Template Builder User Guide

Version 2.3

Disclaimer

Vizrt provides this publication "as is" without warranty of any kind, either expressed or implied. This publication may contain technical inaccuracies or typographical errors. While every precaution has been taken in the preparation of this document to ensure that it contains accurate and up-to-date information, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained in this document. Vizrt's policy is one of continual development, so the content of this document is periodically subject to be modified without notice. These changes will be incorporated in new editions of the publication. Vizrt may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time. Vizrt may have patents or pending patent applications covering subject matters in this document. The furnishing of this document does not give you any license to these patents.

Technical Support

For technical support and the latest news of upgrades, documentation, and related products, visit the Vizrt web site at www.vizrt.com.

Created on

2022/04/11

Contents

1 Introduction

Template Builder lets you make customized templates using scene import or existing templates from Viz Pilot's Template Wizard. This user guide shows you how to customize templates.

> **Info:** A key feature is that you can add custom HTML panels to templates, giving full control over the template through custom scripting and logic.

1.1 Workflow

A simplified version of the workflow follows below:

- Scenes are made in Viz Artist.
- The scenes are imported into Template Wizard, where templates are made.
- Templates are edited and new templates can be made in Template Builder.
- The template is saved in the Viz Pilot system and is available to newsroom and control room systems for playout.

> **Note:** Changes made to a template in Template Builder are not be available when opening the template in Template Wizard.

1.2 Related Documents

The templates customized in Template Builder can be used by other Vizrt products such as Viz Pilot Edge, Viz Story and Viz Multiplay. For more information about all Vizrt products, visit:

- www.vizrt.com
- Vizrt Documentation Center
- Vizrt Training Center
- Vizrt Forum

1.3 Feedback

We welcome your feedback and suggestions regarding Vizrt products and this documentation. Please contact your local Vizrt customer support team at http://www.vizrt.com.

2 Setup And Configuration

This section covers the following topics:

2.1 Software Requirements

- Graphic Hub 3.4.1 or above
- Pilot Data Server 8.6.0 or above
- Preview Server 4.4.0 or above
- Viz Artist 3.14.2 or above (see note below)

Note: Viz Artist 4.2 and Viz Engine 4.2 are required for transition logic and combo template support. See Configuring Viz Artist below. Viz Artist and Viz Engine 4.2 are therefore recommended.

2.2 Browser Requirements

If running inside a browser, the following minimum requirements apply:

- Microsoft Internet Explorer 11 +
- Chrome 64 +
- Safari 11.0 +

2.3 Opening Template Builder

Template Builder opens as a web application in your default browser.

The URL to access Template Builder, if hosted on the Pilot Data Server, is: **http://pds-hostname:8177/app/templatebuilder/TemplateBuilder.html** .

2.4 Configuring Viz Artist

If the Geom of a scene is outdated or empty when creating a transition logic template, Template Builder will block use of the scene.
To fix this, save or update the scene in Viz Artist 4.2.

> **Important:** The feature below must be enabled in the Viz Artist config (see the Viz Artist User Guide for more info):

- Enable automatic creation of merged geometries when saving a transition logic scene: AutoExportTransitionLogicGeometries = 1

2.5 Configuring Preview Server

Preview Server manages one or more Viz Engines, providing frames for thumbnails and snapshots in an ongoing preview process.

Preview Server must be configured in the Pilot Data Server:

1. Access the Pilot Data Server Web Interface: **http://pds-hostname:8177** .
2. Click the **Settings** link.
3. Select the **preview_server_uri** setting, and add the URL for the machine on which you installed the Preview Server (ie. **http://previewserver-hostname:21098**).
4. Click **Save**.

> **Note:** All applications with a connection to the database will now have access to Preview Server.

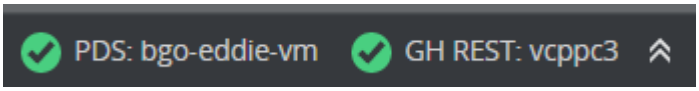
2.6 Specifying A Graphic Hub Endpoint

If you're using multiple Graphic Hubs, the one used to store your scenes must be configured in the Pilot Data Server:

1. Access the Pilot Data Server Web Interface: **http://pds-hostname:8177** .
2. Click the **Settings** link.
3. Select the **graphic_hub_url** setting, and add the URL for the machine on which your scenes are stored (ie. **http://gh-hostname:19398**).
4. Click **Save**.

2.7 Monitoring Graphic Hub Status

Since some users have multiple Graphic Hubs (GHs) for design, distribution, testing and production, **green icons** at the bottom of the interface show you which GH and which database you're currently connected to:



Note: GH REST status info is based on the `graphic_hub_url` parameter mentioned above - not Graphic Hub's search provider settings.

3 Creating Templates With Scene Import

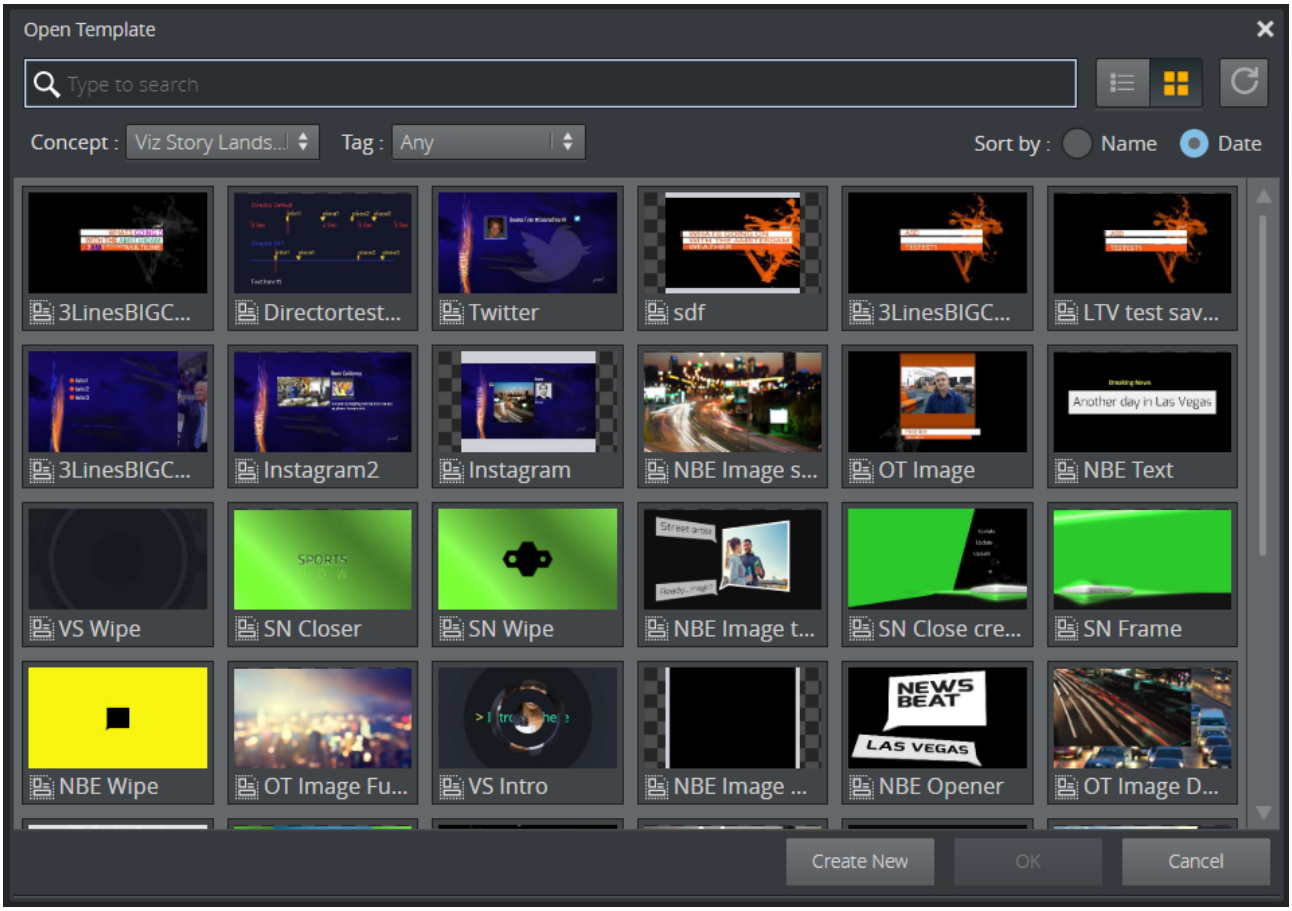
Create templates using the scene import feature.

This section covers the following topics:

3.1 Opening A Template

- Click **Open**, or **CTRL + O**, to open a dialog containing templates available within the Pilot system.
- In the **Open Template** dialog, use **Concepts** and **Tags** to filter templates. The search can also be narrowed down by searching for the template name in the **Type to filter...** field at the top of the dialog.
- Select a template and click **OK** or double-click it to open.

> **Note:** Template Builder can detect if there is an unsaved state while opening an old template. If this occurs, the following message will show: "The old template was updated and needs saving". Save the template and continue your workflow.



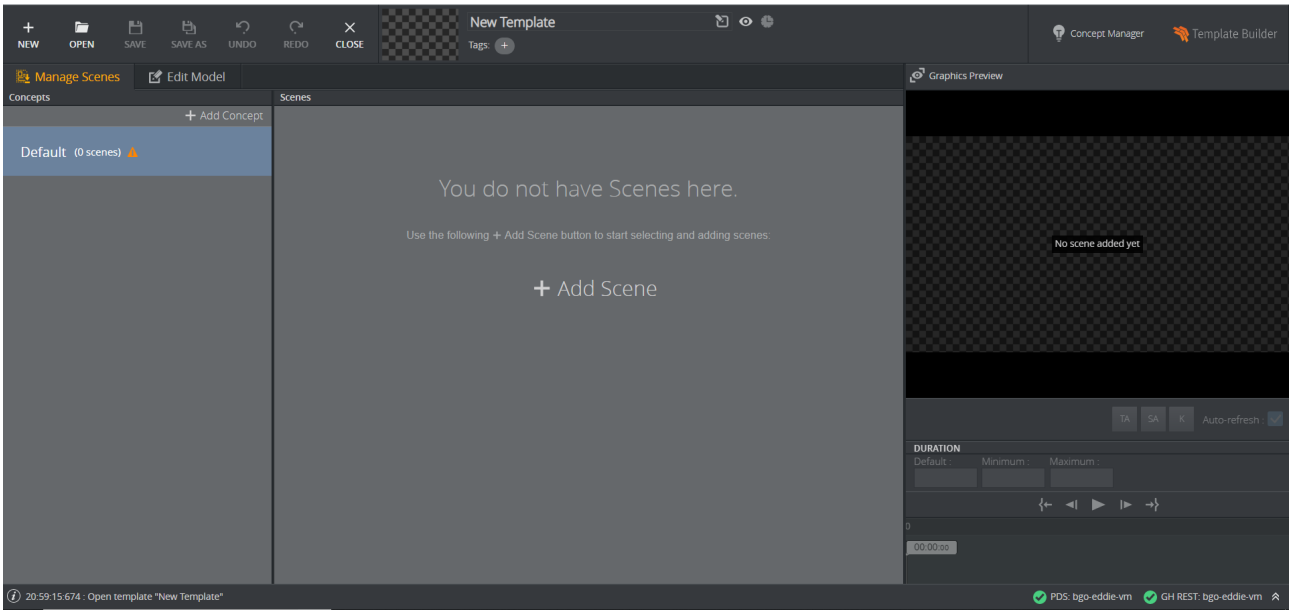
3.2 Creating A New Template

- Click **Create New** in the **Open Template** menu.

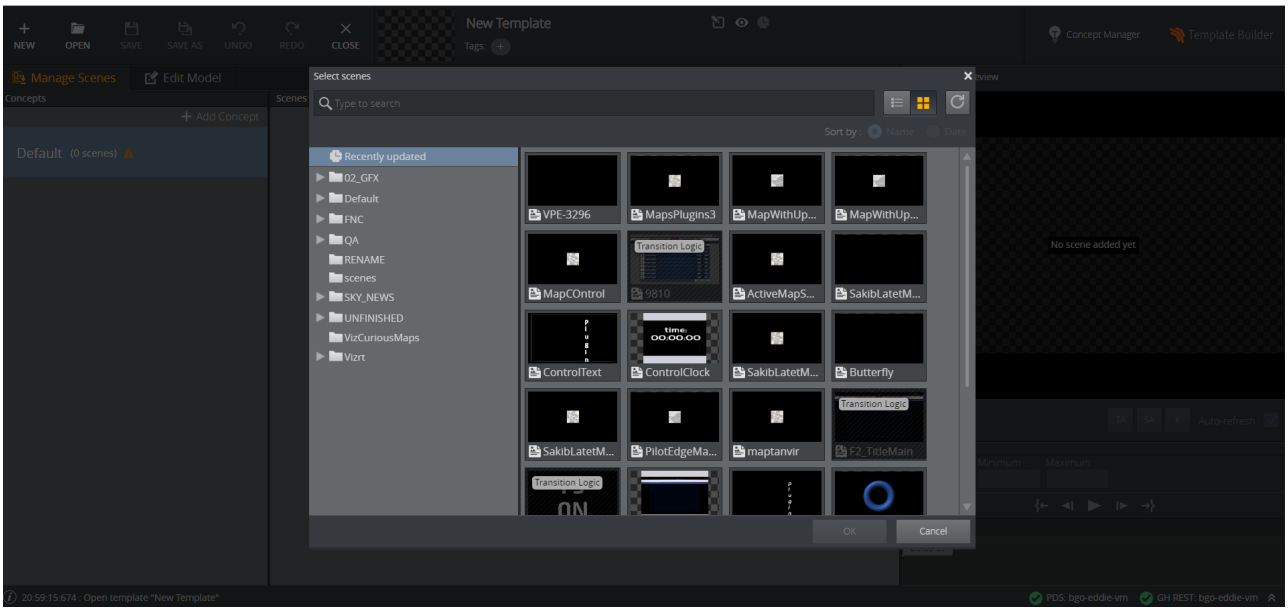
> **Note:** If you don't see this option, make sure that the required programs are up-to-date.

You now need to add a scene to your new, empty template:

- Go to **Manage Scenes** at the top left of the interface.
- Select **+Add Scene.**

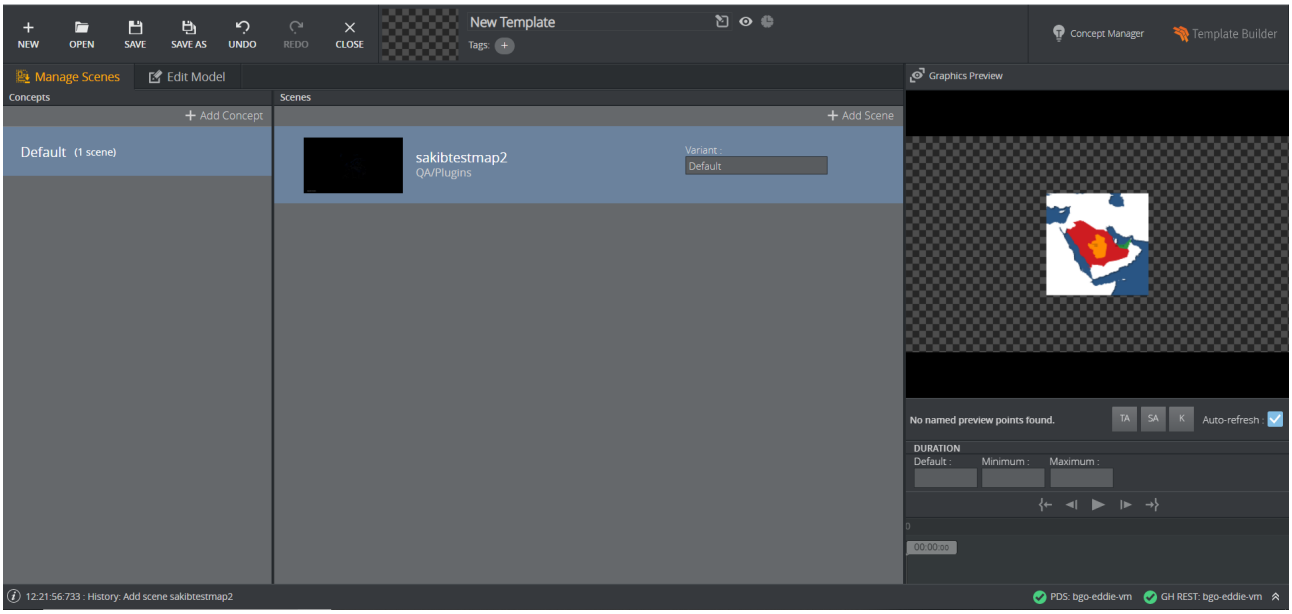


The **Select scenes** menu appears, containing all of the scenes stored in the Graphic Hub to which you are connected:

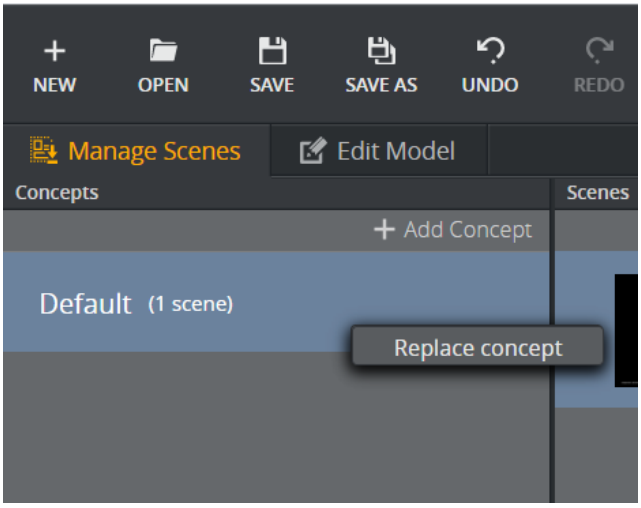


Note: The Graphic Hub containing your scenes is specified through the **graphic_hub_url** setting in Pilot Data Server.

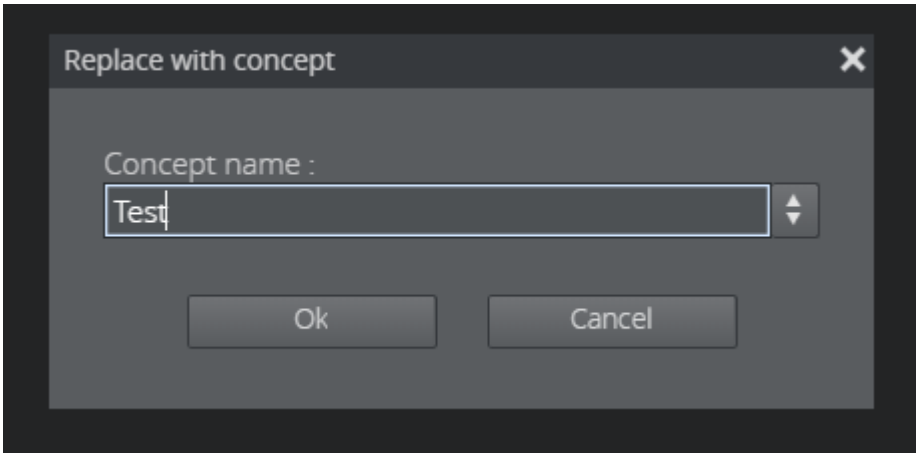
- Enter a search term or browse the folder structure. Once you have selected the correct scene or scenes, press **OK** to add them to the template:



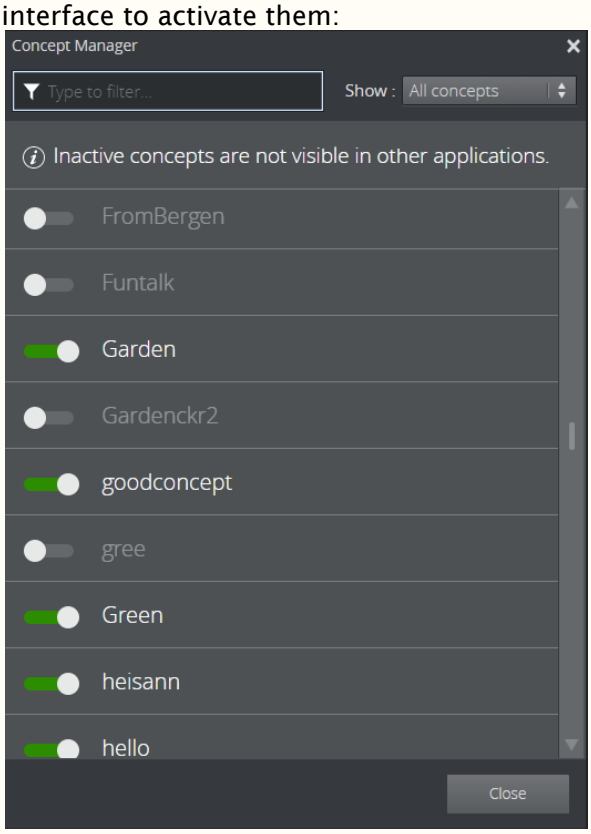
- If you want to rename a concept, right-click it and **Replace concept**:



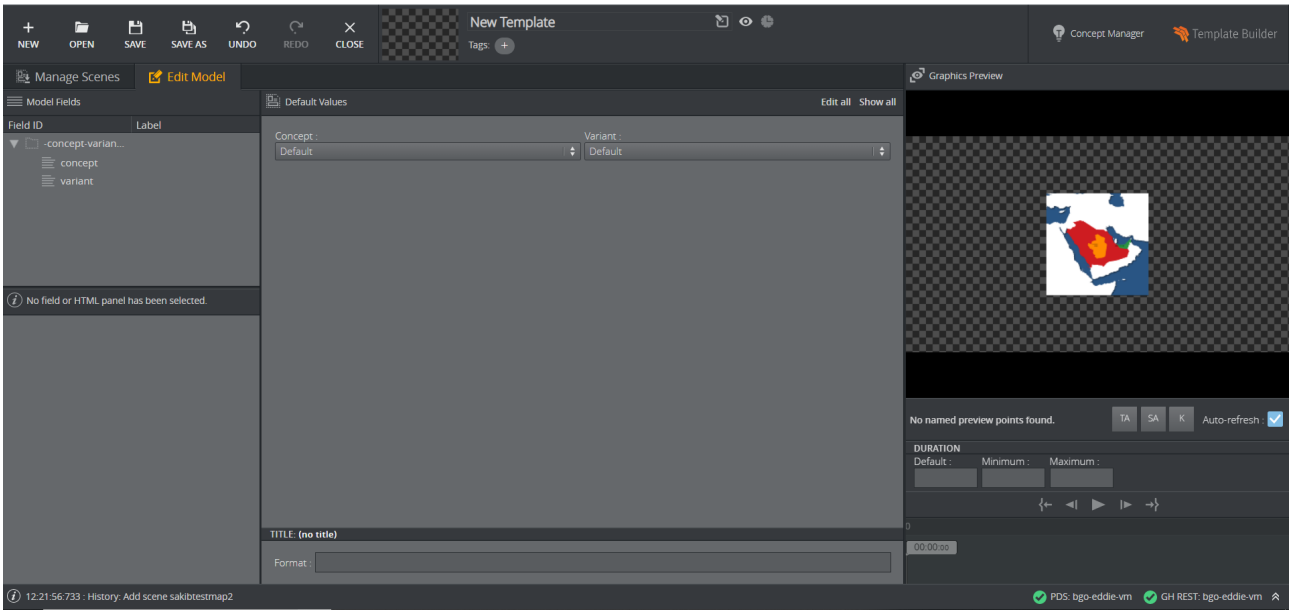
- Enter a **Concept name** and click **Ok**:



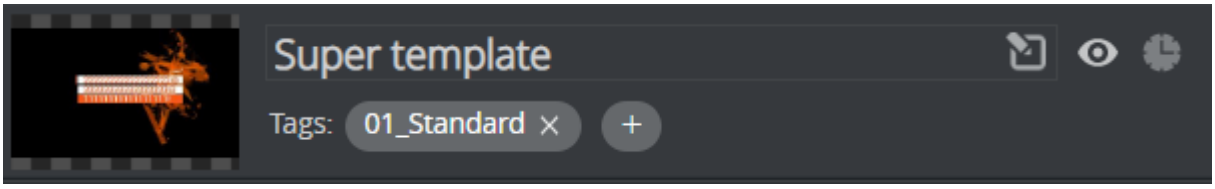
Note: New concepts are inactive by default, which means they won't be visible in other applications.
Once the template has been saved, use the **Concept Manager** at the top right of the interface to activate them:



- Click the **Edit Model** tab to see a preview of the template:



The menu at the top center of the interface allows you to:



- Rename the template.
- View and edit its **Tags**.
- Hide it from other applications by clicking the **Eye icon**.
- Decide whether Director should open it using Pilot Edge or Viz Pilot News (a legacy setting) by clicking the **Clock icon**:
- **Clock lit** - Legacy template: opens in Delphi in Viz Pilot 8.6 or later.
- **Clock grayed out** - New template: opens in Pilot Edge 1.6 or later.
- Finally, to save the template, simply click **SAVE** at the top left of the interface.

> ⚠ **Note:** The updated template will overwrite the existing template in Pilot Data Server.

4 Customizing Templates

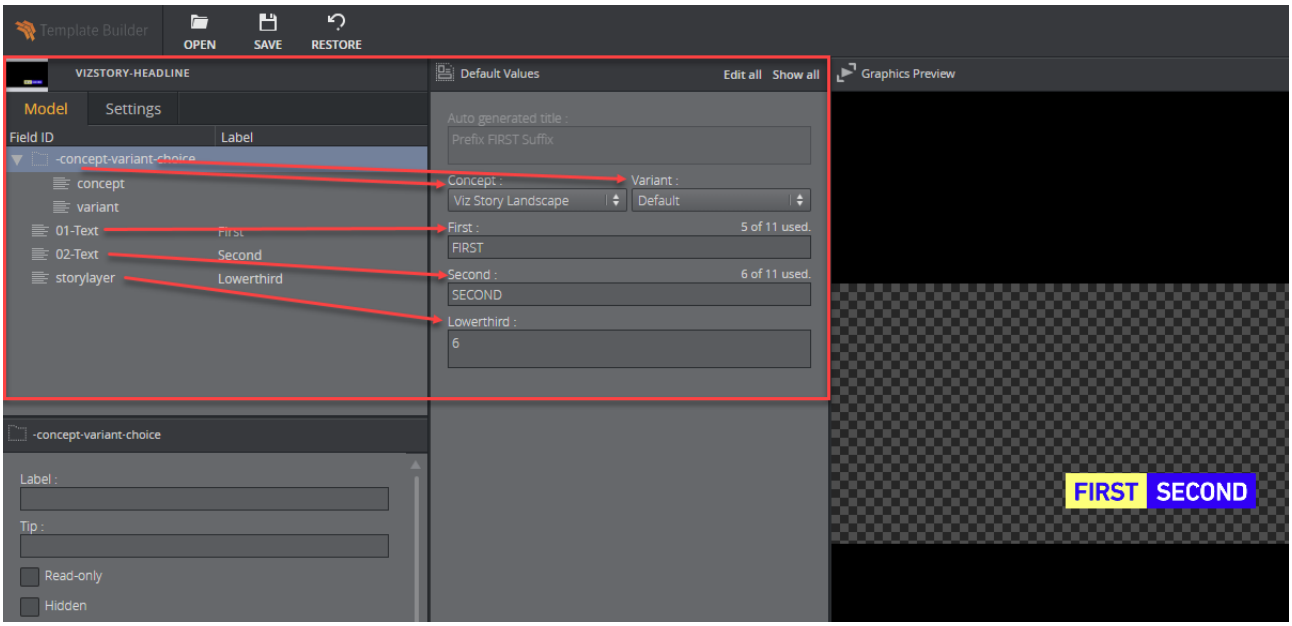
Customize templates in the **Template Builder** window at the left of the application:

- Model
  - Field Tree
  - Multi-selection
  - Field Properties
- Settings
  - Duration
  - Track
  - Title Generation

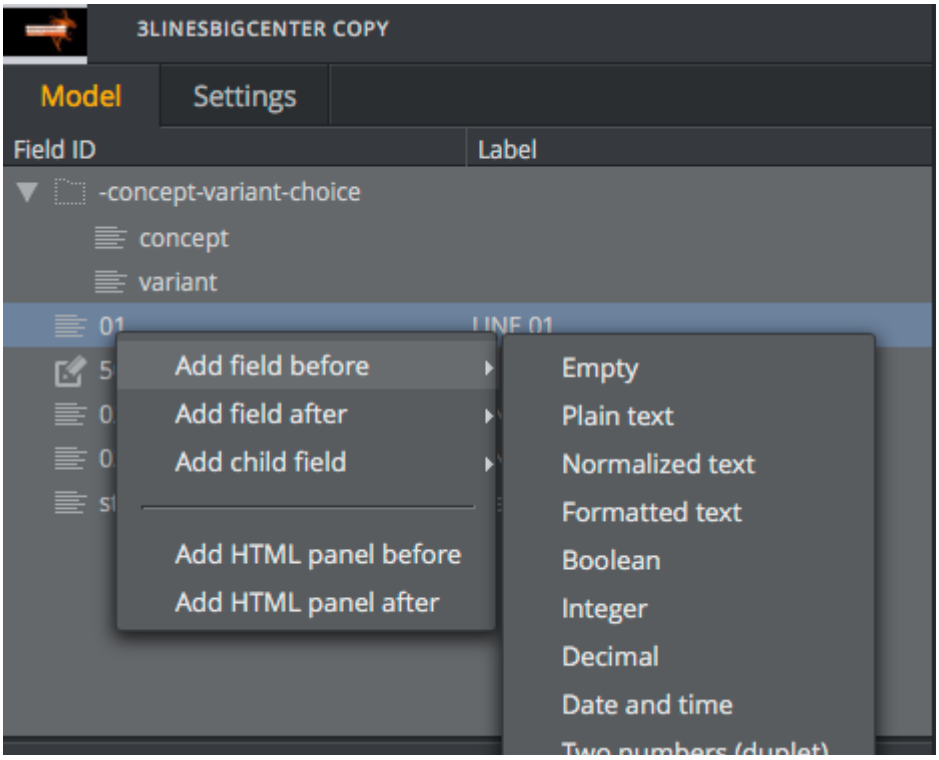
4.1 Model

4.1.1 Field Tree

The Field Tree contains the **Field ID** and **Label**, which are also shown in the Fill In Form, see the red arrows in the figure below. The icon beside each line in the tree indicates the Type of content in the field.



