

Configuration Guide

Model

In the process of configuring a robust and versatile system for managing clinical visits, it is crucial to understand the architecture of the involved classes. This model is based on a clear and interconnected organization of four fundamental elements: **Visit Structure**, **Agendas**, **Types**, and **Viewers**.

Visit Structure is the core of the System, linking and coordinating the other components. A Visit Structure is associated with a specific *Type* that precisely defines the data structure used to represent *Clinical Visits*.

Agenda defines the clinical studies associated with each *Visit Structure*. This part of the system allows for the organization of visits based on specific clinical or research contexts, offering a structured and orderly view of the scheduled activities.

Type defines the data structure in order to accurately represent information related to Clinical Visits. Each *Visit Structure* is associated with a specific *Type* that determines the way of managing and storing such structured data, allowing a detailed and precise representation of the information collected during the clinical visit.

Lastly, **Viewers** are the tools used to view and interact with the data defined by the *Types*. These components provide an intuitive and customizable user interface to access and interpret information related to Clinical Visits, enabling users to efficiently and accurately view and analyze the stored data.

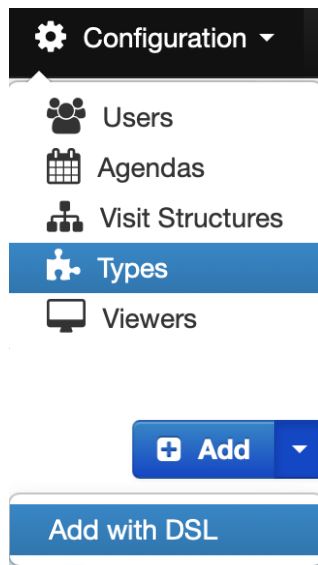
Types e Viewers are defined using a domain-specific language (DSL), a programming language that mimics the terms, idioms, and expressions used by experts in a particular domain.

These data structures can be defined through an interface that allows the insertion, modification, and deletion of *Types* and *Viewers*. When a *Type* or a *Viewer* is defined, it can be validated or previewed before saving, to ensure it is correct.

Viewers' templates consist of both DSL and CSS, to customize the display of data contained in the structures defined by *Types*. These templates can be configured in a manner similar to that described for the *Types*, but there are also scripts available for their automatic generation based on the *Types*.

The following sections illustrate how to insert each of these elements, starting from *Types* up to the *Visit Structure*, which will gather all the previously defined information, linking them together and defining the structure of the *Clinical Visit*.

Type Definition



The Type can be added by navigating to the "Configuration" tab, then "Types".

This will open a list of all the Types already present. A new Type can be added by going to "Add", then "Add with DSL".

At this point, the "Create Type" screen will open with two main fields: "Name" and "Definition".

The image shows a form titled 'Create Type'. In the top right corner, there are two buttons: 'Cancel' and 'Validate'. The form has two main input fields. The first field is labeled '* Name' and contains the text 'Dermatology Clinical Visit'. The second field is labeled '* Definition' and is a large, empty text area for defining the type.

Both fields are necessary for the definition of the Type, and the Name must be unique. To save a Type, it must first be validated by clicking on "Validate". If you try to validate a Type without specifying its definition, an error message will be displayed.

'Definition' is a required field!

Create Type

* Name

* Definition

By adding a simple definition (more detailed explanations will be provided below), you can see how the validation is successfully carried out, allowing the Type to be saved.

Validation successfully completed!

Create Type

* Name

* Definition

ct{
 "Visit Date" : dt
}

Within the "Definition" field, the structure of a type can be defined as if it were a tree. Below are the elements that can be defined:

- **ct composite type**, which can contain other elements inside. They can be nested and group other elements within them. The syntax with which they are defined is as follows: ("Nome_Composite" can also be omitted)

```
["Nome_Composite"] : ct {...}
```

- **qt quantitative type**, is defined by specifying two values: unit of measurement and a pair of values (integer, decimal). For example:

```
"Quante sigarette al giorno" : qt {"(Undefined)"(2,0)},  
"Da quanto ha smesso di fumare" : qt{"month"(2,0)}
```

- **ql qualitative type**, is defined by specifying a list of value strings and -optionally- "ordered". For example:

```
"Diagnosi" : ql ordered { "Common Nevus",  
                          "Atypical Nevus",
```

```
        "Melanoma",
        "Epithelioma"},
    "Smoking": ql ordered {"Yes", "No", "Former Smoker"},
```

