# Eclipse Scout

## *Release Notes*

Scout Team

Version 7.1

# Table of Contents

# About This Release

Eclipse Scout 7.1 is a preview version of the Eclipse *Photon* release. It will be released in June 2018 (release schedule). The latest version of this release is: None released yet.

If you are upgrading from version 6.1, please also read the migration guide for the 7.0 (*Oxygen*) release:
https://eclipsescout.github.io/7.0/migration-guide.html

- [PLACEHOLDER]

You can see the detailed change log on GitHub.

# Service Releases

The following changes were made after the initial 7.1 release (Eclipse Photon release). The following notes relate to a *service release*.

### Photon.1 (7.1.100) Release expected on September, 2018

**Attention:** The here described functionality has not yet been released and is part of an upcoming release.

# Obtaining the Latest Version

### Runtime (Scout RT)

Scout RT artifacts are distributed via Maven:

- 7.1.0.001_Photon on *Maven Central*
- 7.1.0.001_Photon on *mvnrepository.com*

Usage example in the parent POM of your Scout application:

```
<dependency>
    <groupId>org.eclipse.scout.rt</groupId>
    <artifactId>org.eclipse.scout.rt</artifactId>
    <version>7.1.0.001_Photon</version>
    <type>pom</type>
    <scope>import</scope>
</dependency>
```

### Eclipse IDE Tooling (Scout SDK)

You can download the complete Eclipse IDE with Scout SDK included (EPP) here:
Eclipse for Scout Developers

To install the Scout SDK into your existing Eclipse IDE, use this update site:
http://download.eclipse.org/scout/releases/7.1/7.1.0/001_Photon/

## Demo Applications

The demo applications for this version can be found on the features/version/7.1.0.001_Photon branch of our docs repository on GitHub.

If you just want to play around with them without looking at the source code, you can always use the deployed versions:

- https://scout.bsi-software.com/contacts/
- https://scout.bsi-software.com/widgets/
- https://scout.bsi-software.com/jswidgets/

# Java 8 required

The required Java Runtime Environment (JRE) to run an Eclipse Scout application has changed: Starting with Eclipse Scout 7.1, a Java 8 runtime is required.

The Scout 7.1 Runtime does not support Java 9 yet. The Java 9 support is planned for Eclipse *Photon* release (Scout 8.0) in summer 2018.

# New SDK Feature in Eclipse: Search for missing NLS keys

If NLS keys are used in the code that do not exist in a properties file, an ugly placeholder is displayed to the user. To find such missing translations the new Menu `Scout → Search missing text keys⋯` may be handy. The result is listed in the Eclipse `Search` view.

The search also takes the scope of each NLS key into account. So that the key is considered to be available there must be a `TextProviderService` with that key on the classpath of that module.

Reported false positives can be suppressed using the following comment at the end of the corresponding line: `NO-NLS-CHECK`. Matches on that line are then not reported in future searches anymore.

# Config Properties

## Descriptions

Config properties based on `org.eclipse.scout.rt.platform.config.IConfigProperty` include a description text. This description is stored in the new `description()` method.

The class `org.eclipse.scout.rt.platform.config.ConfigDescriptionExporter` can be used to export these descriptions. By default an AsciiDoctor exporter is included.

All Scout properties have been extended with descriptions. The same text is also part of the technical documentation.

## Default value

Config properties based on `org.eclipse.scout.rt.platform.config.IConfigProperty` include a default value. The default value is stored in the `getDefaultValue()` method.

The method was moved from `org.eclipse.scout.rt.platform.config.AbstractConfigProperty<DATA_TYPE, RAW_TYPE>` to the interface. Therefore the visibility has changed from protected to public.

## Validation

The concrete implementation `org.eclipse.scout.rt.platform.config.ConfigPropertyValidator` which validates the configuration of `config.properties` files will also check if a configured value matches the default value. In case it does a info message (warn in development mode) will be logged but platform will still start. To minimize configuration files such entries should be removed from `config.properties` files.

