Learning object creation

This manual explains how to create learning objects and learning paths. We provide concrete examples of how a learning object can be created. These examples are marked in green.

Definitions

Learning object: A learning object is a unit of didactical information with its corresponding metadata which has the option of being recursively defined.

This definition is broad and allows for many different types of content, with different granularities to be defined as a learning object. We consciously chose this broad definition because the structure of the information that can be identified as a learning object is highly dependant on the context. For example, in a course about construction, we can have a learning object explaining how to build walls that are at a right angle. This learning object can contain an image illustrating the 3-4-5 rule and some information about how to apply it. On the other hand, in a course about the Pythagorean theorem, there can be a learning object explaining the Pythagorean theorem and using the image of the 3-4-5 rule as an example application of the theorem. Since the image can be used on its own, it can also be defined as a learning object. In this case, it can be reused by integrating it into other learning objects.

Learning path: A learning path is the definition of a logical progression between learning objects. It can be represented as a graph in which the nodes are learning objects and the edges are possible transitions between the nodes in the graph.

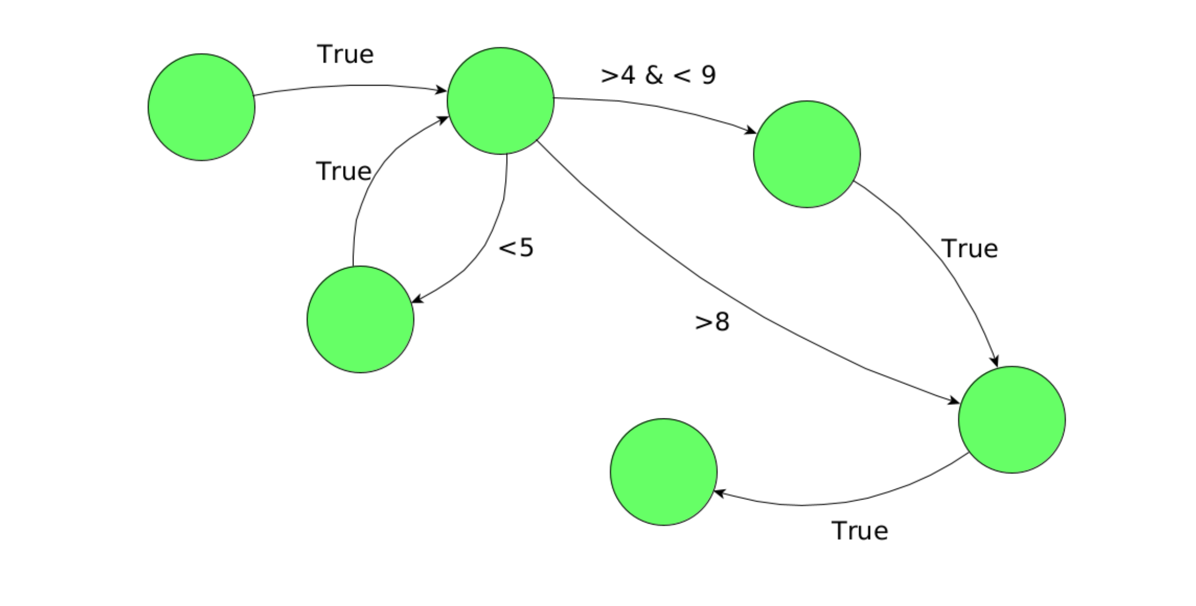

Figure 1: Visual representation of a learning path. Green nodes represent learning objects. Edges have transition conditions that are checked against the outcome of the source learning object.

Defining a learning object

A learning object consists of two parts: (1) Metadata associated with the learning object. (2) The content of the learning object itself. Both metadata and content have to adhere to some rules to be valid.

In the following sections we first describe the metadata format, thereafter we provide examples of how different types of content can be added as a learning object.

Learning object metadata

Learning object metadata is specified in YAML format (https://yaml.org/). YAML uses key value pairs to define the different types of metadata associated with the learning object. In the table below we give and overview of the different keys and their possible values.