- Fields can be rearranged by drag-and-drop within the Field tree.
- Right-click a field to open a menu where additional fields and HTML panels can be added.
- The Fill In Form updates immediately when any changes are made.



> **Info:** Only fields created in Template Builder can be deleted and given a new ID.

4.1.2 Multi-selection

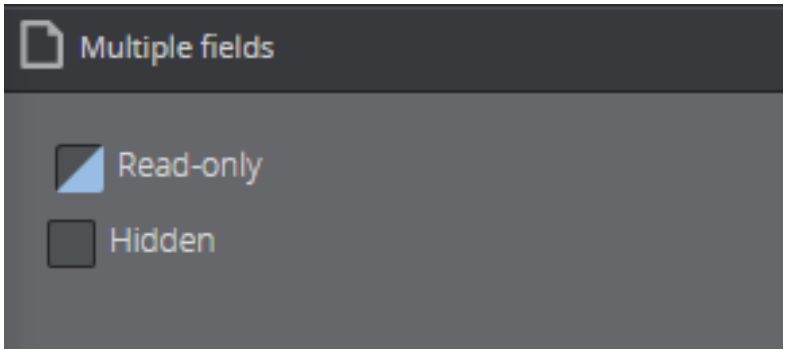
Multi-select fields in the Field Tree by pressing **CTRL + click**. Although it's not possible to move or rename multiple fields at the same time, multiple fields can be deleted and some of their properties can be changed in the **Field Properties** window.

4.1.3 Field Properties

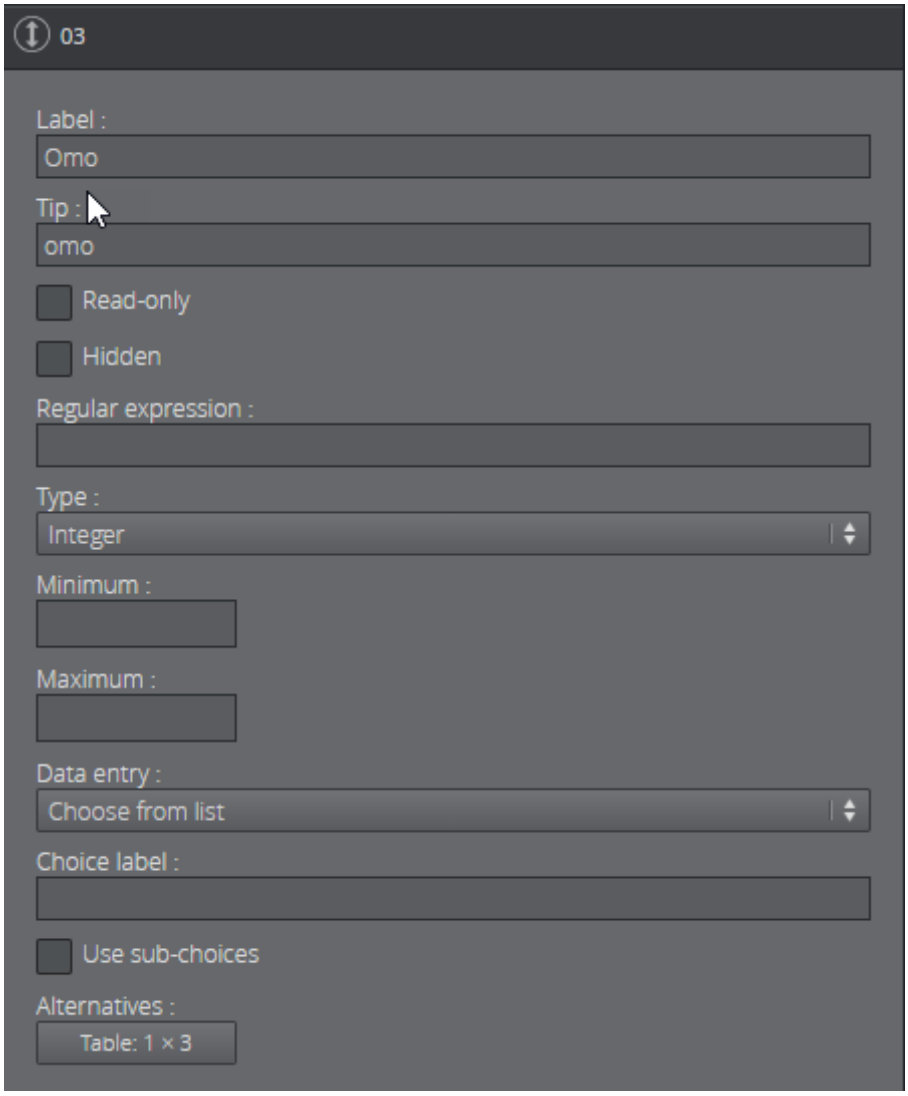
The **Field Properties** window is located below the Field Tree window. It displays the properties of a selected fields in the Field Tree.

- **Multi-selection:** If several fields are selected in the Field Tree, a subset of the field properties is displayed. If the selected fields have different field property values, the Field Properties window displays a multiple values state. Changes made in the Field Properties window are

immediately applied to all the selected fields.



- **Single-selection:** All properties for the selected field are displayed. Note that the set of properties depends on the type of the field.



- **Label:** Specifies the label of the field in the Fill-In Form.
- **Tip:** A tooltip text can be entered to provide more information about the field.
- **Read-only:** The field remains visible, but is grayed out in the Fill-In Form.

- **Hidden:** Hides the field in the Fill-In Form.
- **Regular expression:** Defines constraints of the value in the field.

Type

The type of content allowed in the field in **Default Values** is set using the drop-down list under **Type**. Depending on the type selected, different sub-options become available, as specified in the table below.

There are two main field type categories: *scalar* and *list*. Fields of all types apart from the list type are referred to as *scalar fields*. Fields using the list type are referred to as list fields.

The following types are available:

Type	Icon	Media Type (XSD Type)*	Comments
Empty			Makes the field unavailable. Typically used as a container for other fields.
Multi-line text		text/plain (string)	**Max length**: Sets the maximum number of characters allowed in the field.
Single-line text		text/plain (normalizedString)	**Max length**: Sets the maximum number of characters allowed in the field.
Formatted text		application/vnd.vizrt.richtext+xml	**Max length**: Sets the maximum number of characters allowed in the field. **Single-line**: Check this box to specify that the rich-text editor allows one line of text only.
Boolean		text/plain (boolean)	Creates a checkbox that has two states: true and false.
Integer		text/plain (integer)	This field is an integer field. **Minimum**: Sets the minimum value allowed in the field. **Maximum**: Sets the maximum value allowed in the field.
Decimal		text/plain (decimal)	This field allows decimal numbers. **Minimum**: Sets the minimum value allowed in the field. **Maximum**: Sets the maximum value allowed in the field.

Type	Icon	Media Type (XSD Type)*	Comments
Date and time		text/plain (dateTime)	Use the **Date Chooser** in **Default Values** to select date and time in this field.
Date		text/plain (date)	Use the **Date Chooser**, or the individual editors for day, month and year in **Default Values**, to select the date in this field.
Two numbers (duplet)		application/vnd.vizrt.duplet	This field allows two numbers (decimal numbers are allowed). **Minimum**: Sets the minimum value allowed for both numbers. **Maximum**: Sets the maximum value allowed for both numbers.
Three numbers (triplet)		application/vnd.vizrt.triplet	This field allows three numbers (decimal numbers are allowed). **Minimum**: Sets the minimum value allowed for all three numbers. **Maximum**: Sets the maximum value allowed for all three numbers.
Image		application/atom+xml;type=entry;media=image	Makes the field available for an image. **Image Constraints**: Enable this option if you want to set constraints on the image. **Minimum Size of the image (pixels)**: Specifies the minimum allowed image dimensions in pixels. Both width and height must be at least this big. **Aspect Ratio (width x height)**: Specifies the aspect ratio of the image. **Allowed Error (%)**: Specifies the maximum stretch limit for both the width and height of the image, in relation to its defined aspect ratio.
Video		application/atom+xml;type=entry;media=video	Makes the field available for a video.
Font		application/vnd.vizrt.viz.font	Makes the field available for a font.

Type	Icon	Media Type (XSD Type)*	Comments
Geometry		application/vnd.vizrt.viz.geom	Makes the field available for a geometry.
Material		application/vnd.vizrt.viz.material	Makes the field available for a material.
Map		application/vnd.vizrt.curious.map	Makes the field available to present and edit a map.
Custom			Lets you freely specify the media and XSD type.
List			Lists may be modified by adding and removing columns in the Field Tree. · To add columns to a list - right-click the columns node under the list field node in the Field Tree and select **Add column**. · To remove a column - select the column field in the Field Tree and press **Delete**, or right-click it and select **Delete field**. **Note:** List fields are fundamentally different from scalar fields. It's therefore not possible to change a list type to a scalar type and vice versa. **Minimum number of rows**: Defines the minimum allowed number of rows in the list. **Maximum number of rows**: Defines the maximum allowed number of rows in the list.

* For more information on media types, see: Overview of Media Types.

Note: Be aware of available control plugins in the template that have been exposed by the scene designer in Viz Artist.

Data Entry

There is a Data entry drop-down list available for all scalar field types that contains three options:

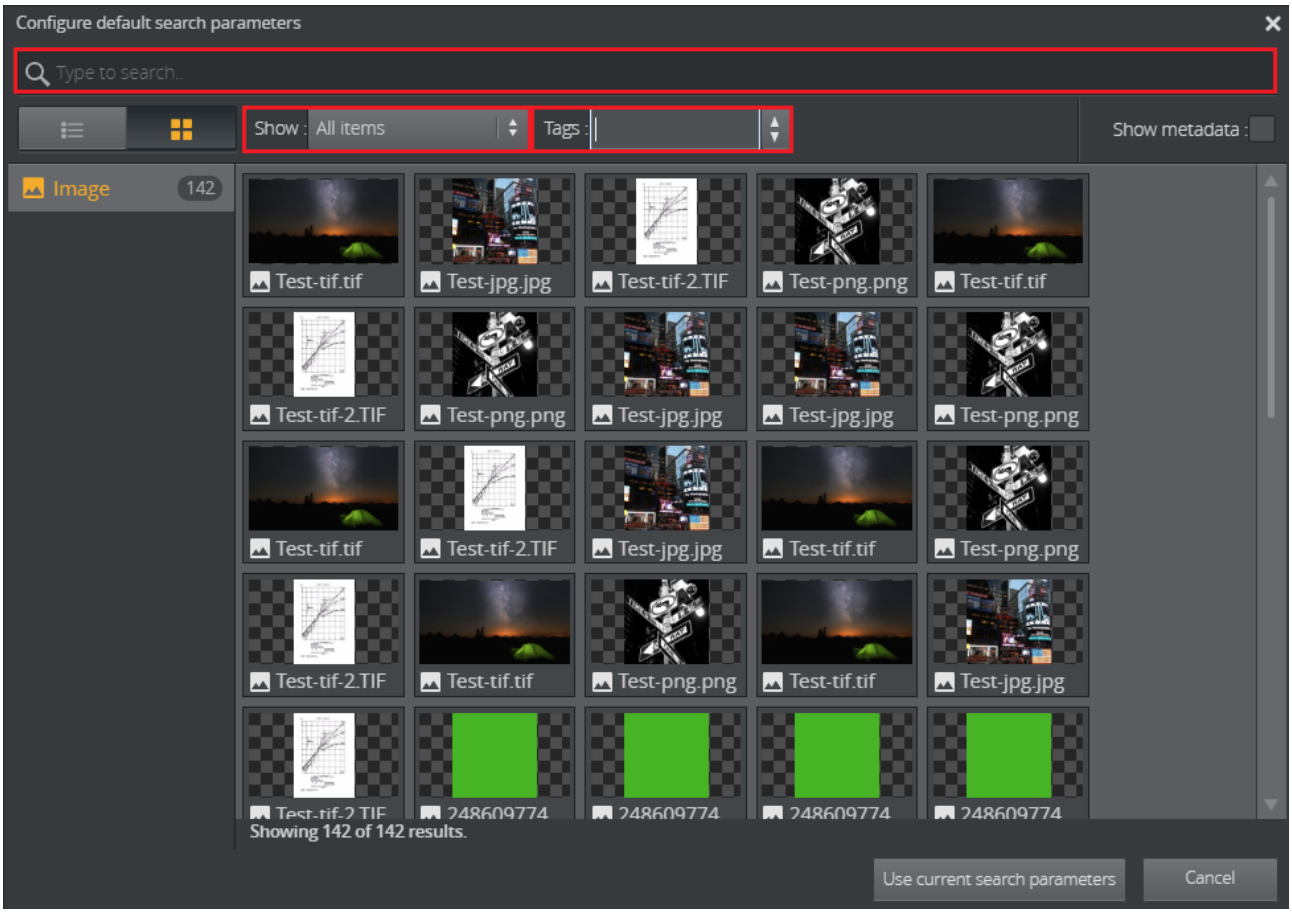
- **Manual:** Does not permit any additional settings for the field, see Manual.
- **Choose from list:** Allows the template designer to present the right content in a drop-down list, see Choose from list.
- **Enable feed browser:** Allows you to browse for an entry, see Enable feed browser/Parent feed browser.

For more information, see Data Entry.

Default Search Parameters

For the types **Image**, **Video**, **Font**, **Geometry**, and **Material**, it's possible to define default search parameters that will be used by the media search that is launched when you click on the field:

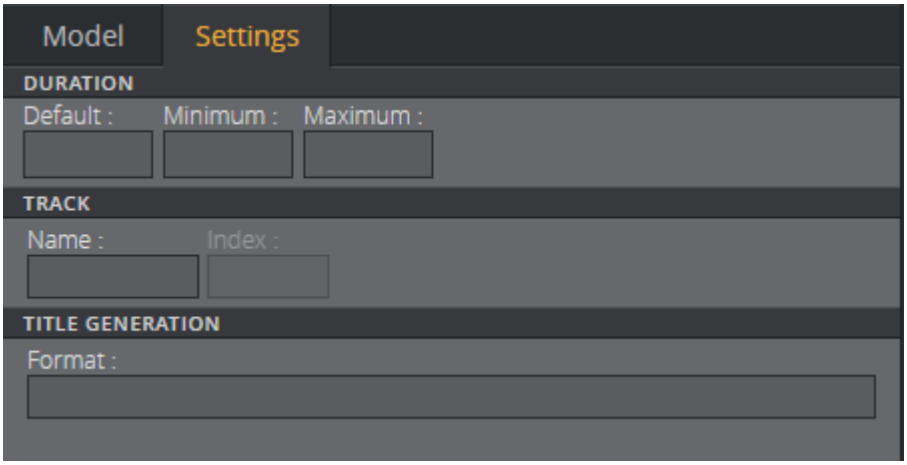
1. Click the **Set** button to open a media search window.
2. Enter text in the search field, selecting whether to show all items or to limit by time from the **Show** drop-down list, and/or selecting a tag from the **Tags** drop-down.
3. Save by clicking **Use current search parameters**.



4.2 Settings

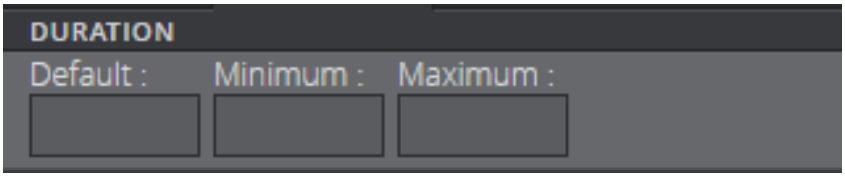
The following template settings are available in the **Settings** tab:

- Duration
- Track
- Title Generation



4.2.1 Duration

Specify the duration of a graphic on a timeline using **Minimum** and **Maximum** values; if these are set to the same number the item will be assigned a fixed duration.



Note: A default value is used if you don't specify duration.

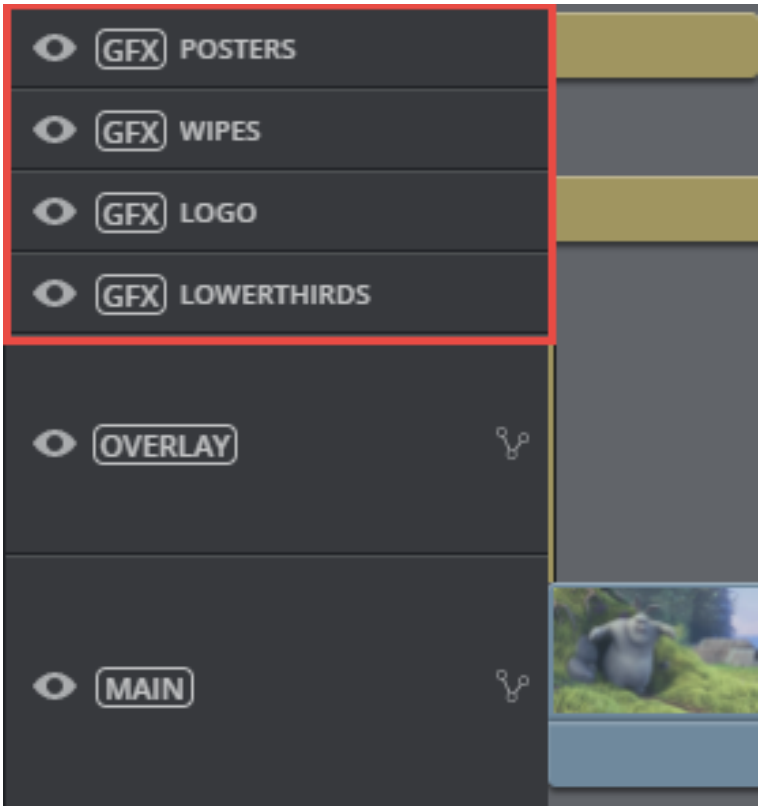
4.2.2 Track

Setting a **Name** in the track settings allows graphics to be grouped in the timeline editor, see Track settings displayed in the timeline editor. **Index** sets the position of the group in the timeline editor, where 0 is the lowest position.

Warning: All templates with the same track name must have the same index.

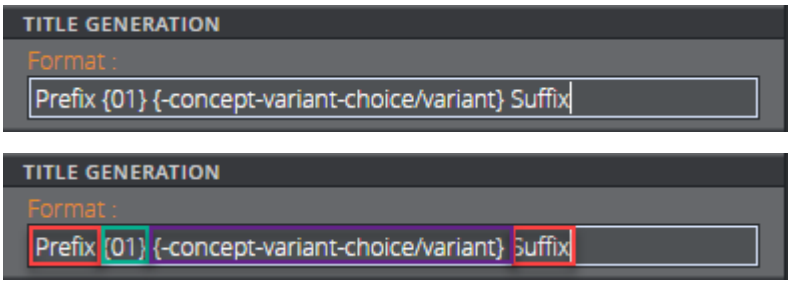
Track settings displayed in the timeline editor

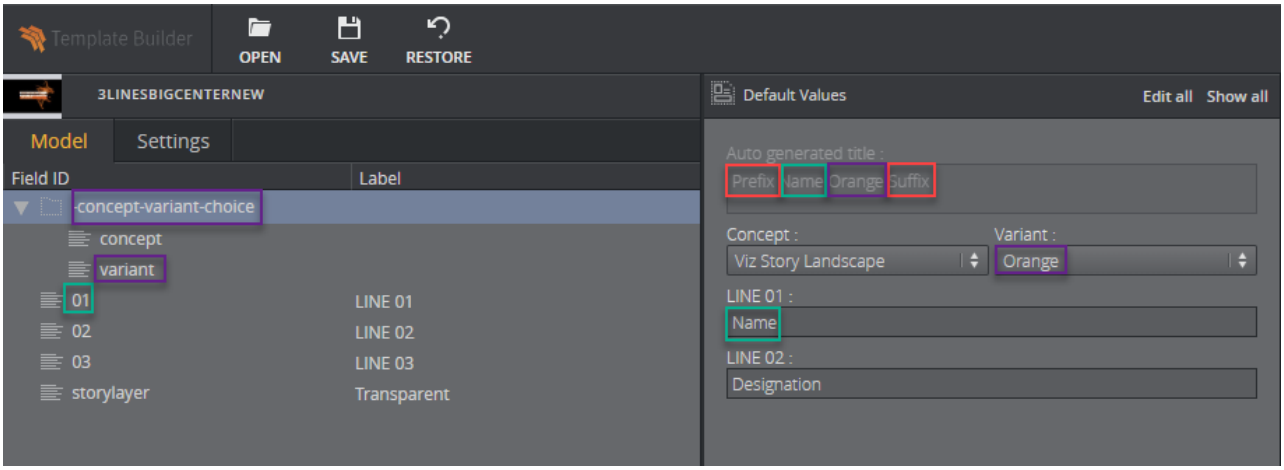
The red rectangle highlights the different tracks in the timeline editor. The position of these tracks is set using the Track setting.



4.2.3 Title Generation

The title generation setting provides auto-generation of the title. The title can be plain text or it can be a placeholder for one or several field values, or it can be a combination of these. The placeholder is the {Field ID}. An example using a combination of plain text, field name, and sub-field name is shown below:





A template title can be auto-generated by combining one or several of these options:

- **Normal text:** Plain text (red).
- **{Field ID}:** Substituted with the value of the field (green).
- **{Field ID/subfield ID}:** Substituted with the value of the subfield (purple).
- **{listfieldname/#index/cellname}:** Substituted with the value of the field in a row in a list. Note that the index is zero-based.

> **Warning:** The auto-generated title's length is not shortened in Vizrt web clients. However, if the title is longer than 128 characters it will be reduced when dragging out the MOS XML file due to size constraints. This affects the element title in the newsroom system.

4.3 Data Entry

The Data Entry field property specifies how users should fill in field values:

- Manual
- Choose From List
- Enable Feed Browser/Parent Feed Browser

4.3.1 Manual

Selecting **Manual** in the Data entry drop-down list does not give access to any additional settings for the field.

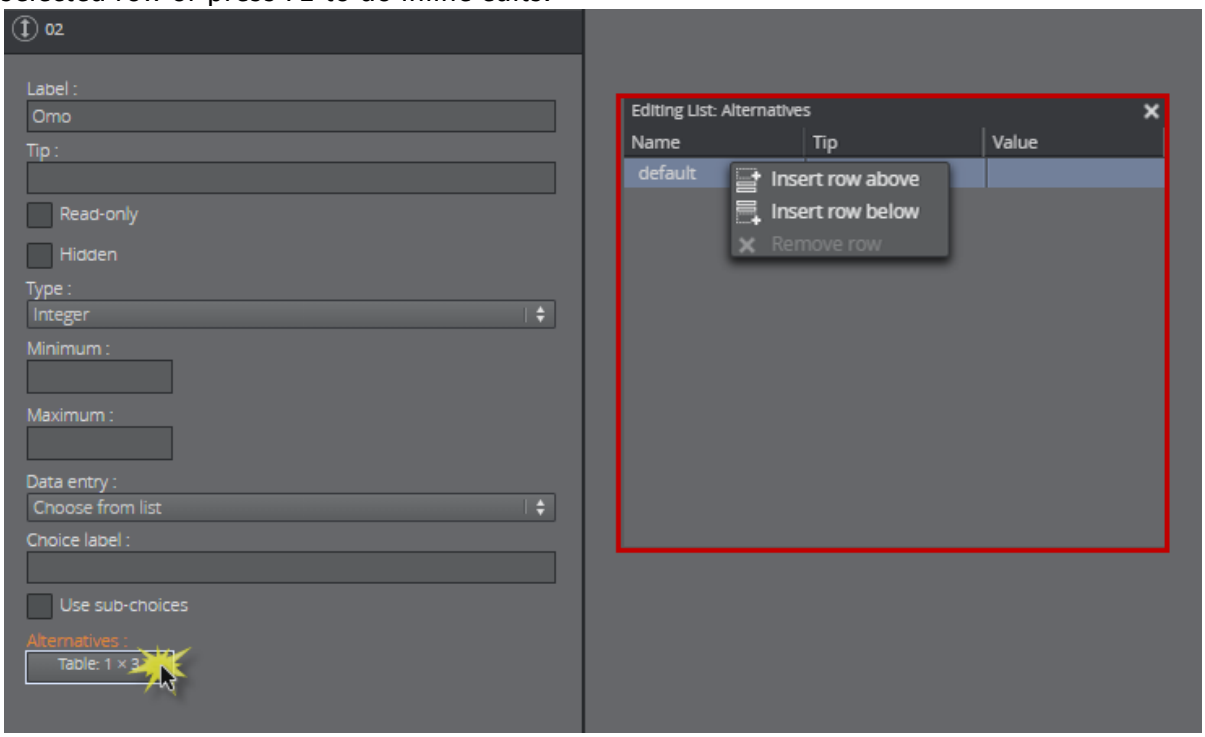
4.3.2 Choose From List

Selecting **Choose from list** lets you see the content in a drop-down list, which may in some cases make it easier and less error-prone to fill the template in with the right content.

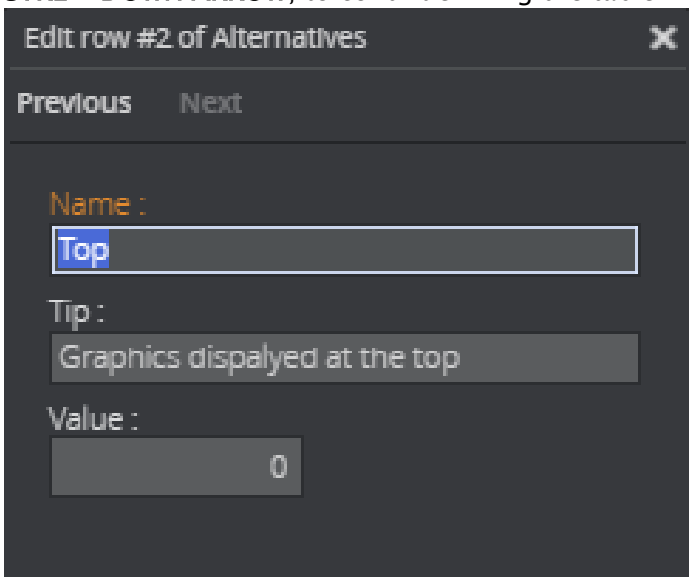
For example, when a Control Object moving (Omo) plugin is accessible in the template, scenes using Omo plugins are originally presented as integer values for the different elements in the Fill In Form. The **Choose from list** option can assign text to these values to make it easier to select the right element.

The example below contains a scene that can be displayed at the top, in the middle or at the bottom in the graphics. For the Omo plugin, these positions correspond to the values 0, 1 and 2 respectively. To assign text to these values:

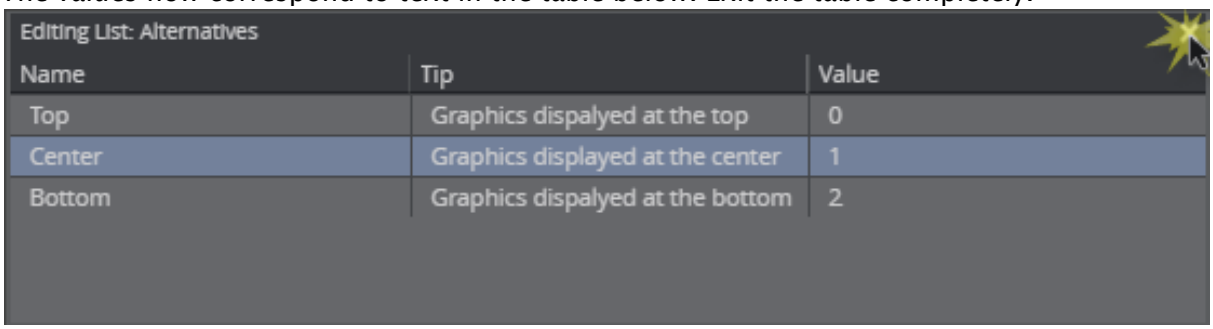
- Mark the **Omo** Field ID in the Field Tree.
- Select **Choose from list** in the **Data entry** drop-down list.
- Click **Alternatives**. A new window appears. Right-click to insert or remove rows. Click a selected row or press F2 to do inline edits.