# GroupBox enhancements

## Layout Configuration

It is now possible to adjust the parameters of how the group box will be layouted. The following parameters may be set:

- hgap: the horizontal gap in pixels to use between two logical grid columns
- vgap: the vertical gap in pixels to use between two logical grid rows
- columnWidth: the width in pixels to use for a grid column
- rowHeight: the height in pixels to use for a grid row
- minWidth: the minimum width of the group box. If this width is > 0 a horizontal scrollbar is shown when the group box gets smaller than this value.

These values may be set using `getConfiguredBodyLayoutConfig`.

# Introducing Widget.java

On JavaScript side, there has been a class `Widget.js` for a long time now. With this release the counterpart `Widget.java` has been added. This gives all existing widgets like `FormField`, `Form`, `MessageBox`, `Menu` etc. a new common base class. It also helps creating widgets which aren't necessarily form fields.

# New Widget 'Tiles'

The new `Tiles` widget arranges `Tile` s in a grid by using the `LogicalGridLayout`. This is the same layout as used for a `GroupBox`, so the same `GridData` object may be used to configure how the individual tiles should be arranged.

A `Tile` directly extends `Widget` and is not much more than a `<div>` with the CSS class `tile`. In order to customize your tile you have to create a custom widget, which is easier than it sounds. Just create a JS class lets say `CustomTile.js` which extends from `Tile.js`, create a Java class `CustomTile.java` which extends from `AbstractTile.java` and add some glue code to link them together. See the code of the demo widgets on GitHub for details. You could also use existing widgets as tiles. In that case instead of extending `AbstractTile` you would extend `AbstractWidgetTile` or `AbstractFormFieldTile` and set the property `tileWidget` accordingly.

In order to add the `Tiles` to a form, you can use the class `TilesField` which is basically a simple `FormField` wrapping the `Tiles`. You cannot use the `Tiles` directly because a `GroupBox` only accepts `FormField` s.

A demo of the widget may be found here: https://scout.bsi-software.com/widgets/?dl=widget-tilesfield.

And here for the JS only version: https://scout.bsi-software.com/jswidgets/#tiles.