- **dt** *temporal type*, used to define dates. For example:

```
"Visit Date" : dt
```

- **tx** *textual type*, used to specify textual fields. For example:

```
"Conclusions" : tx
```

Summarizing all the various fields and collecting them into a single composite type, a tree structure can then be obtained, as in the example Type below.

```
ct {
    "Visit Date" : dt,
    "Diagnosis" : ql ordered { "Common Nevus",
        "Atypical Nevus",
        "Melanoma",
        "Epithelioma"},
    "Personal History": ct {
        "Smoking": ql ordered {"Yes", "No", "Former Smoker"},
        "How many cigarettes per day" : qt
    },
    {"(Undefined)"(2,0)},
    "How long since quitting smoking" : qt{"month"(2,0)}
},
"Conclusions" : tx
}
```

Validation successfully completed!

Create Type

Cancel Validate Save

* Name Dermatology Clinical Visit

```
* Definition
ct {
    "Visit Date" : dt,
    "Diagnosis" : ql ordered { "Common Nevus",
        "Atypical Nevus",
        "Melanoma",
        "Epithelioma"},
    "Personal History": ct {
        "Smoking": ql ordered {"Yes", "No", "Former Smoker"},
        "How many cigarettes per day" : qt {"(Undefined)"(2,0)},
        "How long since quitting smoking" : qt{"month"(2,0)}
    },
    "Conclusions" : tx
}
```

Once saved, the "Dermatology Clinical Visit" Type will be viewable in the list of Types under Configuration->Types.

Type successfully saved!

Types

Add

Active Filters +

All3 results found

Composite

Enumerated

Quantitative

Name ^	Description	Recurrent	Actions
Dermatology Clinical Visit			<div></div> <div></div> <div></div>
HeightWeightBMI			<div></div> <div></div> <div></div>

By clicking on the magnifying glass, you can see the details of the Type, its structure, and the fields it contains.

Type Detail

EditClose

Name

Dermatology Clinical Visit

Categoria

Composito

Description

Validità

Sola lettura

Ricorrente

Struttura

Visit Date

Diagnosis

Personal History

Smoking

How many cigarettes per day

How long since quitting smoking

Conclusions

Viewer Definition

Once the Type is defined, it is possible to generate a basic structure of viewers, according to needs. Viewers are distinguished into "Viewer Output" or "Viewer Edit". "Viewer Output" does not allow for the specification of fields, they display what has already been written within the structure of the visit. On the other hand, "Viewer Edit" are used when the structure of the visit is being compiled, and each field will have its own appropriately associated element.

Configuration

Users

Agendas

Visit Structures

Types

Viewers

The Viewer can be added by navigating to the "Configuration" tab, then "Viewers".



After that, a list of all the existing *Viewers* will open. A new *Viewer* can be added by clicking on “Add”.

Create Viewer

Cancel Validate

* Name

Apply to

* Definition **CSS**

Creating a *Viewer* involves specifying a Name, a Definition, and a Type to apply it to. Once the Type of the *Viewer* is specified, the interface allows for the automatic generation of Edit or Output *Viewers*, and related CSS, via the “Generate” button.

Create Viewer

Cancel Validate

* Name

Apply to

Generate ▾

* Definition **CSS**

Generate Edit Viewer
Generate Output Viewer

Below, we can see the result applied to the *Viewer Edit* of the previously defined Type.

Create Viewer

Cancel Validate Preview Save

* Name Dermatology Clinical Visit - VIEWER_EDIT

Apply to Dermatology Clinical Visit

Generate

* Definition

CSS

```
"Dermatology Clinical Visit"
box {
  : label "Dermatology Clinical Visit"
  : grid {
    : grid spaced_horizontal {
      "Visit Date" : outputPath
      "Visit Date" : inputTemporal
    }

    : grid spaced_horizontal {
      "Diagnosis" : outputPath
      "Diagnosis" : combo
    }

    : box {
      : label "Personal History"
      "Personal History" : grid {
        : grid spaced_horizontal {
          "Smoking" : outputPath

```

A default definition of the Viewer's CSS is also provided, which can be further customized if necessary.