Name (*=required)	**Type**	**Description**
hruid*	string	Human readable unique id. This identifier is used to uniquely identify the learning object in combination with its version and language. When creating new learning objects it is advised to use a name associated with your institution. For example: *org-dwengo-3-4-5-rule*
version*	int	The version of that specific learning object. Unique in combination with hruid and language.
language*	string	The language of the learning object.
The items above (marked in green) are used together to uniquely identify a learning object.		
title*	string	Short description of the learning object.
description*	string	Long description of the learning object.
content_type*	string	One of: “text/markdown”, “image/image”, “audio/mpeg”, “application/pdf”, “extern”, “blockly” This is how the content is added to our learning object repository, it does not determine the format for the user of the API. API users always get the content in HTML format.
keywords	array<string>	Descriptive keywords related to the learning object. We do not define a vocabulary for the keywords. They are used when a user searches for learning objects. Consequently, providing as many relevant keywords as possible is advised.
target_ages	array<int>	List of target ages for the learning objects
teacher_exclusive	boolean	If the learning object is for teachers only. Learning objects can be created by educators and contain

		information specifically for teachers and not students. This field allows you to specify if the learning object is only for teachers.
skos_concepts*	array<string>	List of uri string referring to skos metadata concepts (https://ilearn.ilabt.imec.be/Skosmos/nl/)
educational_goals	array<Object>	Depending on the context, combination of the source of the educational goals (string), for example a government website and a unique id for the educational goal in that context (string).
copyright	string	Copyright (ex. CC BY)
licence	string	licence (ex. GPL)
difficulty	number	Difficulty on scale of one to five
estimated_time*	number	Estimated time in minutes
return_value	Object	Used for defining a return value. - callback_url: The url to send the response of the learning object to. - callback_schema: A json schema defining the format of the return value.
available	boolean	If the object is available or not
content_location	string	Only contains relevant information if the content_type “extern”. “extern” content types refer to a website hosted somewhere online. In this case the content_location attribute contains the link to that external content.

Below you can see an example of a metadata definition in YAML format.

```
hruid: org-dwengo-3-4-5-rule
version: 1
language: en
title: “Image illustrating the 3-4-5 rule.”
description: This is a description
keywords: [“3-4-5-rule”, “Pythagoras”]
educational_goals: [
   {source: “https://onderwijsdoelen.be/”, id: 17},
   {source: “https://onderwijsdoelen.be/”, id: 12}
]
copyright: CC by dwengo vzw
licence: GPL
content_type: extern
available: true
target_ages: [4, 3]
difficulty: 3
```

```
return_value: {
    callback_url: callback-url-example,
    callback_schema: {
        att: test,
        att2: test2
    }
}
content_location: "https://kiks.ilabt.imec.be/jupyterhub/?id=1"
estimated_time: 50
skos_concepts: [
    'http://ilearn.ilabt.imec.be/vocab/curr1/c-andere-talen',
    'http://ilearn.ilabt.imec.be/vocab/ondniv/sec-gr2-doorstroom-aso'
]
teacher_exclusive: true
```

Learning object content

To define a learning object all we need to do is create a folder on your computer containing one of the following files: **index.md, metadata.md, or metadata.yaml**. Once the folder contains **one and only one** of these files, it can be processed as a learning object. When to use which file will become clear from the examples below.

For now learning objects with the following content types are supported: “text/markdown”, “image/image”, “audio/mpeg”, “application/pdf”, “extern”, “blockly”

Markdown

The markdown data type is the main format for defining learning content. At the same time, it is the simplest method of defining a learning object since both metadata and content can be defined in the same file. Perform the following steps to create a markdown learning object:

- **Create a folder in which you will put all the files related to the learning object.**
- **Create an index.md file.**
- **Add the metadata of your learning object to the top index.md file in YAML format (notice the three dashes at the start and end of the metadata definition in the example below, these are required).**
- **Add your content to the same file, below the metadata definition, using markdown syntax. (https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet)**
- **If you have images or other content you want to embed into your markdown, you can add them to the folder of the learning object and refer to them from your markdown using a relative path.**

In the example below, the learning object folder contains a folder containing the file pyth_theorem.png.

