file. The links in the model are relative and not automatically updated.

19.3.8 Why is copy-pasted logic not working?

Copy-pasting logic is not supported. The references to the logic will be broken and this logic will be perceived as textual items or images. This means that after copy-pasting in the Sketch editor the logical section still looks like logic, but no longer acts like logic. Copy-pasting in general is not recommended in Sketch.

Textual copy-pasting is possible however. To do so, click **Edit > Paste Special... > Paste Unformatted Text** to avoid lingering markup characteristics.

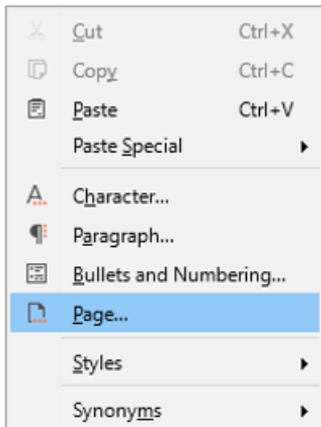
19.4 HouseStyle templates

19.4.1 How can I automatically standardize my format?

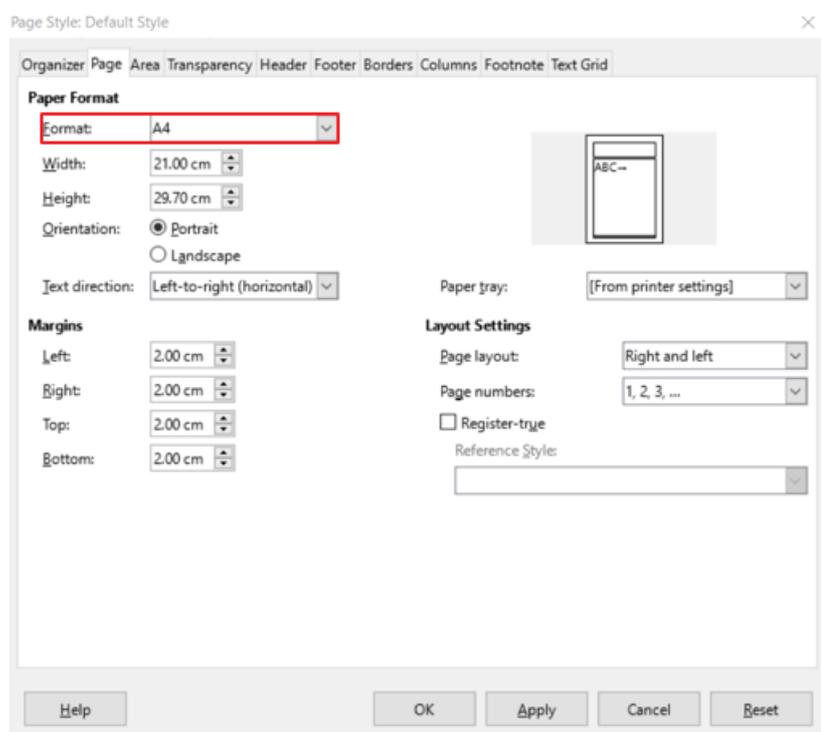
To automatically standardize the format of a HouseStyle model, e.g. to A4 format, take the following steps.

If you are using an HouseStyle model:

- Check out your HouseStyle model.
- Right-click your page, and click **Page** in the context menu.



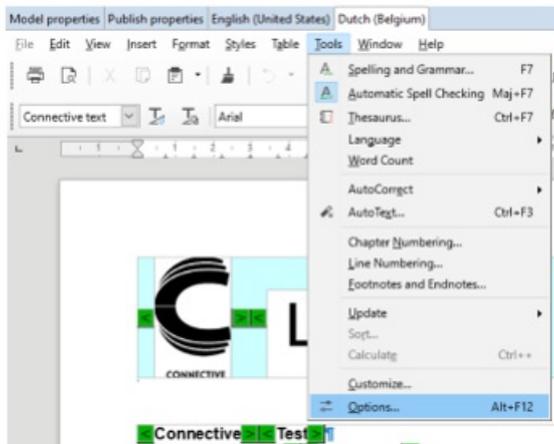
- The **PageStyle** window now opens.
- Click the **Page** tab.
- In the **Format** drop-down list, select the desired standard format for your documents.



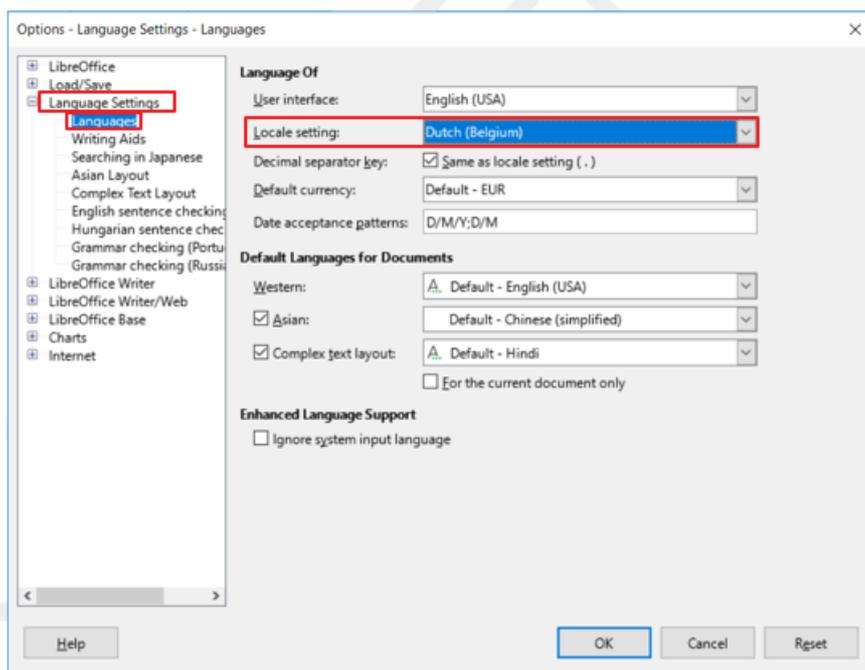
- Then click **Apply**.
- Now check in your HouseStyle model, and refresh the ModelStore. All existing models and new models linked to this HouseStyle will take over the HouseStyle.
- Note that all existing models must be republished after this modification.

If you are not using a HouseStyle model:

- Set the Locale settings to Dutch (Belgium). These settings use A4 as standard format. To do so:
- Open your language model.
- In the Editor region, click the **Tools** tab > **Options**.



- Expand **Language Settings** in the left menu.
- Click **Languages**.
- Select **Dutch (Belgium)** in the **Locale setting** drop-down list. The document format will be automatically changed to A4.



19.5 XML

19.5.1 What to do when empty fields cause an error during preview or generation?

19.5.2 What to do in case of the following error message: The specified path was not found?

19.5.1 What to do when empty fields cause an error during preview or generation?

If the error message "Cannot convert string to a double" is displayed, it is probably caused by a field in numeric type (decimal, integer,..) which contains no value. Numeric fields cannot contain blanks or strings, only numeric values are allowed.

19.5.2 What to do in case of the following error message: The specified path was not found?

- Open the Sketch [log files](#) and verify that the path is available in Sketch.
- Restore your path.

19.6 XSD

19.6.1 How do I create a new XSD?

19.6.2 How do I insert a new variable?

19.6.3 How do I insert a new data block?

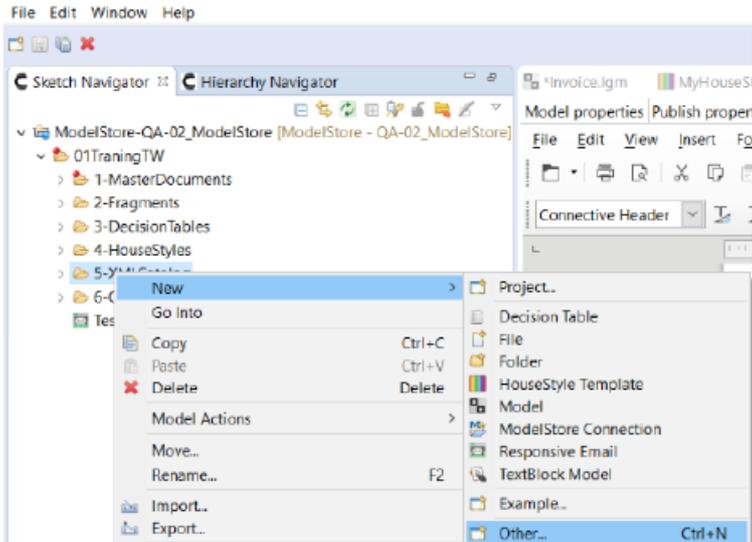
19.6.4 How do I move variables and data blocks in between other variables or data blocks?

19.6.5 How do I remove a linked XSD from a model?

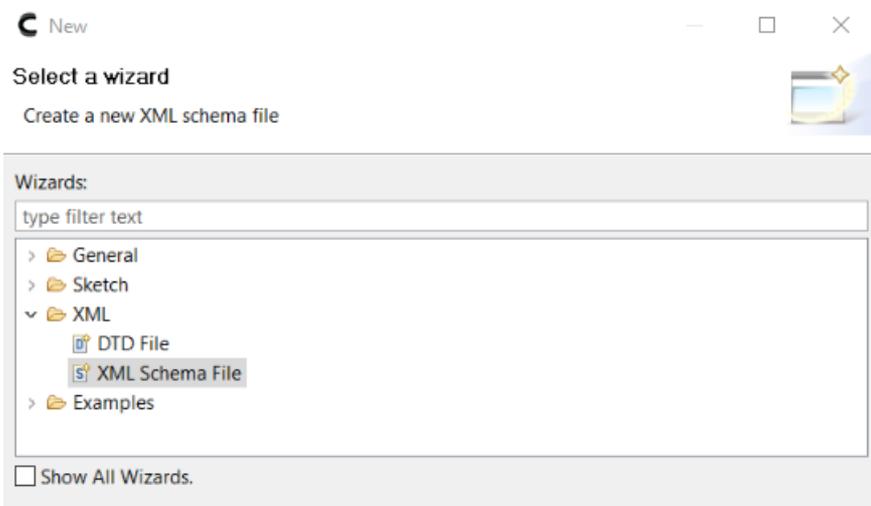
19.6.6 Why is my XSD not overwritten when pasting a new one?

19.6.1 How do I create a new XSD?

- Right-click the folder where you want the XSD to be located.
- In the context menu click **New > Other**.



- The **New** wizard appears.
- Click **XML > XML Schema File**.
- Click **Next**.
- Enter the file name.
- Click **Finish**.