Create Viewer

Cancel Validate Preview Save

* Name Dermatology Clinical Visit - VIEWER_EDIT

Apply to Dermatology Clinical Visit

Generate

* Definition

CSS

```
.fieldset {background-color: #f9f9f9; margin: 5px;}
.fieldset .fieldset {background-color: white;}
.fieldset .fieldset .fieldset {background-color: #f9f9f9;}
.label_text {font-weight: bold;}
.outputPath_text {font-weight: bold;}
.grid_table_grid_cell_0 {width: 25%; text-align: right; padding-right: 15px; vertical-align: top;}
.grid_cell_1_grid_table {width: auto;}
.grid_cell_1_grid_table_grid_cell_0 {padding-right: 0px; width: auto;}
.grid_cell_1_grid_table_grid_cell_1 {padding-left: 6px;}
.box_label {font-weight: bold; background: transparent; background-image: -webkit-linear-gradient(bottom, #f9f9f9 50%, transparent 50%); padding-left: 6px; padding-right: 6px;}
.fieldset .fieldset .box_label {font-weight: bold; background: transparent; background-image: -webkit-linear-gradient(bottom, transparent 50%, #f9f9f9 50%); padding-left: 6px; padding-right: 6px;}
.fieldset .fieldset .fieldset .box_label {font-weight: bold; background: transparent; background-image: -webkit-linear-gradient(bottom, #f9f9f9 50%, transparent 50%); padding-left: 6px; padding-right: 6px;}
.fieldset {padding: 10px 0 3px 0;}
```

Using the “Preview” button at the top, it is also possible to see how the fields will be displayed, filled with random data.

Create Viewer

Cancel Validate Preview Save

* Name Dermatology Clinical Visit - VIEWER_EDIT

Apply to Dermatology Clinical Visit

Generate

* Definition CSS

```
"Dermatology Clinical Visit"
box {
  : label "Dermatology Clinical Visit"
  : grid {
    : grid spaced_horizontal {
      "Visit Date" : outputPath
      "Visit Date" : inputTemporal
    }

    : grid spaced_horizontal {
      "Diagnosis" : outputPath
      "Diagnosis" : combo
    }

    : box {
      : label "Personal History"
      "Personal History" : grid {
        : grid spaced_horizontal {
          "Smoking" : outputPath

```

Preview

Close

Dermatology Clinical Visit

Visit Date 20/02/2024

Diagnosis Atypical Nevus

Personal History

Smoking Former Smoker

How long since quitting smoking 25 month(s)

Conclusions cuafwtspmi

The structure of a Viewer is composed of *grid* and *box* elements nested within each other. In case of more extensive Clinical Folders, It is also possible to have a *tabbedPanel*, the outermost element, to specify a tab structure.

In the following example It is possible to see the **Viewer Edit** of the previously defined Type “Dermatology Clinical Visit”.

```
"Dermatology Clinical Visit"
box {
  : label "Dermatology Clinical Visit"
```

```

: grid {
  : grid spaced_horizontal {
    "Visit Date" : outputPath
    "Visit Date" : inputTemporal
  }

  : grid spaced_horizontal {
    "Diagnosis" : outputPath
    "Diagnosis" : combo
  }

  : box {
    : label "Personal History"
    "Personal History" : grid {
      : grid spaced_horizontal {
        "Smoking" : outputPath
        "Smoking" : combo
      }
      : conditionalPanel {
        "Smoking" : "Yes"
        clear "How many cigarettes per day"
        : grid spaced_horizontal {
          "How many cigarettes per day" : outputPath
          "How many cigarettes per day" : inputText
        }
      }
      : conditionalPanel {
        "Smoking" : "Former Smoker"
        clear "How long since quitting smoking"
        : grid spaced_horizontal {
          "How long since quitting smoking" : outputPath
          "How long since quitting smoking"
          : grid spaced_horizontal {
            : inputText
            : combo
          }
        }
      }
    }
  }

  : grid spaced_horizontal {
    "Conclusions" : outputPath
    "Conclusions" : inputText
  }
}
}

```

Below, the Viewer Output

```

"Dermatology Clinical Visit"
box collapse {
  : label "Dermatology Clinical Visit"

