

Software Engineering

### ESOF – 3MIEIC05

T3 – Open Source

Duarte Nuno Esteves André Lima de Carvalho – up201503661@fe.up.pt

João Álvaro Cardoso Soares Ferreira – [up201605592@fe.up.pt](mailto:up201605592@fe.up.pt)

[João Augusto dos Santos Lima](https://sigarra.up.pt/feup/pt/fest_geral.cursos_list?pv_num_unico=201605314) – [up201605314@fe.up.pt](mailto:up201605314@fe.up.pt)

Índice

Table of Contents

[ESOF – 3MIEIC05 1](#__RefHeading___Toc5863_1682680128)

[Issue #15009 3](#__RefHeading___Toc5865_1682680128)

[Issue Description 3](#__RefHeading___Toc5867_1682680128)

[Requirements 3](#__RefHeading___Toc5869_1682680128)

[User Story: 3](#__RefHeading___Toc5871_1682680128)

[Source Code Files 4](#__RefHeading___Toc5873_1682680128)

[System Architecture 4](#__RefHeading___Toc5875_1682680128)

[Design of the Fix 5](#__RefHeading___Toc5877_1682680128)

[Fix of the Issue 5](#__RefHeading___Toc5879_1682680128)

[Source Code Changes 6](#__RefHeading___Toc5881_1682680128)

[Validation 6](#__RefHeading___Toc5883_1682680128)

[class\_projectsettings.rst (line 1828): 7](#__RefHeading___Toc5885_1682680128)

[class\_projectsettings.html 7](#__RefHeading___Toc5887_1682680128)

[Submission of the Fix 8](#__RefHeading___Toc5889_1682680128)

[Godot Issue #24183 9](#__RefHeading___Toc1129_2928827938)

[Issue 9](#__RefHeading___Toc1131_2928827938)

[Requirements 9](#__RefHeading___Toc1133_2928827938)

[User story 9](#__RefHeading___Toc1135_2928827938)

[Source code files 9](#__RefHeading___Toc1137_2928827938)

[System architecture 10](#__RefHeading___Toc1139_2928827938)

[Fix design 11](#__RefHeading___Toc1141_2928827938)

[Fix source code 11](#__RefHeading___Toc1143_2928827938)

[Validate the fix 12](#__RefHeading___Toc1145_2928827938)

[Submission of the fix 12](#__RefHeading___Toc1147_2928827938)

# Issue #15009

## Issue Description

Issue link: https://github.com/godotengine/godot/issues/15009

In the generation of the documentation for the project, there is a mistake regarding how the documents display a specific interaction with special characters. If the XML, for whatever reason, has parenthesis after brackets, the documentation displays an error.

## Requirements

As a user, I want to be able to read documentation that is well formatted and displayed correctly so that I can understand how the program works. Syntax errors that omit information, such as the one in Issue #15009, prevent that.

The documentation for Godot is displayed in two different ways - online and in the program itself. The online version of the documentation has a display error for specific characters.

### User Story:

When we look at [this page of Godot's online documentation](https://docs.godotengine.org/en/3.0/classes/class_projectsettings.html?highlight=projectsettings), looking at the member function description we can see the following:

*Add a custom property info to a property. The dictionary must contain: name:String), and optionally hint:int), hint\_string:String.*

This is incorrect and it's apparent that in the process of creating the HTML files for the documentation, something went wrong with the escaping for parenthesis. The documentation for Godot is generated from XML files that contain all the information about the specific elements of the program. In the XML file for a page (ProjectSettings.xml), that line shows up as the following:

*Add a custom property info to a property. The dictionary must contain: name:[String](the name of the property) and type:[int](see TYPE\_\* in [@GlobalScope]), and optionally hint:[int](see PROPERTY\_HINT\_\* in [@GlobalScope]), hint\_string:[String].*

That is how it should be displayed in the HTML files as well.

## Source Code Files

The main source code file we'll need to alter is "makerst.py". This is the file responsible for generating RST files out of the XML files created to document the program. These RST files, in turn are copied over to the godot-docs software where, from them, the HTML files get generated.

As the XML files are correct, and the generation of HTML files from RST files doesn't change the content at all but merely converts it (as both are very similar), the error to correct can be pinpointed to the "makerst.py" file. Specifically, in the condition after line 285:

*`if escape\_post and post\_text and post\_text[0].isalnum():`*

This condition should also escape for parenthesis.