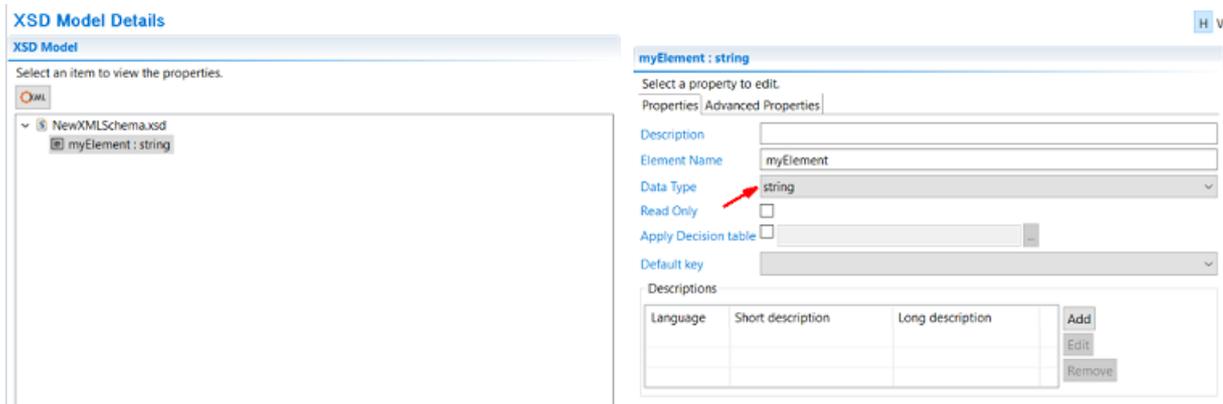


- Your new data structure file is created in the selected folder. The XSD is checked out. Double-click the file to open the XSD in the Editor region.

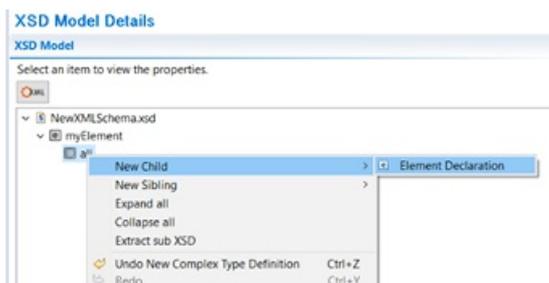
19.6.2 How do I insert a new variable?

Before you can insert a new variable you must first add a data block to the XSD:

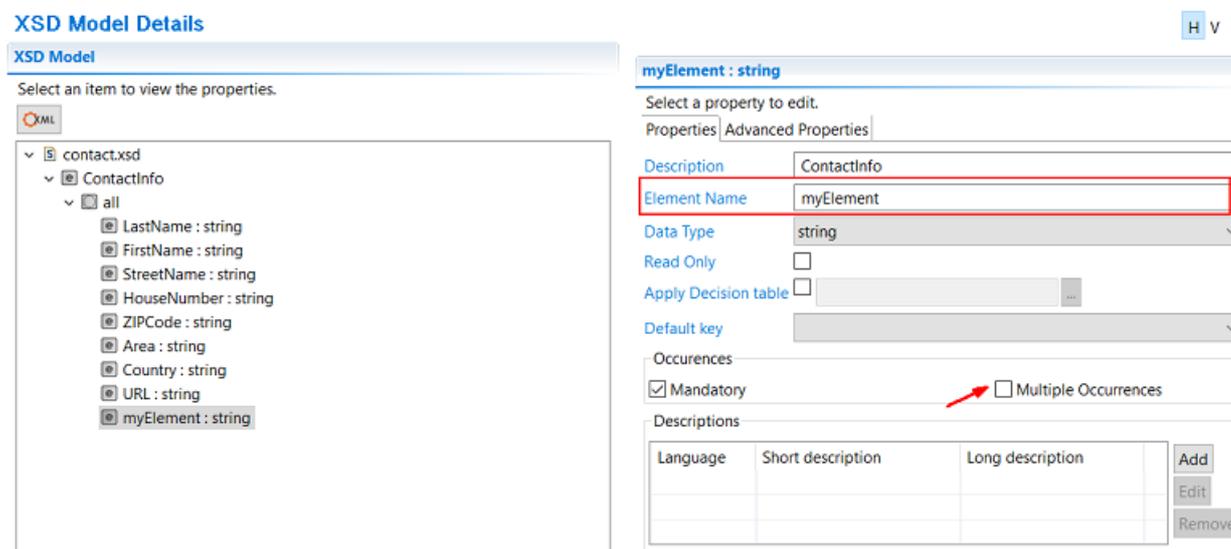
- Right-click on the root element, and click **New Child > Element Declaration**.
- Click the element you created.
- In the **Element Name** field, type a name for the variable.
- Select a variable type from the **Data Type** drop-down list (or **Browse** if not listed).



- Right-click the element you created, and click **New Child > Complex Type Definition**.
- The "all" block is now added beneath the element.
- Right-click the "all" block and click **New Child > Element Declaration**.



- In the **Element Name** field, type a name for the variable.
- Select a variable type from the **Data Type** drop-down list (or **Browse** if not listed).
- Check the **Mandatory** check box if the variable needs to be mandatory.
- Check the **Multiple Occurrences** check box if the variable needs to be repeated.



19.6.3 How do I insert a new data block?

To add a **new data block** at the **same level** of an existing data block:

- Right-click on a data block.
- Select **New Sibling > Element Declaration**.
- A new element is added.
- Right-click on this new element.
- Select **New Child > Complex Type Definition**.

To add a **new data block nested** in an existing data block:

- Right-click on the **ALL** node of a data block.
- Select **New Child > Element Declaration**.
- A new element is added.
- Right-click on this new element.
- Select **New Child > Complex Type Definition**.

Now you can fill in the appropriate fields in the **Properties Section**. Save by doing a check-in of the file.

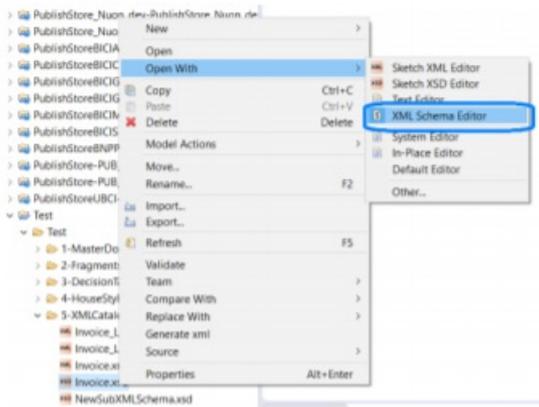
XSD Model Details

The screenshot displays the 'XSD Model Details' interface. On the left, a tree view shows the project structure: 'NewXMLSchema.xsd' containing 'NewVariable : string', 'myElement', and 'all'. Under 'myElement', there is an 'all' child. The right pane shows the 'myElement' properties. The 'Properties' tab is active, showing fields for Description, Element Name (myElement), Data Type (myElement__type), Read Only (unchecked), Apply Decision table (unchecked), and Default key. The 'Occurrences' section has 'Mandatory' checked and 'Multiple Occurrences' unchecked. Below is a 'Descriptions' table with columns for Language, Short description, and Long description, and buttons for Add, Edit, and Remove.

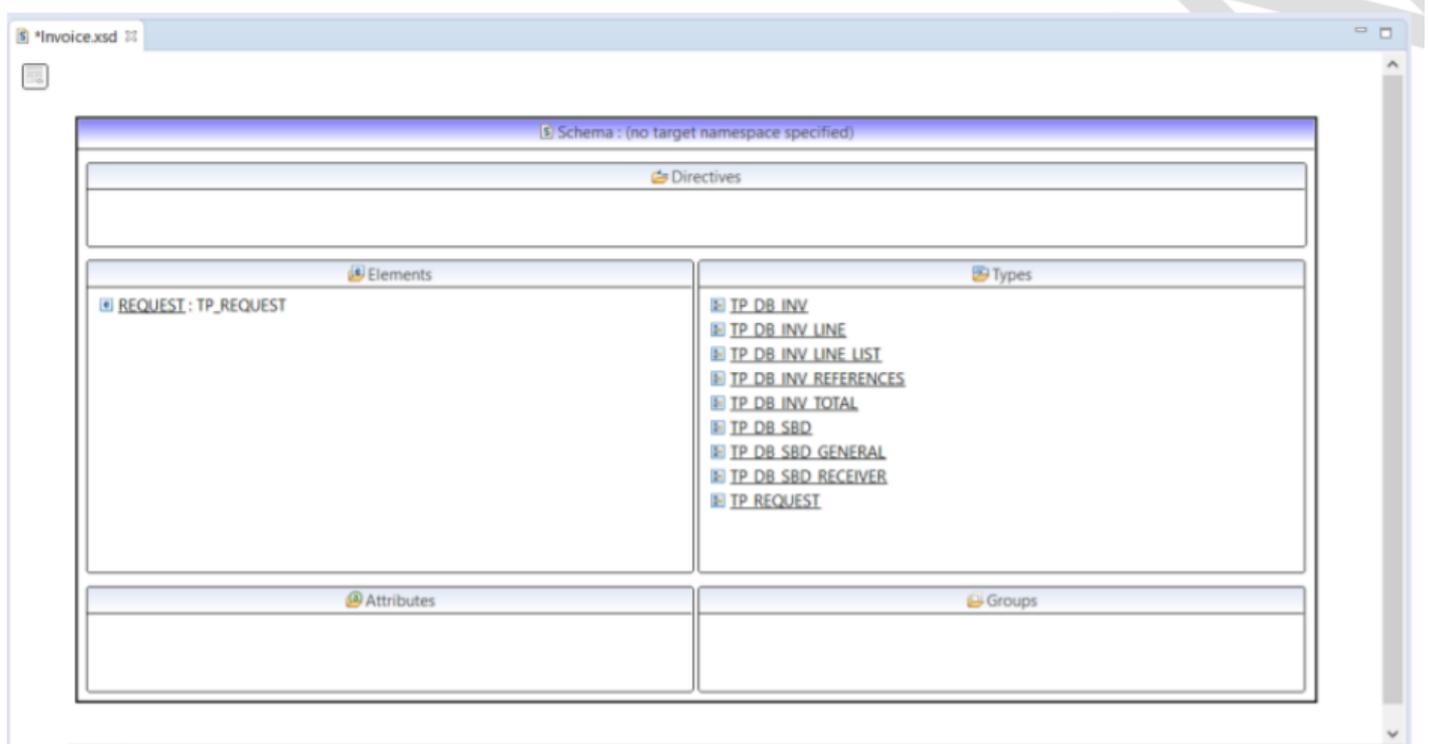
| Language | Short description | Long description | |
|----------|-------------------|------------------|--------|
| | | | Add |
| | | | Edit |
| | | | Remove |

19.6.4 How do I move variables and data blocks in between other variables or data blocks?

- Right-click the required XSD, and click **Open with > XML Schema Editor**.