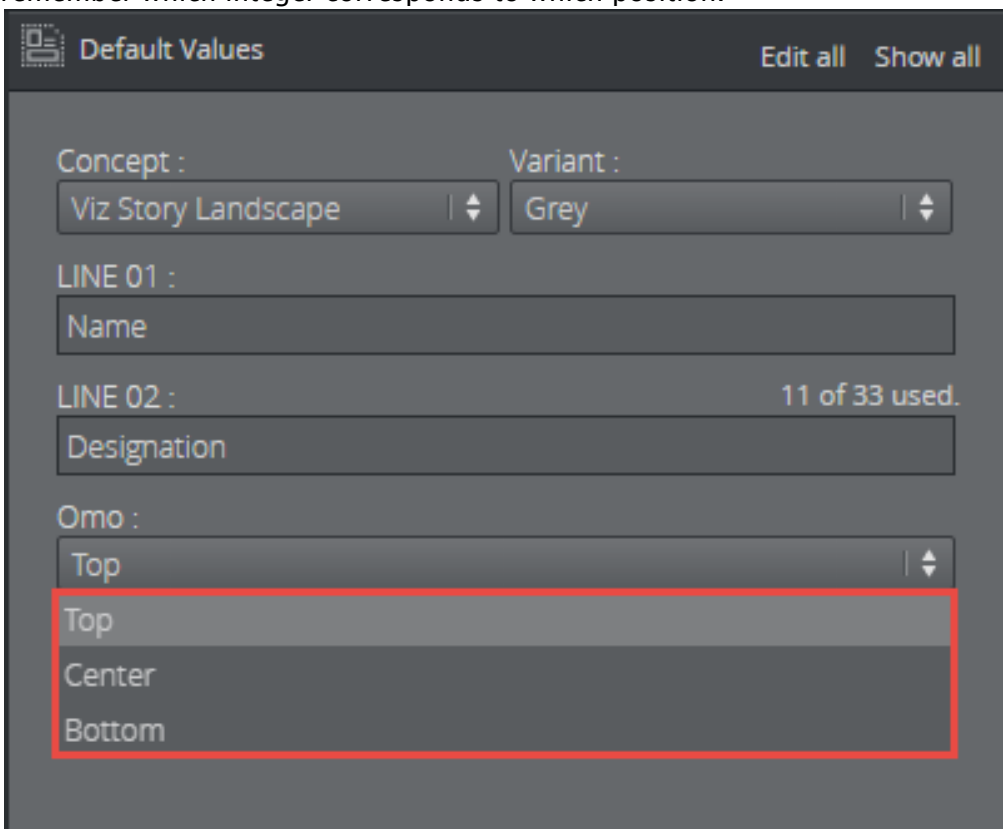
- Double-click the table or press Return to insert **Name**, **Tip**, and **Value**. Click **Next,** tab or **CTRL + DOWN ARROW**, to continue filling the table.



- The values now correspond to text in the table below. Exit the table completely.



- The **Omo** field in the Fill In Form now contains a drop-down list containing the alternatives created above as text, as opposed to an integer field where the user would have to remember which integer corresponds to which position.



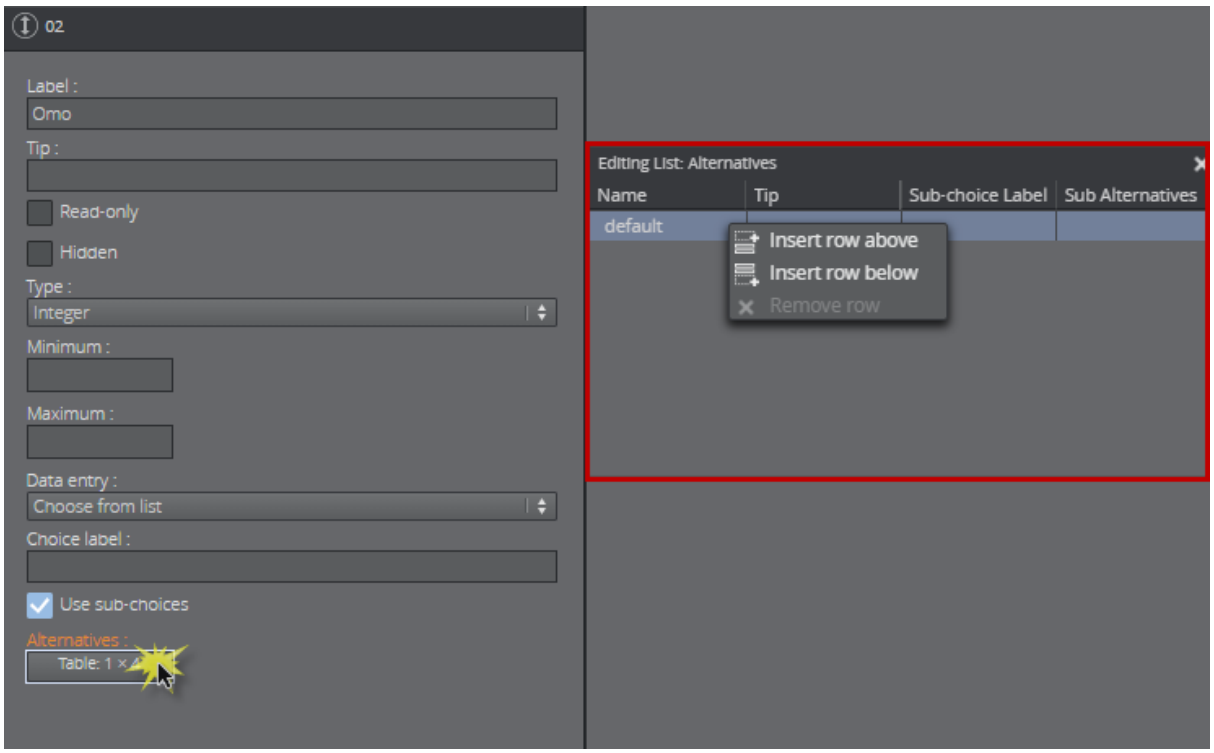
4.3.3 Using Sub-Choices

If you select the **Choose from list** option, a checkbox is made available called **Use sub-choices**, which lets you set multiple sub-choices for each choice.

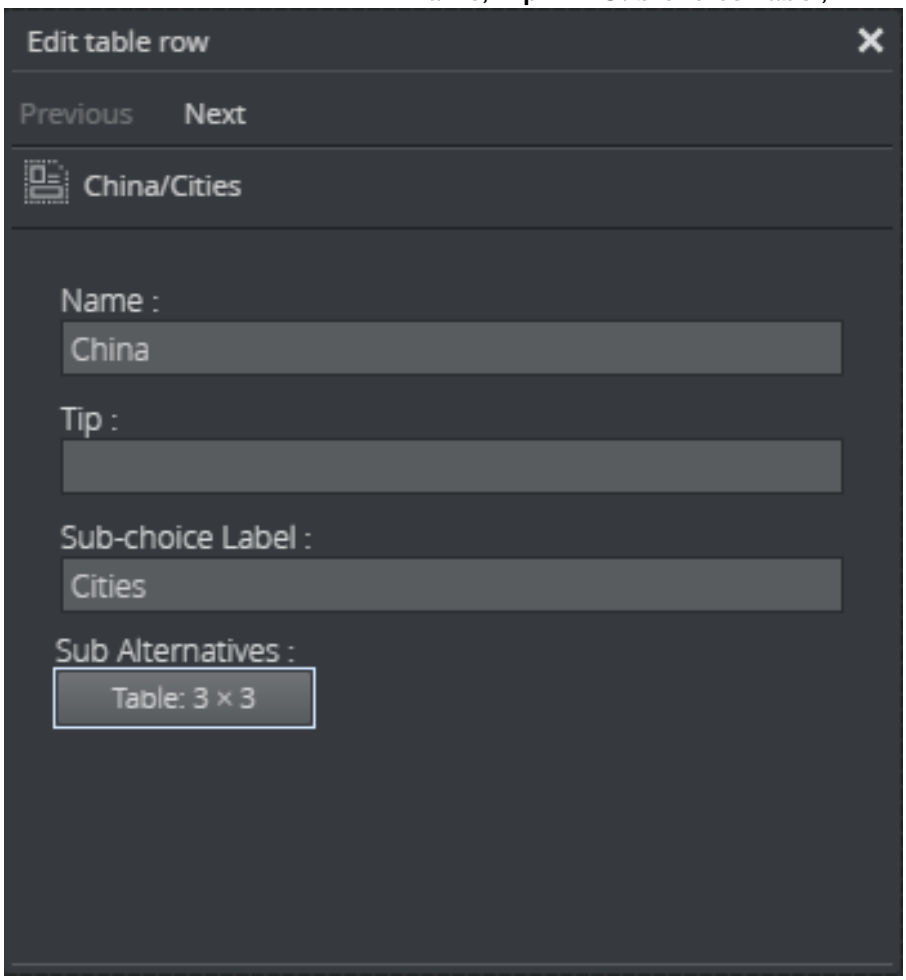
For example, if the choices list different countries, sub-choices could list cities in each of the countries.

- Mark the desired Field ID in the Field Tree.

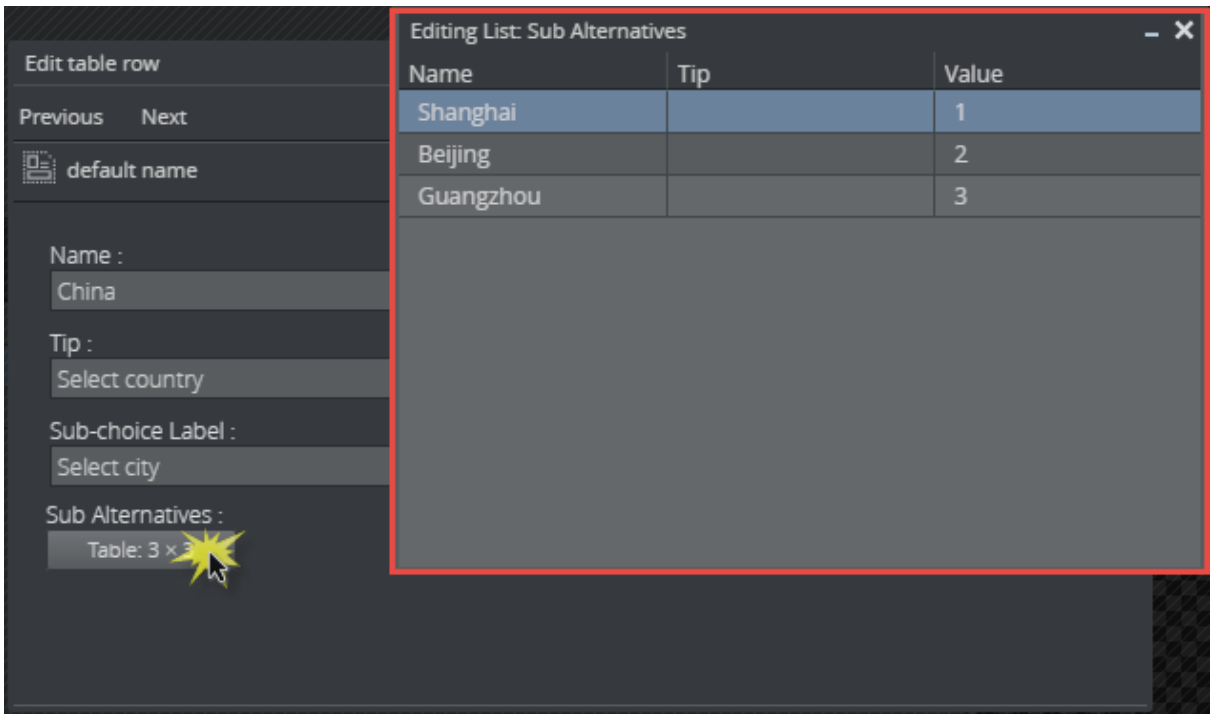
- Select **Choose from list** and tick the **Use sub-choices** check box.
- When clicking on the **Alternatives** button, a new window appears. Right-click to insert or remove rows:



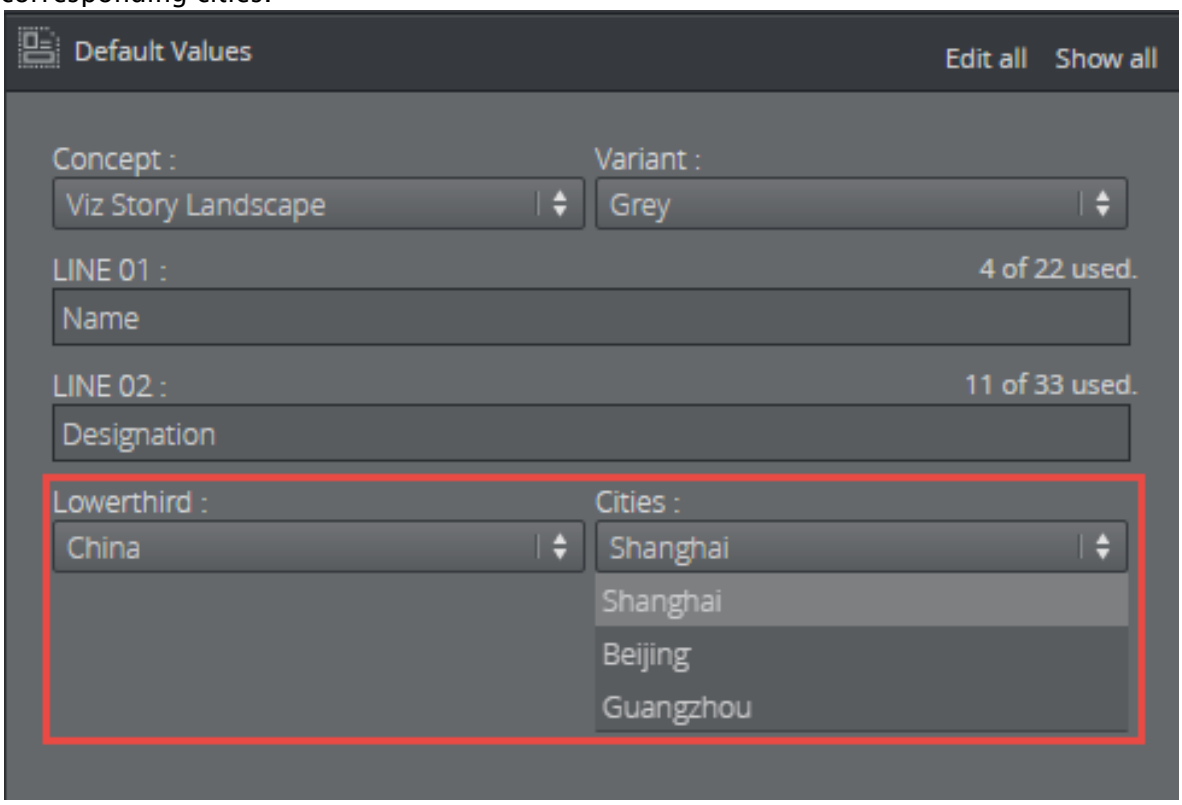
- Double-click the table to insert **Name**, **Tip** and **Sub-choice Label**, in this example **Cities**:



- Click the **Sub Alternatives** button to add the sub-alternatives.



- Exit the tables when complete.
- Instead of a text field in the Fill In Form, the field now contains two drop-down lists: the main choices, which in this case is a list of countries, and sub-choices, with corresponding cities.



4.3.4 Enable Feed Browser/Parent Feed Browser

This option specifies that the field should get its value from a property of an atom feed entry. If the field is a sub-field of another field that has enabled feed browser, the option is named *Parent feed browser*. Otherwise, it is named *Enable feed browser*.

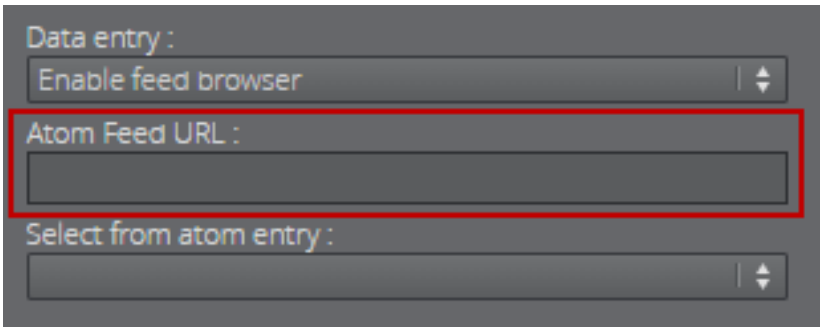
- If the Enable feed browser option is selected, a **Browse** button appears next to the field in the fill-in form.
- Click **Browse** to open the **Feed Browser** dialog.
- In the Feed Browser, the atom entries of the feed are presented (with thumbnails, if available), and one of the entries can be selected.
- Information from the selected atom entry is used to fill in the feed browser enabling field and its subfields.

Note: In order to be able to fill in multiple fields from a single selection in the feed browser, fields must be subfields of the field that enables the feed browser.

4.3.5 Atom Feed URL

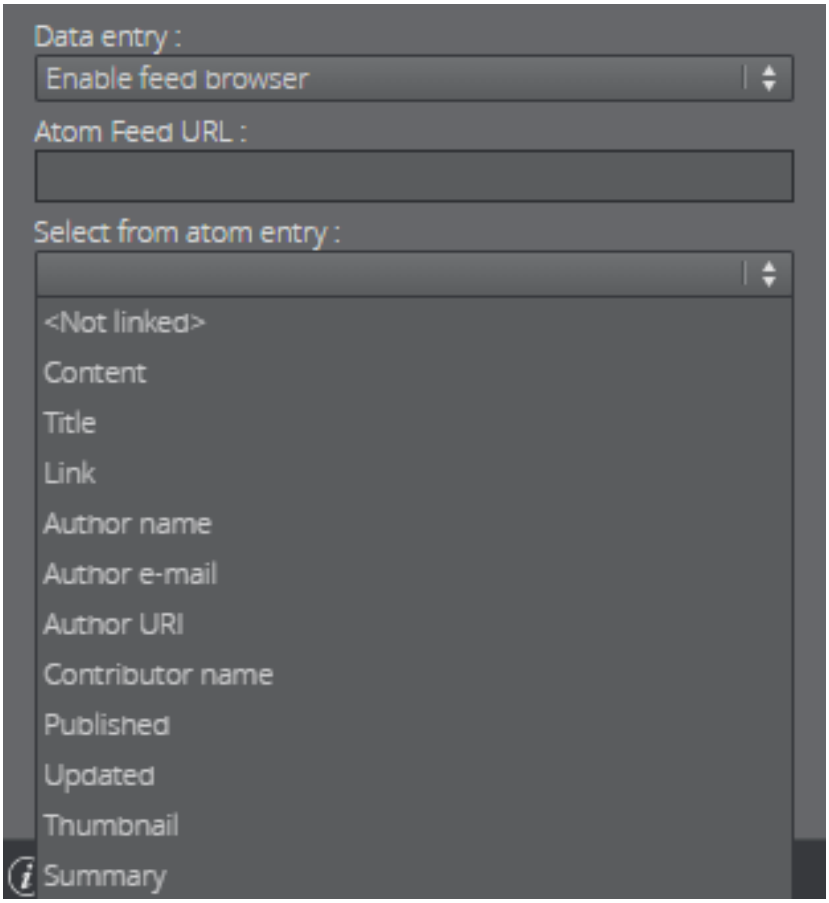
Note: This field property is available only for feed browser enabling fields (not for fields using *parent feed browser*).

Specify the atom feed for the field and a selection of its subfields:



4.3.6 Select from Atom Entry

Note: The options available for a given field depend on the type of the field (the atom namespace prefix represents the http://www.w3.org/2005/Atom namespace, and the media namespace represents the http://search.yahoo.com/mrss/ namespace).



- **<Not linked>:** Not linked to the atom feed, and must be filled in manually.
- **Content:** Linked to the content of the *atom:content* element in the atom entry.
- **Title:** Linked to the content of the *atom:title* element in the atom entry.
- **Link:** Linked to the *href* attribute of the *atom:link* element in the atom entry. Which link entry to pick depends on the *Link-rel in atom entry* property and the type of the field - the first link with a correct rel attribute and a type that matches the type of the field is chosen.
- **Entry:** Linked to the atom entry itself.
- **Author name:** Linked to the content of the *atom:name* element inside the relevant *atom:author* element. If the entry itself contains an *atom:author* element, that is used. Otherwise the *atom:author* element of the feed is used.
- **Author e-mail:** Linked to the content of the *atom:email* element inside the relevant *atom:author* element. If the entry itself contains an *atom:author* element, that is used. Otherwise the *atom:author* element of the feed is used.
- **Author URI:** Linked to the content of the *atom:uri* element inside the relevant *atom:author* element. If the entry itself contains an *atom:author* element, that is used. Otherwise the *atom:author* element of the feed is used.
- **Contributor name:** Linked to the content of the *atom:name* element inside the *atom:contributor* element in the atom entry.
- **Published:** Linked to the content of the *atom:published* element in the atom entry.
- **Updated:** Linked to the content of the *atom:updated* element in the atom entry.
- **Thumbnail:** Linked to the *url* attribute of the *media:thumbnail* element in the atom entry.

- **Summary:** Linked to the content of the *atom:summary* element in the atom entry.
- **Link-rel in Atom Entry:** Only available if **Link** is selected in the Select from atom entry property; it specifies the `rel` attribute of the link element in the atom entry.

> **Note:** A linked field may also be filled in manually if it's not hidden or read-only.

4.4 The HTML Panel

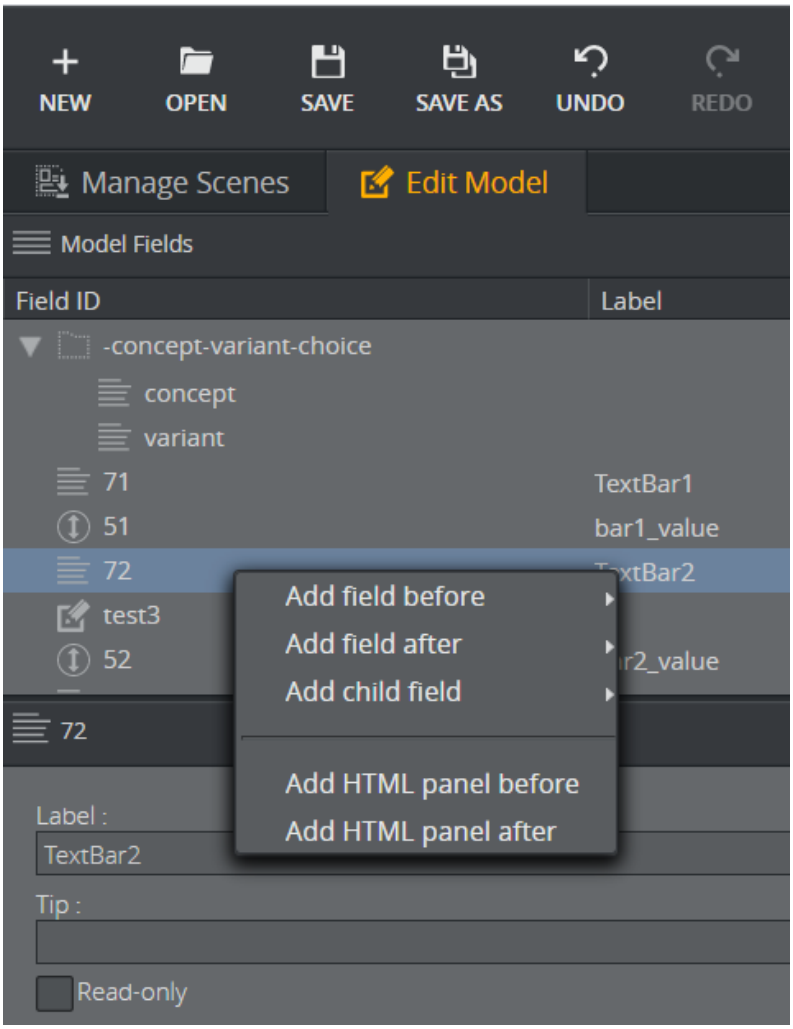
You can add an HTML panel to the template as part of the template customization workflow, giving you full control through custom scripting and logic when building the template.

> **Note:** The template can host a web page if you enter a web address in Template Builder. The aim is usually not to host a single web page, but rather to make a data integration HTML template.

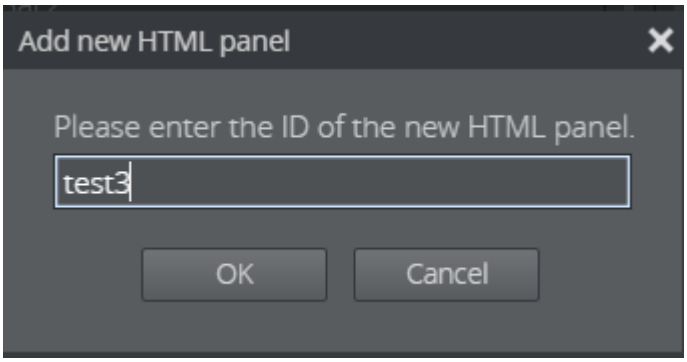
> **Note:** Creating HTML Templates contains examples of how to use HTML templates.

4.4.1 Adding an HTML Panel

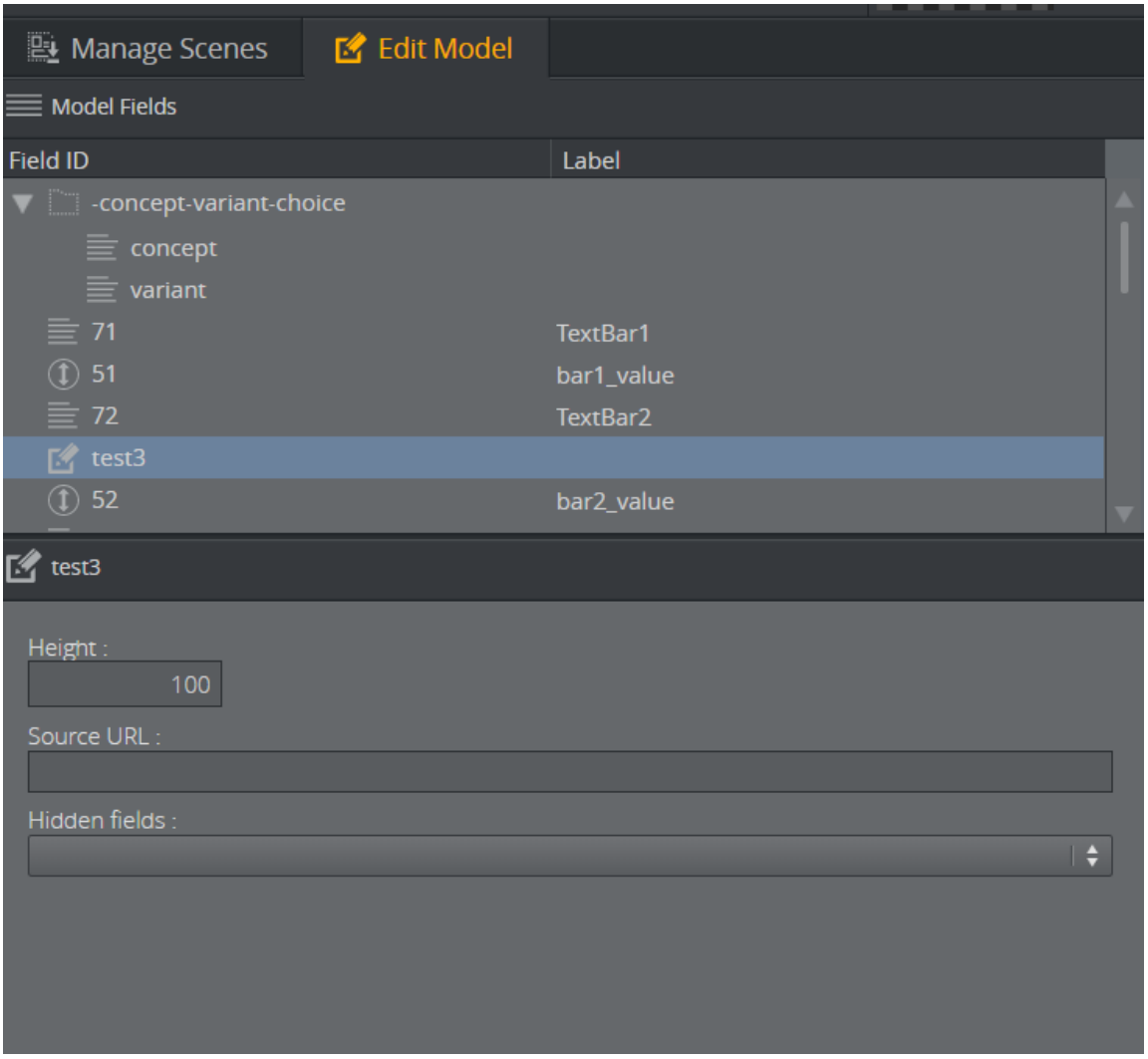
- Right-click the field ID list and select **Add HTML panel before/after**:



- Enter the ID of the of the new panel in the dialog that appears:



- Select the new **html-panel** field ID in the field tree to show its properties in the Field Properties window:



- Adjust the size of the HTML panel shown in the Fill In Form using the **Height** field.
- In **Source URL**, enter the web address.
- The **Hidden fields** drop-down list allows the user to hide available fields in the Fill In Form.

