**SADARA Instructor graphics integration**

|  |  |
| --- | --- |
| Version | Date |
| Creation | 26 March 2014 |
| Modification | April 1, 2014 |

|  |  |  |
| --- | --- | --- |
| Written by | Checked by | Approved by |
| L.Cariteau | F. Poindron |  |

This document contains confidential data and information proprietary to RSI. Unauthorized reproduction is strictly prohibited.

[1 Requirements 3](#_Toc384155734)

[1.1 Software requirements 3](#_Toc384155735)

[1.2 Input data requirements 3](#_Toc384155736)

[2 Workflow 1](#_Toc384155737)

[2.1 Global workflow description: 1](#_Toc384155738)

[2.2 Steps description 1](#_Toc384155739)

[3 Graphic files preparation 2](#_Toc384155740)

[3.1 Xgml to Xaml 2](#_Toc384155741)

[3.2 Graphic object implementation specification 2](#_Toc384155742)

[3.2.1 Graphic object definition 2](#_Toc384155743)

[3.2.2 CSV file generation 3](#_Toc384155744)

[3.2.3 Automatic graphic object instantiation 4](#_Toc384155745)

[4 Graphics authoring in blend 6](#_Toc384155746)

[4.1 Setups 6](#_Toc384155747)

[4.2 Blend preparation 6](#_Toc384155748)

[4.2.1 Loading project 6](#_Toc384155749)

[4.2.2 Configuration customization 8](#_Toc384155750)

[4.3 Display the graphic to modify 9](#_Toc384155751)

[4.4 Focus on graphic layout components 11](#_Toc384155752)

[4.5 Behind the Scene (advanced)… 12](#_Toc384155753)

[5 Graphics in viewer 13](#_Toc384155754)

# Requirements

## Software requirements

* IndissPlus v1.9.4
* Viewer 2.0
* Blend v4.0
* RSI Blend extensions
* SADARA Emulation Libraries Setup

## Input data requirements

* .IndissPlus simulator file
* IPCSxxx project files
* FOD/BL/FAILURE implementation documents, delivered by **the project Lead Engineer**.
  + Excel file : FOD/BL/FAILURE object enumeration
  + PDF Files : FOD positions

# Workflow

## Global workflow description:

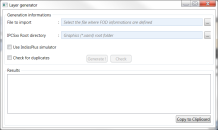
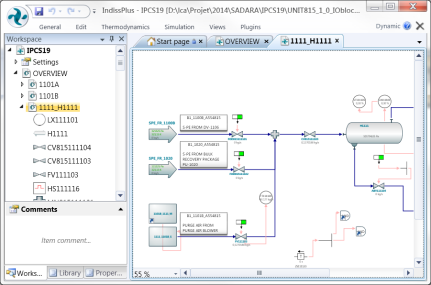
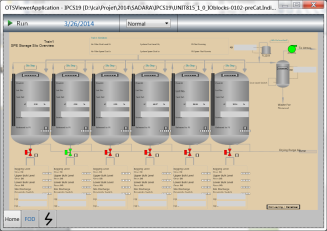
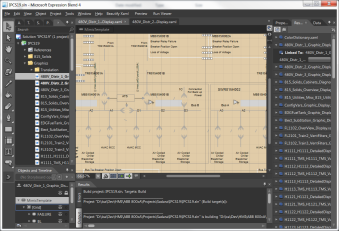


ABB xgml graphic files

xgml to xaml translator

xaml and project files (.sln .csproj)

Graphic object implementation (.csv)

Blend 4 and RSI addins and libraries

Communication

Possible

communication

Possible

communication

IndissPlus and simulator

Layer Generator

Viewer

1

2

3

4

5

6

7

## Steps description

The whole process can be recursive. Here is a short description of steps involved (will be detailed in the next chapters):

1. ABB xgml files are exported graphics from ABB DCS
2. Those files are translated into xaml files. Xgml files should be sent to **RSI R&D team** for translation
3. Project lead engineer generates a csv file that enumerates the different graphic object to implement in the different Layer (FOD/BL/FAILURE).
4. The “layer generator” application takes as input the graphic object implementation csv file and instantiate automatically those graphics in the corresponding graphic pages and layer
5. Xaml, solution and project files are ready to be authored in Blend
6. Graphic objects need to be positioned correctly and FOD lines needs to be drawn according to P&ID specification. This is done within Blend
7. Blend generates the graphic dlls that will be loaded into the IndissPlus viewer

# Graphic files preparation

## Xgml to Xaml

RSI R&D has developed a tool that translates ABB exported graphics (xgml) into graphics that IndissPlus viewer handles (xaml).

RSI R&D will directly manage this translation.