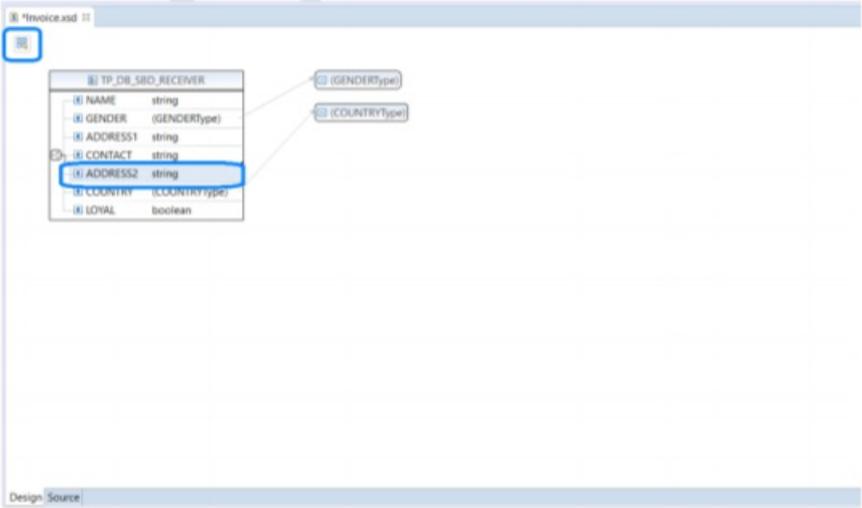


- Double-click the data block that must be edited for a detailed view.



- Click and drag the elements to the desired location in the structure.

Tip: you can switch between the detailed view and the general overview using the button at the top left.



19.6.5 How do I remove a linked XSD from a model?

- Check out the model.
- Click the **Model properties** tab.
- Click the recycle bin icon next to the XSD in order to remove the XSD and the linked XML's.

Model properties | Publish properties | English (United States) | Dutch (Belgium)

Model properties

▼ Descriptions

| Language | Description | |
|-------------------------|------------------|--------|
| English (United States) | Invoice Training | Add |
| Dutch (Belgium) | Factuur Training | Edit |
| French (Belgium) | Facture Training | Remove |

▼ Data structures

| | | | |
|----------------------|---|---|--------|
| Data structure (XSD) | ../5-XMLCatalog/Invoice.xsd |  | ... |
| Sample XML | ../5-XMLCatalog/Invoice Connective.xml
../5-XMLCatalog/Invoice Other.xml |  | Add |
| | | | Remove |



19.6.6 Why is my XSD not overwritten when pasting a new one?

To solve this issue:

- Check out the old XSD in the destination folder.
- Paste the new XSD in the destination folder.
- The old XSD is now successfully overwritten.

19.7 Fragments

19.7.1 How to update a fragment?

19.7.2 How to update all fragments?

19.7.3 What is the reason a fragment with an image cannot be inserted?

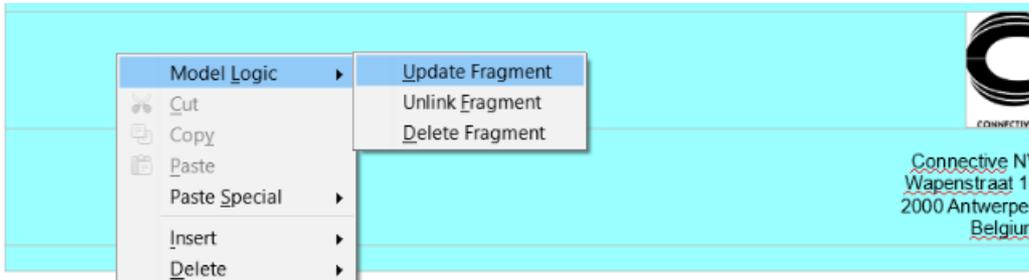
19.7.4 How to automatically update a model after changing an inserted fragment?

19.7.5 How do I add text at the beginning of the page when there is a fragment placed at that position?

19.7.1 How to update a fragment?

To update a **single fragment**:

- Make sure the master document is **checked out**.
- In the Model Editor, place the cursor in the document where you want to update the fragment.
- Right-click and click **Model Logic > Update Fragment**.



- The selected fragment is updated and the other fragments have not been modified.

19.7.2 How to update all fragments?

After editing a fragment, you should republish all the templates that make use of this fragment.

- Check the Hierarchy navigator to see the dependencies.
- Publish all linked models.
- Right-click your selection and click **Model options > Publish**. Publishes are updated, models aren't. To update your models, follow the steps described below.

Update all fragments:

- Make sure the master document is checked out.
- In the upper left corner, click the **Update Fragments** icon.



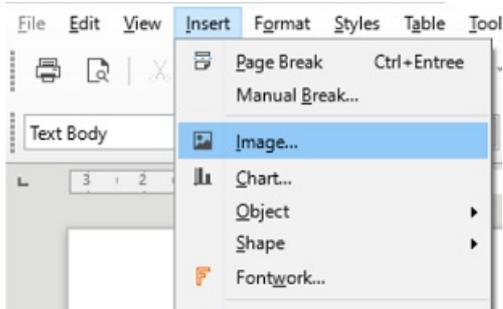
- Click **Yes** to confirm. All fragments are updated at once.
- Now save and check in the model. The updated versions of the fragments will also be saved.

19.7.3 What is the reason a fragment with an image cannot be inserted?

The way the image has been inserted determines its use in Sketch. Depending on how you did it, Sketch will see the image as an object. In this case problems will occur in Writer.

The best way to insert an image is as follows:

- Click the **Insert** tab.
- Click **Image**.



- Browse for the required image file, select it and click **Open**.

Note: if you copy-pasted the image, save the image as a file first, and then follow the steps described above.

19.7.4 How to automatically update a model after changing an inserted fragment?

- Open the Hierarchy navigator to check the dependencies.
- Publish all linked models.
- Right-click your selection and click **Model options > Publish**. Publishes are updated, models aren't.

19.7.5 How do I add text at the beginning of the page when there is a fragment placed at that position?

It is impossible to add text in front of a fragment that is placed on the first paragraph of a page. The best way to handle this situation is to place an enter at the top of your page before adding the fragment. If the fragment was already inserted, follow the steps below:

- Remove the fragment.
- Add your text or place an enter.
- Reinsert your fragment.

19.8 Conditions

19.8.1 Why can't I select a function in the condition designer?

19.8.2 Why is my condition not working in other language models?

19.8.3 What if my condition does not work as expected?

19.8.4 Why am I unable to add a condition where bullet texts are used?

19.8.5 How do I apply a condition to all rows in a table?

19.8.1 Why can't I select a function in the condition designer?

- Check if the format of the variable is correct for the function.
- Sketch won't allow you to select conditions of a different type (e.g. strings and integers).

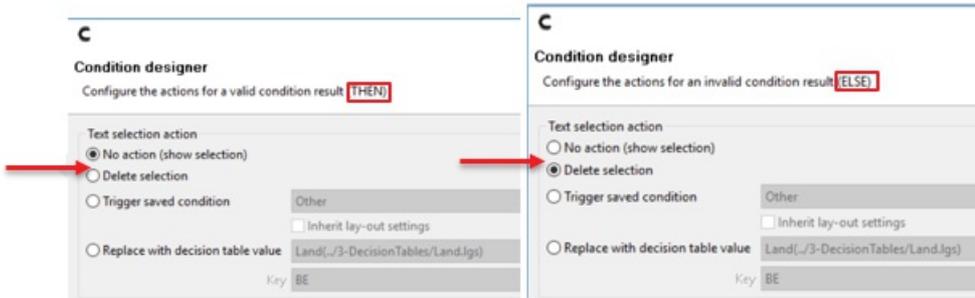
19.8.2 Why is my condition not working in other language models?

Check if the condition has not been copy-pasted to this position.

- If so, all variables/conditions/reusables/loops will have to be recreated. The logic will appear as images, considering the links from the variables/conditions/reusables/loops cannot be copied.
- If this is not the case and the model is copied from another language model, the links of the variables/conditions/reusables/loops are copied correctly.
- If both language models already exist, a condition from the first language model can be saved and reused in the other language model (as saved condition).

19.8.3 What if my condition does not work as expected?

- Check the used logic.
- Check in the condition designer if your configuration is correct for the condition.



19.8.4 Why am I unable to add a condition where bullet texts are used?

It is impossible to add something before the bullet, this is standard LibreOffice (the underlying word processor) behavior.

To solve this issue:

- First add the condition to the rows.
- Then bullet the text.

19.8.5 How do I apply a condition to all rows in a table?

- Select the entire table.
- Add the condition. The first condition symbol should now contain a 'T' (for 'Table'), which means the condition is applied to the entire table. 

19.9 Calculated variables

19.9.1 What are calculated variables?

19.9.2 How can I calculate values that are not defined in the XSD?

19.9.3 How do I create a calculated variable?

19.9.4 For what types of calculations can calculated variables be used?

19.9.1 What are calculated variables?

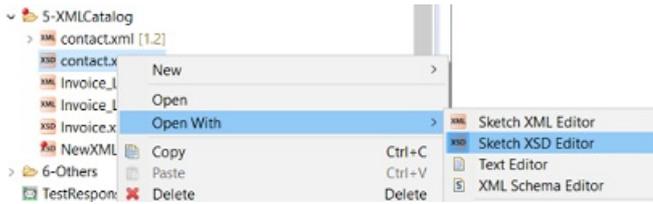
A calculated variable executes a function on an existing variable that has been defined in the XSD. Example: based on the variable 'Cost Price per year', the calculated variable 'Cost Price per month' can be calculated, by dividing the price per year by 12.

19.9.2 How can I calculate values that are not defined in the XSD?

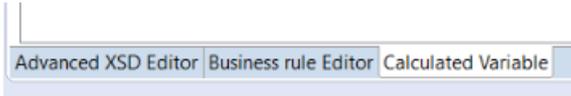
You can insert calculated variables. These variables have been defined in the data structure (XSD) and will be replaced by a calculated value at generation time or after a preview.