4.4.2 Browser Caching

You may experience browser caching behavior when trying to update and display changes in the custom HTML template in Template Builder; this is standard behavior. Template Builder does not control caching resources included in the HTML file itself. To prevent caching:

1. Ensure the URLs to the resources are unique upon reload.
2. Optionally configure the web server serving the resources to send Expiry headers set to 0.
3. Disable caching on the browser side.

> **Note:** A detailed description of how caching works is unfortunately beyond the scope of this documentation.

4.4.3 Creating HTML Templates

A few examples of how to create HTML templates are shown below:

- Setting Up a Simple Custom HTML Template
- Connecting a Custom HTML Template to a Viz Pilot Template
- Connecting a Custom HTML Template to a Viz Pilot Template - Advanced
- Creating a List of Functions Where You Can Bind Fields
- Redesigning Concept/Variant Fields
- Controlling the Auto-generated Fill In Form from the HTML Template

Note: To fully understand the workflow in these examples, some basic knowledge about web technology (javascript, HTML, CSS) is required. Although detailed description of the content in the files used will not be provided, API documentation bundled with the product describes the API to which a web developer creating custom HTML templates has access. This can be found at *http://<pilotdataserverhost>:8177/app/templatebuilder/js/doc/index.html*.

Note: jQuery is used in the examples for brevity, but is not mandatory when creating your own HTML template.

Note: The following examples are a proof of concept and show only some of what can be done using the customized workflow; more advanced use allows full control of the template using custom scripting and logic. More samples can be found at *http://<pilotdataserverhost>:8177/app/templatebuilder/samples/html_panels*.

- Use simple HTML <input> or <textarea> fields that contain an id="field_<fieldid>" that will automatically have a bi-directional connection.
- Take control of function mapping, and use JavaScript to do virtually whatever you wish.

Setting Up a Simple Custom HTML Template

The example below uses a template that shows the message **Hello world** when opened in a browser.

The following three files, including the HTML template, must be located in the same folder (*C:\Program Files\Vizrt\Pilot Data Server\app\mytemplates*):

1. **customTemplate_sample.html:** The custom HTML template.

```
< script  src = "https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/
jquery.min.js" ></ script >
< script  type = "text/javascript"  src = "./payloadhosting.js" ></
script >
< script  type = "module"  src = "./customTemplate_sample.js" ></
script >
```

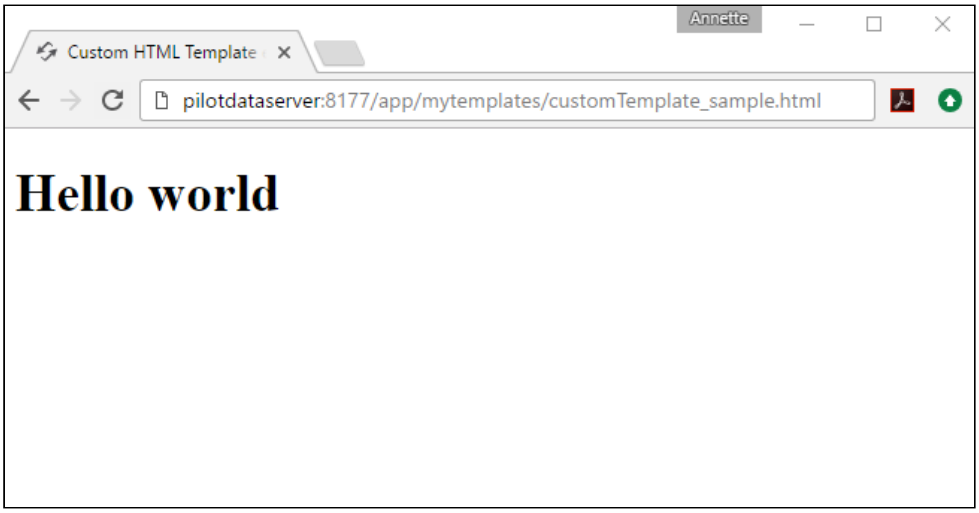
```
< head ></ head >
< body >
    < h1 >Hello world</ h1 >
</ body >
```

2. **customTemplate_sample.js:** The JavaScript part of the template.

```
$(document).ready( function  () {
    console.log( "Hello world" );
});
```

3. **payloadhosting.js:** This connects everything. (Get it from http://*<pilotdataserverhost>*:8177/app/templatebuilder/js/payloadhosting.js).

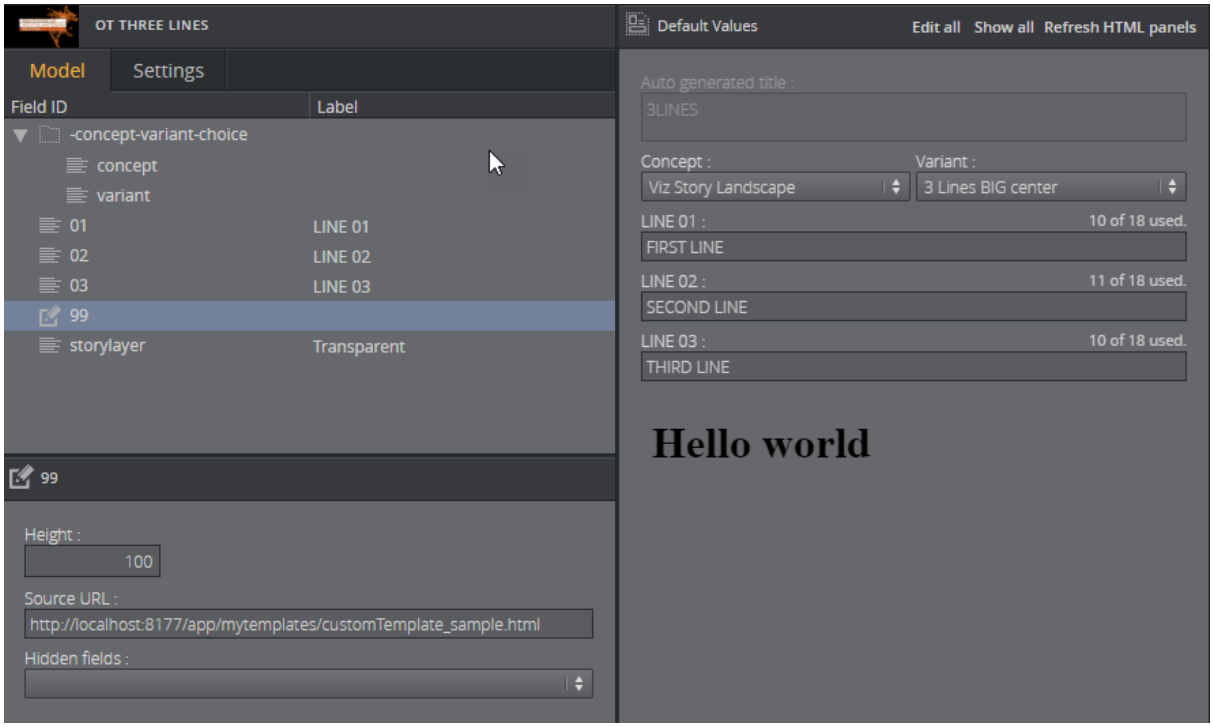
The URL in the image below points to the location of the *.html* file mentioned above:



Viewing a Custom HTML Template in Template Builder

- Open a template and add an HTML panel as described here.
- In the URL field, enter the URL of the custom HTML template. In this example, the URL from the picture above.

- **Hello world** now appears in the Fill In Form:

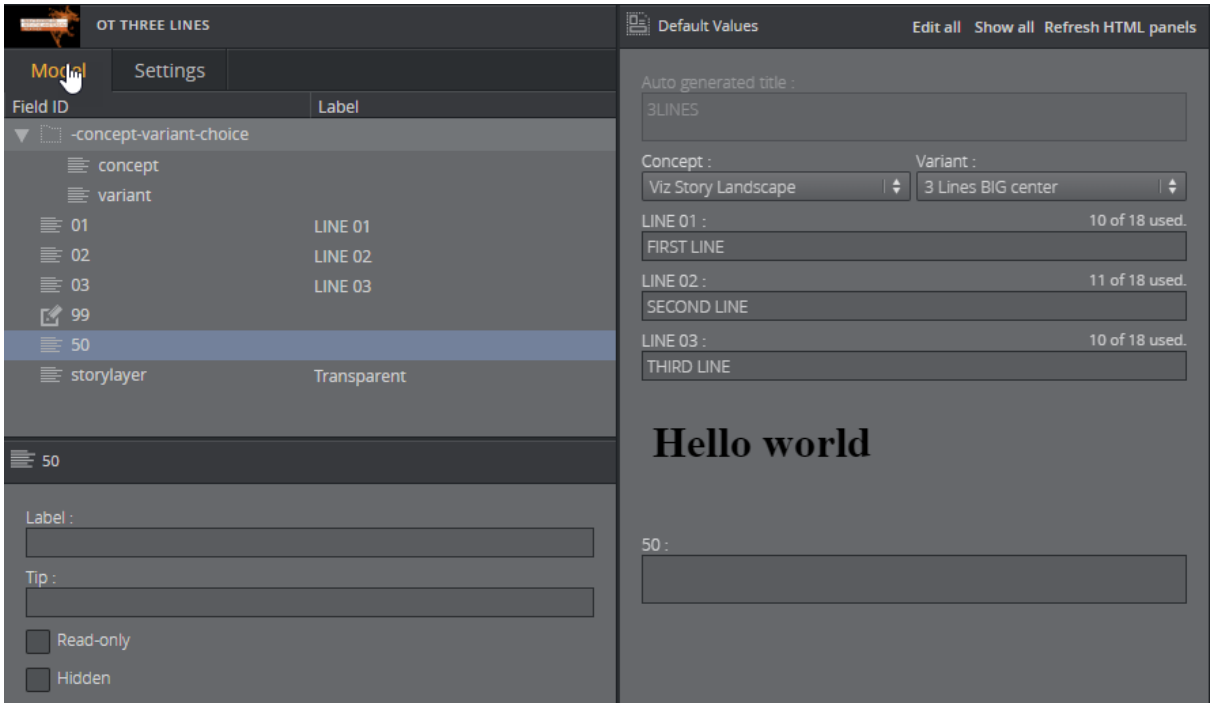


Connecting a Custom HTML Template to a Viz Pilot Template

Following the example above, we can establish a two-way communication, or *bind fields*, between the HTML template and the opened pilot template. This provides a simple way of setting up a binding field. Add a new field to the template:

- Right-click in the HTML panel field, choose **Add field before/after** and select **Plain text.**
- Give it the ID **50.**

- A new field appears:



The <body> block in the custom HTML template (*Template_sample.html*) that previously contained:

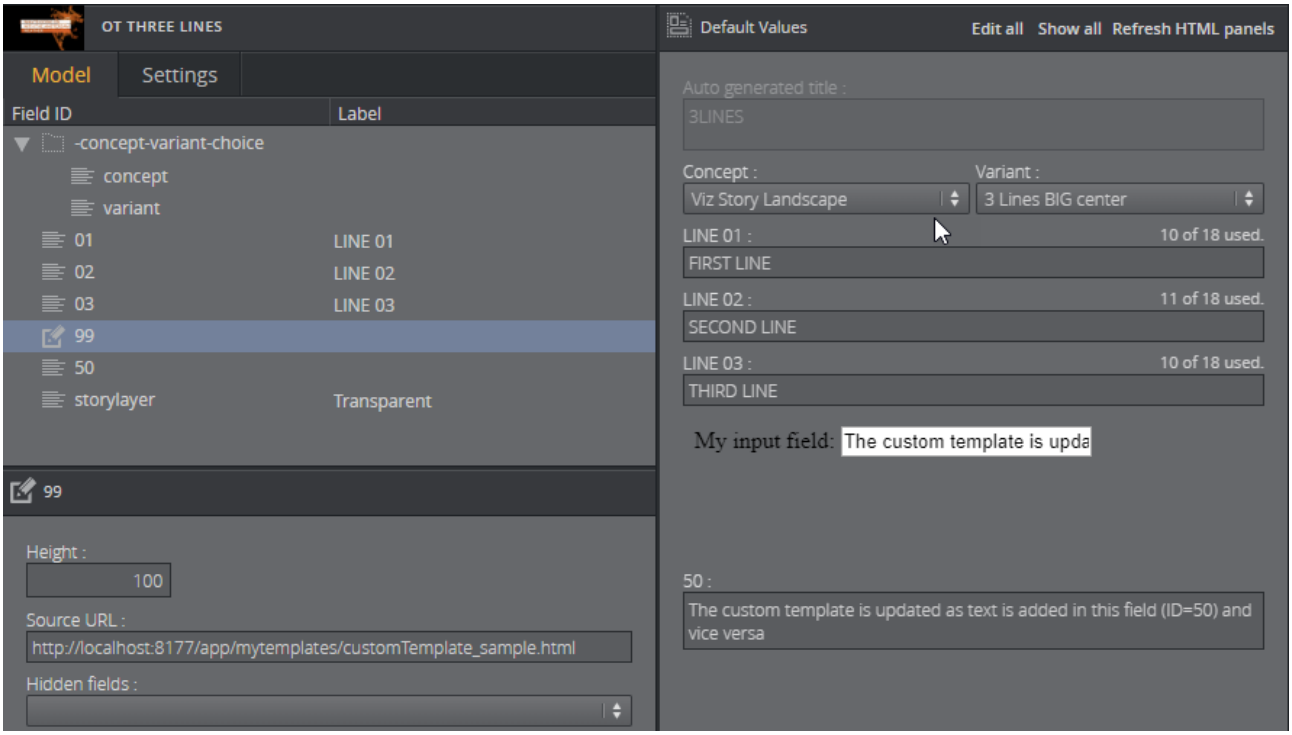
```
<body>
    <h1>Hello world</h1>
</body>
```

must then be replaced with:

```
<body onload= "vizrt.payloadhosting.initialize()" >
    <label  for = "field_50" >My input field:</label>
    <input name= "field_50"  type= "text"  id= "field_50" >
</body>
```

Saving the HTML file and clicking **Refresh HTML panels** reloads the custom HTML template with the changes just made. A bi-directional connection between the custom template and pilot template has now been established. If you now type inside either the template or the field with ID 50, both fields are updated at the same time.

> **Note:** This way of binding fields works for any HTML fields that have value support, typically `<input>` types and `<textarea>`.



The JavaScript file automatically seeks input elements in the HTML that match the ID of fields inside the template. Adding the `id="field_50"` to the <input> element inside the HTML template is all that is needed for the two-way communication to be set up since a field with ID 50 was added above. An unlimited number of these binding fields can be established in the exact same way, since they are mapped via ID.

> **Note:** Updating the `<input>` elements programmatically still sends data back and forth, which is useful for automated data integration such as fetching live sports data.

> **Tip:** Use the **Hidden fields** setting inside the HTML panel settings to prevent two editors for the same field being visible at the same time.

Connecting a Custom HTML Template to a Viz Pilot Template – Advanced

The following example will go more into detail than the example above, and use a more scripting to give you 100% control over the template. The three files mentioned in the Setup a simple custom HTML template example are also used here.

Creating a List of Functions Where You Can Bind Fields

By adding the following above the *document.ready()* function in the *customTemplate_sample.js* file:

```
// Will be called when the field with id "50" changes
function on50Changed(value) {
```

```
}
```

and the following inside the $(document).ready function:

```
var pl = vizrt.payloadhosting;
pl.initialize();
pl.setFieldValueCallbacks({  "50" : on50Changed });
```

you set up a way for a custom JavaScript function to be called upon detecting a change. When field_50 receives a change from the host, the function will be called with its new value as a parameter.

Some changes will be made to the HTML file below to demonstrate that we can use custom HTML/JavaScript to do something with these values.

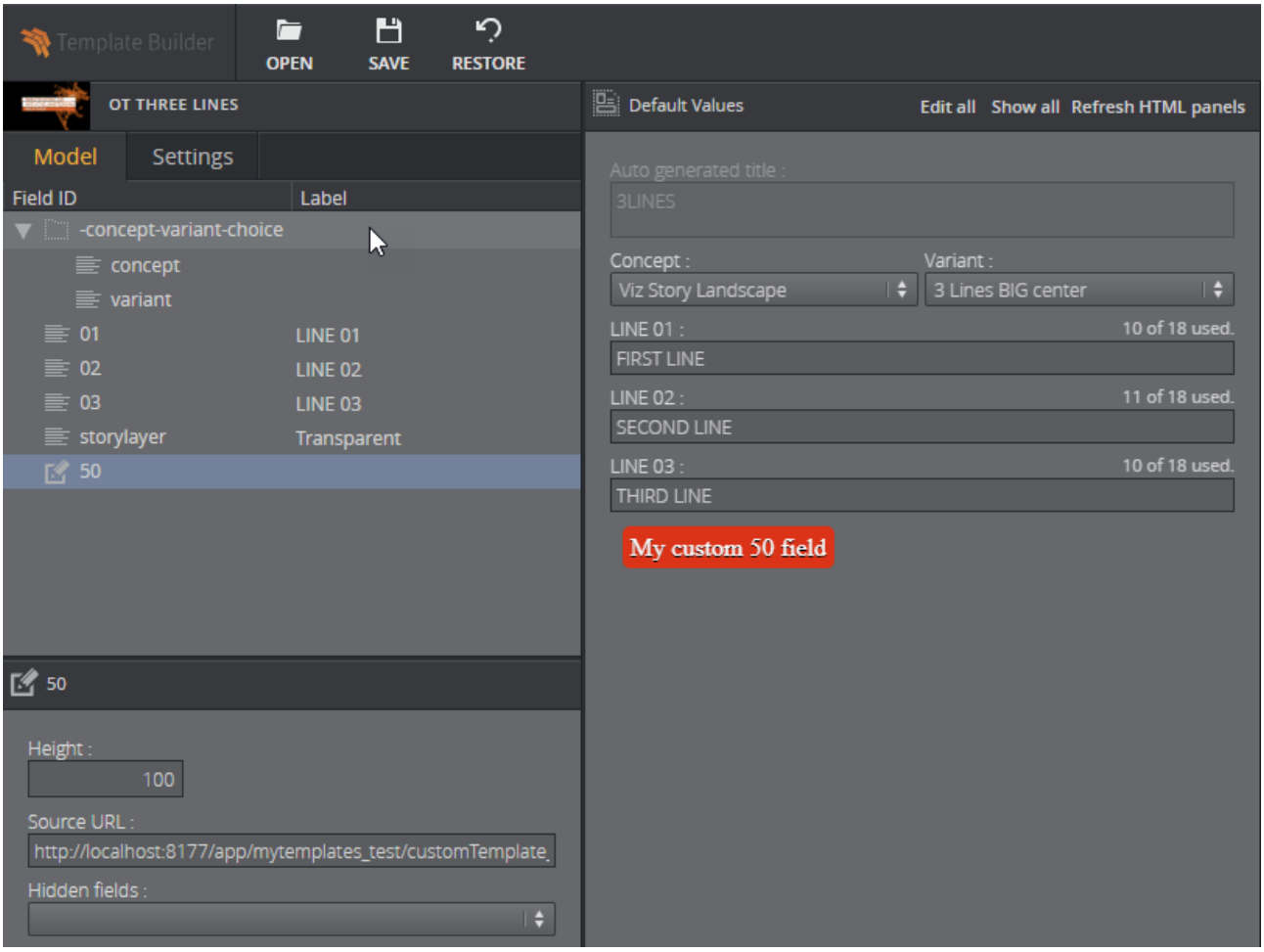
Inside the HTML file, the entire body is replaced with:

```
<body>
    <span id= "myfield"   class = "sample red" >My custom  50  field
</body>
```

To add some CSS to style the text, add the style tag after the closing </head> tag and before the <body> tag:

```
<style>
     .sample {
          padding:5px;
          color:white;
          border-radius:5px;
          text-shadow: 0  1px  0  black;
          background:red;
    }
   .green {
          background:green;
    }
</style>
```

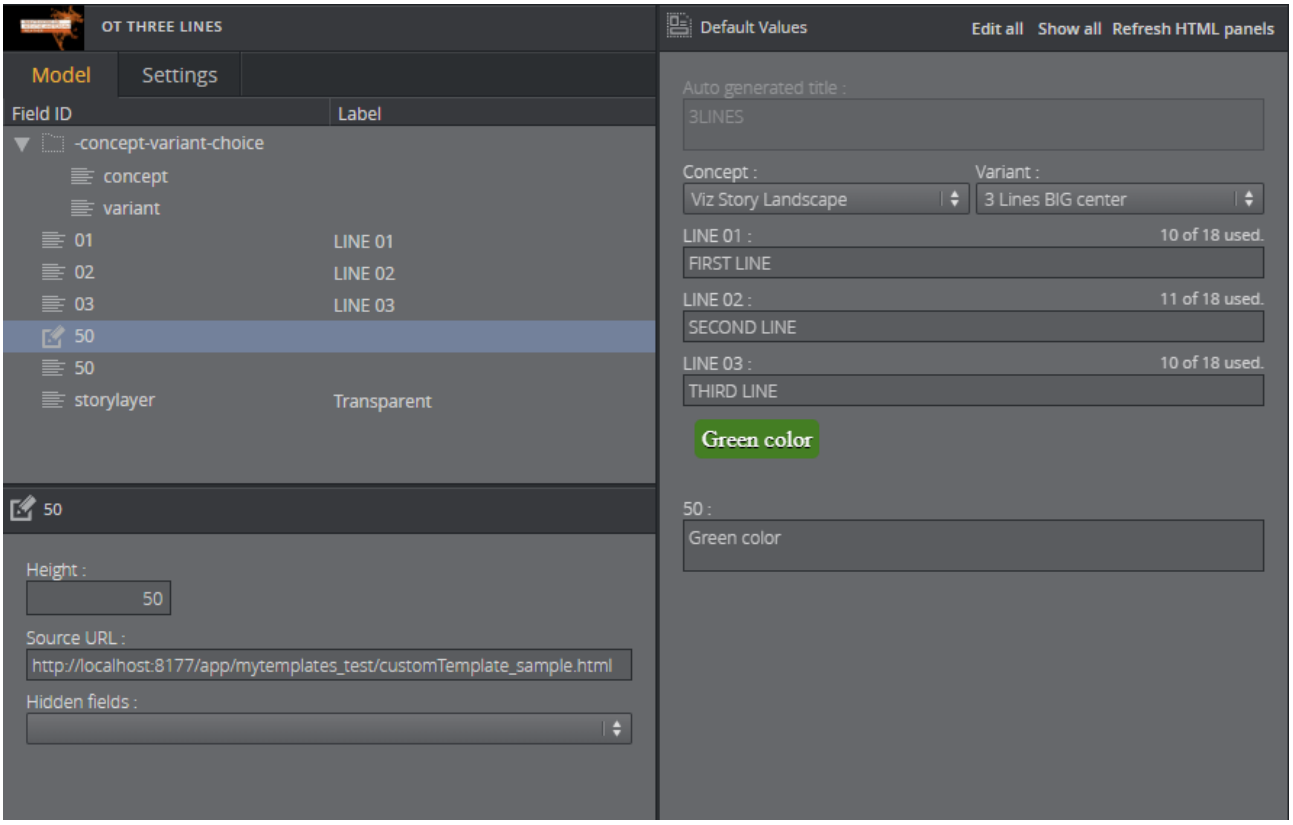
This provides the following output in Template Builder:



Adding a bit more custom logic, we will make the background color green when there is a text value that is longer than five or shorter than 20 characters. The function is expanded by adding the following function:

```
function on50Changed(value) {
     var myField = $("#myfield");
     myField.text(value);
     if (value.length > 5 && value.length < 20) {
              myField.addClass("green");
     } else {
              myField.removeClass("green");
     }
}
```

After refreshing the HTML panel, the background color should change to green dynamically when typing.

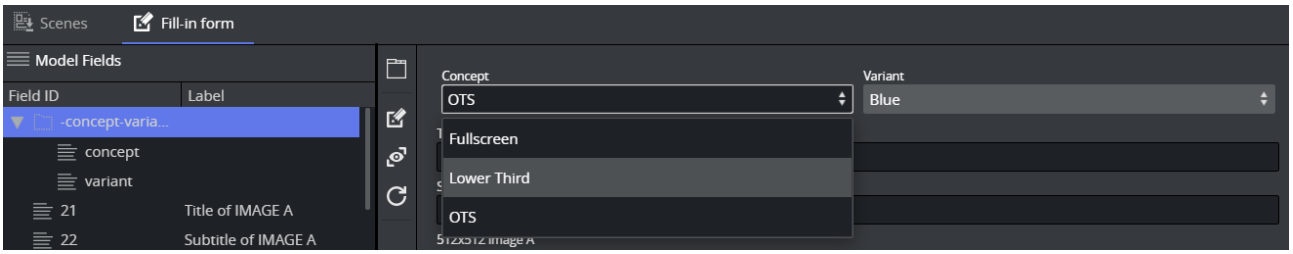


Redesigning Concept/Variant Fields

This example shows how to present the concepts and variants in a template in a different way.

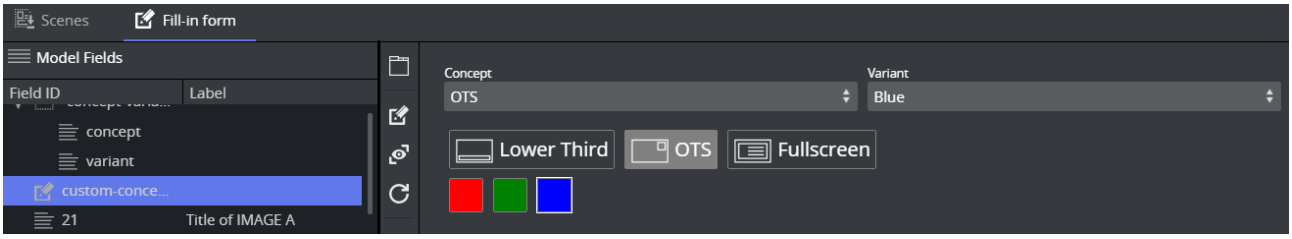
The full HTML / JavaScript code is available at *http://<pilotdataserverhost>:8177/app/templatebuilder/samples/html_panels/concept_variant.*

Let's consider a template with concepts **Fullscreen, Lower Third, OTS** and variants **Red, Green, Blue** available as drop-down lists in the Fill In Form:

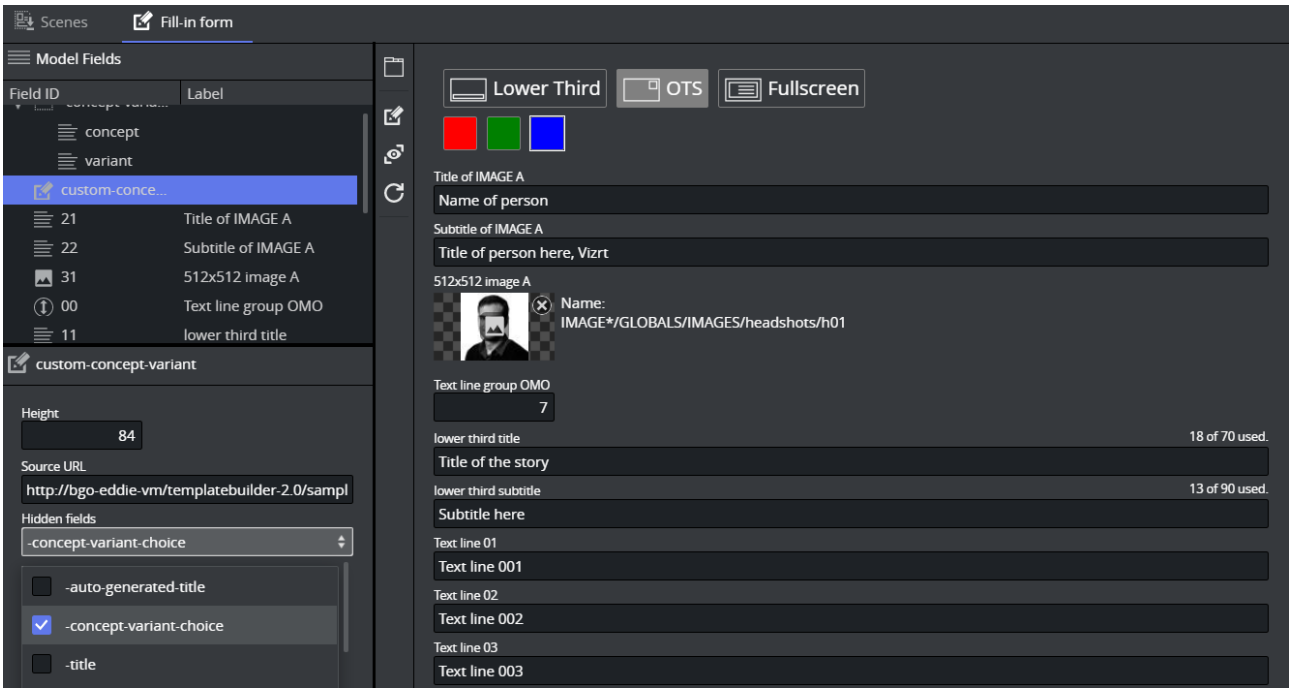


The field **-concept-variant-choice** actually contains 2 subfields, **concept** and **variant**.
You can access their value using slash "/" to navigate in the list. For example, to access the concept use **-concept-variant-choice/concept**.