index.md

```
---
hruid: org-dwengo-pythagoras-explained
version: 1
language: en
title: “Explanation of the Pythagorean theorem.”
description: “Explanation of the Pythagorean theorem”
keywords: [“Pythagoras”, “maths”, “mathematics”, “geometry”]
content_type: “text/markdown”
estimated_time: 50
skos_concepts: [
    'http://ilearn.ilabt.imec.be/vocab/curr1/c-andere-talen',
    'http://ilearn.ilabt.imec.be/vocab/ondniv/sec-gr2-doorstroom-aso'
]
---

# The Pythagorean theorem

In mathematics, the Pythagorean theorem, or Pythagoras' theorem, is a fundamental
relation in Euclidean geometry among the three sides of a right triangle. It states that the
area of the square whose side is the hypotenuse (the side opposite the right angle) is
equal to the sum of the areas of the squares on the other two sides. This theorem can be
written as an equation relating the lengths of the sides a, b and c, often called the
Pythagorean equation. [source](https://en.wikipedia.org/wiki/Pythagorean_theorem, “Link
to Wikipedia article about the Pythagorean theorem.)

![Illustration of the principle of the Pythagorean theorem](images/pyth_theorem.png
"Pythagorean theorem")
```

Aggregation

The markdown content type can be used to aggregate learning objects into a larger learning object. Learning objects can be integrated as links or as direct content. The syntax for integrating learning objects is as follows:

- **Link to a learning object**:
  [link tekst](@learning-object/blockly-test-1/nl/1)
- **Directly integrated content:**
  ![blockly](@learning-object/blockly-test-1/nl/1)

Referring to a learning object is done using the **@learning-object** directive followed by the **hruid**, **language**, and **version** of the learning object you want to link to separated by **slashes**.

Audio

To add an audio fragment as a learning object use the following steps:

- Create a folder for your learning object
- Add a metadata.md file with the YAML metadata
- Copy the audio fragment to the folder where your metadata.md file is located

Below you can see an example

metadata.md

```
---
hruid: org-dwengo-audio-example
version: 1
language: nl
title: “Example audio fragment”
description: “Example audio fragment as illustration of how to add audio as learning object”
keywords: [“voorbeeld”, “voorbeeld2”]
content_type: “audio/mpeg”
target_ages: [4, 3]
content_location: example-location
estimated_time: 20
skos_concepts: [
    'http://ilearn.ilabt.imec.be/vocab/curr1/c-andere-talen',
    'http://ilearn.ilabt.imec.be/vocab/ondniv/sec-gr2-doorstroom-aso'
]
---
```

Images

To add an image as a learning object use the following steps:

- Create a folder for your learning object
- Add a metadata.md file with the YAML metadata
- Copy the image to the folder where your metadata.md file is located

Below you can see an example of the metadata associated with an image of the Shazam logo.

metadata.md

```
---
hruid: org-dwengo-jommeke-images-shazam-logo
version: 1
language: nl
title: "Shazam"
description: "Shazam is een service die het mogelijk maakt om op basis van een geluidsopname het overeenkomstige nummer te herkennen."
keywords: ["Muziek", "AI-systeem", "artificiële intelligentie"]
content_type: "image/image"
estimated_time: 1
skos_concepts: [
    'http://ilearn.ilabt.imec.be/vocab/vak1/informatica-wetenschappen',
    'http://ilearn.ilabt.imec.be/vocab/curr1/s-computers-en-systemen',
    'http://ilearn.ilabt.imec.be/vocab/curr1/s-mediawijsheid',
    'http://ilearn.ilabt.imec.be/vocab/tref1/ict',
    'http://ilearn.ilabt.imec.be/vocab/curr1/c-digitale-competenties-en-mediawijsheid',

'http://ilearn.ilabt.imec.be/vocab/onddoel/sec-gr1-astroom-digitale-competenties-en-mediawijsheid-4.5',
]
copyright: "CC BY dwengo"
target_ages: [10, 11, 12, 13]
---
```

Remark: Images can also be added directly into markdown using classical markdown syntax. You can link an image using the relative path from the index.md file.