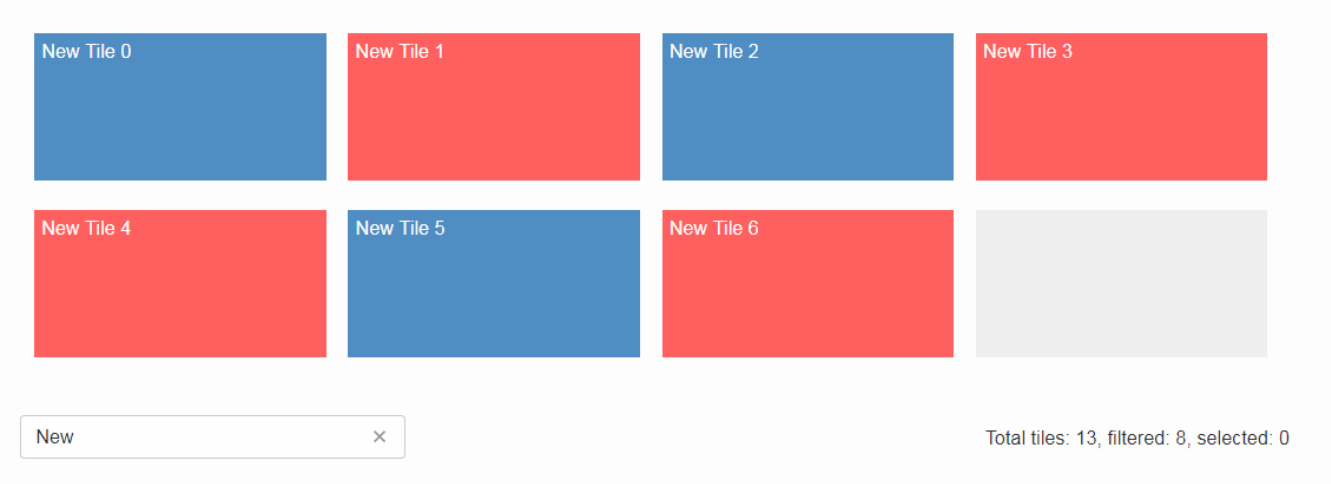


*Figure 1. Tiles*

# New Widget 'Accordion'

The `Accordion` displays several collapsible `Group` s. The default behavior is to collapse every other group if one group is expanded. Because that is not in any case desired, the behavior may be disabled by setting the property `exclusiveExpand` to false.

The `Group` is a simple widget containing of a header and a body. The body may be any other widget like the new `Tiles`. Because having tiles in an accordion is a typical use case, there is a widget called `TilesAccordion` which helps creating the groups and provides some delegate methods to easily access the tiles of every group.

A demo of the widget may be found here: https://scout.bsi-software.com/widgets/?dl=widget-accordionfield.

And here for the JS only version: https://scout.bsi-software.com/jswidgets/#accordion.