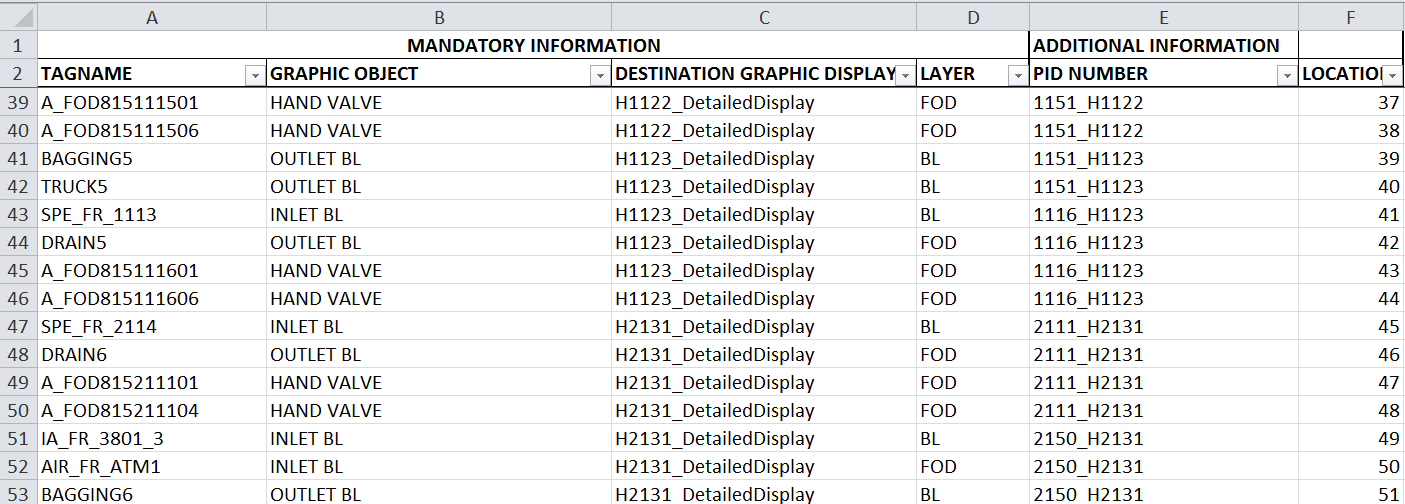
## Graphic object implementation specification

### Graphic object definition

The project lead engineer needs to generate a CSV file (from an Excel files) that specify the different graphic object to implement in the different Layer (FOD/BL/FAILURE). Object properties must, at least, contain the following information:

* Graphic display destination name
* Tag Name (from IndissPlus simulator)
* Object type (FOD/BL/FAILURE)

Sample:



An Excel template file is provided.

Types of graphic objects are:

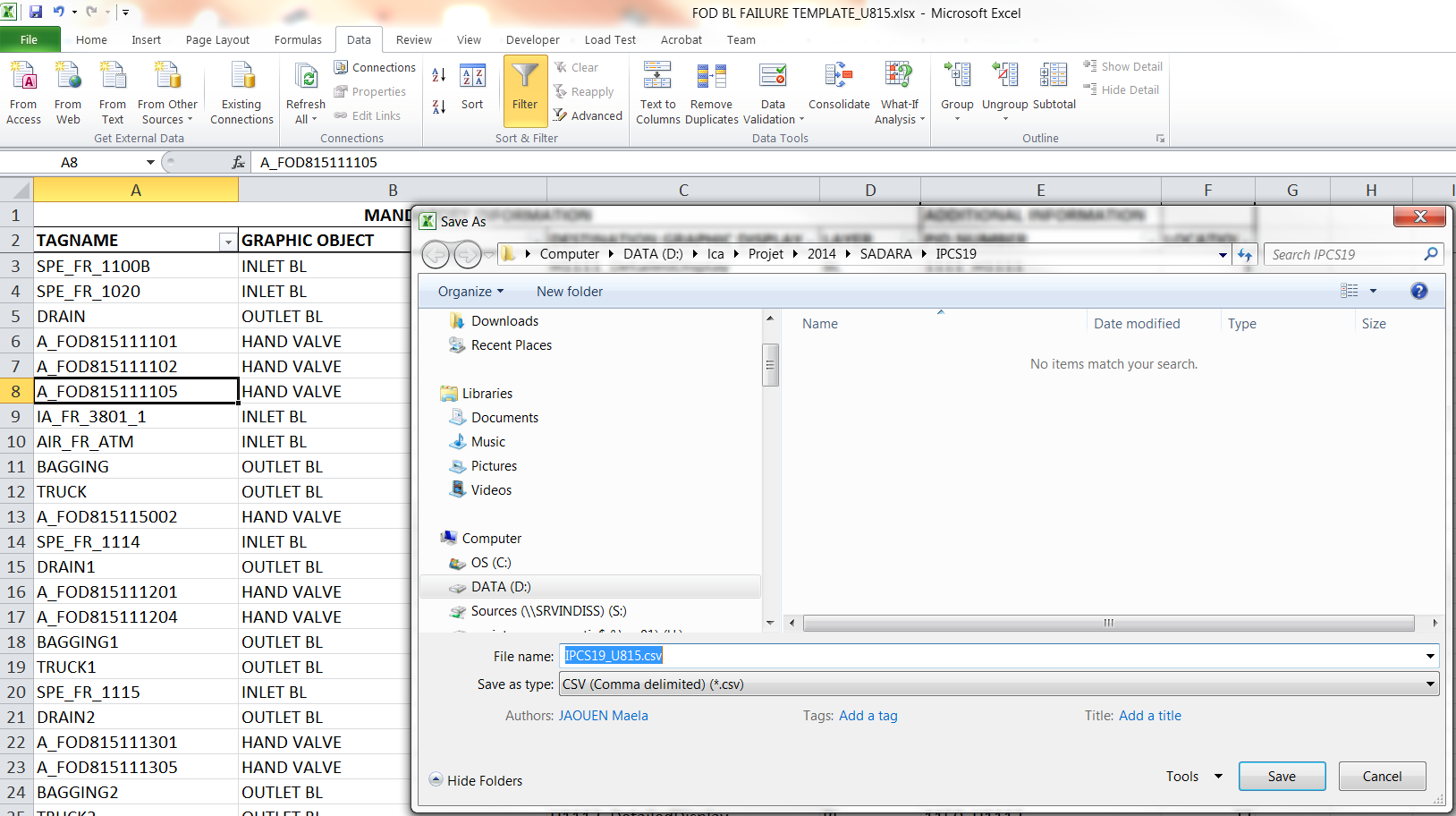
* HAND VALVE
* BLIND FLANGE
* PSV
* REGULATOR
* INLET BL
* OUTLET BL
* EXCHANGER
* PUMP
* AIRCOOLER
* FILTER
* PUSH BUTTON