Pdf documents

To add a pdf document as a learning object use the following steps:

- Create a folder for your learning object
- Add a metadata.md file with the YAML metadata
- Copy the pdf document to the folder where your metadata.md file is located

Below you can see an example

metadata.md

```
---
hruid: org-dwengo-pdf-as-lo
version: 1
language: nl
title: "This is a titlee"
description: "This is a description"
keywords: ["voorbeeld", "voorbeeld2"]
copyright: CC BY dwengo"
content_type: "application/pdf"
target_ages: [4, 3]
difficulty: 3
estimated_time: 20
skos_concepts: [
    'http://ilearn.ilabt.imec.be/vocab/vak1/informatica-wetenschappen',
    'http://ilearn.ilabt.imec.be/vocab/curr1/s-computers-en-systemen',
    'http://ilearn.ilabt.imec.be/vocab/curr1/s-mediawijsheid',
    'http://ilearn.ilabt.imec.be/vocab/tref1/ict',
    'http://ilearn.ilabt.imec.be/vocab/curr1/c-digitale-competenties-en-mediawijsheid',

'http://ilearn.ilabt.imec.be/vocab/onddoel/sec-gr1-astroom-digitale-competenties-en-mediawijsheid-4.5',

]
---
```

External content

External content is all content not hosted by the learning object repository itself. This can be any other publicly available resource on the internet. An example are youtube videos.

To add an external link as a learning object use the following steps:

- Create a folder for your learning object
- Add a metadata.md file with the YAML metadata
- Add the *content_location* metadata to the metadata.md file

Below you can see an example

metadata.md

```
---
hruid: org-dwengo-jommeke-video-deep-fake
version: 6
language: nl
title: "Deep fake"
description: "AI-systemen op basis van diepe neurale netwerken zijn in staat om gezichten in video's te vervangen."
keywords: ["deep fake", "AI-systeem", "artificiële intelligentie"]
content_type: "extern"
estimated_time: 1
skos_concepts: [
    'http://ilearn.ilabt.imec.be/vocab/vak1/informatica-wetenschappen',
    'http://ilearn.ilabt.imec.be/vocab/curr1/s-computers-en-systemen',
    'http://ilearn.ilabt.imec.be/vocab/curr1/s-mediawijsheid',
    'http://ilearn.ilabt.imec.be/vocab/tref1/ict',
    'http://ilearn.ilabt.imec.be/vocab/curr1/c-digitale-competenties-en-mediawijsheid',

'http://ilearn.ilabt.imec.be/vocab/onddoel/sec-gr1-astroom-digitale-competenties-en-mediawijsheid-4.5',

]
copyright: "CC BY dwengo"
target_ages: [10, 11, 12, 13]
content_location: "https://www.youtube.com/embed/I8JC2R3sbsk"
---
```

Blockly xml

Learning objects support the visualisation of blockly code for the Dwenguino simulator (https://blockly-backend.dwengo.org/dwenguinoblockly/). To get an xml definition of a set of blocks you can go to the dwenguinoblockly simulator, create a program, and download the xml file using the *save blocks* button in the top right corner.

To add a blockly xml as a learning object use the following steps:

- Create a folder for your learning object
- Add a metadata.md file with the YAML metadata
- Copy the blockly xml file to the folder where your metadata.md file is located

Below you can see an example.

metadata.md

```
---
hruid: org-dwengo-blockly-hello-dwenguino
version: 1
language: nl
title: "Hello dwenguino on lcd"
description: "This program shows the text hello dwenguino on the lcd screen of the dwenguino microcontroller"
keywords: ["dwenguino", "blockly", "program", "code", "microcontroller", "lcd", "hello world"]
copyright: "CC BY dwengo"
content_type: blockly
target_ages: [10, 11, 12, 13]
difficulty: 2
estimated_time: 1
skos_concepts: [
   'http://ilearn.ilabt.imec.be/vocab/vak1/informatica-wetenschappen',
   'http://ilearn.ilabt.imec.be/vocab/curr1/s-computers-en-systemen',
   'http://ilearn.ilabt.imec.be/vocab/curr1/s-mediawijsheid',
   'http://ilearn.ilabt.imec.be/vocab/tref1/ict',
   'http://ilearn.ilabt.imec.be/vocab/curr1/c-digitale-competenties-en-mediawijsheid',

'http://ilearn.ilabt.imec.be/vocab/onddoel/sec-gr1-astroom-digitale-competenties-en-mediawijsheid-4.5',

]
---
```