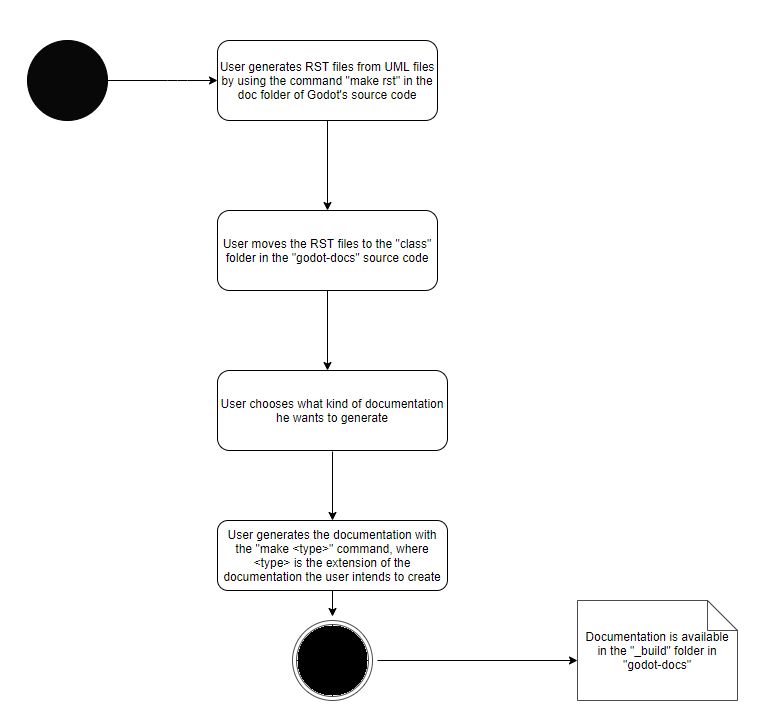
## System Architecture

The architecture of godot as a whole is not relevant for the fixing of this issue, as this issue only deals with a system that works in parallel to the software itself – the generation of the documentation. The UML files to describe Godot's elements are written by hand (allowing for easy editing whenever something gets added, removed or changed) and, from them, we can generate Doxygen documentation or RST files - for Doxygen the Makefile refers to the Doxyfile for RST it calls makerst.py, a Python file that generates RST files by interpreting the XML files in the classes folder.

There used to also be the ability to generate Markdown files directly via a file called makemd.py, but it was removed in [this commit](https://github.com/godotengine/godot/commit/0cc3aff8edd5f63436814a6b2219044d94eb8201" \l "diff-9a09b4dfda82e3e665e31092d1c3ec8d) after being deemed not useful due to a previous rework in how documentation for Godot was generated and stored.

The RST files generated from makerst (stored in the \_build folder) then need to be copied over to the classes folder in the [Godot-Docs](https://github.com/godotengine/godot-docs) software, which is developed specifically to generate Godot's Documentation. This program can, from the RST files, create documentation in plenty of different formats, such as HTML, LaTeX, JSON, Pickle, text, a manual, or even turn them back into XML. There we generate the HTML files. For this, Godot-Docs uses [Sphinx](https://github.com/sphinx-doc/sphinx), software created specifically to generate documentation from RST files.

Simplifying this explanation, we have various different folders and files that need to interact with each other over two different programs. Here tis an action diagram explaining how documentation (of any kind) is generated in Godot:



## Design of the Fix

Considering the small scale of the issue at hand and how direct the fix is (adding an extra condition for escaping characters), nothing in the structure of the program is changed. We will edit the functions responsible for the syntax regarding the handling of parenthesis and brackets and alter them so that they interpret the cases where the symbols appear in succession correctly.

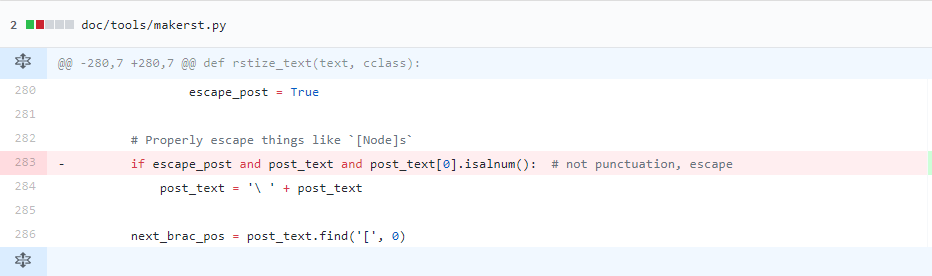
## Fix of the Issue

The fix was done as detailed in the "Design of the Fix" section and the results were as expected. Developing unit tests for this fix is not practical due to the nature of the issue at hand - it requires files changing folders changes and executing multiple programs and python files, such as makerst and Sphinx, only to then check the HTML output, which can be easily be done manually. So we will display both the changes in code and the changes in the output, both in the RST files and in the webpages.