By setting up the HTML panel hosted at *http://<pilotdataserverhost>:8177/app/templatebuilder/samples/html_panels/concept_variant*, the concepts and variants are now presented as buttons. This example has mutual binding support for both concept and variant - clicking on the new buttons updates the original drop-downs and vice versa:

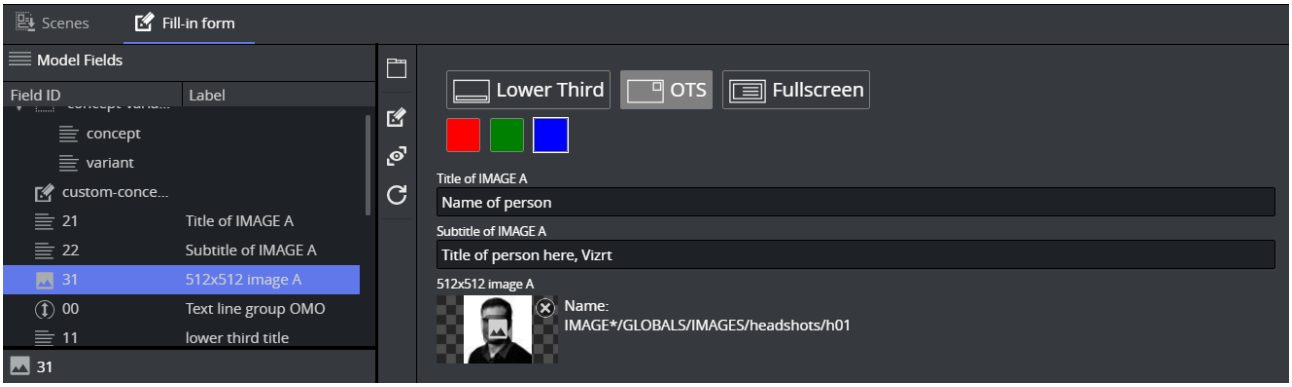


The drop-down lists are no longer needed and can be set as a **Hidden field** in the HTML panel properties window:



Controlling the Auto-generated Fill In Form from the HTML Template

It's possible to dynamically set visibility and read-only attributes, so you can filter the auto-generated form based on the custom HTML template.
In the following example, the **31** image field should only be visible when the **Fullscreen** or **OTS** concept is active:



In the JavaScript used in the example, there is a function called updateActiveConcept which is called when the concept changes.

Adding the following line inside the updateActiveConcept method block checks which concept is chosen. If it isn't **Lower Third**, it displays the field with ID **31** in the Fill In Form:

```
pl.setFieldVisibility( "31" , conceptValue !=  "Lower Third" );
```

If you now click on the **Lower Third**, the image field with ID **31** disappears, but is displayed if the **OTS** or **Fullscreen** concept is selected.

> **Note:** This is a powerful feature that lets you customize available editing options based on certain conditions set in the template.

4.5 Form Customization Scripts

If dynamic or advanced customization of the fill-in form is needed, Pilot Edge allows this through template scripting.

Scripts are written in TypeScript, and access the template via a provided Script API named `vizrt`.

It allows users to customize templates look and behavior, as well as fill in values from external sources. This section describes how to use the script editor, and access the values of the fields supported in Template Builder.

> Check out the Quick Start Examples for some quick hands-on experience.

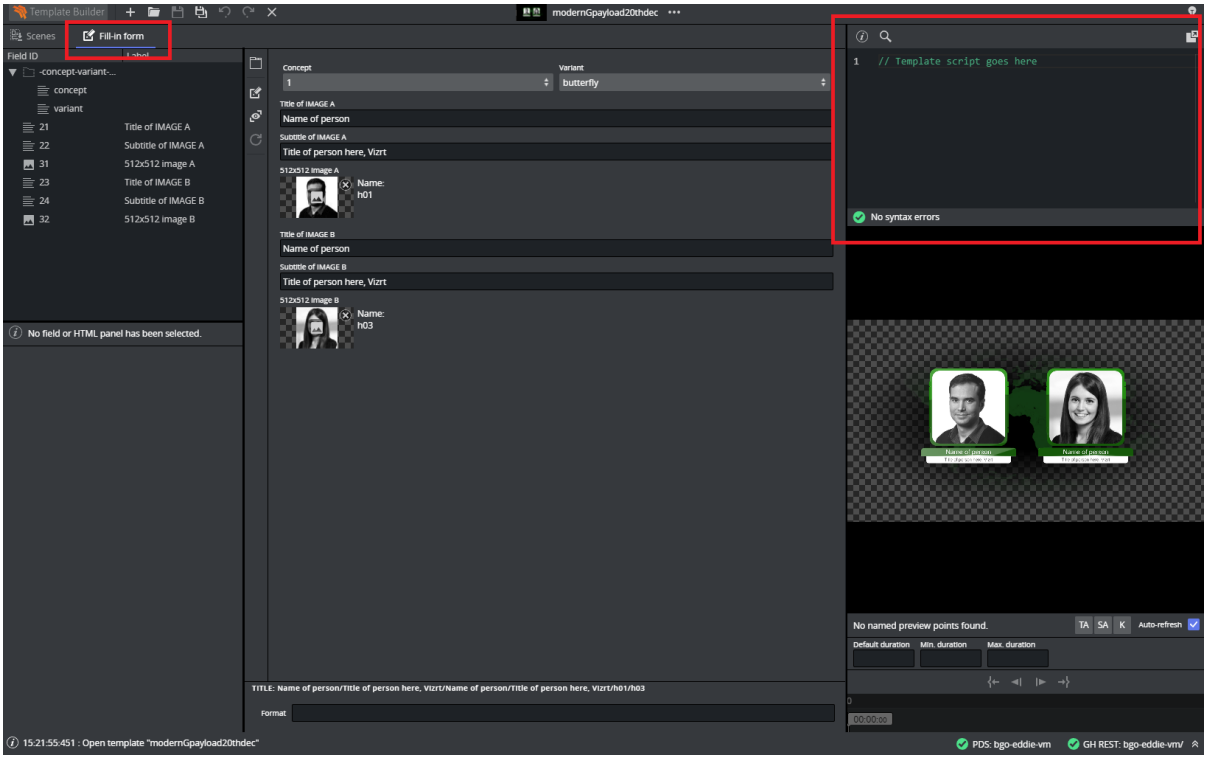
> **Note:** For testing API endpoints, please use the following:
> - HTTP://<PDS-HOST>:8177/testing/fakepremierleague/
> - HTTP://<PDS-HOST>:8177/testing/fakepersonsearch/

These are the following topics:

- The Script Editor
- Field Access
  - External Sources
  - Read Only Fields
  - Unsupported Fields

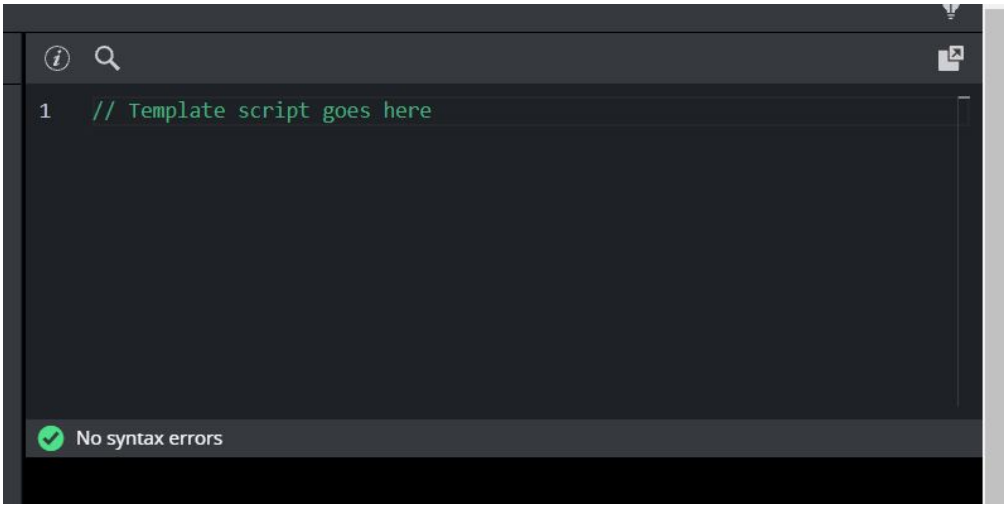
4.5.1 The Script Editor

The script editor is available in the "Fill-in Form" tab of Template Builder.



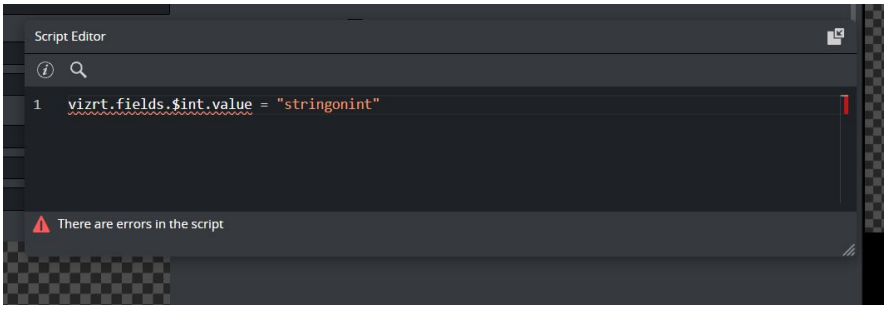
It can be used docked or undocked from the Template Builder. While undocked, you can adjust the size of the window for a smoother experience.

There is a search option to search within the script code, and you can access it by clicking the icon within the script editor.

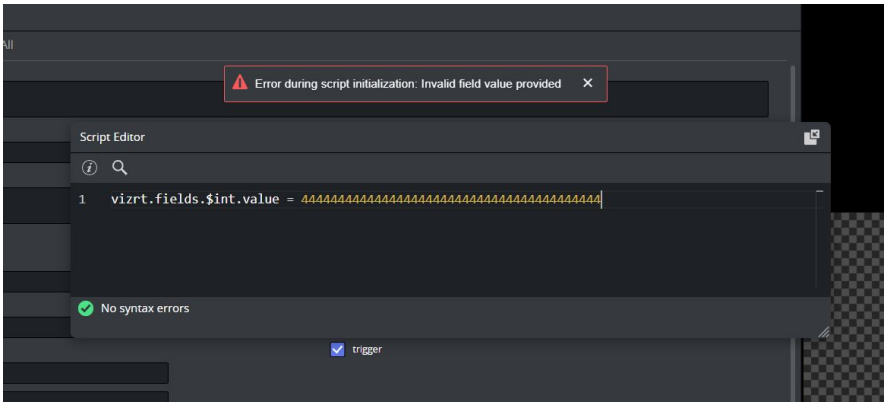


The script editor also provides error messages depending on the problem. It will show compile or run time errors.

Compile Error



Run Time Error



4.5.2 Field Access

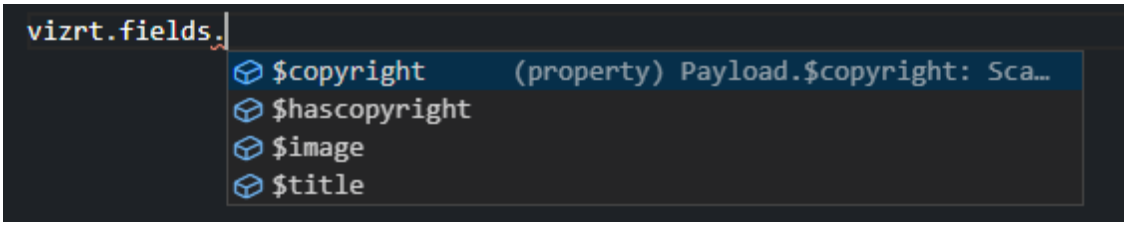
When using the scripting tool in the template, the individual fields must be accessed through the global name space *vizrt.fields* (for example, *vizrt.fields.$singleline.value*).

> **Note:** Writing *$singleline.value* instead of *vizrt.fields.$singleline.value* will not work, and will give a Compile Error.

The script executes when a graphic element is opened or created with the scripted template in Pilot Edge.

In Template Builder, the script is also re-loaded and restarted when there are changes made to it.

By typing *vizrt.fields*, the editor's autocomplete will show you the available fields to choose from.



You can read and write field values, as well as react to value changes from outside the script.

You can also access the properties **read-only** and **hidden** of the `vizrt` fields.

```
vizrt.fields.$image.
                    error          (property) EmptyField.error: string
                    hidden
                    onChanged
                    readOnly
                    tip
                    value
```

- **onChanged:** A property on fields that you can set as a function, and if you do so, this function is called whenever the value of the fields changes, and gets the new value as an argument. If this is not set, it will be *null*.

  > **Note:** Changes done to field values by the template script will *not* trigger the **onChanged** function to be called.

- **readOnly:** Read and write *boolean* access, to whether the field should be editable in the form or not. If *false*, the field and its input elements are editable in the UI. If *true*, they are read-only and greyed out in the UI, but are accessible, saved and loaded as part of the payload.
- **hidden:** Read and write *boolean* access, to whether the field should be editable in the form or not. If *false*, the field and its input elements are present and visible in the UI. If *true*, they are hidden from the UI but are accessible, saved and loaded as part of the payload.

> **Note:** Dashes cannot be used in Typescript with the dot syntax, instead you can use *vizrt.fields["$01-week"]* syntax to be able to access it.

External Sources

Whether on template load, or as a reaction to a field change, you can initiate HTTP, HTTPS or REST calls to fetch values from third party or external services.

This can easily be done via the browser's built-in *fetch* API: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API.

> See the Quick Start Examples section for a short example of a REST call triggered by an *onChanged* event.

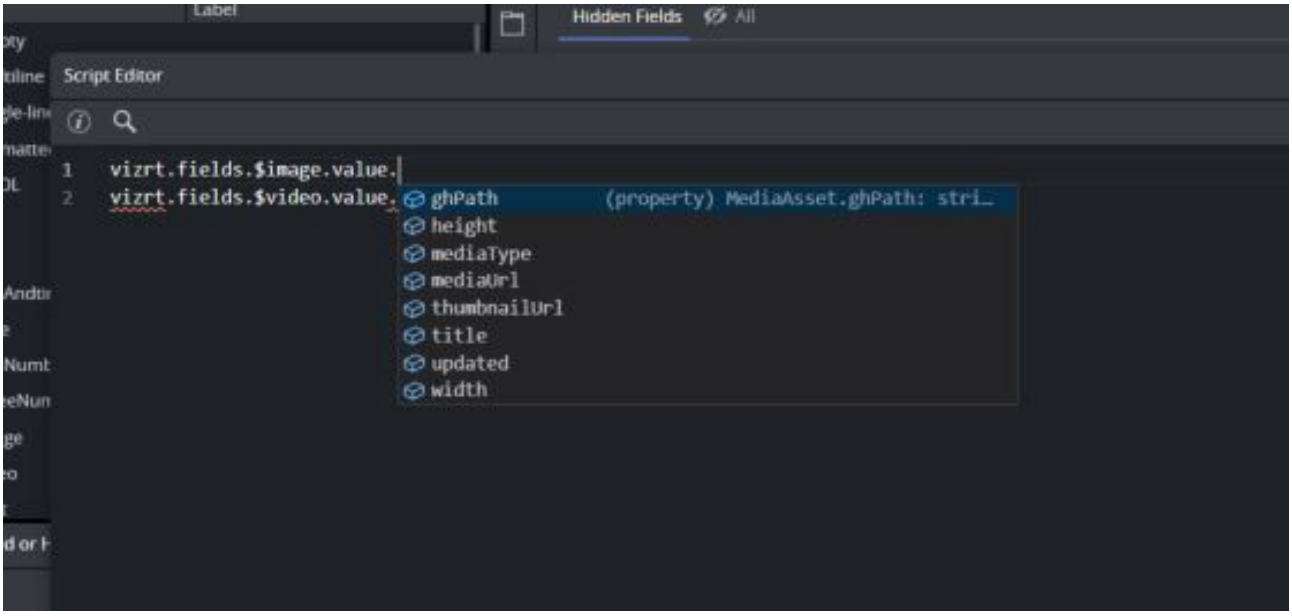
Read Only Fields

Some fields are currently supported only for read-access by the scripting API. These are the following:

- Duplet
- Triplet
- Map
- Image

- Video

For the **image** and **video** fields, the script is able to access some metadata from the file. These can be explored through the editor's autocompletion:



Unsupported Fields

As of now, all **List** and **Table** fields are unavailable from the scripting API.

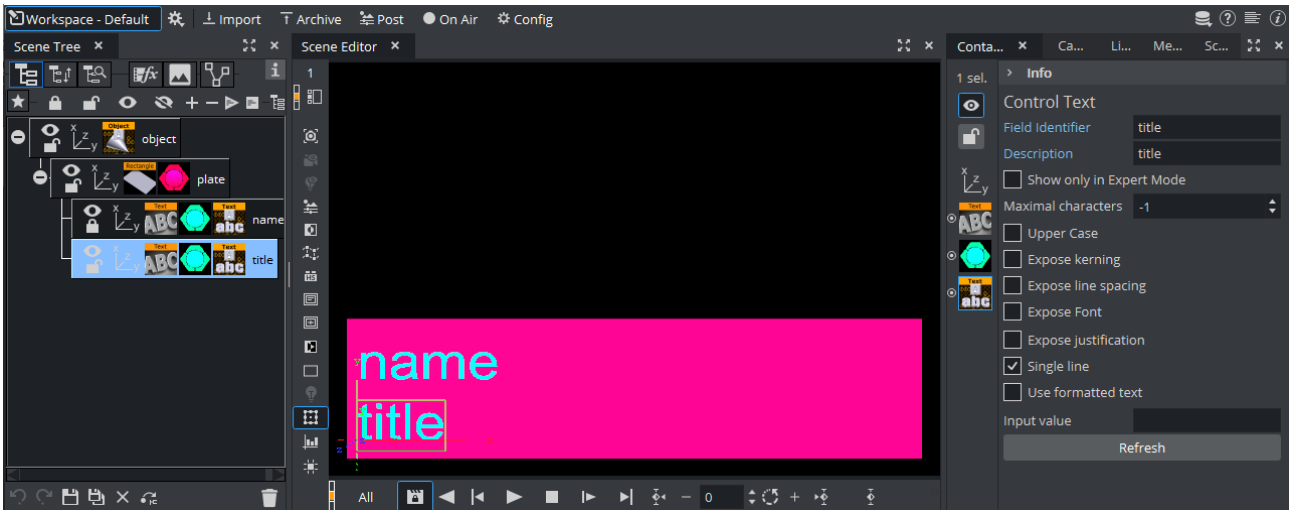
4.5.3 Quick Start Examples

> ⓘ In this section you can find short examples of how to use the scripting functionality.

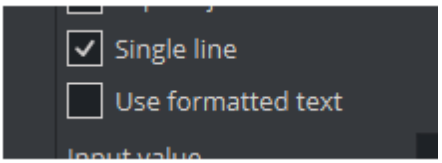
- Automatically Clear Title Field
- Fetch Title from REST Service

In Viz Artist, create and save a regular Pilot template scene with two text control fields:

- name
- title



Make sure to uncheck **Use formatted text** in the Control Text properties for both fields, which is easier to work with.

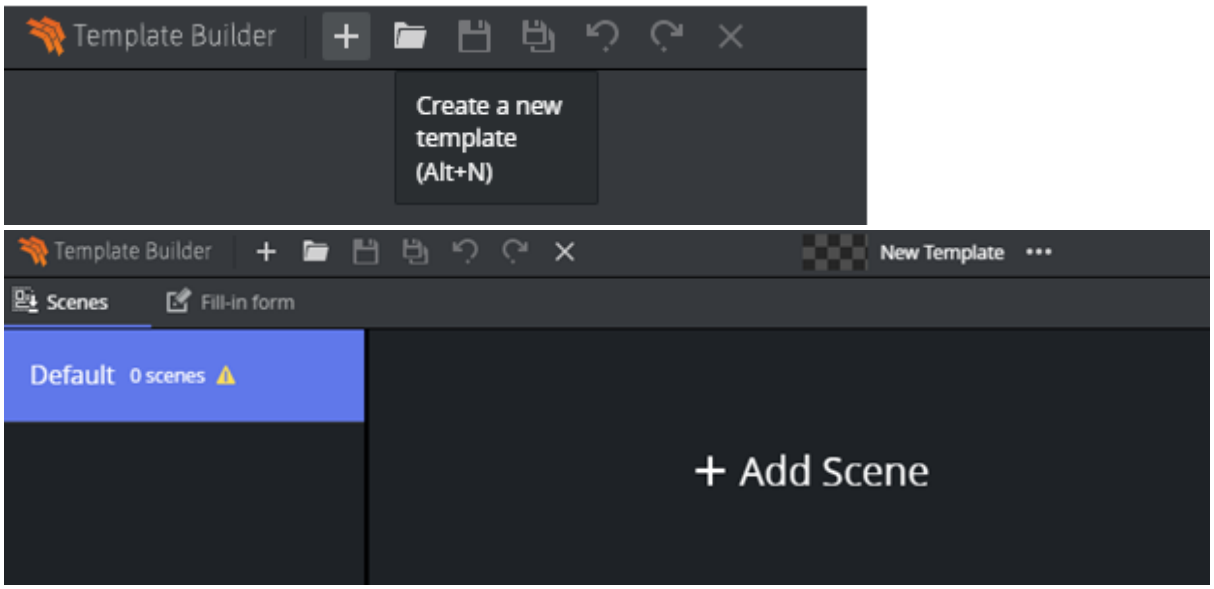


Automatically Clear Title Field

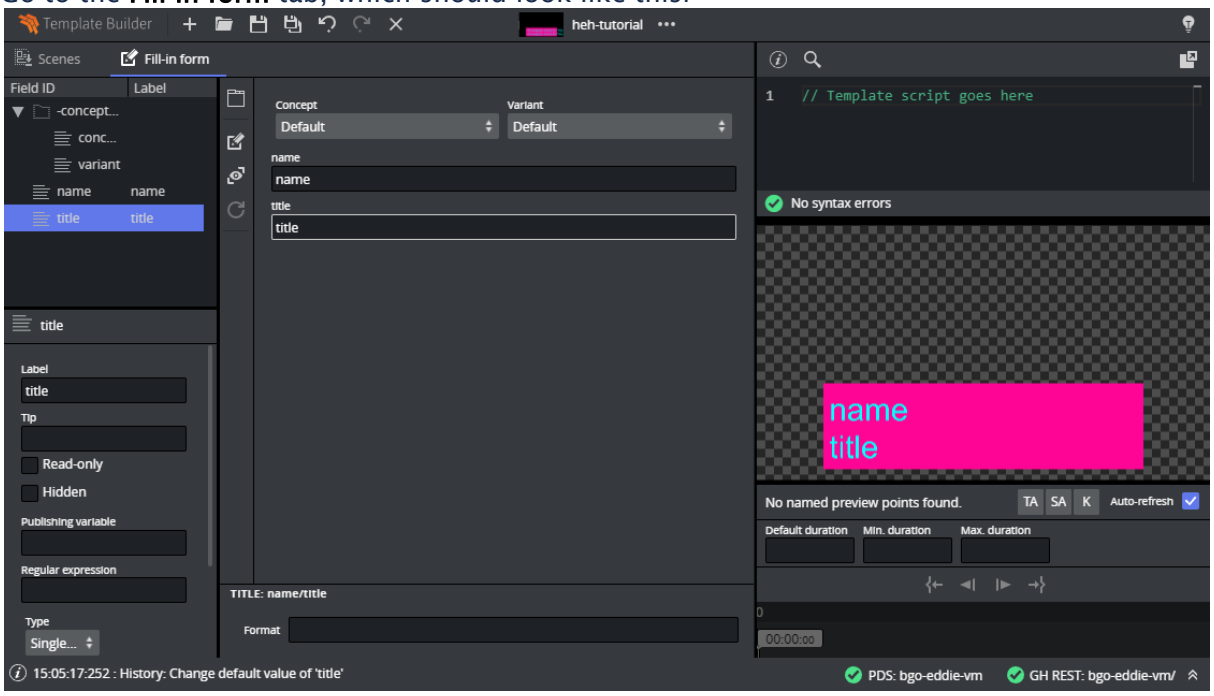
In this example, the following basic features are shown:

- Script that executes on template load.
- Reacting to user changes to the fields.
- Modifying fields from the script.

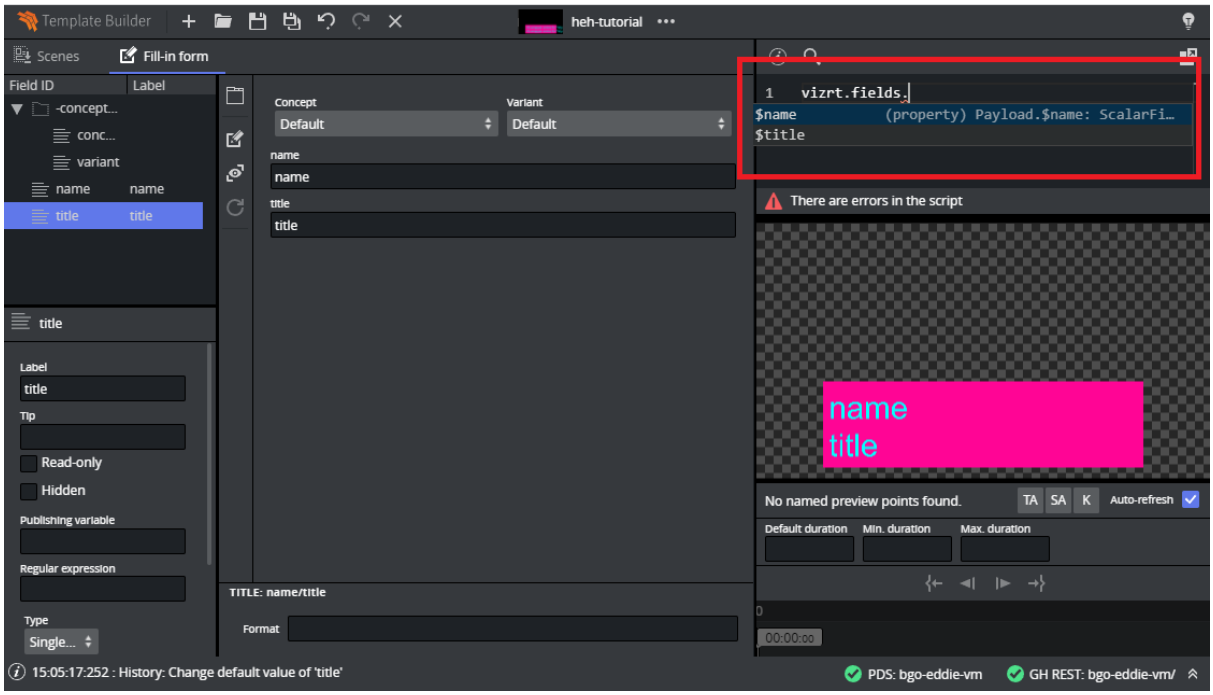
1. Create a new template based on this in Template Builder, by choosing **Create a new template** and adding your newly created scene via "Add Scene".



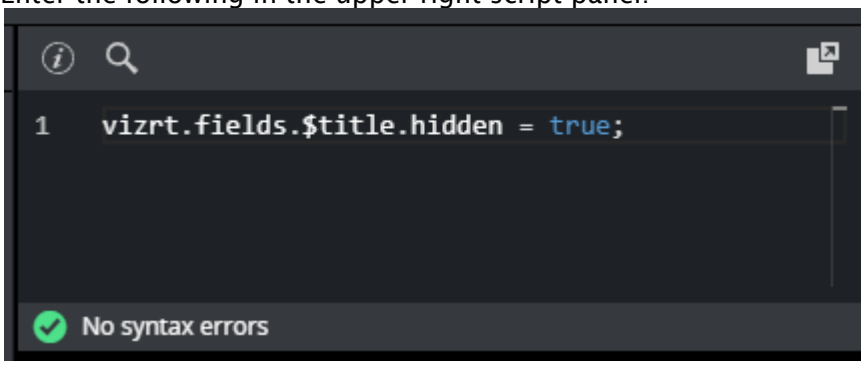
2. Go to the **Fill-in form** tab, which should look like this:



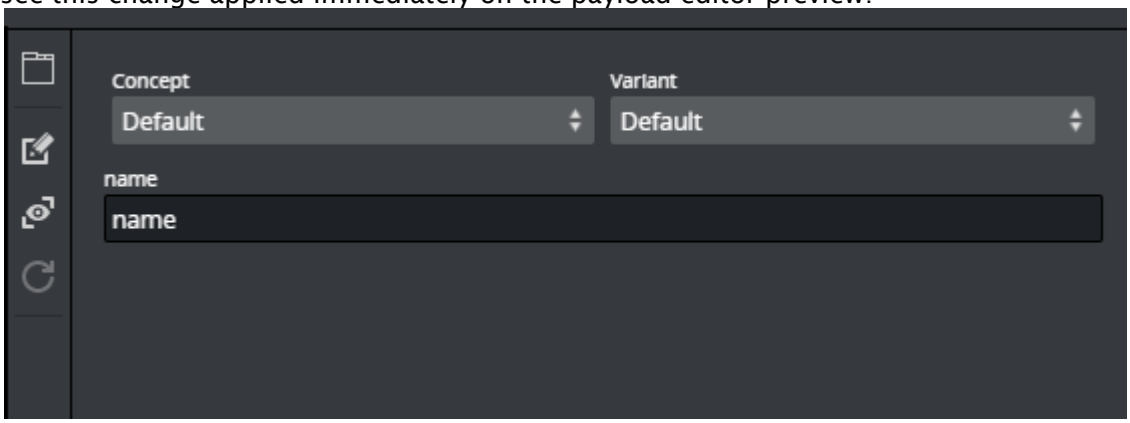
3. In the scripting tab, you can verify that the template fields are available by typing *vizrt.fields* and looking at the autocompletion:



4. Enter the following in the upper right script panel:



This script causes the template to hide the **title** field when the template loads. You should see this change applied immediately on the payload editor preview:



5. The script is now extended to react to changes made in **name**. When *name* is empty, *title* should be hidden and empty, but not otherwise:

```
vizrt.fields.$title.hidden = true;

vizrt.fields.$name.onChanged = (name) => {
  if (name == "") {
    vizrt.fields.$title.value = "";
    vizrt.fields.$title.hidden = true;
  } else {
    vizrt.fields.$title.hidden = false;
  }
}
```

No syntax errors

   The *title* field will now hide and clear when *name* is cleared and reappear when something is entered into name.
6. **Save** the template and observe this action in the Pilot Edge client.