Defining a learning path

Learning paths define a logical progression through a set of learning objects. Learning paths are defined using json. The json object for the learning path has to be validated by the following json schema:

learning path json schema

```
{
   "$id": "https://dwengo.org/learning_track.schema.json",
   "$schema": "http://json-schema.org/draft-07/schema",
   "title": "Schema for definition of learning track",
   "type": "object",
   "properties":{
      "language": {
         "title": "The language used in the learning track",
         "type": "string"
      },
      "title": {
            "title": "The title of the learning track",
            "type": "string"
        },
        "description": {
            "title": "The description of the learning track",
            "type": "string"
        },
        "image": {
            "title": "Base64 encoding of cover image",
            "type": "string"
        },
        "nodes": {
            "type": "array",
            "items": {
                "type": "object",
                "properties": {
                    "learningobject_id": {
                        "type": "string"
                    },
                    "language": {
                        "type": "string"
                    },
                    "version": {
                        "type": "string"
                    },
                    "instruction": {
                        "type": "string"
                    },
                    "start_node": {
                        "type": "boolean"
                    },
                    "transitions": {
                        "type": "array",
                        "items": {
                            "type": "object",
                            "properties": {
                                "condition": {
                                    "type": "string"
                                },
                                "next": {
                                    "type": "string"
                                },
                                "default": {
                                    "type": "boolean"
                                }
                            },
                            "required": ["next"]
                        }
                    }
                },
                "required": ["learningobject_id", "language", "version"]
            },
            "minItems": 1
        }
```

```
    },
    "required": ["language", "nodes"]
}
```

Implementation details on the transition condition still have to be worked out, however, this string will contain a jsonpath expression which is matched against the return value of the learning object. The transition condition will then check the result of this jsonpath expression against a certain condition.

Below you can find an example definition of a simple linear learning path:

learning path definition example

```
{
    "hruid": "jommeke-vooroordelen-van-ai",
    "language": "nl",
    "title": "De vooroordelen van AI-systemen",
    "description": "AI-systemen worden krachtiger en krachtiger, dit geeft de mogelijkheid
om krachtige systemen te ontwikkelen die bepaalde menselijke taken overnemen. Toch
zijn er ook gevaren bij het toepassen van AI-systemen. In dit leerpad zoomen we in op
een aatal beperkingen van AI-systemen en leggen uit waarom het belangrijk is om zich
bewust te zijn van deze problemen.",
    "image": "$base64_encoding_of_image",
    "nodes": [
      {
        "learningobject_hruid": "org-dwengo-jommeke-definitie-ai-systeem",
        "language": "nl",
        "version": "1",
        "start_node": true,
        "transitions": [
          {
            "default": true,
            "next": {
              "hruid": "org-dwengo-jommeke-voorbeelden-ai-systemen",
              "version": "1",
              "language": "nl"
            }
          }
        ]
      },
      {
        "learningobject_hruid": "org-dwengo-jommeke-voorbeelden-ai-systemen",
        "language": "nl",
        "version": "1",
        "transitions": [
          {
            "default": true,
            "next": {
```

```
      "hruid": "org-dwengo-jommeke-fouten-van-ai-systemen",
      "version": "1",
      "language": "nl"
     }
    }
   ]
  },
  {
   "learningobject_hruid": "org-dwengo-jommeke-fouten-van-ai-systemen",
   "language": "nl",
   "version": "1",
   "transitions": [
     {
       "default": true,
       "next":{
         "hruid": "test-pdf-as-lo",
         "version": "1",
         "language": "nl"
       }
     }
   ]
  },
  {
    "learningobject_hruid": "test-pdf-as-lo",
    "language": "nl",
    "version": "1"
  }
 ]
}
```