Graphic layers:

* BL
* FOD
* FAILURE
* FOD+FAILURE

### CSV file generation

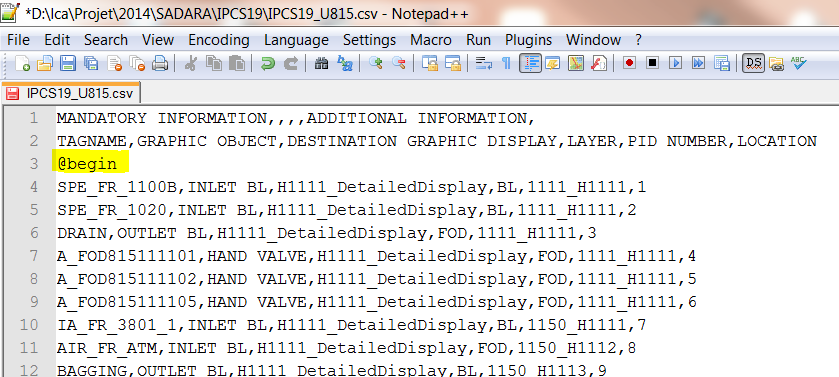
From the Excel file, select “Object List” sheet and go to save as and select type “CSV (Comma delimited)”

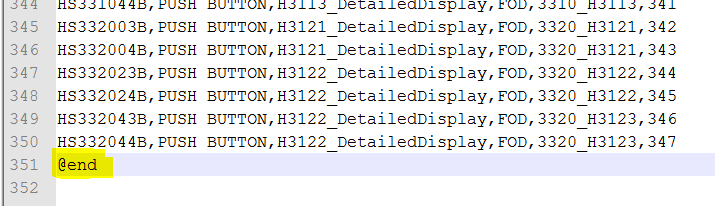


**Important**:

When the csv file is generated, open it in a text editor and add after the header definition, a specific tag: @begin

At the end of the graphic object definition, a specific tag needs to be inserted: @end

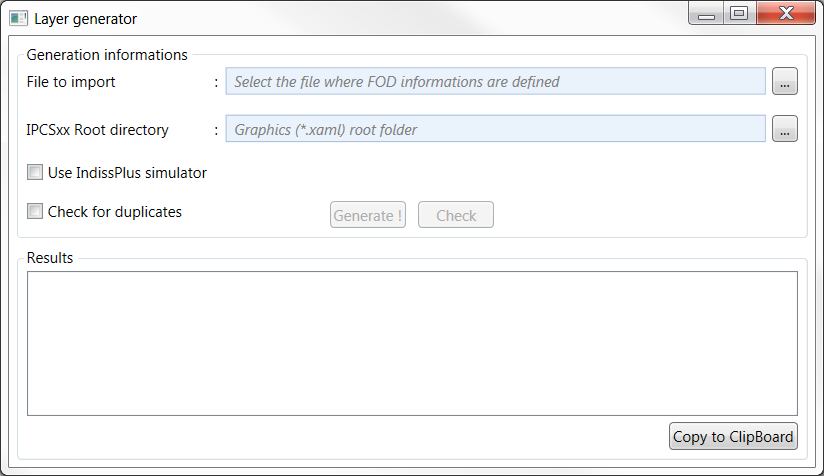




### Automatic graphic object instantiation

The “layer generator” application takes as input the graphic object implementation csv file and instantiate automatically those graphics in the corresponding graphic pages and layer.