```



```

: grid collapse {
  : grid spaced_horizontal collapse {
    "Visit Date" : outputPath
    "Visit Date" : outputValue
  }

  : grid spaced_horizontal collapse {
    "Diagnosis" : outputPath
    "Diagnosis" : outputValue
  }

  : box collapse {
    : label "Personal History"
    "Personal History" : grid collapse {
      : grid spaced_horizontal collapse {
        "Smoking" : outputPath
        "Smoking" : outputValue
      }

      : conditionalPanel {
        "Smoking" : "Yes"
        clear "How many cigarettes per day"
        : grid spaced_horizontal {
          "How many cigarettes per day" : outputPath
          "How many cigarettes per day" : outputValue
        }
      }

      : conditionalPanel {
        "Smoking" : "Former Smoker"
        clear "How long since quitting smoking"
        : grid spaced_horizontal {
          "How long since quitting smoking" :
outputPath
          "How long since quitting smoking" :
grid spaced_horizontal {
: outputPath
: outputValue
}
}
}
}
}

: grid spaced_horizontal collapse {
  "Conclusions" : outputPath
  "Conclusions" : outputValue
}

}

}

```

The 'final' elements change from edit to output. If the viewer being constructed is an edit viewer, a corresponding element must be placed for the field that is to be filled out. Therefore, we will have:

- *textArea* / *inputText* if the specified field is of type :tx. TextArea is used to handle larger textual fields, such as the conclusions of a Clinical Visit.
- *combo* if it is of type :ql or :qt
- *inputTemporal* if it is of type :dt.

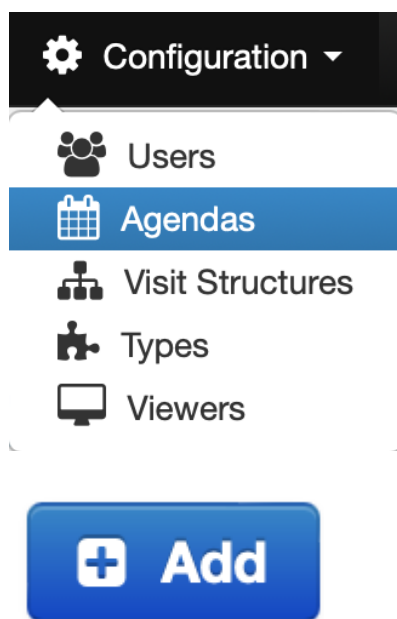
In the case of a Viewer Output, however, we will always have the following structure:

```
"Label": outputPath
"Value": outputValue
```

Unlike the auto-generated Viewers, the ones described previously specify **conditionalPanels**. *ConditionalPanels* are elements, used in combination with '**ql qualitative types**', that allow for the management of the appearance of specific forms when a particular value is entered in the qualitative type. Specifically, we can see in the provided example, that depending on the possible responses to the field "Smoking?" (Yes/Former Smoker/No), different fields appear:

- If the answer is Yes, it is required to specify the number of cigarettes per day, and thus the appropriate form appears: "How many cigarettes per day?", while "How long since quitting smoking" remains hidden and unfillable.
- If the answer is "Former Smoker", the opposite effect occurs: the form related to "How many cigarettes per day?" is hidden, and "How long since quitting smoking" appears instead.
- If the answer is 'No', there is no need to specify anything else, and both fields remain hidden.

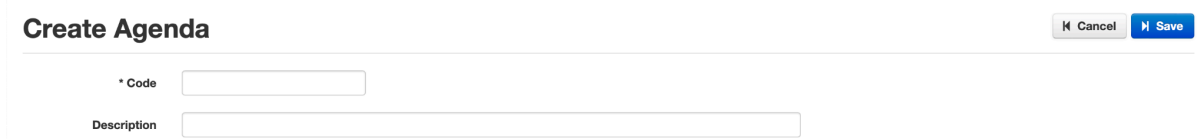
Agenda



Agenda can be added by navigating to the "Configuration" tab, then "Agendas".

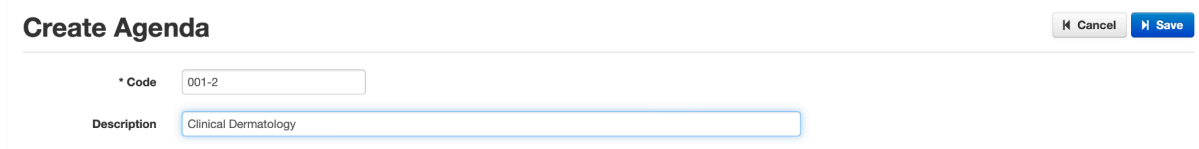
After that, a list of all the existing Agendas will open. A new Agenda can be added by clicking on "Aggiungi" ('Add').

This will open a “Create Agenda” window that will allow for the definition of an Agenda.

A screenshot of a web form titled "Create Agenda". At the top right, there are two buttons: "Cancel" and "Save". Below the title, there are two input fields. The first is labeled "* Code" and is empty. The second is labeled "Description" and is also empty.