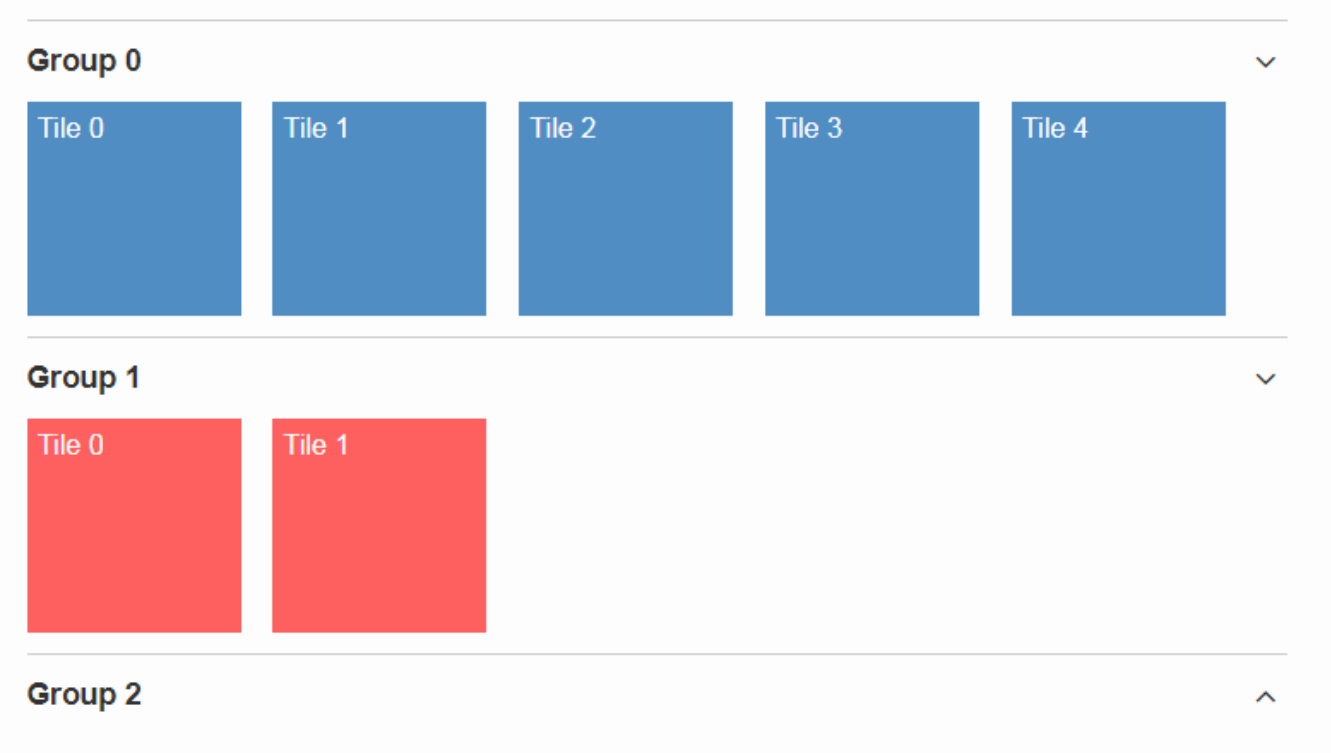


*Figure 2. Accordion*

# Menu, Menubar enhancements

The menubar supports form field menu items (`FormFieldMenu`). On the model side extend `AbstractFormFieldMenu` with a form field as an inner class to use a form field menu in any menu supporting environment.

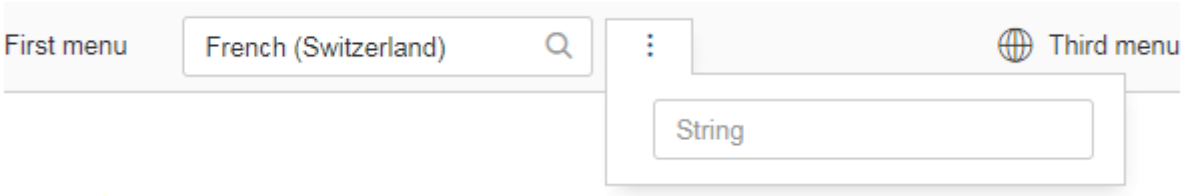


*Figure 3. Menubar with form fields*

The menu property `stackable` defines if a menu is stackable or not. A stackable menu will be moved to the ellipsis dropdown menu when there is not enough space in the menubar. The ellipsis menu is placed after the last stackabel menu in the menubar. Right and left aligned menus will be moved to a single ellipsis menu per menubar. The horizontal alignment of the ellipsis menu is the same as the last stackabel menu in the menubar.

# Tabbox enhancements

- The menubar of a tabbox considers the menu alignemnts *left* or *right*. Menus in the menubar will be moved to an ellisis menu in case there is not enough space in the tabbox header. The tab items are moved to an ellipsis menu when there is not enough space for all tabs. The collapse order is first all menus before the tabs will be collapsed from right to left.
- TabItems got a `sublabel` property with is displayed in a very small font below the title (see also *GroupBox enhancements*).
- The marker of the selected tab is now animated and follows the user or model selection.
- Several bugfixes of pixel issues due to zoom levels.

# GroupBox enhancements

GroupBoxes got a new property called `sublabel`. The sublabel is displayed below the title in a very small font.

# JS Widget clone

The clone function of any widget got an `options` parameter. The options define what properties and events are synchonized between the widget and its clone.

# Desktop splitter position remembered across sessions

The position of the desktop splitter position (between the navigation and the bench) is now persisted across sessions, i.e. the previous setting will be restored even after you closed your browser. The position is stored in the HTML 5 *local storage* provided by the local browser. It is therefore a device-specific setting, which is especially useful when accessing the same application through screens with different resolutions.

In case the splitter position should never be remembered, the feature can be disabled globally by setting the property `cacheSplitterPosition` on the desktop to *false*.

# ImageField: support for SVG images and image URLs

It's now possible to use SVG images in the same way as bitmap images. Simply put the .svg file in the `/icons` folder of the client module and reference the SVG image in any widget that supports the `iconId` property. Example:

```
@Override
protected String getConfiguredIconId() {
  return "person.svg";
}
```

Additionally you can now reference an image by URL, for instance an image hosted on an external server. Use the property ´imageUrl` of the AbstractImageField to reference the image. Note: the AbstractImageField defines a priority for which one of the three image properties is used to render the image in the browser:

1. image (Binary resource)
2. imageUrl
3. imageId

Do you want to improve this document? Have a look at the sources on GitHub.