19.9.3 How do I create a calculated variable?

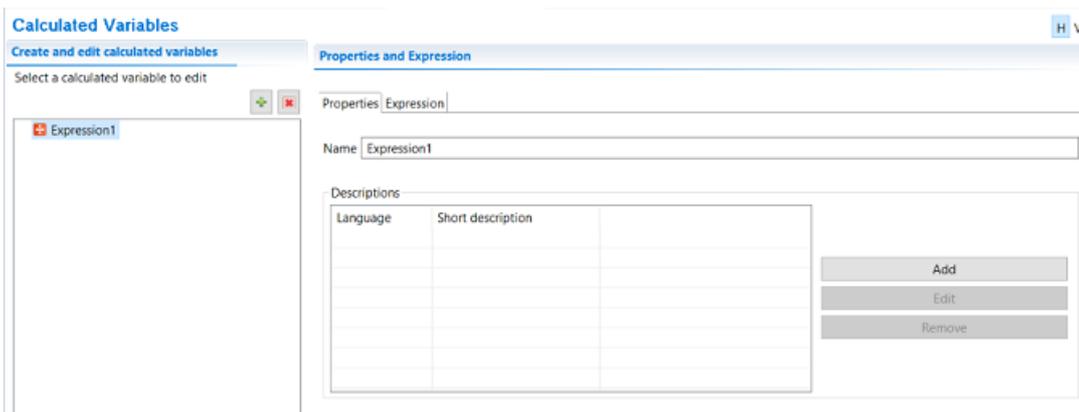
- Check out the XSD.
- Right-click and select **Open With > Sketch XSD Editor**.



- The XSD is opened in the Editor region.
- Open the **Calculated Variable** tab at the bottom of the window.

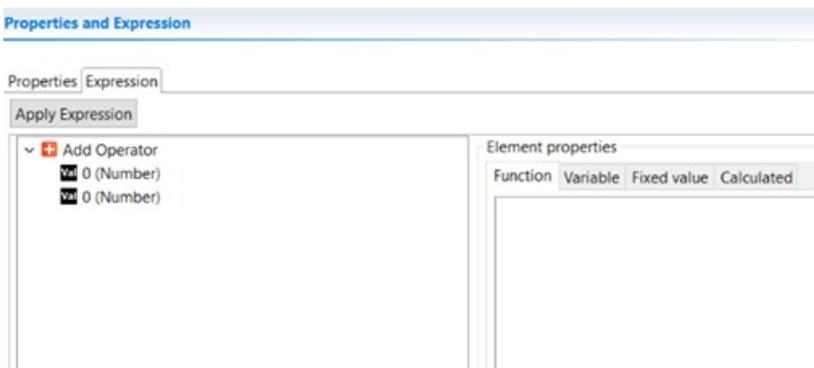


- Select the green '+' button.
- A new variable is added, named "Expression1".
- Select this variable.
- In the right pane, the **Property Section** tab is opened where you can give a name to the calculated variable and add descriptions in different languages.



- When you open the **Expression** tab, you can add the expression for the calculated variable.
- Select **Apply Expression** to finish.

The calculated variable has now been created. Save by doing a check-in of the XSD file.



19.9.4 For what types of calculations can calculated variables be used?

Calculated variables can only be used for simple, straightforward calculations, meaning:

- No combinations of operators (+, -, *, ...).
- No combinations of calculated variables (e.g. no calculated variable derived from a calculated variable).
- Calculations of iterations in loops are possible.

19.10 Preview

19.10.1 Why are my condition tags visible in my preview?

19.10.2 Why do I have a discrepancy between my preview in Sketch and the actual published model?

19.10.3 Why won't my template generate/publish?

19.10.4 Why do I see things in my preview that are not in my logical model?

19.10.5 Why is the automatic numbering of cross-references incorrect in the Sketch preview?

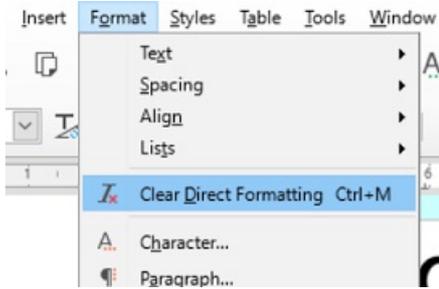
19.10.6 Why is there a discrepancy between the bullets in my preview in Sketch and the bullets in the actual published model?

19.10.1 Why are my condition tags visible in my preview?

They are probably linked to the layout.

To solve this issue:

- Select the texts and condition frames.
- Click **Format > Clear Direct Formatting**.



In case this shouldn't work:

- Select the complete conditional section including the open and close brackets.
- Right-click and click **Styles > Default Character** in the context menu.
- Reapply the desired character formatting.

19.10.2 Why do I have a discrepancy between my preview in Sketch and the actual published model?

- Check the date of the published model in production. Is this the date you expect? (Date/Time of your publish)
- Is it possible to validate the production XML? Are the resulting values what you expect for your condition?
- Is it possible to use the XML from production in the Sketch preview? This way you can try to reproduce the scenario in your Sketch preview.

19.10.3 Why won't my template generate/publish?

This issue might have the following causes:

- Internet connection isn't working.
- The linked XSD/XML is not correct.
- The model is checked out by someone else (and the ModelStore is not refreshed, so it was not visible that it was checked out).
- The Model Key is not filled in in the Publish properties.
- The wrong language definition was selected when creating the language model. This results in a discrepancy between the asked language and the created language model.

19.10.4 Why do I see things in my preview that are not in my logical model?

Check if your logical model is up-to-date. When a language model is opened, the fragments are not automatically updated. When Sketch generates a preview it will search for the most recent information, so you will see the last version of the model. It will be the version that becomes published.

19.10.5 Why is the automatic numbering of cross-references incorrect in the Sketch preview?

A normal preview is shown in read-only mode and the LibreOffice fields are not updated. This is a limitation in the application because we choose to show the document in read-only via LibreOffice. If you preview the file by saving it to the disk (via PDF, DocX, ...) or when you generate the model, the expected fields will be shown for these reference fields.

19.10.6 Why is there a discrepancy between the bullets in my preview in Sketch and the bullets in the actual published model?

Numbers/bullets should be created with the in-application functionality. Manually created bullet/number lists will be dependent on tabs/margins of the local settings and might result in a discrepancy between the local and the generated output.

19.11 PublishStore

19.11.1 How can I publish more than one model at a time?

19.11.2 How do I delete a PublishStore?

19.11.3 What do I do when the following error message is displayed: Publish properties has not been set for the model?

19.11.1 How can I publish more than one model at a time?

- Select all the models you want to publish. **Tip:** use Ctrl+click to do so.
- Right-click and select **Model actions > Publish**.
- The selected models will now be published.

19.11.2 How do I delete a PublishStore?

A PublishStore must first be deleted in Sketch, and then in the Workspace.

Inside Sketch:

- Select the PublishStore you want to delete.
- Click the delete icon .

In Workspace:

- Open 'C:\User\USERNAME\S sketch\WorkspaceX.X
- Delete the corresponding PublishStore folder.

19.11.3 What do I do when the following error message is displayed: Publish properties has not been set for the model?

- Check if all Publish properties have been completed.
- Verify your Model Key, the PublishStore and the Unit. Are the data you filled in correct?

19.12 Layout

19.12.1 Why can't I see new fonts in Sketch/publishes?

19.12.2 Can I use HTML parameters for the generation layout of documents?

19.12.3 Why is my superscript removed when it is converted to HTML?

19.12.4 Is it possible that Sketch removes trailing spaces in the data?

19.12.1 Why can't I see new fonts in Sketch/publishes?

- New fonts should be available both locally for Sketch and on the server itself for the generation.
- After the installation of the new fonts, the Server should be restarted to make sure the service can find the new fonts.

19.12.2 Can I use HTML parameters for the generation layout of documents?

- These parameters are not supported in Sketch. The HTML markers will be placed as HTML tags in the text.
- There is no alternative or workaround for controllable layout. The generator expects a LibreOffice document for which the layout is designed in Sketch.

19.12.3 Why is my superscript removed when it is converted to HTML?

Superscript is not supported when converting from doc to html. Not all possibilities in documents are similar in HTML.

19.12.4 Is it possible that Sketch removes trailing spaces in the data?

There is no regular expression to remove leading or trailing spaces because it might also remove other wanted whitespaces. Delivered data will not be adapted by Sketch.

19.13 Dynamic image

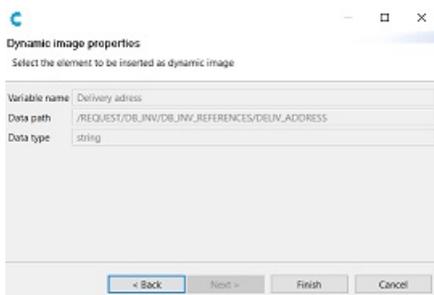
19.13.1 What is a dynamic image?

A dynamic image is in fact a variable image. The size of the image is fixed at design time when you insert the image, but the content of the image is variable and is determined at generation time. The variable image corresponds to a variable in the data structure. This variable should contain the full path to the image that needs to be inserted at generation time.

Example of a path: <SIGNATURE>file:///d:/DynImages/WWilson.jpg</SIGNATURE>

To insert a dynamic image:

- Place the **cursor** in the document where you want to insert the image.
- Right-click the **variable** from the data structure.
- Select **Insert as dynamic image**.
- Click **Finish**.



A placeholder appears where the cursor was positioned.



If needed, you can modify most of the image settings, like size, position, add a border etc. This placeholder will be replaced by the actual image at generation time.

19.13.2 How can I size a dynamic image?

- A dynamic image can be sized as a normal image by double clicking the image.
- The size of a dynamic image can be adjusted by dragging the corners to the desired position.

19.14 Sketch E-mail Editor

19.14.1 What if my Sketch E-mail editor is not working?

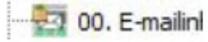
19.14.2 What if I get an error message while trying to open an email in Sketch?

19.14.1 What if my Sketch E-mail editor is not working?

- Check if the file is checked in by the original author.
- Check if your Internet Explorer version is installed and up to date.

19.14.2 What if I get an error message while trying to open an email in Sketch?

If the error message is: "Error retrieving content description for resource /ModelStore/XXX/1-MasterDocuments/email.cem", the Email model is probably locked.



When creating a new model, a placeholder will be created to avoid creation of models with the same name in that location. As long as the model has not been checked in, it is only available on the local workspace of the creator. Trying to retrieve this model will result in the error mentioned above. Placeholders can be recognized by the padlock added to the model's icon.

ConnectR REST API Documentation

[Description](#)

[Authorization](#)

[Translation File Access](#)

[List Templates](#)

[Template Details](#)

[List Email Templates](#)

[Email Template Details](#)

[Create Request](#)

[Get Information about Process](#)

[List Documents](#)

[Get Document](#)

[List Tasks](#)

[Get Task File](#)

[Delete Request](#)

[XML Schema for Generation Request](#)

[XML Schema for Decision Table Structure](#)

[XML Schema for Language Definitions](#)

[XML Schema for Template List / Details](#)

Revisions

| DATE | OWNER | TOPIC |
|------------|-------|---|
| 2015-09-16 | RAM | Initial version |
| 2015-09-21 | RAM | Included Decision Table schema structure for reference. Optional "name" attribute included in FileData response. (To give more info in GetTemplateDetails response) |
| 2015-10-16 | BJA | Overhauled request methods, added more return type schema structures. |
| 2016-09-30 | SJA | Labelled for version 5.0 |

Description

The ConnectR REST API allows the caller to create and remotely manage generation requests through the use of HTTP calls.