Launch the LayerGenerator.exe application:

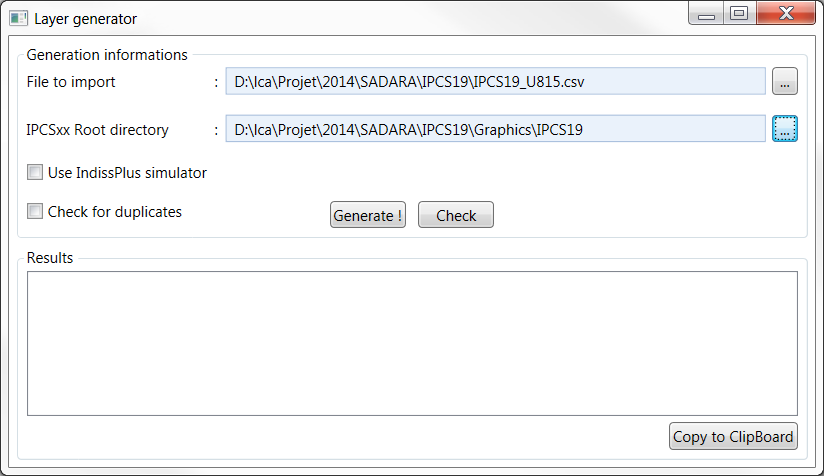


Select the root folder that contains the xaml graphics (from xgml translation)

Select Graphic object definition

CSV file

Once csv file and graphic folder is set, there is the choice to directly generate the graphic object instantiation or to check the validity of the configuration file.

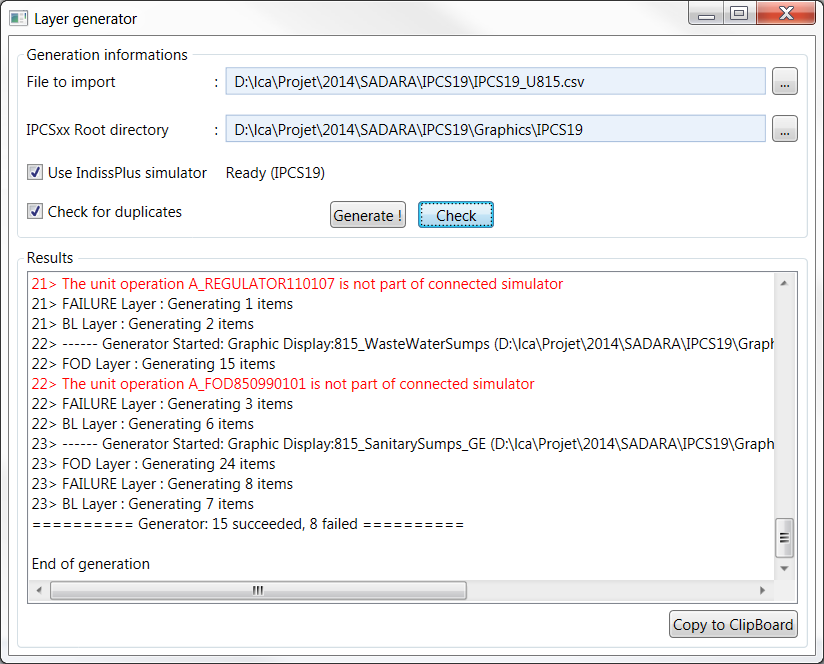


If user selects, the Layer Generator tool will connect to IndissPlus and display the name of the simulator if available (only for local connection). It then will check if tag names specified in the csv file can be found in the simulator

If user selects, the tool will check for duplicate tag names.

User has then the choice to directly generate  or to make a first check 

C:\Users\ludovic cariteau\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\BLXCMZSL\MC900434750[1].png If « Use IndissPlus simulator » option is set and there is an error in the graphic page generation, the whole graphic page is not generated



At the end of the generation, a report is available with the list of succeeded generated object and potential errors. It is possible to copy the report into a text editor for further analysis.