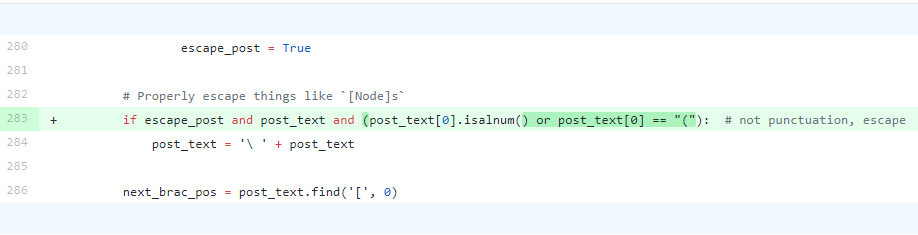
### Source Code Changes

As was stated, the only file changed was \*\*makerst.py\*\*, in the following way:

**Before:**



**After:**



### Validation

We followed the procedure detailed in the "System Architecture" section to get out RST files and our HTML files before and after the code was changed. Here they are the steps done in order to recreate the test:

* Generate RST files using “make rst” in the doc folder, in Godot’s source code
* Copy the RST files that were just generated (in the \_build sub-folder) over to the “classes” folder in Godot-Doc’s source code
* Use the “make html” command in the Godot-Doc’s folder
* The HTML files will be ready to open in the \_build sub-folder. Opening one that contains the kind of syntax that was fixed in this issue will allow you to see if the issue was fixed or not (here we use class\_projectsettings.html, but others where the same syntax is used would also be fitting).

Changes can also be observed in the RST files, although it is not as obvious by reading them directly them if the syntax is correct or not.

We will use the Project Settings class as our example, as it was the class used in the "User Story" section above, by taking a look at the lines mentioned previously:

#### **class\_projectsettings.rst (line 1828):**

**Before:**

Add a custom property info to a property. The dictionary must contain: name::ref:`String<class\_String>`(the name of the property) and type::ref:`int<class\_int>`(see TYPE\\_\\* in :ref:`@GlobalScope<class\_@GlobalScope>`), and optionally hint::ref:`int<class\_int>`(see PROPERTY\_HINT\\_\\* in :ref:`@GlobalScope<class\_@GlobalScope>`), hint\_string::ref:`String<class\_String>`.

**After:**

Add a custom property info to a property. The dictionary must contain: name::ref:`String<class\_String>`\ (the name of the property) and type::ref:`int<class\_int>`\ (see TYPE\\_\\* in :ref:`@GlobalScope<class\_@GlobalScope>`), and optionally hint::ref:`int<class\_int>`\ (see PROPERTY\_HINT\\_\\* in :ref:`@GlobalScope<class\_@GlobalScope>`), hint\_string::ref:`String<class\_String>`.

As you can see, we now have added the "\" character before every parenthesis so that the parenthesis is escaped and thus interpreted properly by the HTML.

Here is also the same excerpt on the [class\_projectsettings.html](https://docs.godotengine.org/ko/latest/classes/class_projectsettings.html) page:

#### **class\_projectsettings.html**

**Before:**

## 

## Submission of the Fix

The pull request was submited [here](https://github.com/godotengine/godot/pull/24266) and, after passing the compilation and organizational codes set up to verify commits, was accepted and merged onto godot's master branch on 11/12/2018. Issue #15009 was then closed.

# **Godot Issue #24183**

## Issue

We chose issue [#24183](https://github.com/godotengine/godot/issues/24183) : [CollisionShape2D Disabled Check Problem](https://github.com/godotengine/godot/issues/24183)

This issue is caused by calculations performed with variables that are not numbers on objects with disabled shapes. This leads to objects not being rendered correctly.

## Requirements

To reproduce this issue you will need the latest release at this moment of Godot: [3.0.6-stable](https://github.com/godotengine/godot/releases/tag/3.0.6-stable).

Then, you just need to do the following:

1. Create a new scene
2. Add RigidBody2D as a root
3. Add CollisionShape2D to RigidBody as a child
4. Check CollisionShape2D disabled feature
5. Play

This steps were pointed out by [cagdasc](https://github.com/cagdasc) the issue creator.