This document explains the available URLs and call formats.

All descriptions show a request method together with a request URI but this is relative to the API REST service URL. For example, if the API REST service URL is <http://connectr.example.org:8080/connectr/api/> and the description is "GET /translations/{path}", the final URL should be <http://connectr.example.org:8080/connectr/api/translations/MS/T/titles.lgs>.

Authorization

All the service requests should include an Authorization header for the HTTP Basic authentication scheme.

When no such header is given, the service will respond with status code **401 Unauthorized** including a challenge for the HTTP Basic authentication and a random realm code. This realm code is generated each time the backend server is restarted.

The server will respond with status code **403 Forbidden** if the user in the Authorization header is not using the right username / password combination or is not allowed access to the service.

Translation File Access

Sketch allows to define “decision tables” to map a given ID to a translated text. The REST API allows to query for this file to use its translations somewhere else. **GET /connectr/api/translations/{path}**

Example Request URL: <http://connectr.example.org:8080/connectr/api/translations/Models/Contoso/0-MainDepartment/3-DecisionTables/Amounttype.lgs>

| TEMPLATE PARAMETER | VALUE | DESCRIPTION |
|------------------------|------------------------|---|
| path
(Mandatory) | String | Relative path of the Decision Table file in the ModelStore. Use Sketch’s “Properties” view to find the “location” property, then use everything after the first /modelstore/ part of that field.

Example: /ModelStore/Development/0-MainDepartment/3-DecisionTables/Amount type.lgs |
| RESPONSE STATUS CODE | CONTENT TYPE | DESCRIPTION |
| 200 OK | application/xml | Request completed successfully, translations in response. See appendix “XML Schema for Decision Table Structure”. |
| 400 Bad request | text/plain | Request specified a file which doesn’t look like a Decision Table. |
| 404 Not found | text/plain | Requested file could not be found. |

Response example

```
<sl:selectionList xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:sl="http://namespace.lettergen.be/letterGen/1.0/selectionList.xsd">
  <keyValuePairs>
    <keyValuePair>
      <key>1</key>
      <value>
        <translation lang_iso="en-US">one</translation>
        <translation lang_iso="fr-FR">un</translation>
      </value>
    </keyValuePair>
    <keyValuePair>
      <key>2</key>
      <value>
        <translation lang_iso="en-US">two</translation>
        <translation lang_iso="fr-FR">deux</translation>
      </value>
    </keyValuePair>
    <keyValuePair>
      <key>3</key>
      <value>
        <translation lang_iso="en-US">three</translation>
        <translation lang_iso="fr-FR">trois</translation>
      </value>
    </keyValuePair>
  </keyValuePairs>
  <description>
    <translation lang_iso="en-US">TestDecisionTable</translation>
  </description>
</sl:selectionList>
```

List Templates

Gets the list of published templates available in the specified unit. For each available template the model.xml is looked up and appended to a large XML **GET /connectr/api/units/{unitId}/templates**

Example Request URL: <http://connectr.example.org:8080/connectr/api/units/4.1/templates?path=/Publish/Contoso/HR>

| TEMPLATE PARAMETER | | VALUE | DESCRIPTION |
|------------------------|------------------------|---|--|
| unit (Mandatory) | | String | Unit code as used in Data Manager. This is currently defined in the Publish Store. |
| QUERY PARAMETER | VALUE | DESCRIPTION | |
| path (Mandatory) | String | Relative path of a Unit in the Publish Store. Use Sketch's "Properties" view to find the "location" property, then use everything after the first /modelstore/ part of that field. Might be optional in the future.
Example: /Publish/Contoso/HR | |
| RESPONSE STATUS CODE | CONTENT TYPE | DESCRIPTION | |
| 200 OK | application/xml | Request completed successfully, templates list in response. See appendix "XML Schema for Template List / Details", this call returns the "Models" element. | |
| 400 Bad request | text/plain | Path is too short or contains illegal characters. | |
| 404 Not found | text/plain | The unit could not be found. | |

Response example

```
<Models StageName="PUB_dev-01" UnitName="Test" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="model.xsd">
  <Model IsPackage="false">
    <Model-Uri>http://dev-01.connective.local:15220/modelstore/PublishStore/PUB_dev-
01/Test/Template_7.3_3.3.0</Model-Uri>
    <XSD-Form-Uri/>
    <LetterWriter-Uri/>
    <Key>Template_7.3_3.3.0</Key>
    <LanguageModel LanguageCode="en-US"/>
    <Description Code="en-GB">Template 7.3</Description>
    <Categories/>
    <SearchKeys/>
    <MetaData>
      <LGM-Uri>http://localhost:15220/modelstore/ModelStore/Contoso/Test/Template_7.3_3.3.0.lgm</LGM-
Uri>
      <LGM-Version>1.1</LGM-Version>
      <LGM-CheckedIn-UserID>usertest1</LGM-CheckedIn-UserID>
      <Published-UserID>usertest1</Published-UserID>
      <LSClient-Version>5.0.0</LSClient-Version>
    </MetaData>
  </Model>
  <Model IsPackage="false">
    <Model-Uri>http://dev-01.connective.local:15220/modelstore/PublishStore/PUB_dev-
01/Test/Template_7.3_3.2.12</Model-Uri>
    <XSD-Form-Uri/>
    <LetterWriter-Uri/>
    <Key>Template_7.3_3.2.12</Key>
    <LanguageModel LanguageCode="en-GB"/>
    <LanguageModel LanguageCode="nl-BE"/>
    <LanguageModel LanguageCode="fr-BE"/>
    <Description Code="en-GB">Template 7.3 US</Description>
    <Categories/>
    <SearchKeys/>
    <MetaData>
      <LGM-Uri>http://localhost:15220/modelstore/ModelStore/Contoso/Test/Template_7.3_3.2.12.lgm</LGM-
Uri>
      <LGM-Version>1.3</LGM-Version>
      <LGM-CheckedIn-UserID>usertest1</LGM-CheckedIn-UserID>
      <Published-UserID>usertest1</Published-UserID>
      <LSClient-Version>5.0.0</LSClient-Version>
    </MetaData>
  </Model>
</Models>
```

Template Details

Gets the details of the specified template file (basically the model.xml as seen in the Publish Store) and inlines the XSD of that template.

GET /connectr/api/units/{unit}/templates/{template}

Example Request URL:

http://connectr.example.org:8080/connectr/api/units/4.1/templates/Template_7.3_3.3.0?path=/Publish/Contoso/HR

| TEMPLATE PARAMETER | VALUE | DESCRIPTION |
|----------------------|------------------------|---|
| unit (Mandatory) | String | Unit code as used in Data Manager. This is currently defined in the Publish Store settings. |
| template (Mandatory) | String | Template name as seen in the Key field of the Template List XML. |
| QUERY PARAMETER | VALUE | DESCRIPTION |
| path (Mandatory) | String | Relative path of a Unit in the Publish Store. Use Sketch's "Properties" view to find the "location" property, then use everything after the first /modelstore/ part of that field. Might be optional in the future.
Example: /Publish/Contoso/HR |
| RESPONSE STATUS CODE | CONTENT TYPE | DESCRIPTION |
| 200 OK | application/xml | Request completed successfully, template details in response. See appendix "XML Schema for Template List / Details", this resource returns the "Model" element. |
| 400 Bad | text/plain | Path is too short or contains illegal characters. |
| 404 Not found | text/plain | The unit or template could not be found. |

Response example

```

<Model xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" IsPackage="false"
xsi:noNamespaceSchemaLocation="model.xsd">
  <Model-Uri>http://dev-01.connective.local:15220/modelstore/PublishStore/PUB_dev-
01/Test/Template_7.3_3.3.0</Model-Uri>
  <XSD-Form-Uri/>
  <LetterWriter-Uri/>
  <Key>Template_7.3_3.3.0</Key>
  <LanguageModel LanguageCode="en-US"/>
  <Description Code="en-GB">Template 7.3</Description>
  <Description Code="en-GB">Template 7.3</Description>
  <Categories/>
  <SearchKeys/>
  <MetaData>
    <LGM-Uri>http://localhost:15220/modelstore/ModelStore/Contoso/Test/Template_7.3_3.3.0.lgm</LGM-Uri>
    <LGM-Version>1.1</LGM-Version>
    <LGM-CheckedIn-UserID>usertest1</LGM-CheckedIn-UserID>
    <Published-UserID>usertest1</Published-UserID>
    <LSClient-Version>5.0.0</LSClient-Version>
  </MetaData>
  <XSD-Name>TestSketch.xsd</XSD-Name>
  <XSD-Contents>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xs:element name="Connective" type="TP_Connective">
        <xs:annotation/>
      </xs:element>
      <xs:complexType name="TP_Connective">
        <xs:all>
          <xs:element minOccurs="0" name="L_Datablock_A" type="TP_L_Datablock_A">
            </xs:element>
          <xs:element minOccurs="1" name="Datablock_B" type="TP_Datablock_B">
            </xs:element>
        </xs:all>
      </xs:complexType>
      <xs:complexType name="TP_L_Datablock_A">
        <xs:sequence>
          <xs:element maxOccurs="unbounded" minOccurs="0" name="Repeating_DB_A" type="xs:string">
            </xs:element>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="TP_Datablock_B">
        <xs:all>
          <xs:element minOccurs="1" name="StringField" type="xs:string"/>
          <xs:element minOccurs="1" name="DateTimeField" type="xs:dateTime"/>
          <xs:element minOccurs="1" name="DateField" type="xs:date"/>
          <xs:element minOccurs="1" name="NumberField1" type="xs:decimal"/>
          <xs:element minOccurs="1" name="NumberField2" type="xs:decimal"/>
          <xs:element minOccurs="1" name="NumberField3" type="xs:decimal"/>
          <xs:element minOccurs="1" name="NumberField4" type="xs:decimal"/>
          <xs:element minOccurs="1" name="BooleanField1" type="xs:boolean"/>
          <xs:element minOccurs="1" name="BooleanField2" type="xs:boolean"/>
          <xs:element minOccurs="1" name="IntegerField" type="xs:int"/>
          <xs:element minOccurs="1" name="DoubleField" type="xs:double"/>
        </xs:all>
      </xs:complexType>
    </xs:schema>
  </XSD-Contents>
</Model>

```

List Email Templates

Gets the list of published email templates available in the specified unit. For each available template the Model.xml is looked up and appended to a large XML.