# Graphics authoring in blend

## Setups

Install:

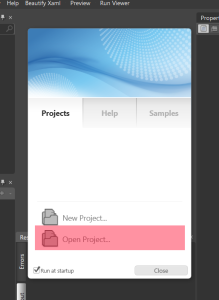
* Microsoft Expression Blend 4
* Blend 4 Extensions\_setup (RSI addins for Blend)
* Sadara Emulation Libraries\_setup

Sadara Emulation libraries setup contains objects definitions that are automatically instantiated by the xgml to xaml translator and the Layer generator. Those libraries will evolve continuously as more objects are created and translated

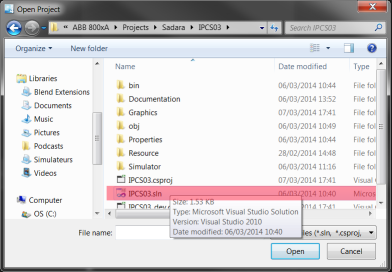
## Blend preparation

### Loading project

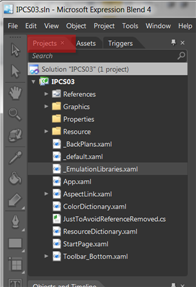
* Start Blend :



* Click “**Open Project** “ and load the **IPCSxx.sln** solution file name :

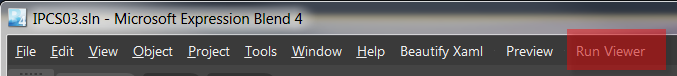


* Once loaded, the project hierarchy is displayed in the **Project** tab item :

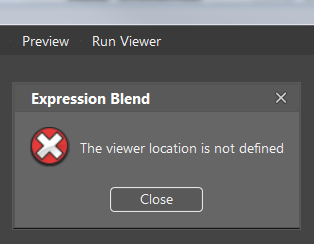


### Configuration customization

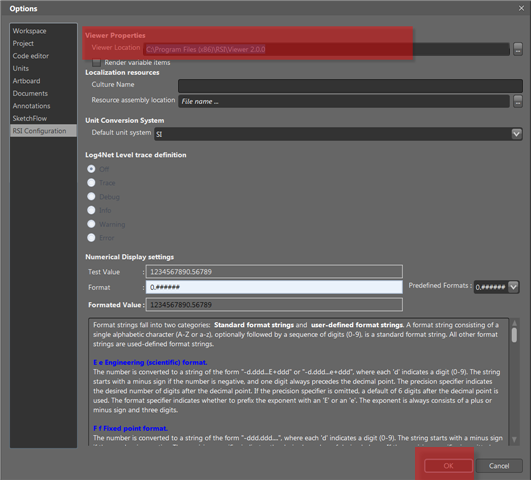
Once “Blend extensions“is installed, a new menu entry “Run viewer” is added to the main menu bar of Blend.



This entry is used to generate graphics binary file and display them through the Viewer 2.0 software. On first activation, the following error message is displayed:

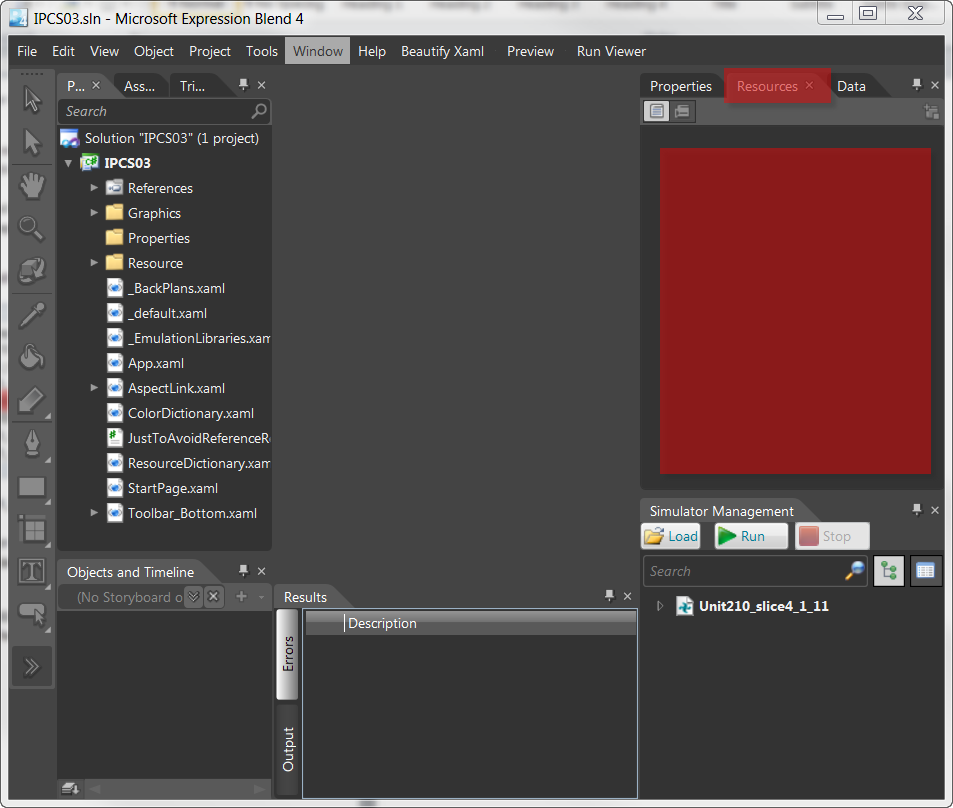


Once closed, the **RSI Configuration** is opened. You now have to define the location where the Viewer 2.0 software is located. Usually, the viewer is installed in **C:\Program Files (x86)\RSI\Viewer 2.0.0** folder:

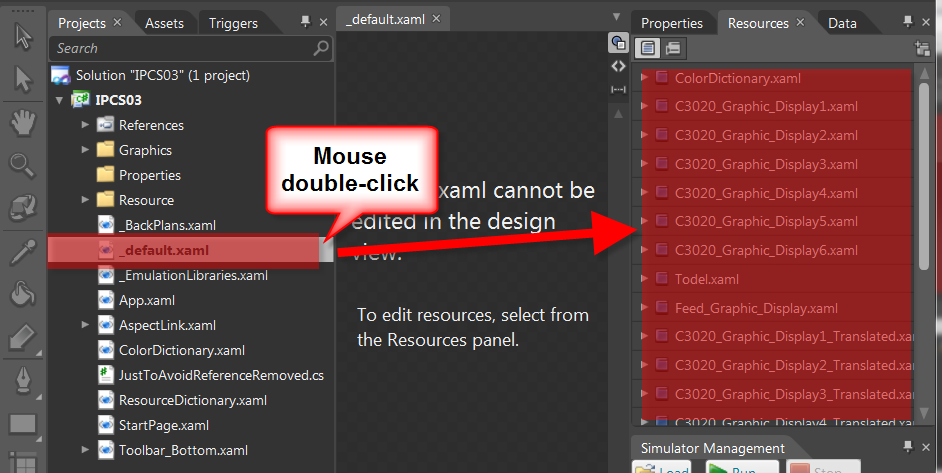


## Display the graphic to modify

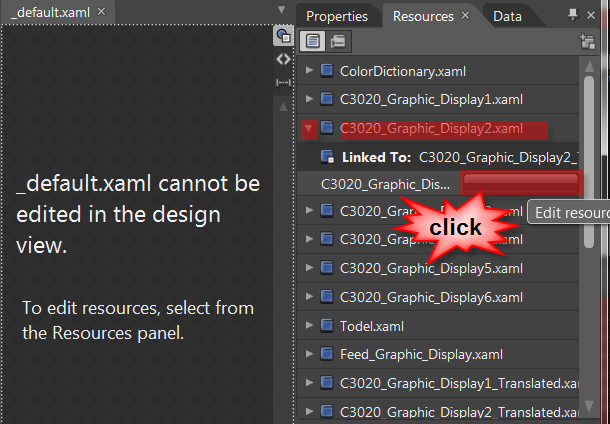
Goal: Locate and display the graphic associated to *C3020Graphic\_Display2* location reference. Displayable graphics (or resources) are displayed in the “Resources” panel. At first, it may be empty.

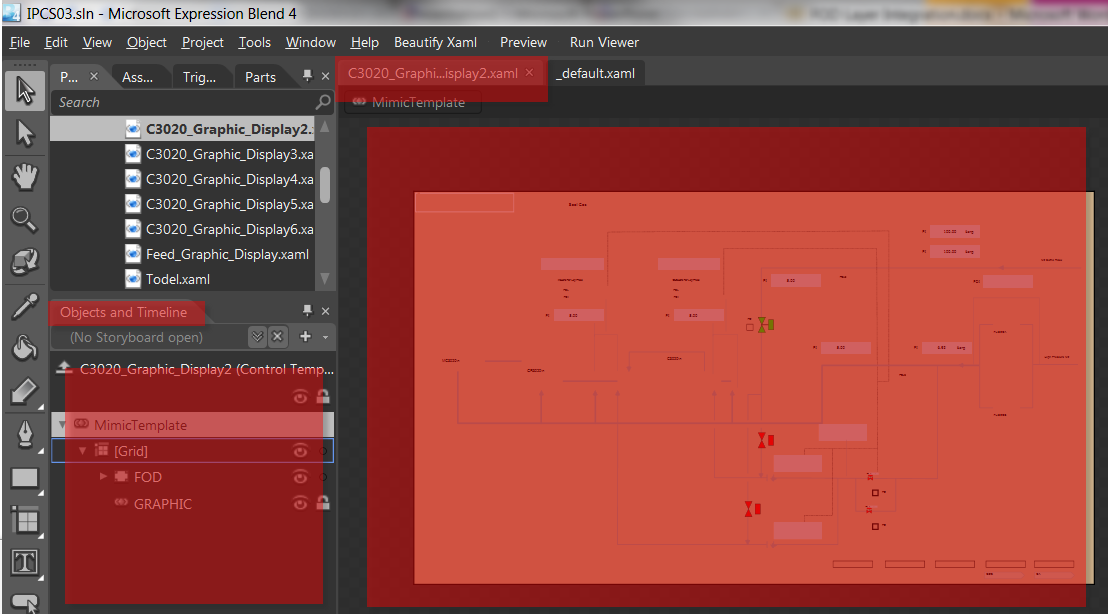


To feed it, just double-click on any .xaml file from the “*Projects*” tree view control:

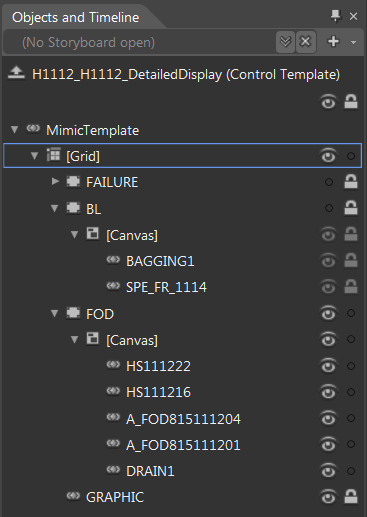


Now that resources are available, locate the graphic associated to *C3020Graphic\_Display2*, expand the item, and Edit the resource



On click, the graphic display is rendered, and the *“Objects and TimeLine”* panel is completed. 

## Focus on graphic layout components

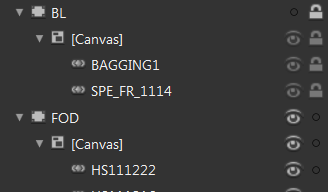


Graphic BL, FAILURE, FOD objects are generated in three separated layers containing their respective objects:

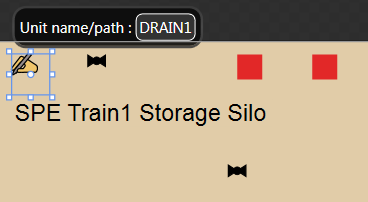
* FAILURE
* BL
* FOD

Translated graphics are located in the “GRAPHIC” layer. This layer is locked and can be regenerated without impact on the other layers.

A good practice is to hide and lock unused layers and show only active layer:

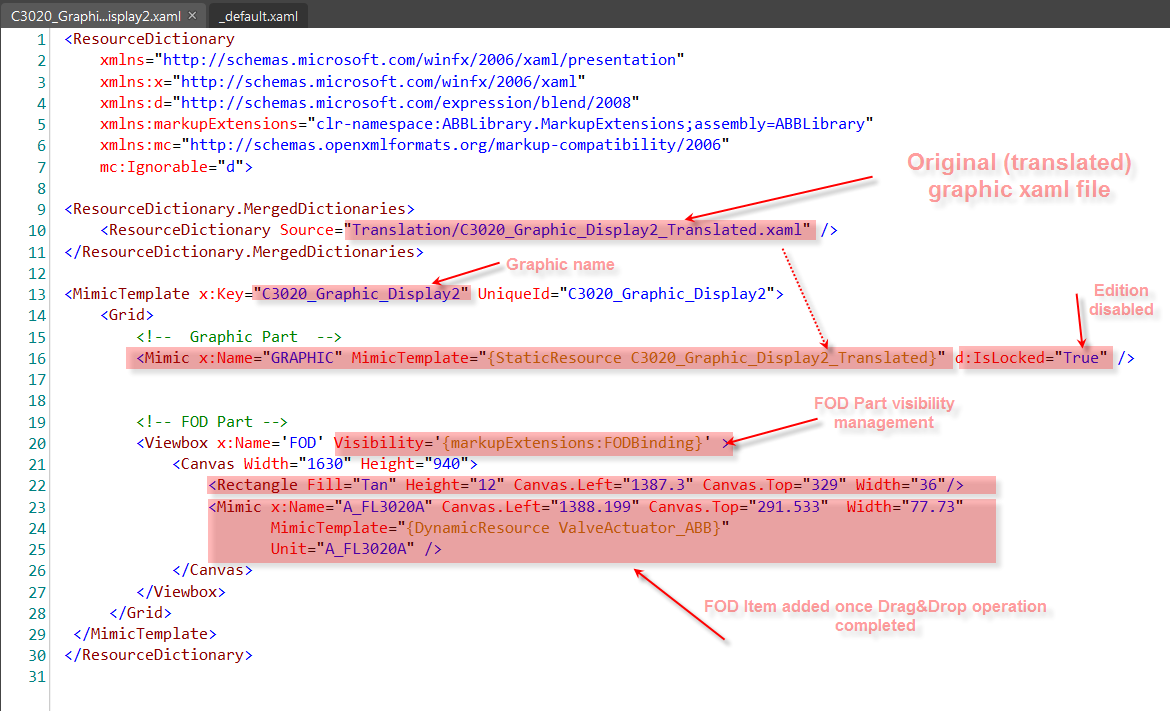


In the edit view, it is then possible to move and place objects:



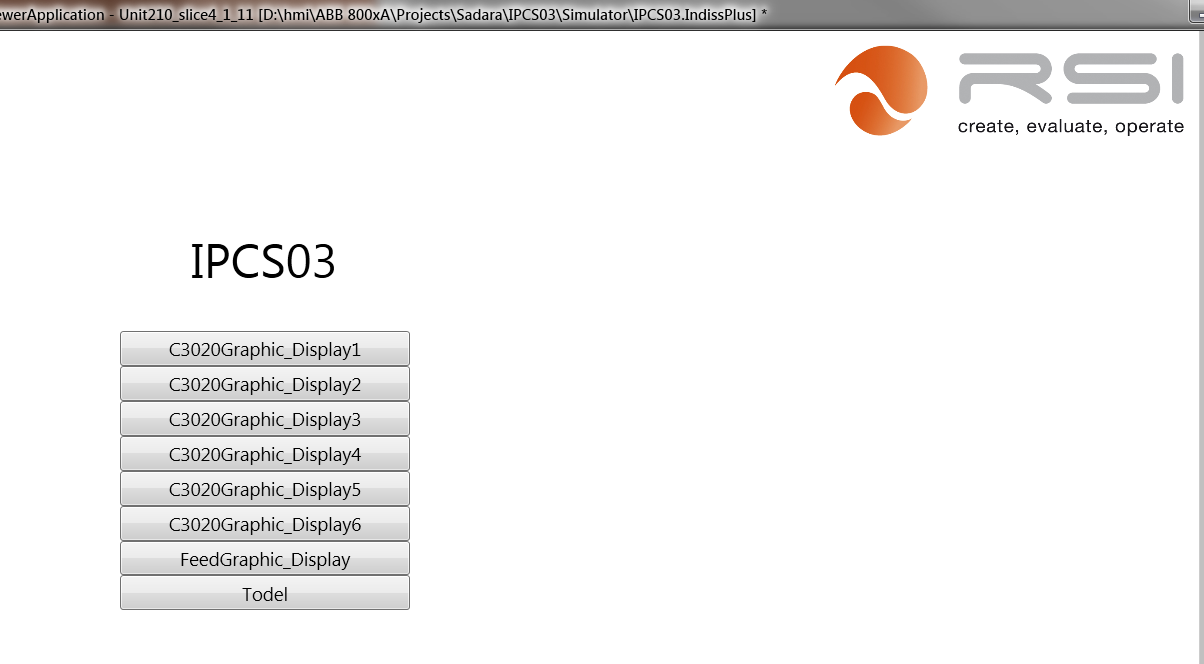
## Behind the Scene (advanced)…

This operative procedure does not require manual .Xaml content edition. However, it may be interesting to know what is done “under the cover”. Here is the Xaml code that we have completed :

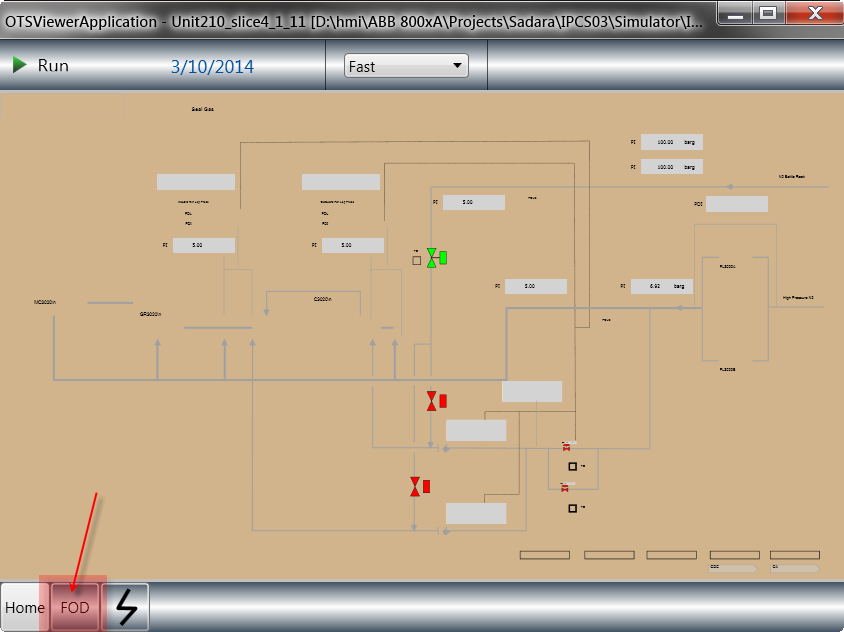


# Graphics in viewer

To check the FOD layer behavior “in real”, you just have to click on the **“Run Viewer”** menu entry. Once “binaries” generation completed, the Viewer 2.0 software is started, and the edited graphics are displayed.



* Navigate to the edited screen (C3020Graphic\_Display2 in our case)



* Check the FOD button to activate the FOD Layer visibility