> **Note:** Dashes cannot be used in Typescript with the dot syntax, instead you can use *vizrt.fields["$01-week"]* to be able to access it. This means, when creating a new scene, you should use camel case notation or underscore (for example, *01thisIsMyField* or *01_week*), to access the field with the dot syntax.

Fetch Title from REST Service

In this example, the script will automatically fill the **title** field by fetching it from a REST endpoint.

This illustrates more advanced features:

- Using the standard browser fetch API.
- Changing field values based on responses from other services.

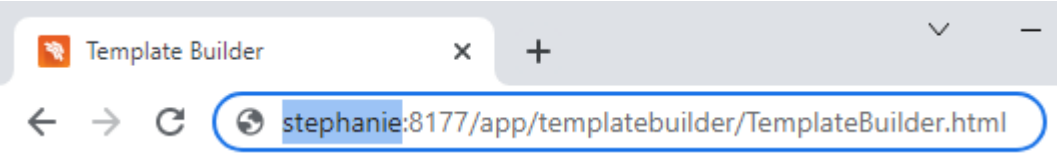
Using the same template as the example above, or creating a new one from the same scene, delete anything in the script tab and write the following:

```
vizrt.fields.$name.onChanged = (name) => {
  fetch("http://HOSTNAME:8177/testing/fakepersonsearch/" + name)
  .then(response => response.json())
  .then(title => {
    vizrt.fields.$title.value = title;
  });
}
```

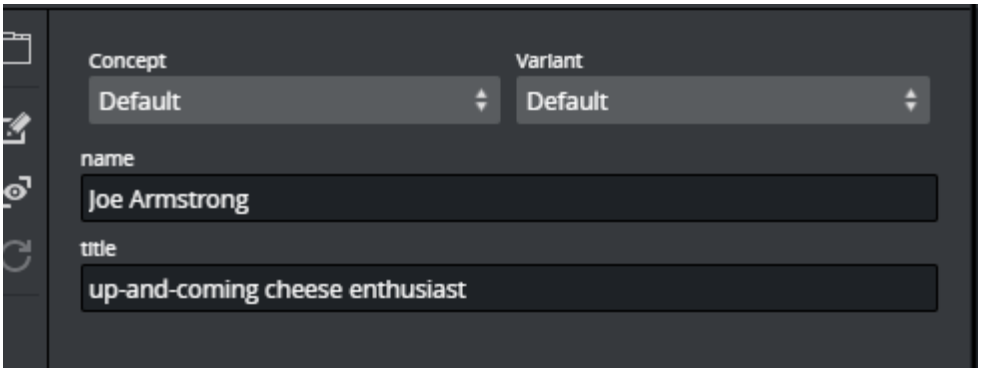
No syntax errors

Remember to replace "HOSTNAME" with the hostname of your Pilot Data Server. In a default install, this should be showing in your Template Builder address bar.

In the following example using Google Chrome, the hostname would be "stephanie", marked in blue:



If everything is working correctly, you should see an autogenerated title appear when you set or change the **name** field.



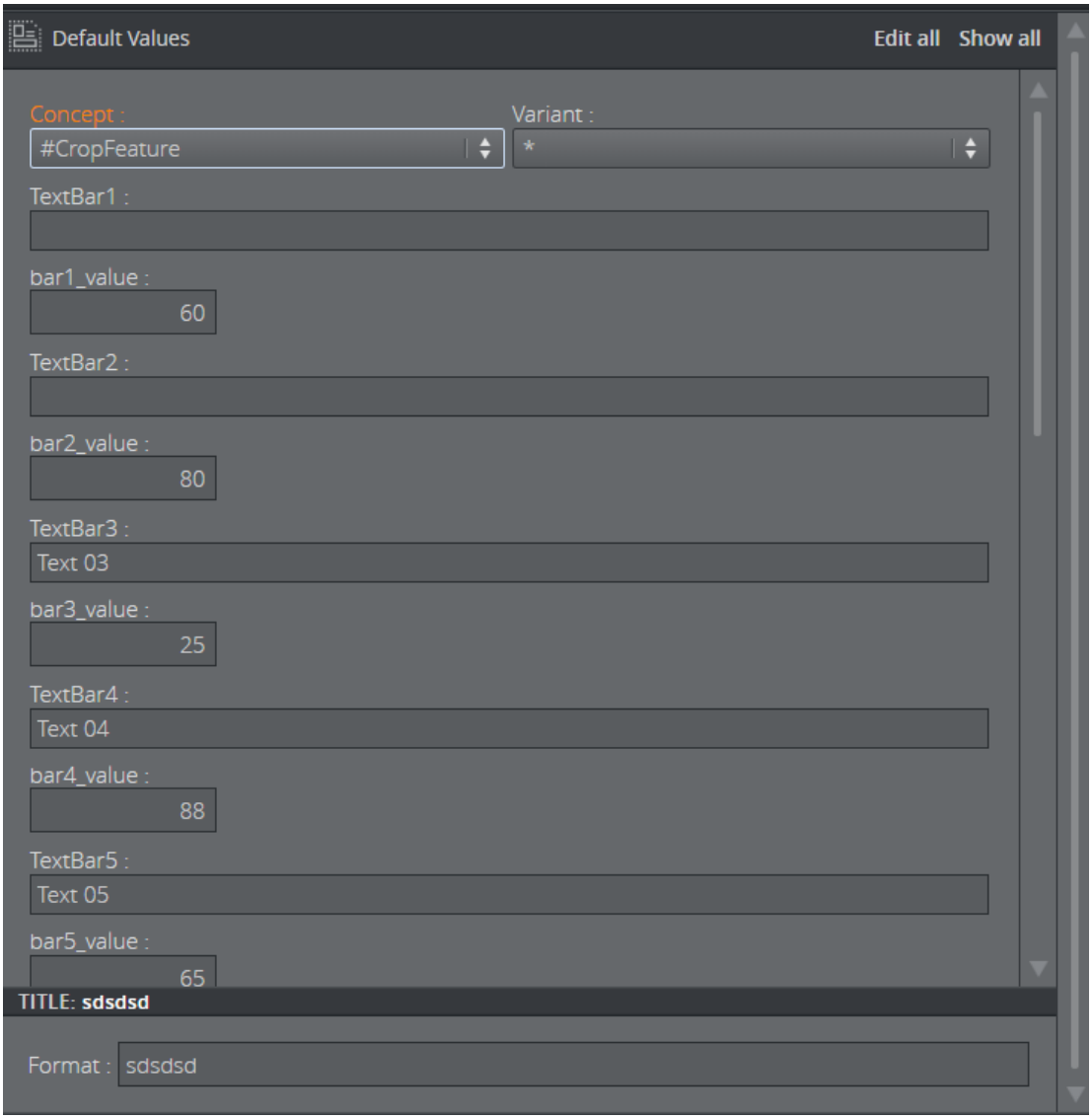
In this case, a specially provided test endpoint was used on the Pilot Data Server, but you can point to any other REST resource. Also, you are not constrained to the fetch API used. All standard JavaScript network mechanisms can be used.

5 The Fill-In Form

Default Values in the middle of the interface contains an auto-generated form for the graphics template (the Fill In Form).

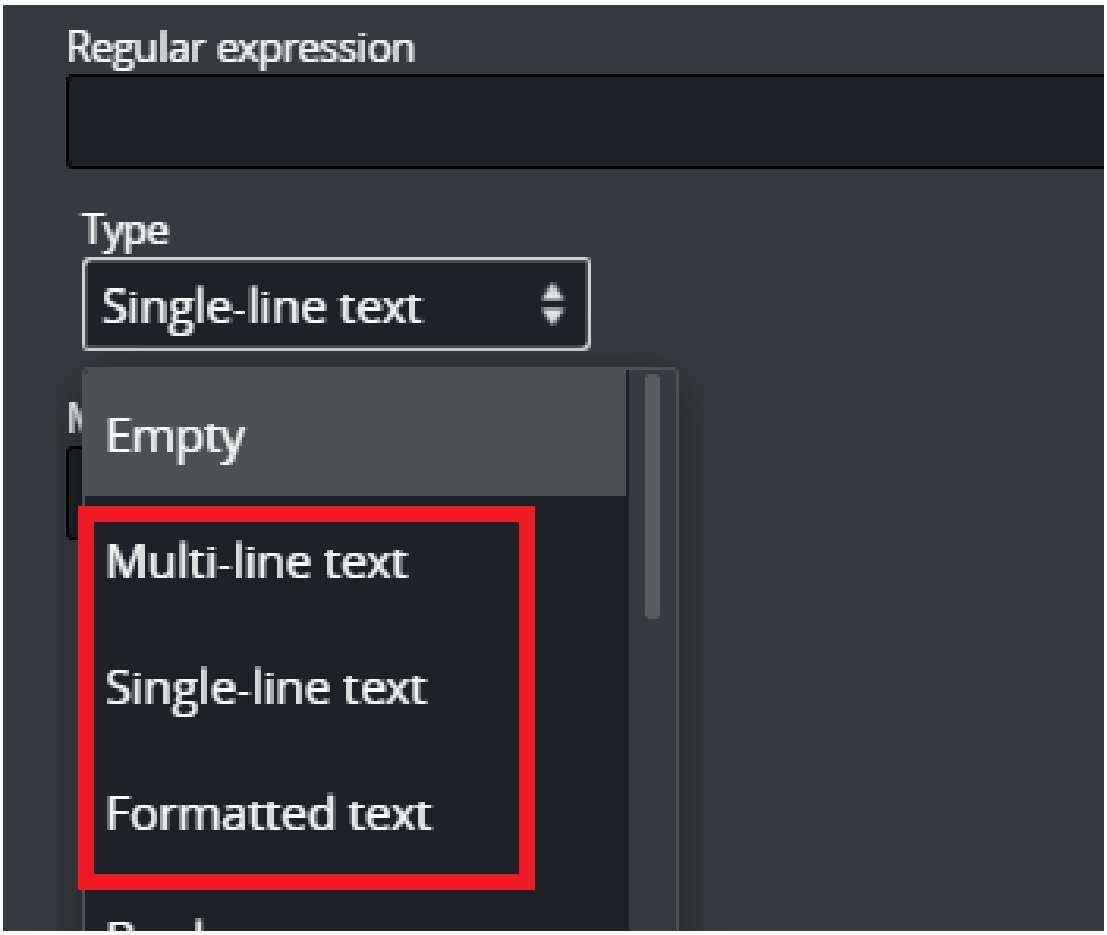
- Add content in the editable fields.
- The type of content allowed in the fields is set in the **Edit Model** window.
- Fields can be restricted: for example, to only include text with a certain amount of characters, numbers within a specific range, or media placeholders for media assets, or be displayed as options in a drop-down list.

> **Info:** Content added will be shown in the Preview Window if the fields in the Fill In Form are exposed controls made by the template designer.



- **Edit all:** Allows editing default values of fields that are set to read-only.
- **Show all:** Displays all fields in the form, including those in hidden mode.

5.1 Text Fields



- **Multi-line text:** Multi-line text supports standard ASCII characters. It does not support any type of text formatting and does not convert any text. It keeps the text as it was typed.
- **Single-line text:** A single-line text field converts any white-space to space. White-space includes space, tab, newline, etc. Single-line text converts any white-space to the space character you get by pressing the spacebar KEY only. For Template Builder, this is also a text field with a single-line entry, unlike multi-line text.
- **Formatted text:** Formatted text refers to the ability to hold formatted text. For example, a formatted text field can show that some of the texts are bold or italic, for example, when a field has Rich-Text functionality.
  Although such a display is not yet completely supported (no Rich-text support yet) on our payload text field. Formatted text is used so that if a field has a formatted type text created in Artist, the field type can also be selected in Template Builder.

5.1.1 See also

- Editing Template Layout

6 Editing Template Layout

Editing a template's layout makes it easy to create fill-in forms for journalist. With drag & drop functionality, creating new fill-in forms is quick and easy.

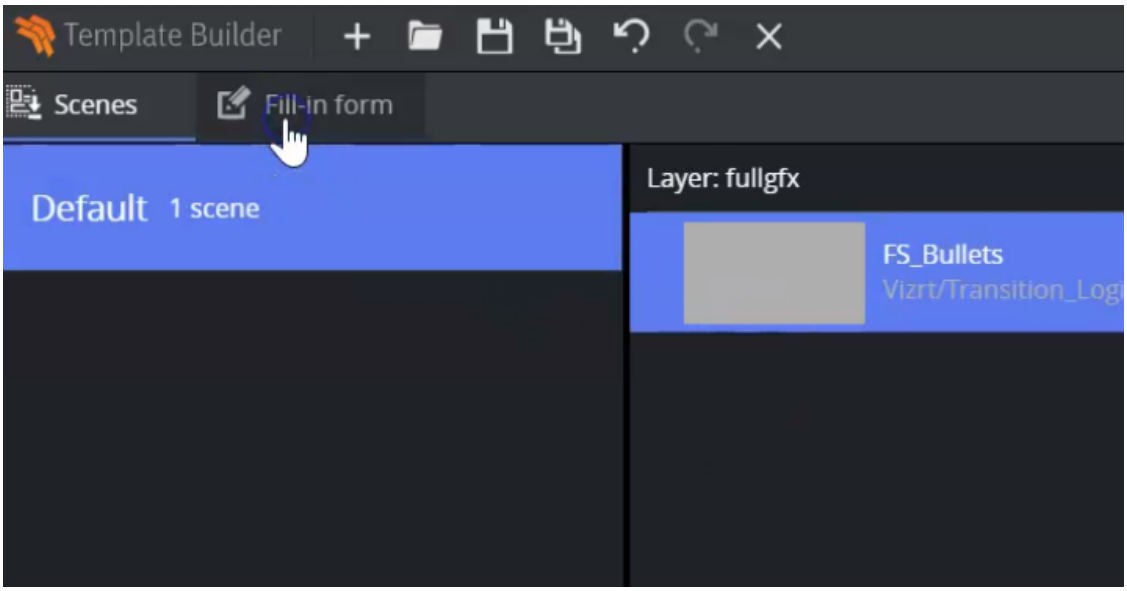
- Adding tabs enables you to quickly create fill-in forms based on selected fields or start with an empty space where you can add your chosen field types.
- The **All** tab gives you access to the classic form that contains all of the template's field types.
- Resize, move, edit, add and delete fields quickly.

This section covers the following topics:

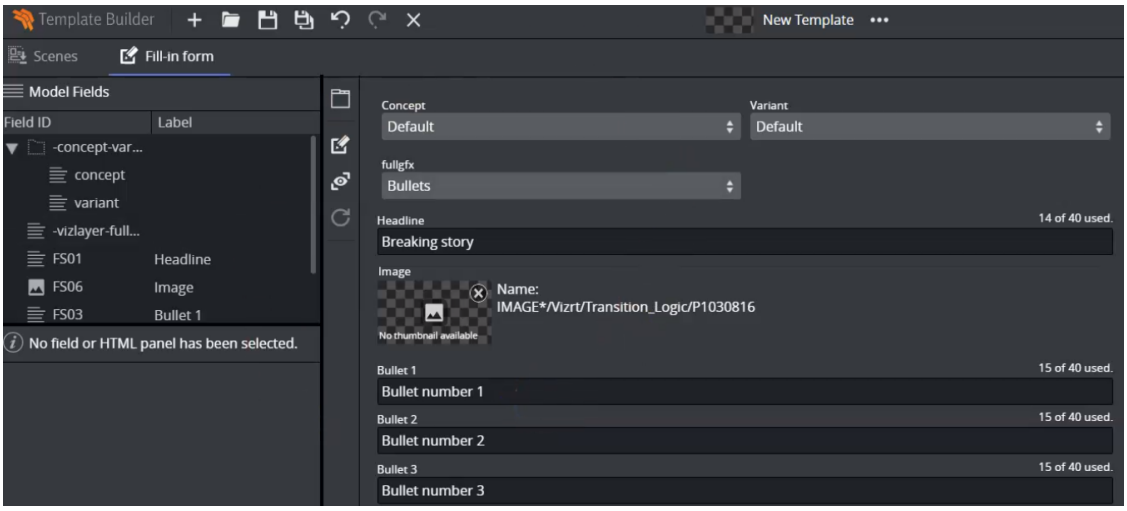
Follow the steps below to get started.

6.1 Accessing Layout Editing

1. Open or create a new template and add a scene.
2. Click **Fill-in form**:



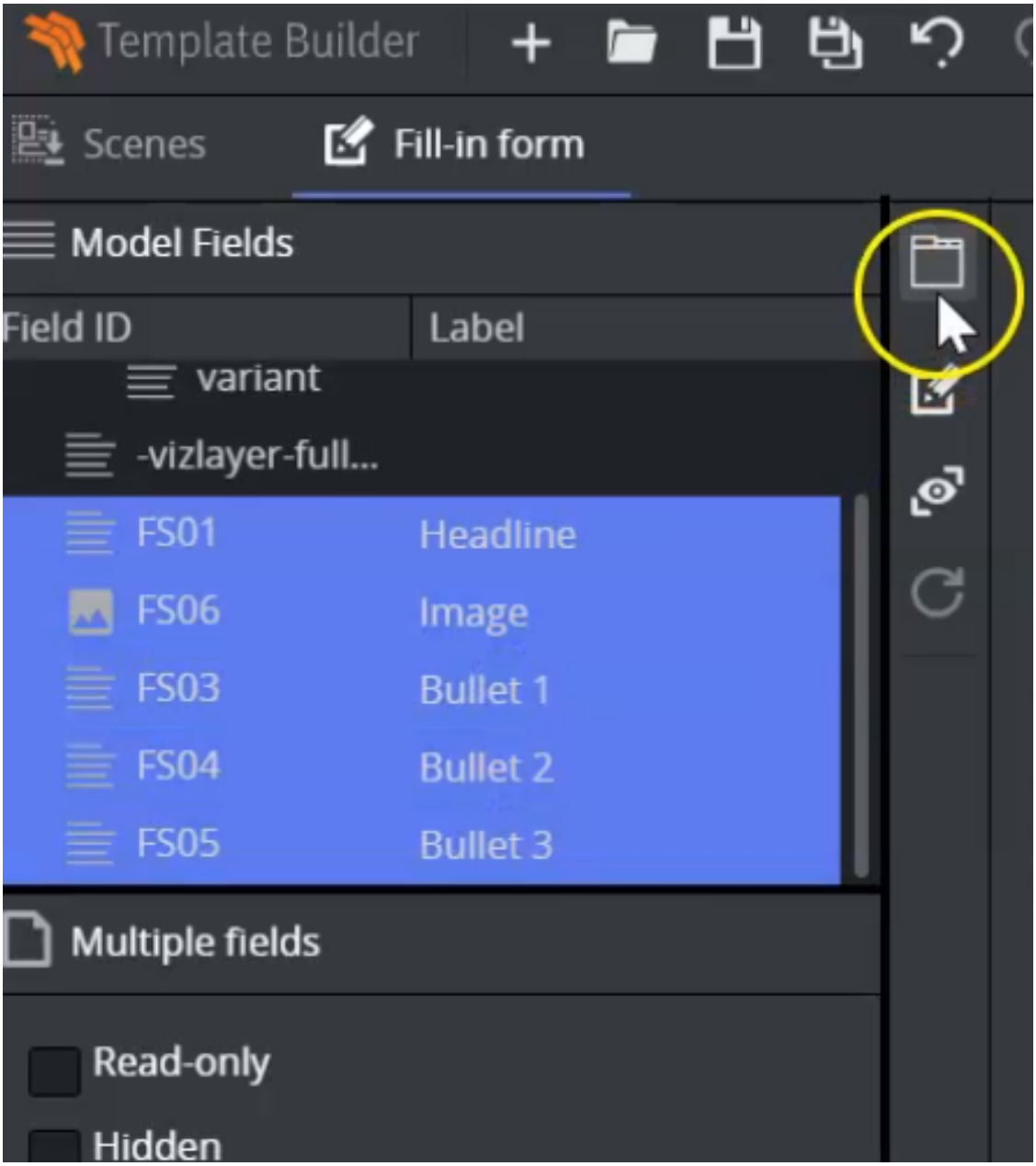
The default view is then displayed:



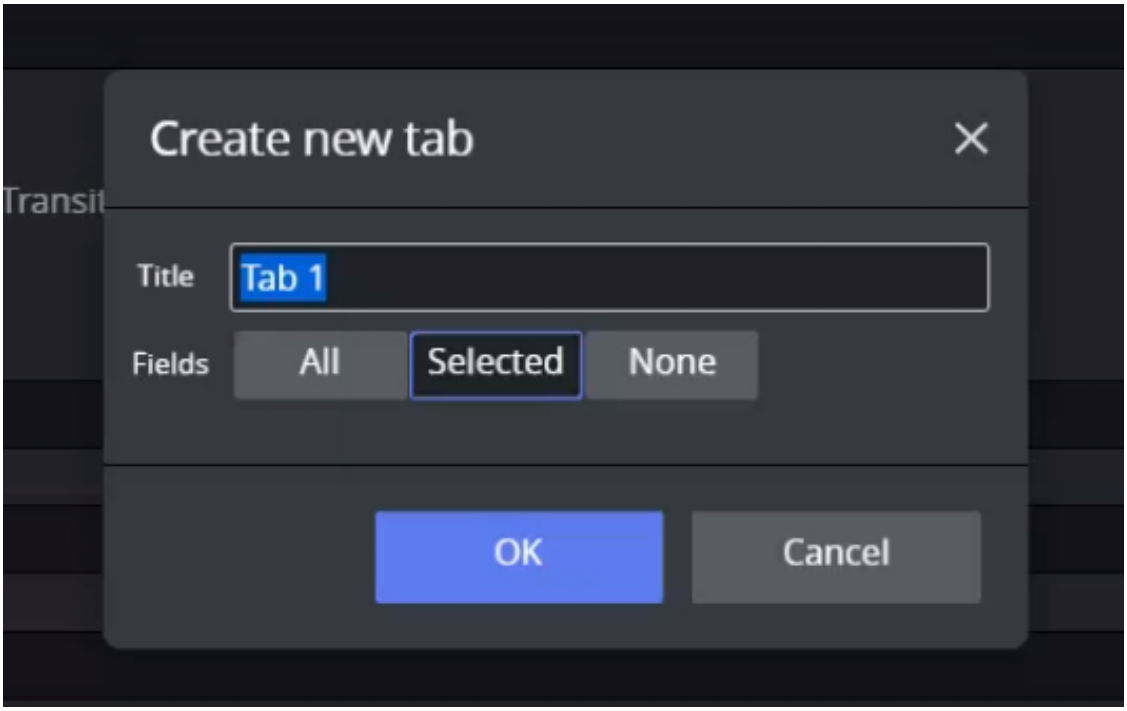
6.2 Creating A Template

6.2.1 Selecting Fields and Creating a New Tab

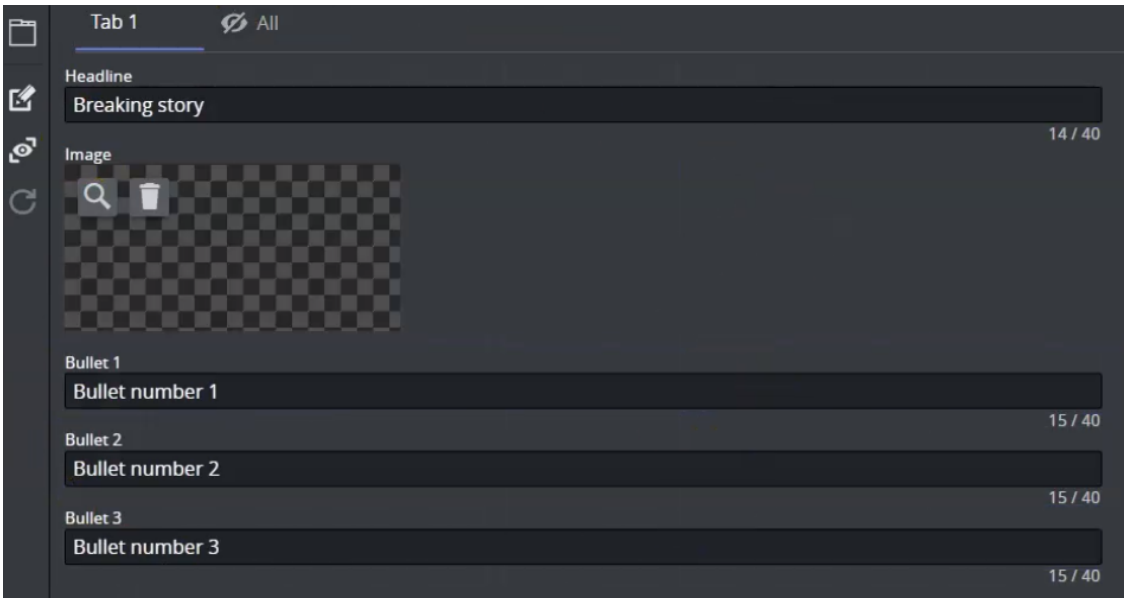
To create a new form for journalists to work with, select fields and click the **Create New Tab** button:



Enter a new for your new tab and click **OK**:



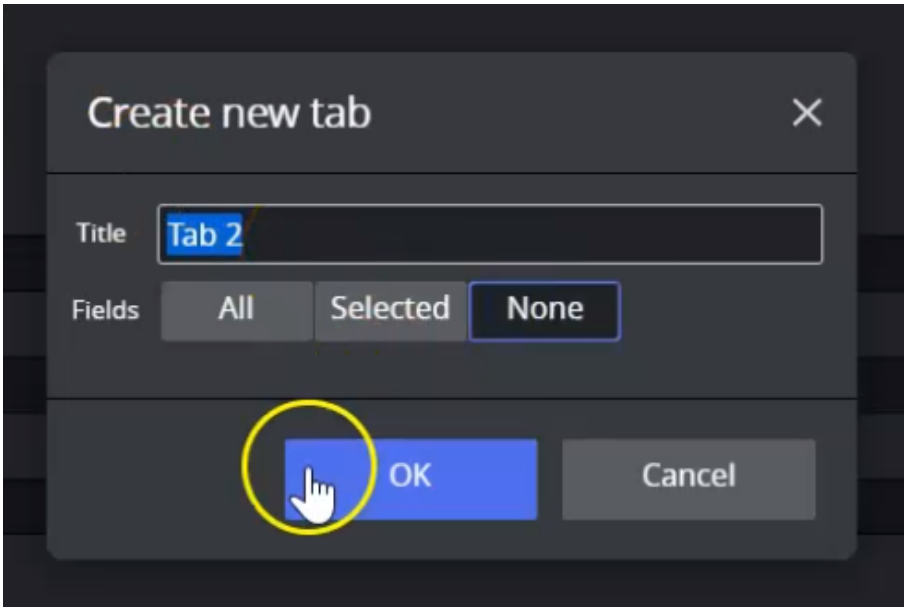
In the example in above, **Selected** is marked as the fields have already been selected. Click **OK**. All of the previously selected fields are now shown:



6.2.2 Creating a Second Tab

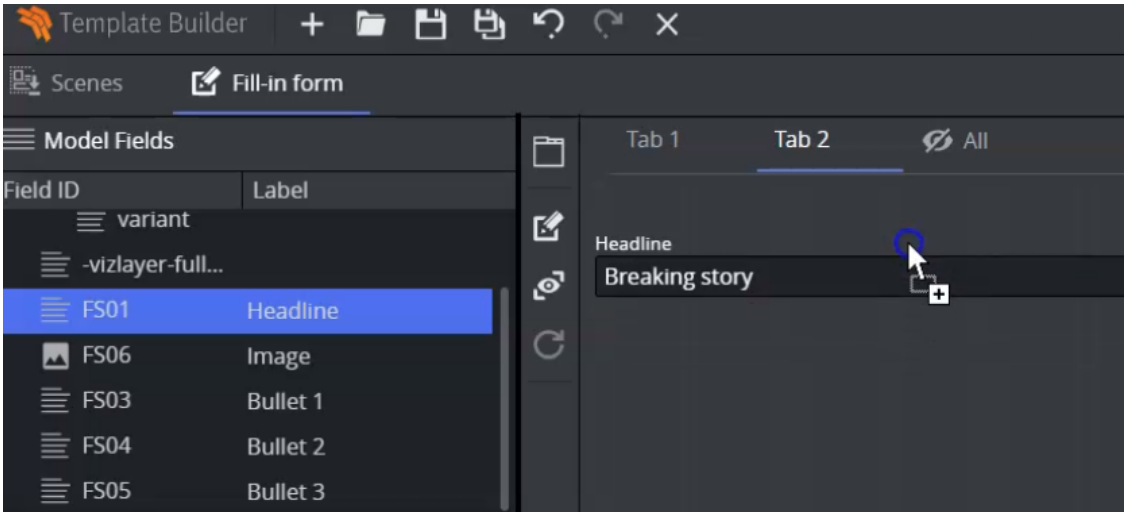
You can use the layout editor to drag and drop, resize and reposition elements in the form. This can be a quick way of doing things if you're only using a few fields.