GET /connectr/api/units/{unitId}/emailtemplates

Example Request URL: <http://connectr.example.org:8080/connectr/api/units/4.1/emailtemplates?path=/Publish/Contoso/HR>

| TEMPLATE PARAMETER | VALUE | DESCRIPTION |
|----------------------|------------------------|---|
| unit (Mandatory) | string | Unit code as used in Data Manager. This is currently defined in the Publish Store settings. |
| QUERY PARAMETER | VALUE | DESCRIPTION |
| path (Mandatory) | String | Relative path of a Unit in the Publish Store. Use Sketch's "Properties" view to find the "location" property, then use everything after the first /modelstore/ part of that field. Might be optional in the future. |
| RESPONSE STATUS CODE | CONTENT TYPE | DESCRIPTION |
| 200 OK | application/xml | Request completed successfully, templates list in response. See appendix "XML Schema for Template List / Details", this call returns the "Models" element. |
| Bad request | text/plain | Path is too short or contains illegal characters. |
| 404 Not found | text/plain | The unit could not be found. |

Response example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EmailModels StageName="PUB_dev-01" UnitName="Test" >
  <EmailModel IsPackage="false">
    <Email-Model-Uri>http://dev-01.connective.local:15220/modelstore/PublishStore/PUB_dev-
01/Test/Template_7.3_3.3.0</Email-Model-Uri>
    <XSD-Form-Uri></XSD-Form-Uri>
    <Key>Template_7.3_3.3.0</Key>
    <LanguageEmailModel LanguageCode="nl-BE"/>
    <Description Code="en-US">Template_7.3_3.3.0</Description>
    <Categories/>
    <SearchKeys/>
    <MetaData>
      <CEM-Uri>http://localhost:15220/modelstore/ModelStore/Contoso/Test/Template_7.3_3.3.0.cem</CEM-
Uri>
      <CEM-Version>1.2</CEM-Version>
      <CEM-CheckedIn-UserID>usertest1</CEM-CheckedIn-UserID>
      <Published-UserID>usertest1</Published-UserID>
      <Published-Comment></Published-Comment>
      <LSClient-Version>6.0.0</LSClient-Version>
    </MetaData>
  </EmailModel>
  <EmailModel IsPackage="false">
    <Email-Model-Uri>http://dev-01.connective.local:15220/modelstore/PublishStore/PUB_dev-
01/Test/Template_7.3_3.2.12</Email-Model-Uri>
    <XSD-Form-Uri></XSD-Form-Uri>
    <Key>Template_7.3_3.2.12</Key>
    <LanguageEmailModel LanguageCode="nl-NL"/>
    <Description Code="en-US">Template_7.3_3.2.12</Description>
    <Categories/>
    <SearchKeys/>
    <MetaData>
      <CEM-Uri>http://localhost:15220/modelstore/ModelStore/Contoso/Test/Template_7.3_3.2.12.cem</CEM-
Uri>
      <CEM-Version>1.2</CEM-Version>
      <CEM-CheckedIn-UserID>usertest1</CEM-CheckedIn-UserID>
      <Published-UserID>usertest1</Published-UserID>
      <Published-Comment></Published-Comment>
      <LSClient-Version>6.0.0</LSClient-Version>
    </MetaData>
  </EmailModel>
</EmailModels>
```

Email Template Details

Gets the details of the specified template file (basically the model.xml as seen in the Publish Store) and inlines the XSD of that template.

GET /connectr/api/units/{unit}/emailtemplates/{template}

Example Request URL: http://connectr.example.org:8080/connectr/api/units/4.1/emailtemplates/Template_7.3_3.3.0?path=/Publish/Contoso/HR

| TEMPLATE PARAMETER | VALUE | DESCRIPTION |
|------------------------|------------------------|--|
| unit (Mandatory) | String | Unit code as used in Data Manager. This is currently defined in the Publish Store settings. |
| template (Mandatory) | String | Template name as seen in the Key field of the Template List XML. |
| QUERY PARAMETER | VALUE | DESCRIPTION |
| path (Mandatory) | String | Relative path of a Unit in the Publish Store. Use Sketch's "Properties" view to find the "location" property, then use everything after the first /modelstore/ part of that field. Might be optional in the future.

Example: /Publish/Contoso/HR |
| RESPONSE STATUS CODE | CONTENT TYPE | DESCRIPTION |
| 200 OK | application/xml | Request completed successfully, template details in response. See appendix "XML Schema for Template List / Details", this resource returns the "Model" element. |
| 400 Bad request | text/plain | Path is too short or contains illegal characters. |
| 404 Not found | text/plain | The unit or template could not be found. |

Response example

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<EmailModel IsPackage="false">
  <Email-Model-Uri> http://dev-01.connective.local:15220/modelstore/PublishStore/PUB_dev-
01/Test/Template_7.3_3.2.12</Email-Model-Uri>
  <XSD-Form-Uri></XSD-Form-Uri>
  <Key>Template_7.3_3.2.12</Key>
  <LanguageEmailModel LanguageCode="nl-BE"/>
  <Description Code="en-US">Template_7.3_3.2.12</Description>
  <Categories/>
  <SearchKeys/>
  <MetaData>
    <CEM-Uri> http://localhost:15220/modelstore/ModelStore/Contoso/Test/Template_7.3_3.2.12.cem</CEM-Uri>
    <CEM-Version>1.2</CEM-Version>
    <CEM-CheckedIn-UserID>usertest1</CEM-CheckedIn-UserID>
    <Published-UserID>usertest1</Published-UserID>
    <Published-Comment></Published-Comment>
    <LSClient-Version>6.0.0</LSClient-Version>
  </MetaData>
  <XSD-Name>TestSketch.xsd</XSD-Name>
  <XSD-Contents>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
      <xs:element name="Connective" type="TP_Connective">
        <xs:annotation/>
      </xs:element>
      <xs:complexType name="TP_Connective">
        <xs:all>
          <xs:element minOccurs="0" name="L_Datablock_A" type="TP_L_Datablock_A">
          </xs:element>
          <xs:element minOccurs="1" name="Datablock_B" type="TP_Datablock_B">
          </xs:element>
        </xs:all>
      </xs:complexType>
      <xs:complexType name="TP_L_Datablock_A">
        <xs:sequence>
          <xs:element maxOccurs="unbounded" minOccurs="0" name="Repeating_DB_A" type="xs:string">
          </xs:element>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="TP_Datablock_B">
        <xs:all>
          <xs:element minOccurs="1" name="StringField" type="xs:string"/>
          <xs:element minOccurs="1" name="DateTimeField" type="xs:dateTime"/>
          <xs:element minOccurs="1" name="DateField" type="xs:date"/>
          <xs:element minOccurs="1" name="NumberField1" type="xs:decimal"/>
          <xs:element minOccurs="1" name="NumberField2" type="xs:decimal"/>
          <xs:element minOccurs="1" name="NumberField3" type="xs:decimal"/>
          <xs:element minOccurs="1" name="NumberField4" type="xs:decimal"/>
          <xs:element minOccurs="1" name="BooleanField1" type="xs:boolean"/>
          <xs:element minOccurs="1" name="BooleanField2" type="xs:boolean"/>
          <xs:element minOccurs="1" name="IntegerField" type="xs:int"/>
          <xs:element minOccurs="1" name="DoubleField" type="xs:double"/>
        </xs:all>
      </xs:complexType>
    </xs:schema>
  </XSD-Contents>
</EmailModel>

```

Create request

Add a new request and optionally start processing it.

A process will only be started if the ACTION and FLOW parameters are filled in.

POST /connectr/api/units/{unit}/requests

Example Request URL: <http://connectr.example.org:8080/connectr/api/units/4.1/requests>

| TEMPLATE PARAMETER | VALUE | DESCRIPTION |
|------------------------|------------------------|--|
| unit (Mandatory) | String | Unit code as used in Data Manager. This is currently defined in the Publish Store settings. |
| REQUEST ENTITY | CONTENT TYPE | DESCRIPTION |
| Request XML | String | A generation request XML. See appendix "XML Schema for Generation Request" for the format. |
| RESPONSE STATUS CODE | CONTENT TYPE | DESCRIPTION |
| 200 OK | application/xml | Request created successfully. See below for the format. |
| 400 Bad request | text/plain | The XML is not well-formed, the METADATA fields might be incorrect or the file could not be saved. The response entity has more details. |
| 404 Not found | text/plain | The unit could not be found. |

Response format

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0">
  <xs:element name="RequestInfo" type="RequestInfoResponse"/>
  <xs:complexType name="RequestInfoResponse">
    <xs:sequence>
      <xs:element name="ext" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:any processContents="lax" namespace="##other" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="requestGuid" type="xs:string"/>
      <xs:element name="processStarted" type="xs:boolean"/>
      <xs:element name="creationDate" type="xs:dateTime"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Example

Suppose we send the following request body:

```

<LETTERGEN>
  <METADATA>
    <REQID>Test_Template_7.3</REQID>
    <GUID/>
    <!-- Following fields are necessary for ConnectR. -->
    <STAGE>4</STAGE>
    <UNIT>4.1</UNIT>
    <FLOW>PDF</FLOW>
    <MODEL>Template_7.3_3.3.0</MODEL>
    <LANGUAGE>en-US</LANGUAGE>
    <ACTION>GENERATE</ACTION>
    <DESCRIPTION>Test_Template_7.3</DESCRIPTION>
    <!-- Must be true for a package generation. -->
    <ISPACKAGE>>true</ISPACKAGE>
    <ISBATCH>>false</ISBATCH>
  </METADATA>
  <DATA>
    <Connective>
      <L_Datablock_A>
        <Repeating_DB_A>Repeat 1</Repeating_DB_A>
        <Repeating_DB_A>Repeat 2</Repeating_DB_A>
        <Repeating_DB_A>Repeat 3</Repeating_DB_A>
      </L_Datablock_A>
      <Datablock_B>
        <StringField>StringField</StringField>
        <DateTimeField>2001-12-31T12:00:00</DateTimeField>
        <DateField>2014-10-01</DateField>
        <NumberField1>0.0</NumberField1>
        <NumberField2>0.0</NumberField2>
        <NumberField3>0.0</NumberField3>
        <NumberField4>0.0</NumberField4>
        <BooleanField1>1</BooleanField1>
        <BooleanField2>>true</BooleanField2>
        <IntegerField>0</IntegerField>
        <DoubleField>0.0</DoubleField>
      </Datablock_B>
    </Connective>
  </DATA>
</LETTERGEN>

```

What a potential response might look like if the response status is 200 OK:

```

<RequestInfo>
  <ext/>
  <requestGuid>5037a3b5-7ec5-4281-89a7-1b352c9bfb95</requestGuid>
  <processStarted>true</processStarted>
  <creationDate>2015-10-22T14:21:40.495Z</creationDate>
</RequestInfo>

```

Get Information about Process

Retrieves more information about the state of a process for a given request.