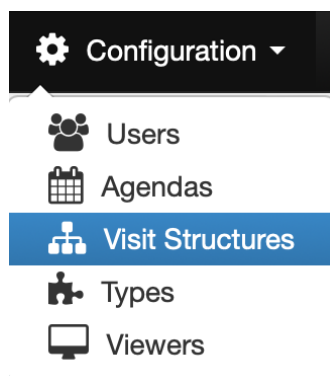
To define an Agenda, it is necessary to specify a code (unique) and a description.

We can fill in the fields as follows:

A screenshot of the "Create Agenda" form, now filled with data. The "* Code" field contains the text "001-2". The "Description" field contains the text "Clinical Dermatology". The "Cancel" and "Save" buttons remain at the top right.

Visit Structure

Once the Types, Viewers, and Agenda have been added, it is possible to gather all this information together by specifying a *Visit Structure*



A Visit Structure can be added by navigating to the "Configuration" tab, then “Visit Structure”.



After that, a list of all the existing *Visit Structures* will open. A new *Visit Structure* can be added by clicking on “Add”.

This will open the usual creation screen that we have already seen for the previous elements.

Create Visit Structure

Cancel Save

* Name	<input type="text"/>
Description	<div></div>
* Time to Live	<input type="text"/> hours
* Data Structure	<input type="text"/>
Agendas	+
Authorizations on Operations	+
Viewers associated with contexts and qualifications	+

To define a Visit Structure, it is necessary to specify:

- *Name*, unique, of the Visit Structure.
- *Description* (optional) explanatory.
- *Time to live* (Automatic closure time of visits): if a Visit is not explicitly concluded by the doctor, this field specifies after how much time It will be automatically closed and considered no longer valid.
- *Data Structure*: the Type defined previously.
- *Agendas*: the agenda(s) defined previously.
- *Authorizations on Operations*: user authorizations regarding certain operations.
- *Viewers associated with contexts and qualifications*: here will be listed the Viewers previously created, they will also be associated with users authorized to compile, manage, and view them. It will be specified in what circumstances one or the other viewer will be used.

Based on the data specified previously in this guide, we can compile the *Visit Structure* in the following way:

Create Visit Structure

Cancel Save

* Name	<input type="text" value="Clinical Dermatology"/>																		
Description	<div>A Clinical Dermatology Visit Structure</div>																		
* Time to Live	<input type="text" value="168"/> hours																		
* Data Structure	<input type="text" value="Dermatology Clinical Visit"/>																		
Agendas	<table><thead><tr><th>Code</th><th>Description</th><th></th></tr></thead><tbody><tr><td>001-2</td><td>Clinical Dermatology</td><td>✕</td></tr></tbody></table>			Code	Description		001-2	Clinical Dermatology	✕										
Code	Description																		
001-2	Clinical Dermatology	✕																	
Authorizations on Operations	<table><thead><tr><th>Operation</th><th>Qualification</th><th></th></tr></thead><tbody><tr><td>end examination</td><td>doctor</td><td>✕</td></tr></tbody></table>			Operation	Qualification		end examination	doctor	✕										
Operation	Qualification																		
end examination	doctor	✕																	
Viewers associated with contexts and qualifications	<table><thead><tr><th>Contexts</th><th>Qualification</th><th>Viewer</th><th></th></tr></thead><tbody><tr><td>Delivery</td><td>doctor</td><td>Dermatology Clinical Visit - VIEWER_EDIT</td><td>✕</td></tr><tr><td>Summary</td><td>doctor</td><td>Dermatology Clinical Visit - VIEWER_VIEW</td><td>✕</td></tr><tr><td>Report</td><td>doctor</td><td>Dermatology Clinical Visit - VIEWER_VIEW</td><td>✕</td></tr></tbody></table>			Contexts	Qualification	Viewer		Delivery	doctor	Dermatology Clinical Visit - VIEWER_EDIT	✕	Summary	doctor	Dermatology Clinical Visit - VIEWER_VIEW	✕	Report	doctor	Dermatology Clinical Visit - VIEWER_VIEW	✕
Contexts	Qualification	Viewer																	
Delivery	doctor	Dermatology Clinical Visit - VIEWER_EDIT	✕																
Summary	doctor	Dermatology Clinical Visit - VIEWER_VIEW	✕																
Report	doctor	Dermatology Clinical Visit - VIEWER_VIEW	✕																