1. Click the Create Tab button (or ALT +T).
2. Enter a name for the tab. Select **None** and click **OK**:

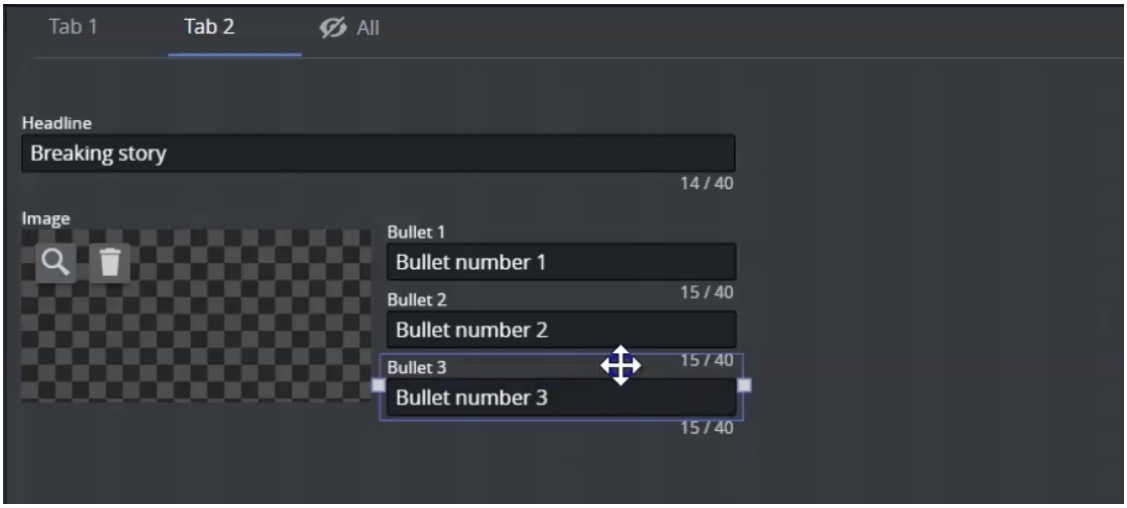


6.2.3 Adding, Moving, and Resizing Fields

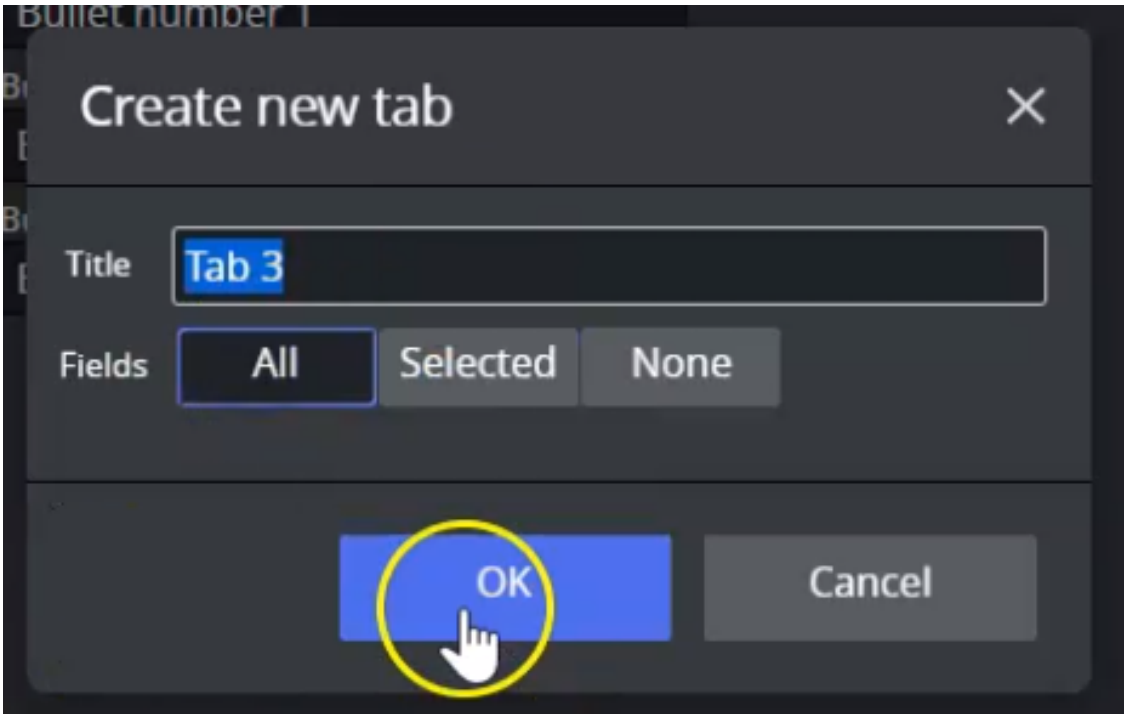
Use drag and drop to add fields:



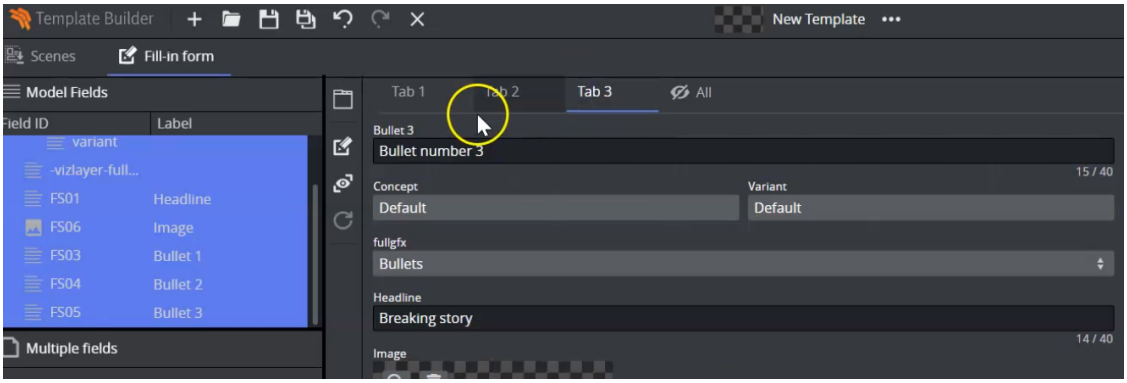
You can then use your cursor to resize and move the elements around:



You can also create a new tab and click **Select All**:

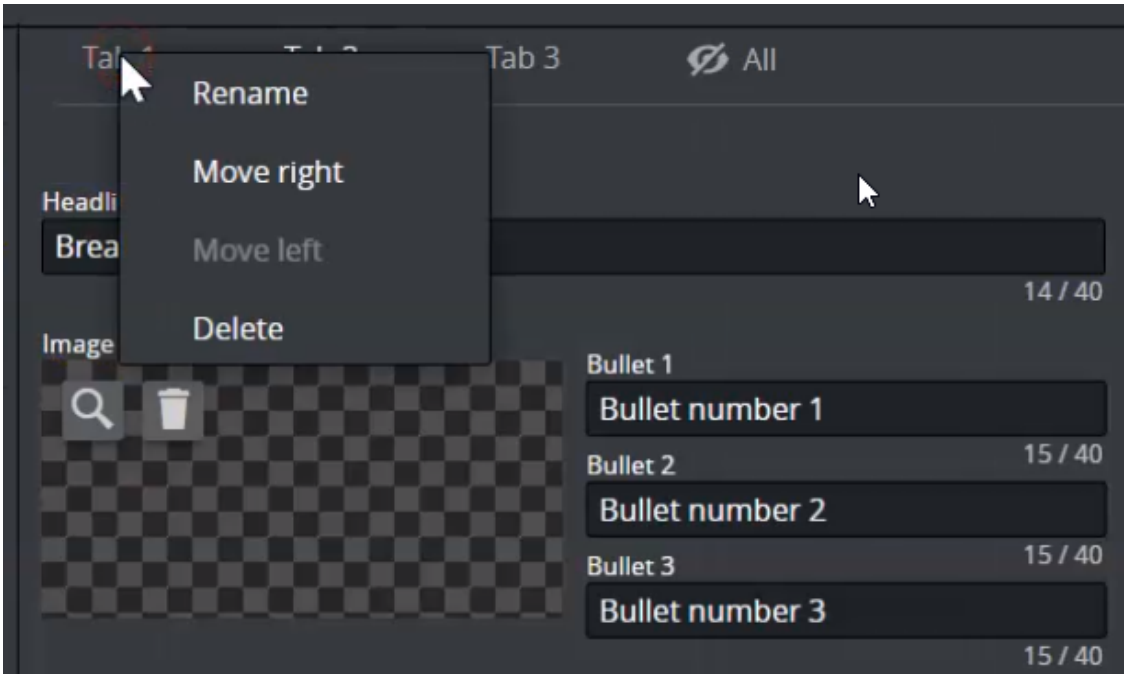


This tab then contains all the fields, and is quite similar to the default view:



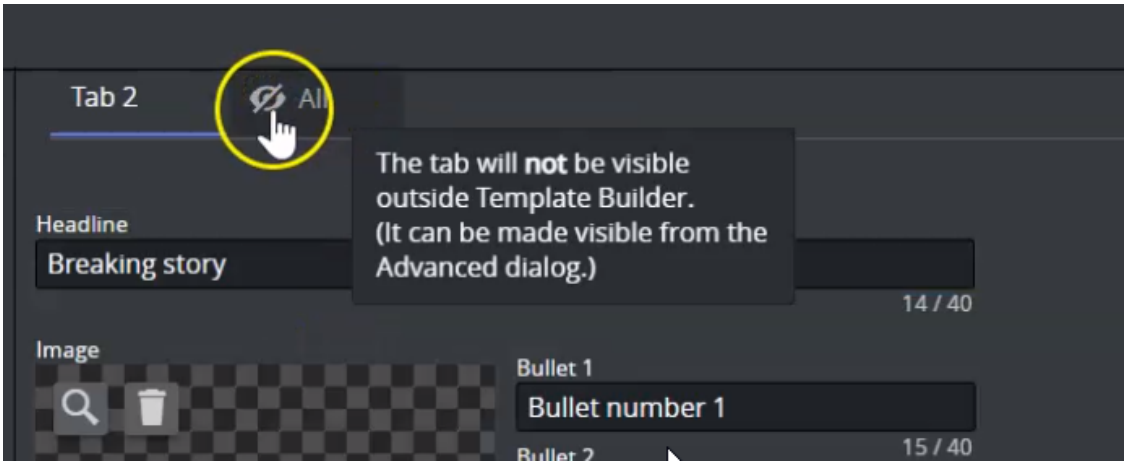
6.2.4 Deleting Tabs

When you're happy with your design, you can **Delete** the tabs that you don't want to use, right-click on the tab:

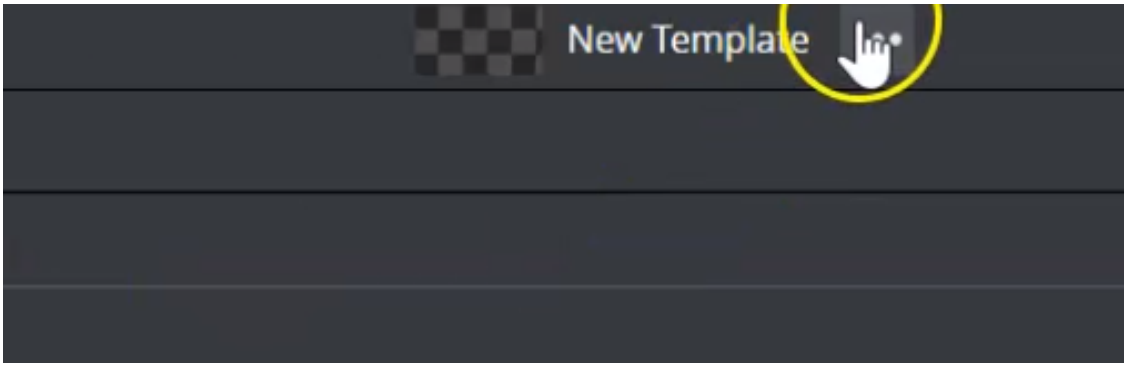


6.2.5 Hiding and Showing Tabs

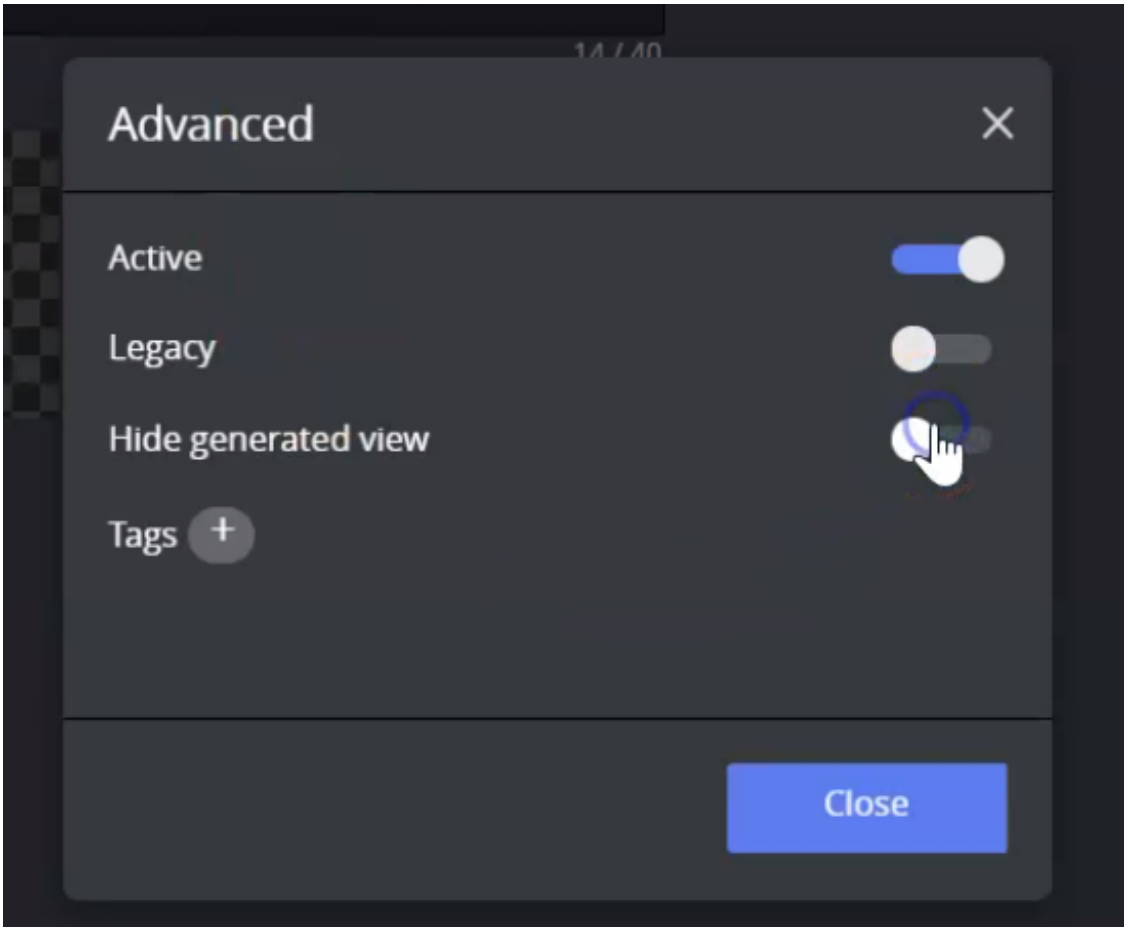
You can then decide what should be visible to the journalist. **All** with a line through the eye icon indicates that a tab is hidden:



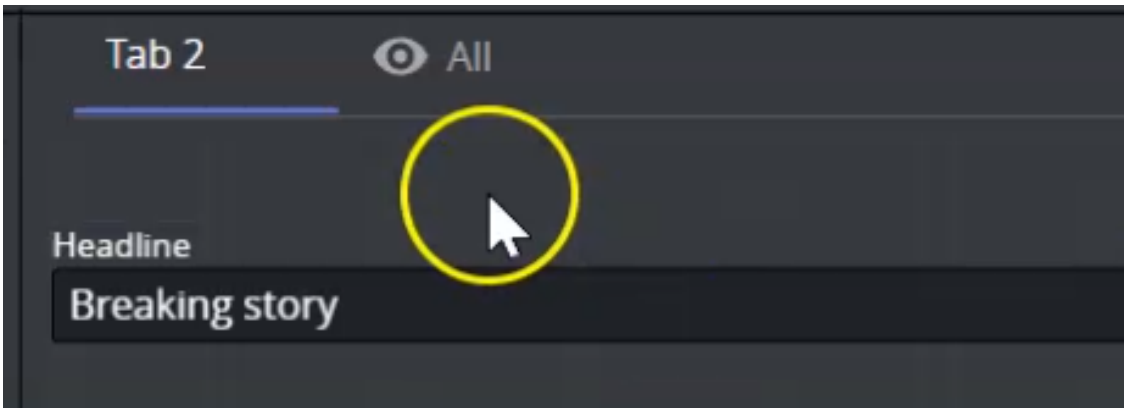
Make a tab visible by clicking the **breadcrumbs** beside **New Template** at the top of the screen:



Deselect **Hide generated view** and click **Close**:

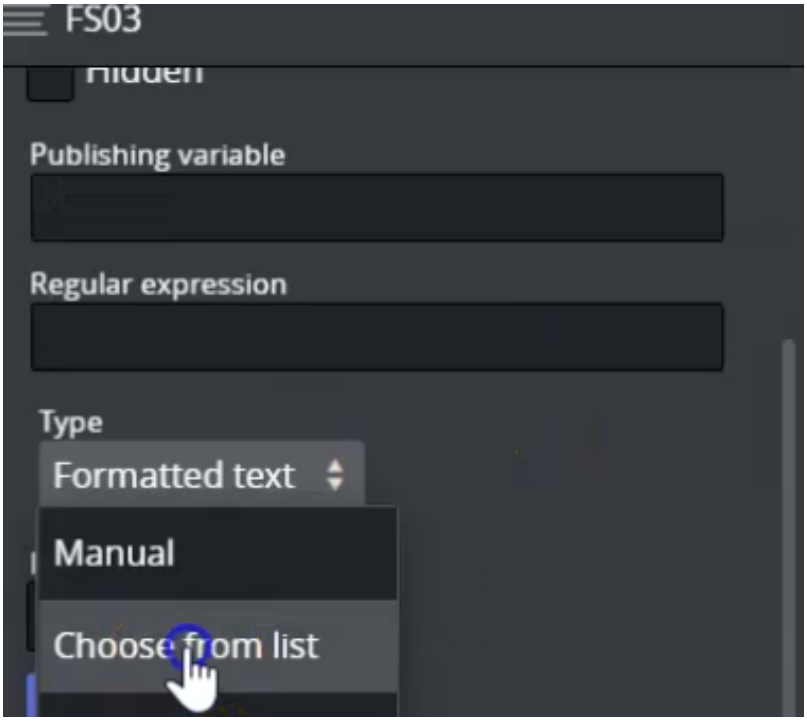


Voilà:

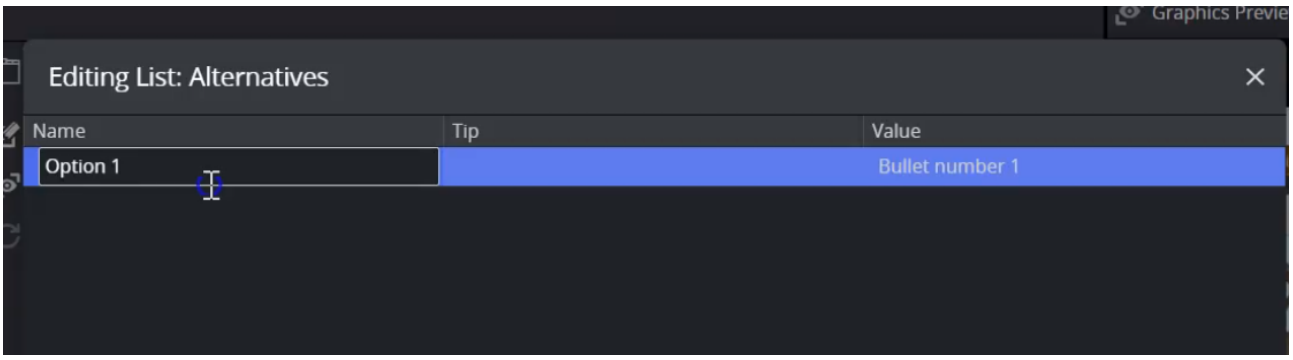


6.2.6 Creating a Drop-down Menu

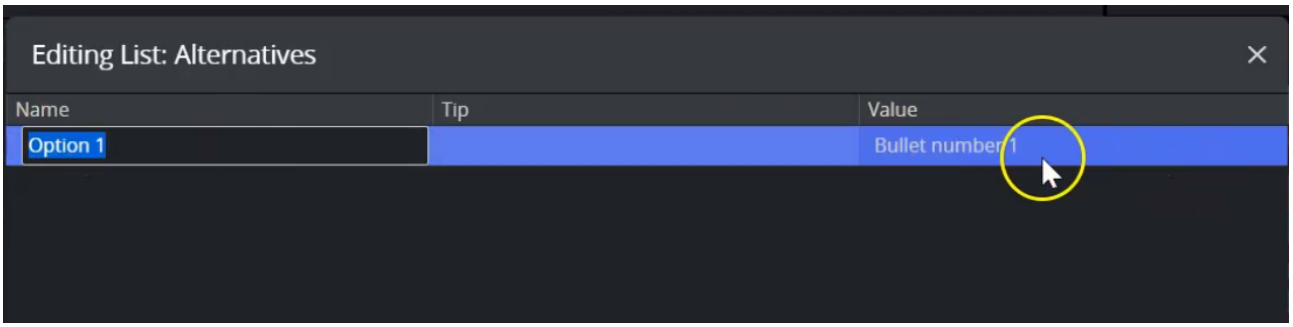
Give the journalist more options to choose from. Click **Data entry** at the bottom left of the screen. Select **Choose from list**:



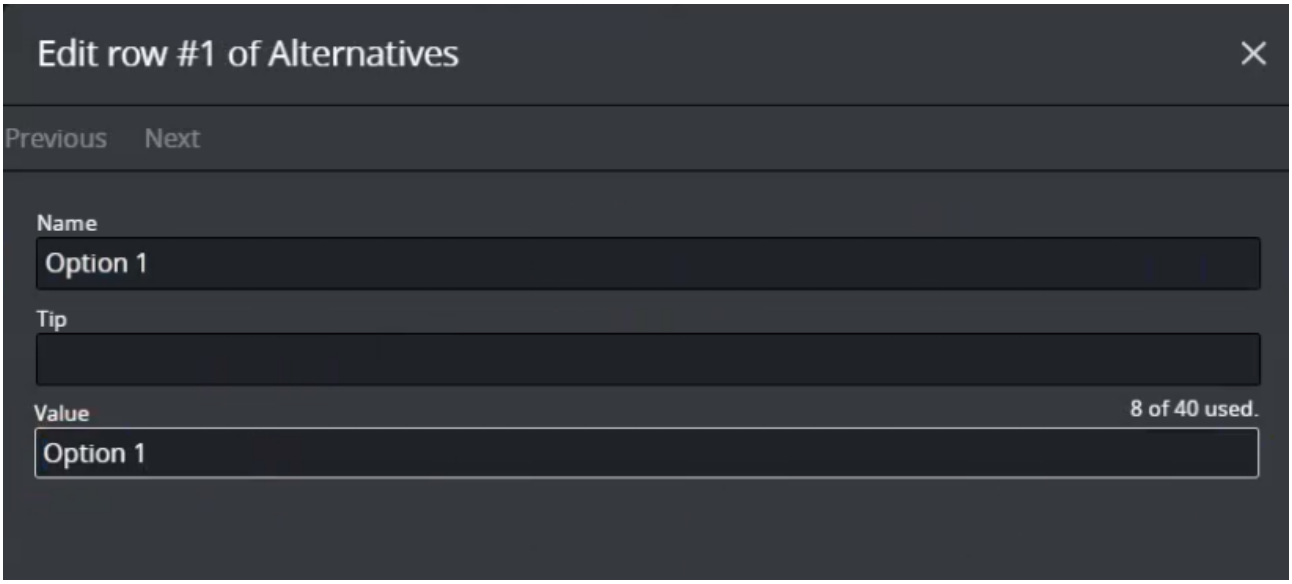
Enter a **Name** in the menu:



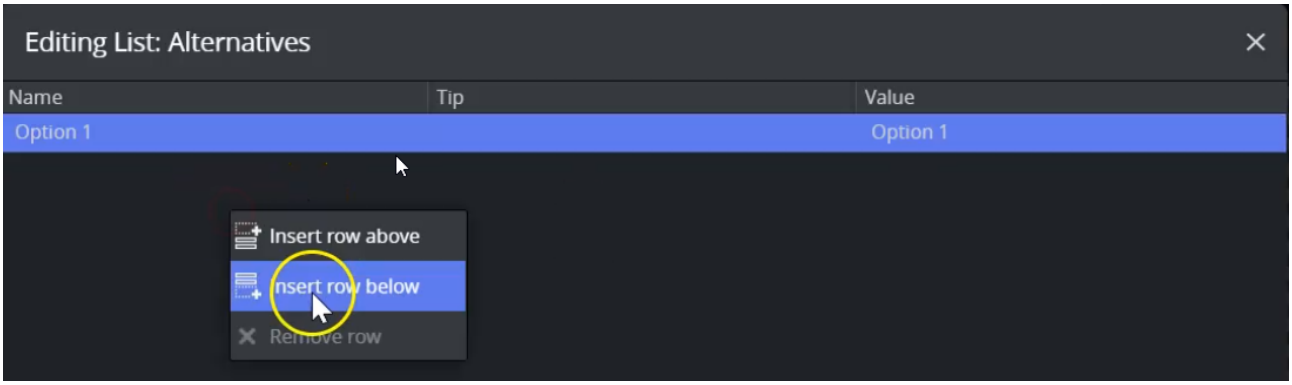
Double-click the **Value** field:



Edit the **Value** box:

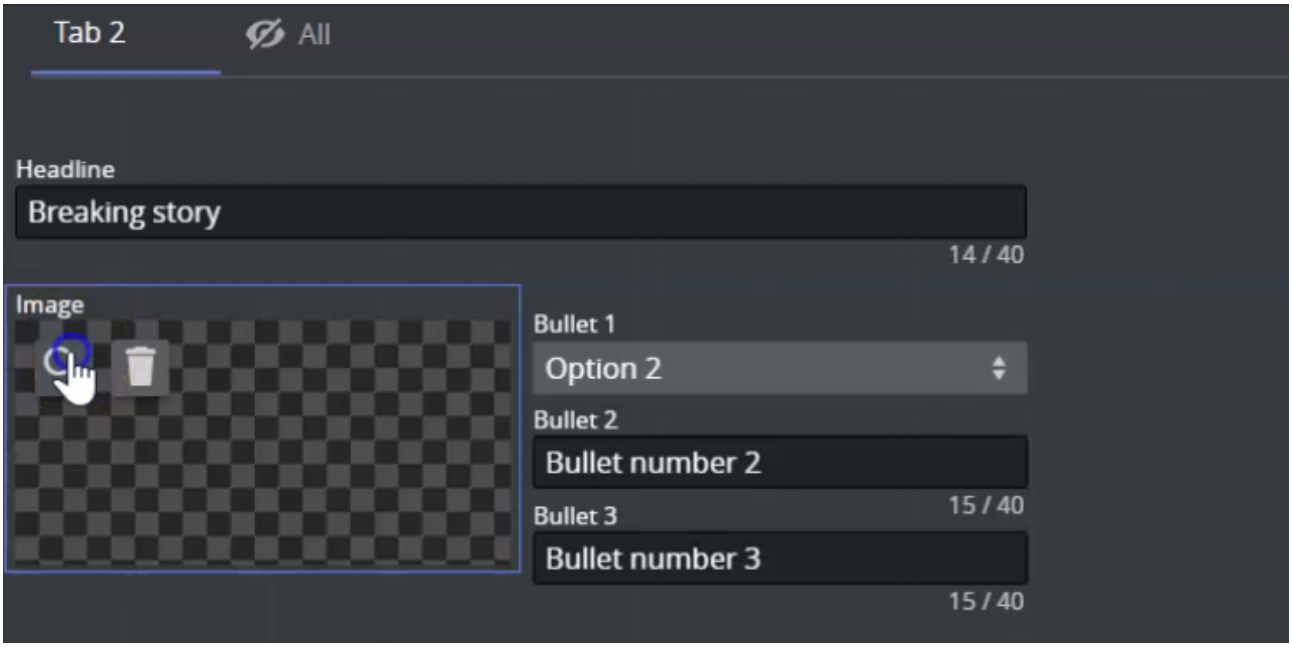


Right click in the menu and select **Insert row below**, to create more rows in the same way:

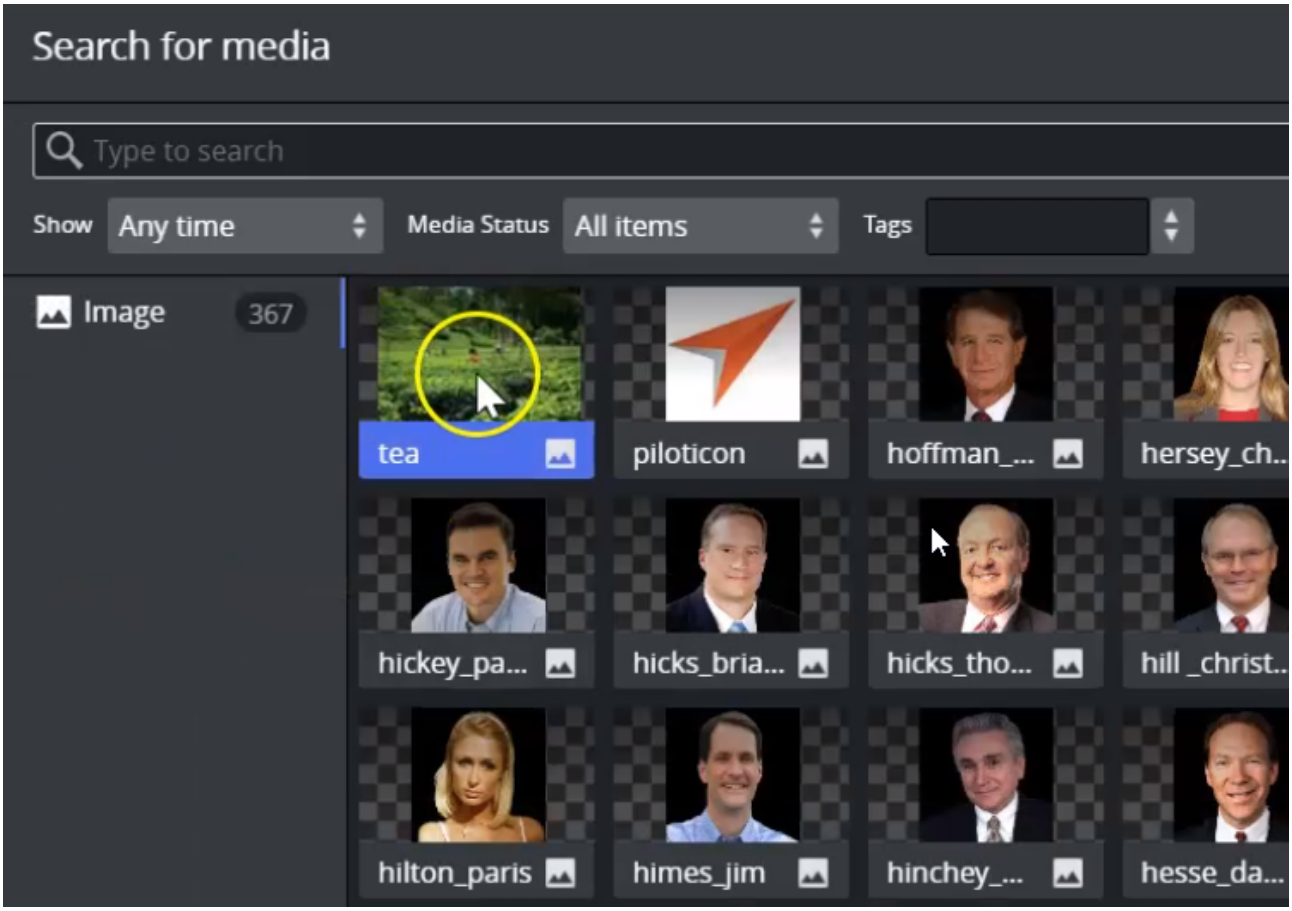


6.2.7 Changing the Image

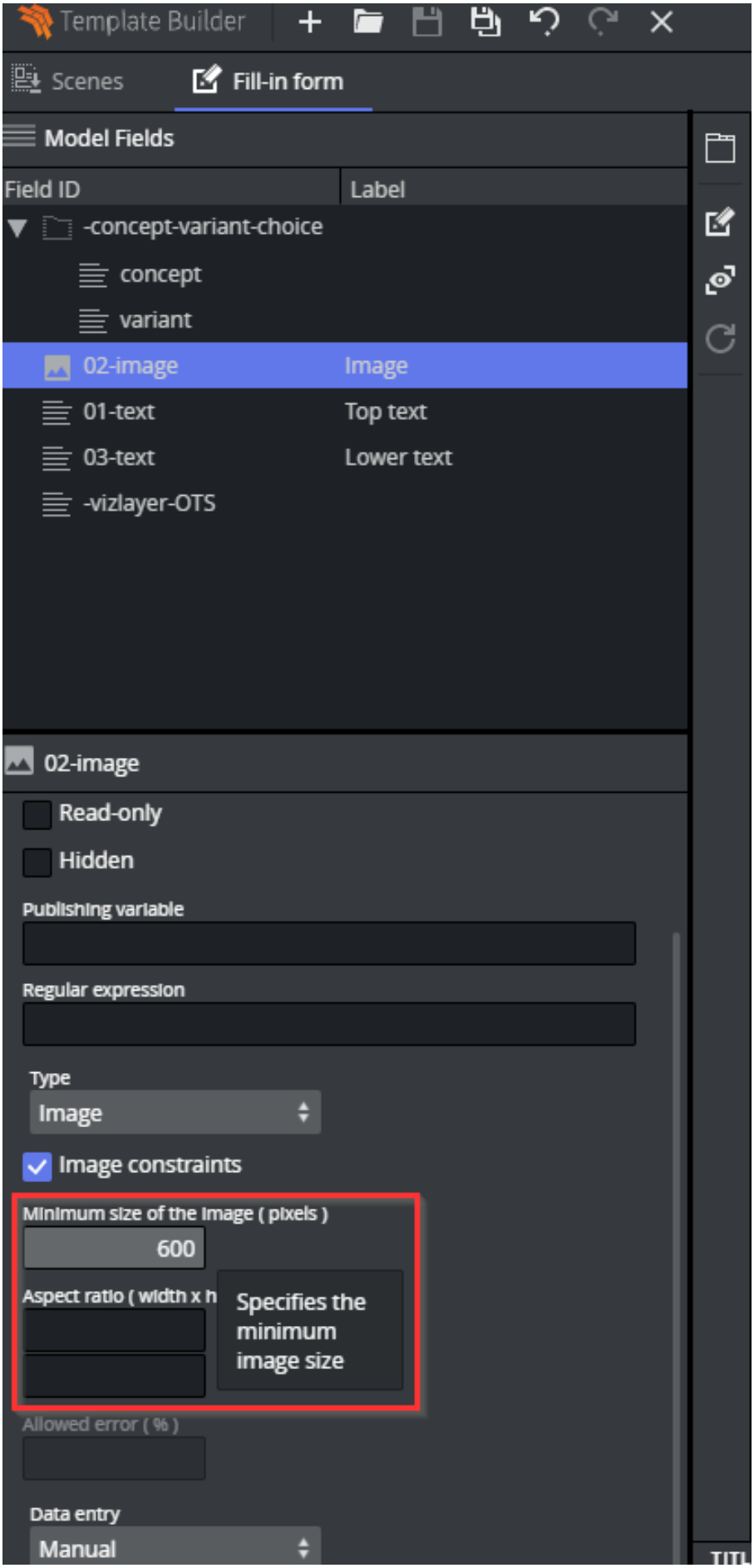
Click the **circle icon** to open a menu to search for a different image:



Select an image and click **OK**:



6.2.8 Image Constraints


Template Builder
Scenes
Fill-in form
Model Fields
Field ID
Label
-concept-variant-choice
concept
variant
02-image
Image
01-text
Top text
03-text
Lower text
-vizlayer-OTS
02-image
Read-only
Hidden
Publishing variable
Regular expression
Type
Image
Image constraints
Minimum size of the image (pixels)
600
Aspect ratio (width x h
Specifies the minimum image size
Allowed error (%)
Data entry
Manual

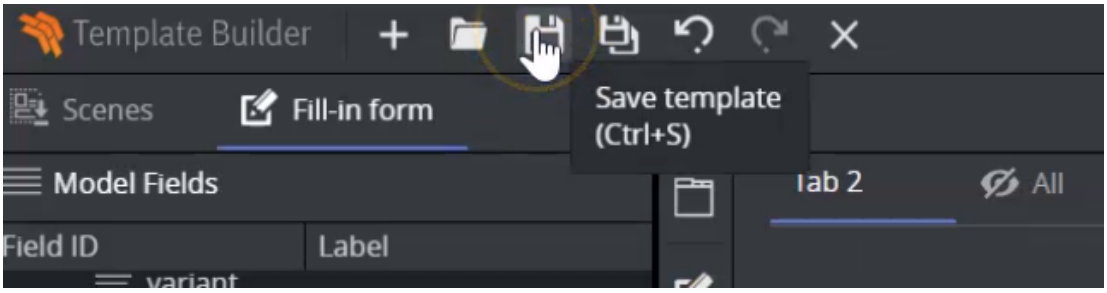
Specify standards for image quality and size using either or both of the following:

- **Minimum size of the image (pixels):** number of pixels, irrespective of aspect ratio.
- **Aspect ratio:** minimum width and height.

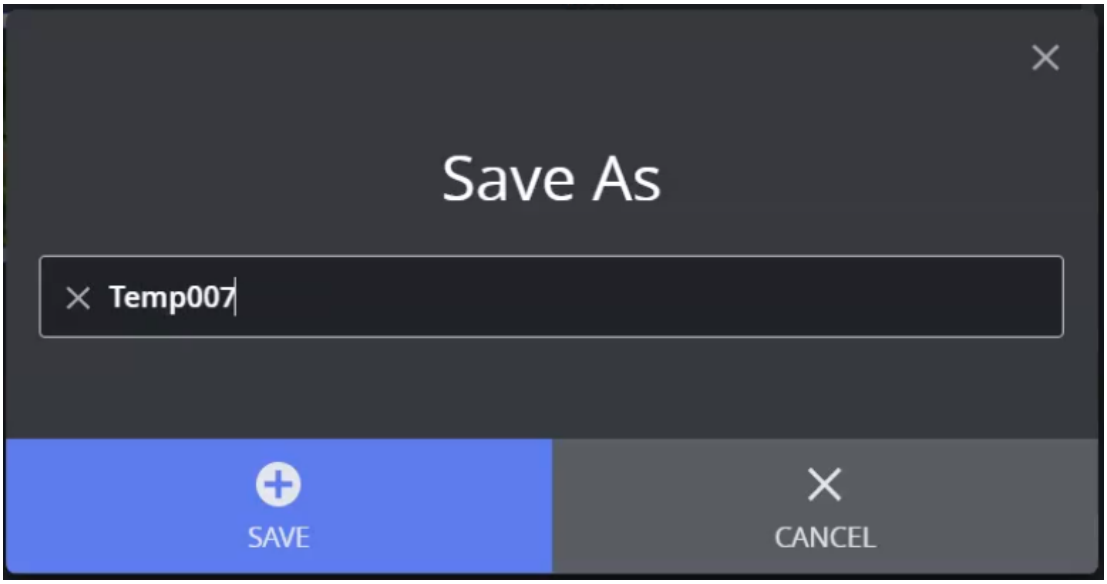
> **Note:** A warning is shown if the image becomes smaller than the minimum permitted.

6.2.9 Saving a Template

Click the **Disk icon** (or shortcut **CTRL + S**) to save:



Enter a template name and click **Save**:



7 Transition Logic And Combo Templates

This section covers transition logic and combo templates, and contains the following topics:

7.1 What Is Transition Logic (TL)?

Transition Logic (TL) is a way of designing a graphics package that lets you maintain the look and feel of the graphics while letting journalists add graphics items to a rundown - without the need for technical knowledge. TL lets you independently control any number of graphics layers, providing a code-free and design-based method for building graphics that gracefully animates in and out, and transitions from one to another automatically.

> **Info:** Transition Logic (TL) can be played out by most Vizrt control applications such as Viz Trio, Viz Pilot and Viz Multichannel.

7.2 How Does TL Work?

7.2.1 Master Scenes

This is accomplished by using a *Master Scene* that coordinates the animation of independently controlled objects which make up the whole. The master scene commonly contains the background items of the graphics package. Such items can be looping backgrounds or the design items of the lower third, over the shoulders, and full-screen graphics. The *variable* or changing content, such as the text in a lower third, is stored separately in *Object Scenes*.

7.2.2 Object Scenes

When a lower third is played On Air, the object scene for the lower third is triggered. This tells the engine to load the master scene, place the object scene inside the master, and animate the timelines. TL handles all of this automatically.

7.2.3 Combo Templates

These are templates that contain multiple layers of TL scenes.

7.2.4 TL Terminology

- **Combo Templates:** A TL template that contains more than one layer of scenes.
- **Master Scenes:** A TL scene is not a single scene, but a set of Viz graphics scenes that consist of a *master scene* that may have multiple layers of graphics that can be On Air at the same time and independently controlled.
- **Object Scenes:** Each layer in the master scene may have multiple referring *object scenes*. However, only one object scene per layer can be active at any given time.
- **Layers:** Layers in the transition logic scene define how many scenes can be on air at the same time. TL layers are conceptual, not spatial.

> **Note:** With Transition Logic scene design, *take in* and *take out* commands are still used as with standalone scene design. Where standalone scene design demands that only a single scene can be On Air at a time, however, Transition Logic allows for more than one scene to be On Air simultaneously. This means using Transition Logic lets you have a graphic covering the lower third of the screen and another graphic covering the left and/or the right side of the screen for over the shoulder graphics On Air at the same time.

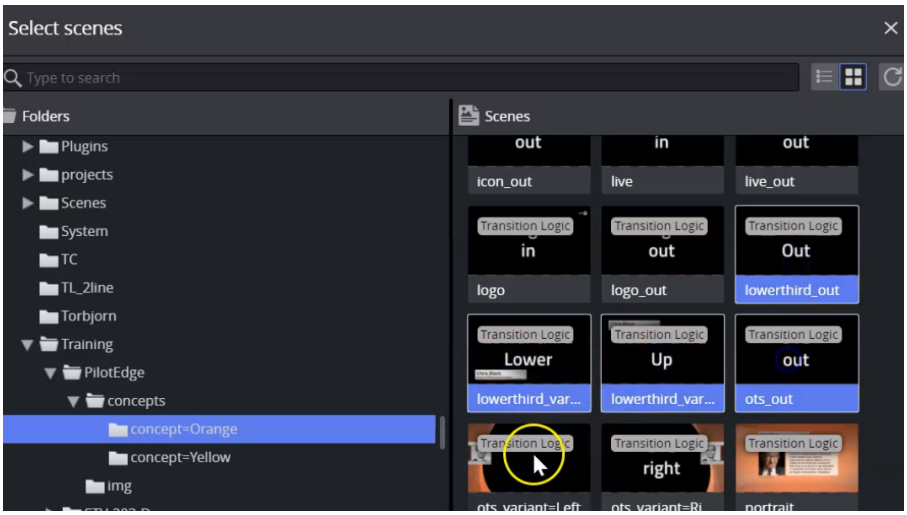
7.3 Working With Transition Logic And Combo Templates

Follow the steps below to get started.

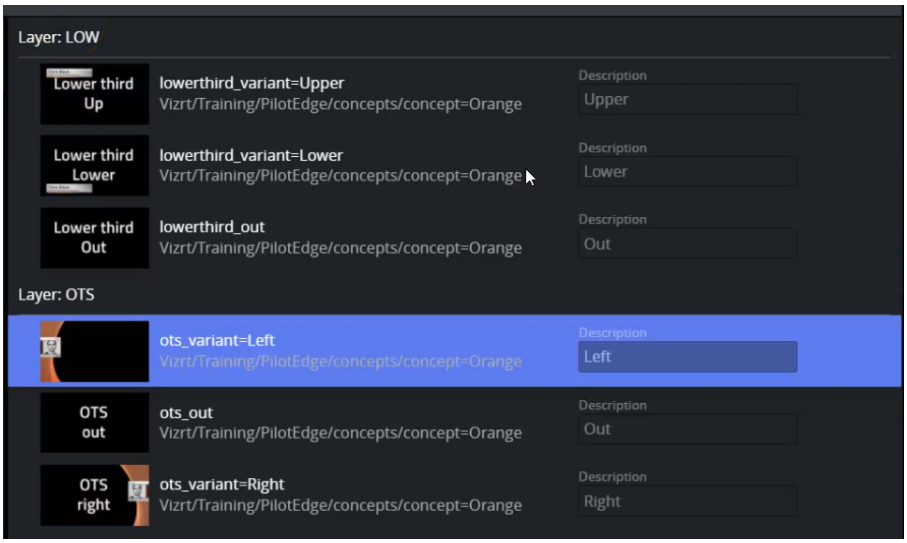
> **Note:** Transition logic and combo templates requires Viz Engine 4.2 or above.

7.3.1 Creating a New Combo Template

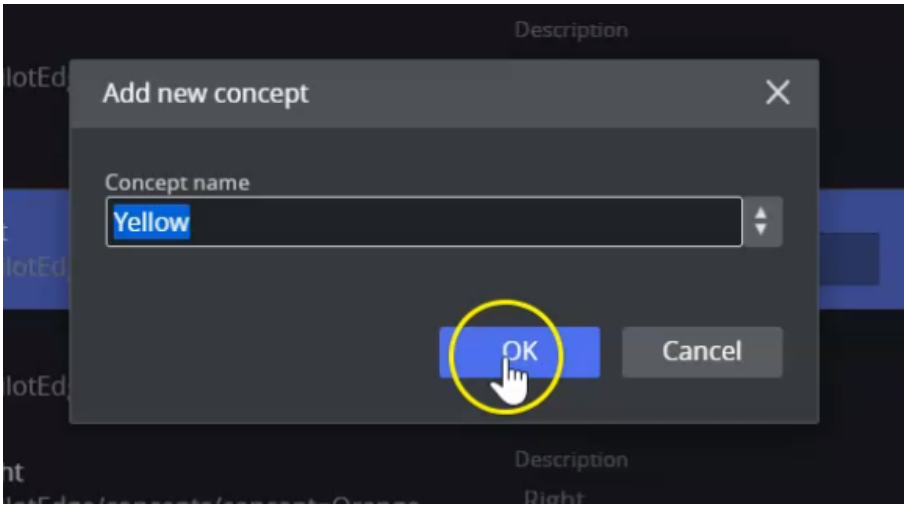
Create a new template and add transition logic scenes. The following example uses **Orange** and **Yellow** concepts. Select scenes and click **OK**.



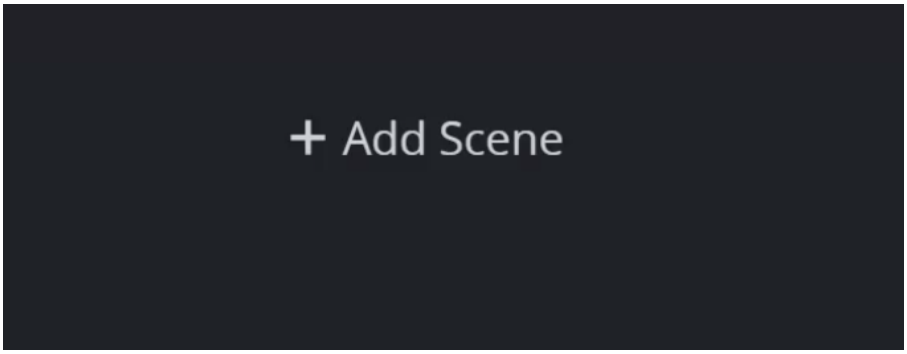
The new template contains transition logic and two layers, and is therefore a combo template:



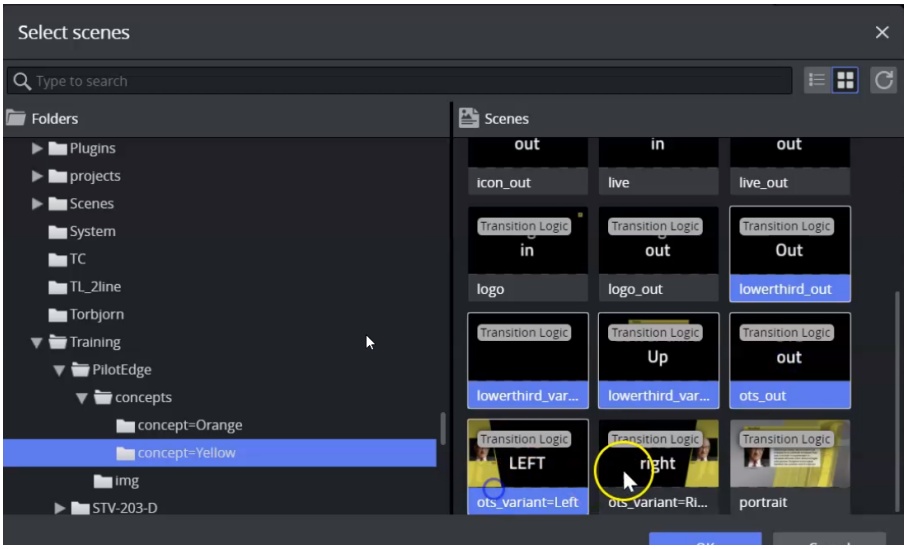
Click **+Add Concept** at the lower left corner of the screen. Enter a name for your new concept and click **OK**.



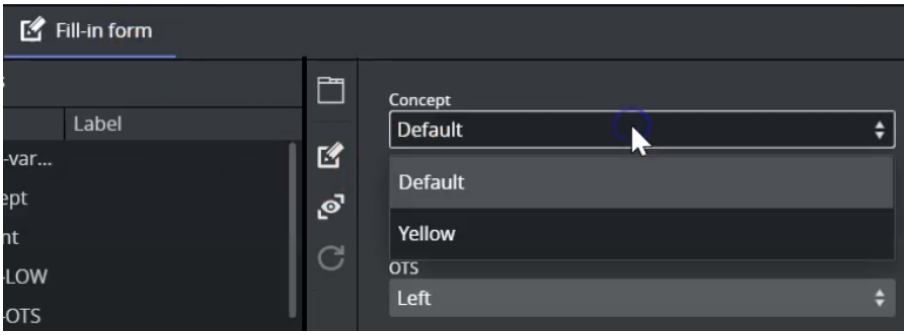
Click the **+Add Scene** button:



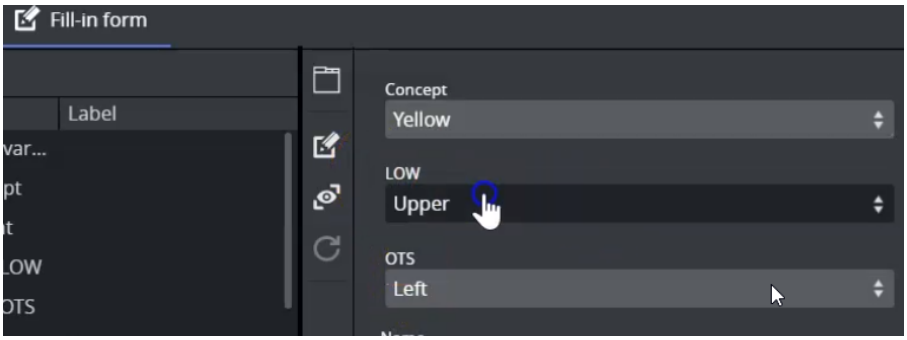
Select the same scenes as those you selected for **Orange**, above:



In the **Fill-in form**, you can now see that the template contains two concepts:



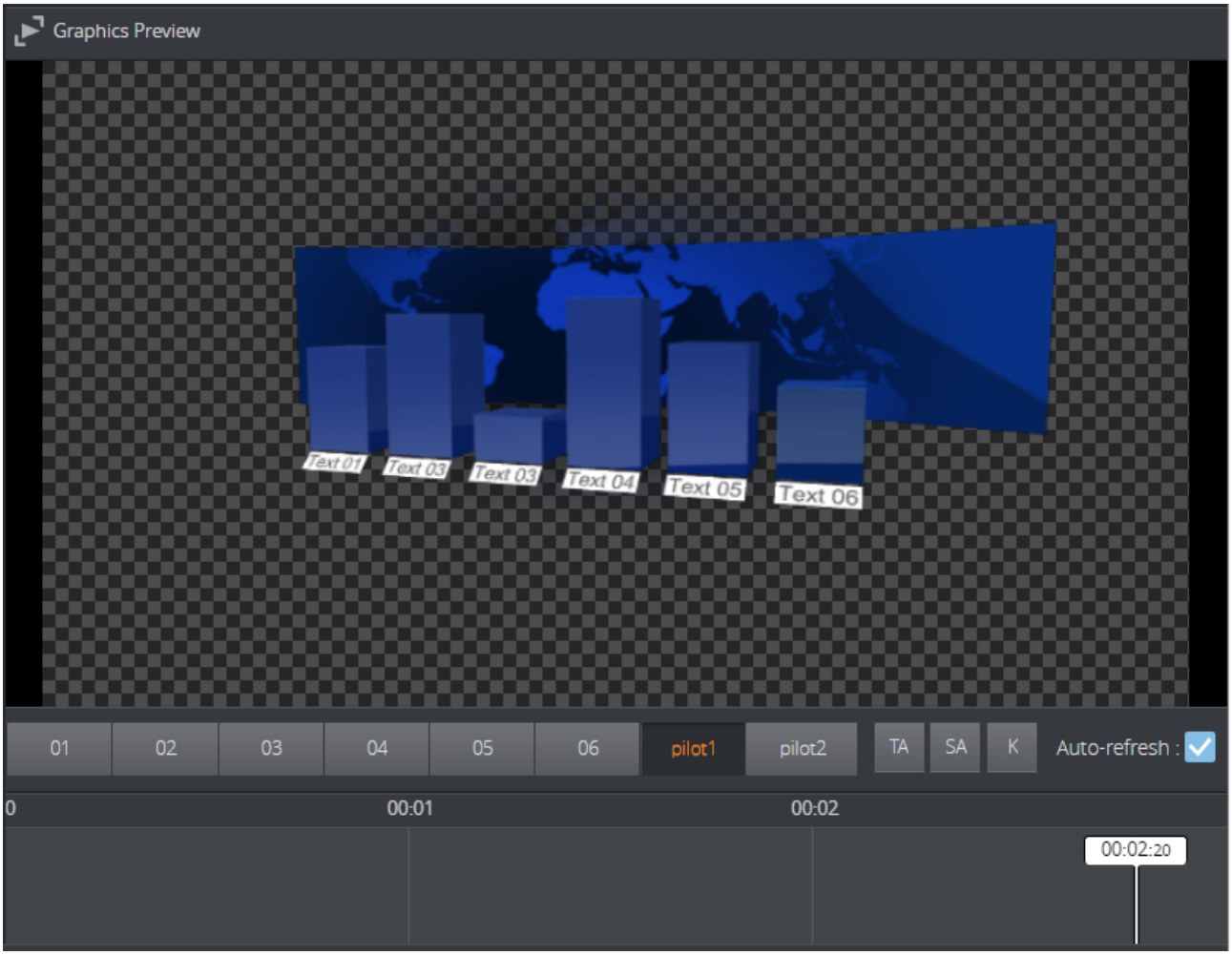
and two layers:



8 Previewing Content

The **Graphics Preview** window is located to the right of the interface. It displays snapshots of the final output in an ongoing preview process, and provides an indication of how the graphics look when played out in high resolution on a Viz Engine.

> **Note:** Template Builder sends requests to Preview Server which manages the Viz Engines that provide the snapshots.



Adjust preview settings using the toolbar at the bottom of the Preview Window:

- **Preview points:** If the scene contains named preview points, such as stop points and/or tags in the Default director, these are displayed as buttons on the toolbar. If there is not enough space for the buttons, they appear in a drop-down list instead. Pressing the buttons or selecting an entry from the drop-down list shows a preview of the scene at the given preview point, and the playhead on the timeline jumps to where the preview point is set.
- **TA:** Show/hide the Title Area.
- **SA:** Show/hide the Safe Area.
- **K:** Show the key signal for the graphics.

- **Load:** Loads the animation of the graphics. Once loaded, this is indicated by a green line at the bottom of the timeline editor; media controls for controlling the graphics animation in the Preview Window appear.
- **Scrub:** Back and forth by clicking on the timeline or moving the playhead. If the scene does not have a director called Default or the Default director does not have a duration, the timeline is disabled. Setting the minimum duration in Template Builder enables the timeline. Setting the default duration in Template Builder changes the duration of the timeline if it is enabled.
- **Auto-refresh:** Enabled by default.

> **Info:** Clicking on a preview point to request a preview sends a snapshot request with a named position to Preview Server. Clicking on the timeline sends a snapshot request with an absolute position to Preview Server. For more information, see the Preview Server REST API documentation.