GET /connectr/api/units/{unit}/requests/{requestguid}

Example Request URL: <http://connectr.example.org:8080/connectr/api/units/4.1/requests/5037a3b5-7ec5-4281-89a7-1b352c9bfb95>

| TEMPLATE PARAMETER | VALUE | DESCRIPTION |
|-------------------------|--------|---|
| unit (Mandatory) | String | Unit code as used in Data Manager. This is currently defined in the Publish Store settings. |
| requestguid (Mandatory) | String | Request guid as seen in the output of the request creation. |

| RESPONSE STATUS CODE | CONTENT TYPE | DESCRIPTION |
|----------------------|------------------------|---|
| 200 OK | application/xml | Retrieval of the request info was successful. See below for the format. |
| 404 Not found | text/plain | The unit or request could not be found. |

Response format

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="ProcessInfo" type="ProcessInfo"/>

  <xs:complexType name="ProcessInfo">
    <xs:sequence>
      <xs:element name="status" type="xs:string"/>
      <xs:element name="flow" type="xs:string"/>
      <xs:element name="primaryDocumentGuid" type="xs:string" minOccurs="0"/>
      <xs:element name="owner" type="xs:string"/>
      <xs:element name="isPackage" type="xs:boolean"/>
      <xs:element name="startDate" type="xs:dateTime"/>
      <xs:element name="lastModificationDate" type="xs:dateTime"/>
      <xs:element name="ext" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:any processContents="lax" namespace="##other" minOccurs="0" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Response example

```
<ProcessInfo>
  <status>GENERATION_FINISHED</status>
  <flow>PDF</flow>
  <isPackage>true</isPackage>
  <startDate>2015-10-22T14:21:40.497Z</startDate>
  <lastModificationDate>2015-10-22T14:21:43.293Z</lastModificationDate>
  <ext/>
</ProcessInfo>
```

Response example with error message returned (available when configured in the workflow)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProcessInfo>
  <status>GENERATION_FAILED</status>
  <flow>PDF</flow>
  <isPackage>>false</isPackage>
  <startDate>2018-07-25T14:17:28.887Z</startDate>
  <lastModificationDate>2018-07-25T14:17:29.793Z</lastModificationDate>
  <ext>
    <FLOW xmlns:ns2="be.lettergen.RequestMetaData" xmlns:ns3="http://www.lettergen.com/datastore"
xmlns:xmime="http://www.w3.org/2005/05/xmlmime">
      <FILES/>
      <ERROR component="DirMon">be.btr.lettergen.exception.ProcessingException: Failed to generate
document ; nested exception is com.lettergen.generatorextension.v3.GeneratorServiceException: Failed to get
input data - Source URI:http://localhost:15220/modelstore/ModelStore/Publish/UnitName/myDocument/nl-
NL/compiled.lgp | TimeStamp:
1532528249230http://localhost:15220/modelstore/ModelStore/Publish/UnitName/myDocument/nl-
NL/compiled.lgp</ERROR>
    </FLOW>
  </ext>
</ProcessInfo>
```

List Documents

Retrieves a list of all documents associated with a given request.

GET /connectr/api/units/{unit}/requests/{requestguid}/documents

Example Request URL:

<http://connectr.example.org:8080/connectr/api/units/4.1/requests/f4dc0da49ad14e768f84afb953807e04>

| TEMPLATE PARAMETER | VALUE | DESCRIPTION |
|-------------------------|------------------------|--|
| unit (Mandatory) | String | Unit code as used in Data Manager. This is currently defined in the Publish Store settings. |
| requestguid (Mandatory) | String | Request guid as seen in the output of the request creation. |
| RESPONSE STATUS CODE | CONTENT TYPE | DESCRIPTION |
| 200 OK | application/xml | Retrieval of the document list was successful. See below for the format. |
| 404 Not found | text/plain | The unit or request could not be found. |
| 409 Conflict | text/plain | The process associated with the request has not finished yet so the list cannot be built at this time. |

Response format

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0">

  <xs:element name="DocumentList" type="DocumentListResponse"/>
  <xs:complexType name="DocumentListResponse">
    <xs:sequence>
      <xs:element name="ext" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:any processContents="lax" namespace="##other" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="documentInfo" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="documentInfo" type="DocumentInfo" nillable="true" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="DocumentInfo">
    <xs:sequence>
      <xs:element name="documentGuid" type="xs:string"/>
      <xs:element name="infoItems" type="InfoItem" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="InfoItem">
    <xs:sequence>
      <xs:element name="type" type="docInfoType"/>
      <xs:element name="value" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>

  <xs:simpleType name="docInfoType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="CONTENTTYPE"/>
      <xs:enumeration value="MODELTYPE"/>
      <xs:enumeration value="DESCRIPTION"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>

```

Response example

```
<DocumentList>
  <ext/>
  <documentInfo>
    <documentInfo>
      <documentGuid>5aabfa2f-7a8d-4fed-99b4-83807e4a1c85</documentGuid>
      <infoItems>
        <type>DESCRIPTION</type>
        <value>Template_7.3_3.3.0</value>
      </infoItems>
      <infoItems>
        <type>CONTENTTYPE</type>
        <value>application/pdf</value>
      </infoItems>
      <infoItems>
        <type>MODELTYPE</type>
      </infoItems>
    </documentInfo>
  </documentInfo>
</DocumentList>
```

Get Document

Retrieve the document for the specified document guid within a given request.

GET /connectr/api/units/{unit}/requests/{requestguid}/documents/{documentguid}

Example Request URL:

<http://connectr.example.org:8080/connectr/api/units/4.1/requests/f4dc0da49ad14e768f84afb953807e04/documents/35dc0da49aj14e768f84afb953825h5>

| TEMPLATE PARAMETER | VALUE | DESCRIPTION |
|--------------------------|--------------|---|
| unit (Mandatory) | String | Unit code as used in Data Manager. This is currently defined in the Publish Store settings. |
| requestguid (Mandatory) | String | Request guid as seen in the output of the request creation. |
| documentguid (Mandatory) | String | Document guid as seen in the output of the document list call or in the primaryDocumentGuid field of the process info output. |
| RESPONSE STATUS CODE | CONTENT TYPE | DESCRIPTION |
| 200 OK | | Retrieval of the document was successful. |
| 404 Not found | text/plain | The unit, request or document could not be found. |
| 409 Conflict | text/plain | The process associated with the request is in a state which forbids retrieving the document (e.g. a user is editing it). |

No example is given here because the resulting document is simply returned as a stream of bytes.

List Tasks

Retrieves a list of all outstanding tasks for a request. This can only be used if the process status is set to **WAITING**. Also note that the request's FLOW parameter in the METADATA needs to be set to an agreed-upon value to mark that these tasks are not to be handled through DataManager.

GET /connectr/api/units/{unit}/requests/{requestguid}/tasks

Example Request URL:

<http://connectr.example.org:8080/connectr/api/units/4.1/requests/f4dc0da49ad14e768f84afb953807e04/tasks>

| TEMPLATE PARAMETER | VALUE | DESCRIPTION |
|-------------------------|--------|---|
| unit (Mandatory) | String | Unit code as used in Data Manager. This is currently defined in the Publish Store settings. |
| requestguid (Mandatory) | String | Request guid as seen in the output of the request creation. |

| RESPONSE STATUS CODE | CONTENT TYPE | DESCRIPTION |
|----------------------|------------------------|---|
| 200 OK | application/xml | Retrieval of the task list was successful. See below for the format. |
| 400 Not found | text/plain | The unit or request could not be found. |
| 409 Conflict | text/plain | The process associated with the request was not in status waiting or the FLOW code was not valid. |

Response format

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0">

  <xs:element name="TaskList" type="TaskListResponse"/>

  <xs:complexType name="TaskListResponse">
    <xs:sequence>
      <xs:element minOccurs="0" name="ext">
        <xs:complexType>
          <xs:sequence>
            <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##other"
processContents="lax"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element minOccurs="0" name="Tasks">
        <xs:complexType>
          <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="TaskInfo" type="TaskInfo"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="TaskInfo">
    <xs:sequence>
      <xs:element minOccurs="0" name="ext">
        <xs:complexType>
          <xs:sequence>
            <xs:any maxOccurs="unbounded" minOccurs="0" namespace="##other"
processContents="lax"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element minOccurs="0" name="TaskGuid" type="xs:string"/>
      <xs:element minOccurs="0" name="RequestGuid" type="xs:string"/>
      <xs:element minOccurs="0" name="DocumentGuid" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>

```

Response example

```

<?xml version="1.0"?>
<TaskList>
  <ext/>
  <Tasks>
    <TaskInfo>
      <ext/>
      <TaskGuid>84aef09f-7301-48bf-af06-e7d3ccf95acc</TaskGuid>
      <RequestGuid>08c69e27-e0de-4656-8f8a-4749e98ce069</RequestGuid>
      <DocumentGuid>a16cfef8-f338-411d-a65c-3eb140bdbbc83</DocumentGuid>
    </TaskInfo>
  </Tasks>
</TaskList>

```

Get Task File

Retrieves a task file (LDE) to be opened by the Writer client on the user's computer. This can only be used if the process status is set to **WAITING**. Note that this file cannot be cached as the user might save his progress, in which case this file needs to be retrieved from this service again to continue with the changes. Also note that the request's FLOW parameter in the METADATA needs to be set to an agreed-upon value to mark that these tasks are not to be handled through DataManager.

GET /connectr/api/units/{unit}/requests/{requestguid}/tasks/{taskguid}/lde

| TEMPLATE PARAMETER | VALUE | DESCRIPTION |
|-------------------------|--------------------------|---|
| unit (Mandatory) | String | Unit code as used in Data Manager. This is currently defined in the Publish Store settings. |
| requestguid (Mandatory) | String | Request guid as seen in the output of the request creation. |
| taskguid (Mandatory) | String | Document guid as seen in the output of the document list call or in the primaryDocumentGuid field of the process info output. |
| RESPONSE STATUS CODE | CONTENT TYPE | DESCRIPTION |
| 200 OK | application/x-lde | Retrieval of the task file was successful. |
| 404 Not found | text/plain | The unit, request or task could not be found. |
| 409 Conflict | text/plain | The process associated with the request was not in status waiting or the FLOW code was not valid. |

Delete Request

Deletes a request, including its documents and process state. Delete will not return **404 Not Found** error if the request guid doesn't exist; instead it will return "MISSING" instead of "DELETED".

DELETE /connectr/api/units/{unit}/requests/{requestguid}

Example Request URL: <http://connectr.example.org:8080/connectr/api/units/4.1/requests/f4dc0da49ad14e768f84afb953807e04>

| TEMPLATE PARAMETER | VALUE | DESCRIPTION |
|-------------------------|------------------------|---|
| unit (Mandatory) | String | Unit code as used in Data Manager. This is currently defined in the Publish Store settings. |
| requestguid (Mandatory) | String | Request guid as seen in the output of the request creation. |
| QUERY PARAMETER | VALUE | DESCRIPTION |
| forceddelete (Optional) | Boolean | When set to true the deletion will be forceful: the current status is ignored and any open tasks are simply deleted. When false or missing a 409 Conflict status will be returned if there were tasks open. |
| RESPONSE STATUS CODE | CONTENT TYPE | DESCRIPTION |
| 200 OK | application/xml | The request was deleted or missing. |
| 409 Conflict | text/plain | The process associated with the request is in a state which forbids retrieving the document (e.g. a user is editing it). |

Response format

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0">
  <xs:element name="DeleteResponse" type="DeleteResponse"/>
  <xs:complexType name="DeleteResponse">
    <xs:sequence>
      <xs:element name="resultCode" type="xs:string" minOccurs="0"/>
      <xs:element name="ext" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:any processContents="lax" namespace="##other" minOccurs="0"
maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

Response example

```
<DeleteResponse>
  <resultCode>DELETED</resultCode>
  <ext/>
</DeleteResponse>
```

XML Schema for Generation Request

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="LETTERGEN">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="METADATA" type="METADATA" minOccurs="0" />
        <xs:element name="DATA" type="requestData" minOccurs="0" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="requestData" mixed="true">
    <xs:sequence>
      <xs:any processContents="skip" namespace="##any" minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>

  <!-- This element occurs under /LETTERGEN/METADATA/PROCESSINFO -->
  <xs:element name="FLOWINFO" type="flowInfo" minOccurs="0" />

  <xs:complexType name="METADATA">
    <xs:all>
      <xs:element name="REQID" type="xs:string" minOccurs="0"/>
      <xs:element name="GUID" type="xs:string"/>
      <xs:element name="UNIT" type="xs:string"/>
      <xs:element name="MODEL" type="xs:string"/>
      <xs:element name="STAGE" type="xs:int"/>
      <xs:element name="LANGUAGE" type="xs:string"/>
      <xs:element name="USER" type="xs:string"/>
      <xs:element name="FLOW" type="xs:string" />
      <xs:element name="ACTION">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="GENERATE"/>
            <xs:enumeration value="PRINT"/>
            <xs:enumeration value="POSTPROCESS"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:element>
      <xs:element name="URI_MODEL" type="xs:string" />
      <xs:element name="DESCRIPTION" type="xs:string" />
      <xs:element name="ISPACKAGE" type="xs:boolean" />
      <xs:element name="ISBATCH" type="xs:boolean" />
      <xs:element name="URI_RESULT" type="xs:string" minOccurs="0"/>
      <xs:element name="BATCH" type="BATCH" minOccurs="0"/>
      <xs:element name="COPYTITLES" type="COPYTITLES" minOccurs="0"/>
      <xs:element name="PROCESSINFO" type="PROCESSINFO" minOccurs="0"/>
    </xs:all>
  </xs:complexType>

  <xs:complexType name="PROCESSINFO">
    <xs:sequence>
      <xs:any minOccurs="0" maxOccurs="unbounded" processContents="lax" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="BATCH">
    <xs:all>
      <xs:element name="GUID" type="xs:string"/>
      <xs:element name="NUMBER" type="xs:int"/>
      <xs:element name="TOTAL" type="xs:int"/>
    </xs:all>
  </xs:complexType>

```

```

    </xs:all>
</xs:complexType>

<xs:complexType name="COPYTTITLES">
  <xs:sequence>
    <xs:element name="COPYYTITLE" type="COPYYTITLE" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="COPYYTITLE">
  <xs:all>
    <xs:element name="NAME" type="xs:string" />
    <xs:element name="NUMBER" type="xs:int" />
  </xs:all>
</xs:complexType>

<xs:complexType name="flowInfo">
  <xs:sequence>
    <xs:element name="FLOWDATA" type="flowData" minOccurs="0" />

    <xs:element name="FILES" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="FILE" type="fileInfo" minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="ERROR" type="flowError" minOccurs="0" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="flowData" mixed="true">
  <xs:sequence>
    <xs:any processContents="lax" namespace="##any" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="fileInfo">
  <xs:sequence>
    <!-- reserved for future property sub-elements -->
    <xs:any processContents="lax" namespace="##any" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="uri" type="xs:string" use="required" />
  <xs:attribute name="key" type="xs:string" use="required" />
  <xs:attribute name="ext" type="xs:string" />
  <xs:attribute name="description" type="xs:string" />
  <xs:attribute name="model" type="xs:string" />
</xs:complexType>

<xs:complexType name="flowError" mixed="true">
  <xs:sequence>
    <xs:any processContents="lax" namespace="##any" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="component" type="xs:string" />
</xs:complexType>
</xs:schema>

```

XML Schema for Decision Table Structure

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema xmlns:langIso="http://namespaces.lettergen.be/LetterGen/1.0/lang_iso.xsd"
xmlns:s1="http://namespace.lettergen.be/letterGen/1.0/selectionList.xsd"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" ecore:documentRoot="SelectionListSchema" ecore:nsPrefix="s1"
ecore:package="be.lettergen.sketch.selectionList"
targetNamespace="http://namespace.lettergen.be/letterGen/1.0/selectionList.xsd">
  <!-- import and re-use global iso language code -->
  <xsd:import namespace="http://namespaces.lettergen.be/LetterGen/1.0/lang_iso.xsd"
schemaLocation="../../be.lettergen.sketch.lang_iso/model/lang_iso.xsd"/>

  <!-- The top level element. -->
  <xsd:element name="selectionList" type="s1:selectionListType"/>

  <!-- The type definition for the top level element. -->
  <xsd:complexType ecore:name="SelectionList" name="selectionListType">
    <xsd:sequence>
      <xsd:element name="keyValuePairs" type="s1:keyValuePairsType"/>
      <xsd:element name="name" type="s1:translationsType"/>
      <xsd:element name="description" type="s1:translationsType"/>
    </xsd:sequence>
    <xsd:attribute name="version" type="xsd:string"/>
  </xsd:complexType>

  <!--The type used to add translations for other types (like name and description and key value pair).-->
  <xsd:complexType ecore:name="Translations" name="translationsType">
    <xsd:sequence>
      <xsd:choice ecore:featureMap="group" maxOccurs="unbounded">
        <xsd:element ecore:lowerBound="1" ecore:upperBound="-1" minOccurs="0" name="translation"
type="s1:translationType"/>
      </xsd:choice>
    </xsd:sequence>
    <xsd:attribute name="default" type="xsd:string"/>
  </xsd:complexType>

  <!-- The type to represent one translation. -->
  <xsd:complexType ecore:name="Translation" name="translationType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute ecore:name="langIso" name="lang_iso" type="langIso:lang_isoType"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <!-- The type to represent list of key value pair. -->
  <xsd:complexType ecore:name="KeyValuePairs" name="keyValuePairsType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="keyValuePair" type="s1:keyValuePairType"/>
    </xsd:sequence>
  </xsd:complexType>

  <!-- The type to represent key value pair with default and lang specific translation. -->
  <xsd:complexType ecore:name="KeyValuePair" name="keyValuePairType">
    <xsd:sequence>
      <xsd:element name="key" type="xsd:string"/>
      <xsd:element name="value" type="s1:translationsType" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

</xsd:schema>
```

XML Schema for Language Definitions

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore"
xmlns:langIso="http://namespaces.lettergen.be/LetterGen/1.0/lang_iso.xsd"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" ecore:nsPrefix="langIso"
ecore:package="be.lettergen.sketch.lang_iso"
targetNamespace="http://namespaces.lettergen.be/LetterGen/1.0/lang_iso.xsd">
  <xsd:simpleType ecore:name="LangIsoType" name="lang_isoType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration ecore:name="enUS" value="en-US"/>
      <xsd:enumeration ecore:name="enGB" value="en-GB"/>
      <xsd:enumeration ecore:name="nlBE" value="nl-BE"/>
      <xsd:enumeration ecore:name="frFR" value="fr-FR"/>
      <xsd:enumeration ecore:name="itIT" value="it-IT"/>
      <xsd:enumeration ecore:name="ptPT" value="pt-PT"/>
      <xsd:enumeration ecore:name="esES" value="es-ES"/>
      <xsd:enumeration ecore:name="trTR" value="tr-TR"/>
      <xsd:enumeration ecore:name="skSK" value="sk-SK"/>
      <xsd:enumeration ecore:name="deDE" value="de-DE"/>
      <xsd:enumeration ecore:name="csCZ" value="cs-CZ"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

XML Schema for Template List / Details

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="Models">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Model" type="modelType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="StageName" type="xs:string"/>
      <xs:attribute name="UnitName" type="xs:string"/>
    </xs:complexType>
  </xs:element>

  <!-- Model with all details and XSD (if template uses one) -->
  <xs:element name="Model" type="modelAndXSDType"/>

  <xs:complexType name="modelType">
    <xs:sequence>
      <xs:element name="Model-Uri">
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="XSD-Form-Uri">
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="LetterWriter-Uri">
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="Key">
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="LanguageModel" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="LanguageCode" type="xs:string"/>
        </xs:complexType>
      </xs:element>
      <xs:element ref="Description" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute name="Code" type="xs:string"/>
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>
      <xs:element name="Categories">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="Category" minOccurs="0" maxOccurs="unbounded">
              <xs:simpleType>
                <xs:restriction base="xs:string"/>
              </xs:simpleType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

```

```

</xs:element>
<xs:element name="SearchKeys">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="SearchKey" minOccurs="0" maxOccurs="unbounded">
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="MetaData">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="LGM-Uri">
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="LGM-Version">
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="LGM-CheckedIn-UserID">
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="Published-UserID">
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="Published-DateTime" minOccurs="0">
        <xs:simpleType>
          <xs:restriction base="xs:dateTime"/>
        </xs:simpleType>
      </xs:element>
      <xs:element name="LSClient-Version">
        <xs:simpleType>
          <xs:restriction base="xs:string"/>
        </xs:simpleType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="IsPackage" type="xs:boolean" use="optional"/>
</xs:complexType>

<xs:complexType name="modelAndXSDType">
  <xs:complexContent>
    <xs:extension base="modelType">
      <xs:sequence>
        <xs:element name="XSD-Name" type="xs:string" minOccurs="0"/>
        <xs:element name="XSD-Contents" minOccurs="0">
          <xs:complexType>
            <xs:sequence>
              <xs:any namespace="http://www.w3.org/2001/XMLSchema"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexType>

```

```
</xs:complexContent>  
</xs:complexType>  
</xs:schema>
```