### User story

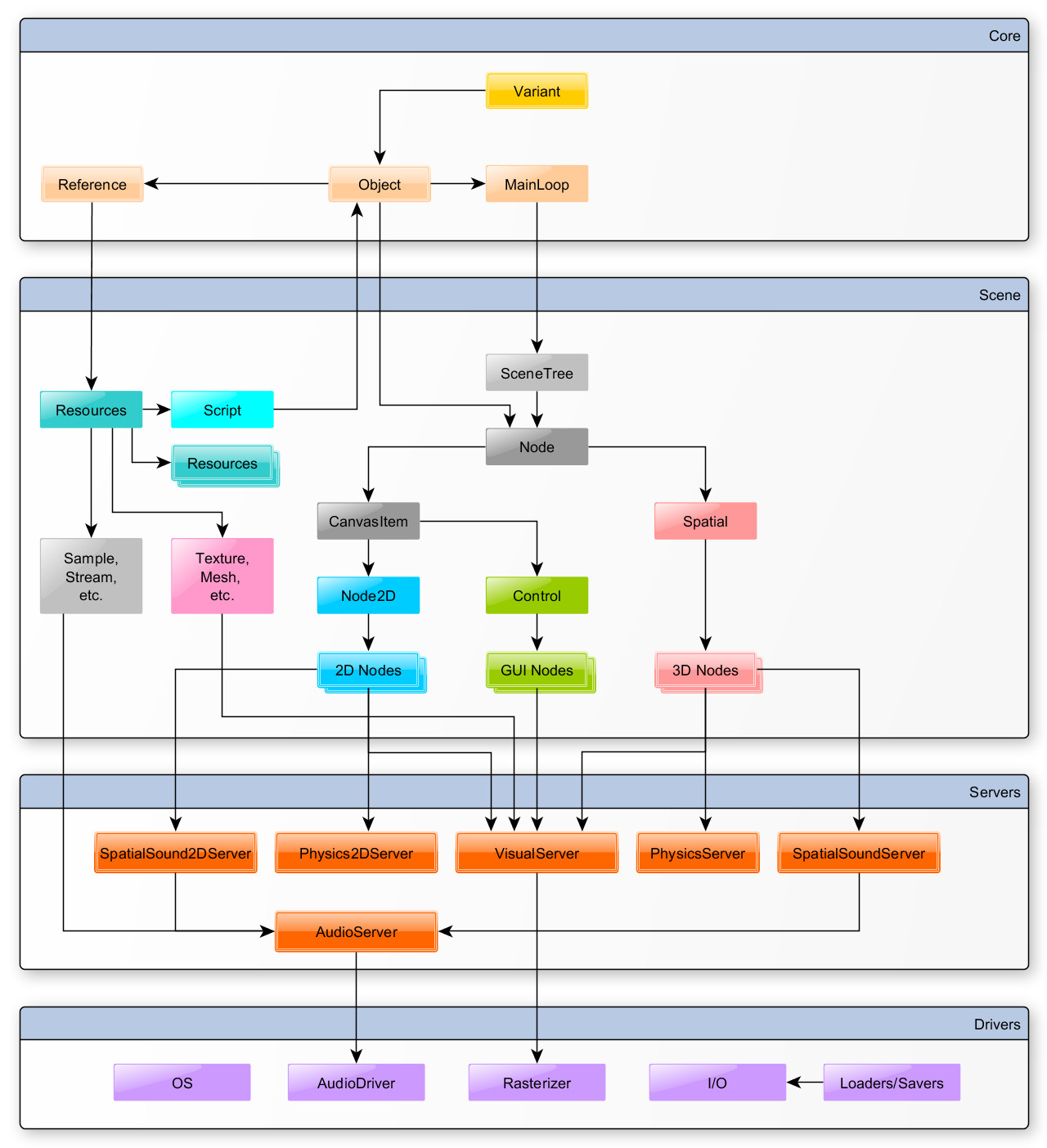
If the user adds a CollisionShape2D to a RigidBody2D and sets it to disabled, the RigidBody2D is not rendered correctly to the screen. If the user unsets the disabled option, everything is rendered correctly.

## Source code files

The error probably is in one of the following files:

* [**servers/physics\_2d/body\_2d\_sw.cpp**](https://github.com/godotengine/godot/blob/master/servers/physics_2d/body_2d_sw.cpp) - probabliy in the function: void Body2DSW::update\_inertias. This file is responsible for handling physics for 2D objects.
* [**servers/physics\_2d/collision\_object\_2d\_sw.h**](https://github.com/godotengine/godot/blob/master/servers/physics_2d/collision_object_2d_sw.h). This file is responsible for handling collisions between 2D objects.

## System architecture

 GODOT provides an UML with the system architecture. It's the following.

As we can see, the engine architecture is splited into four major packages:

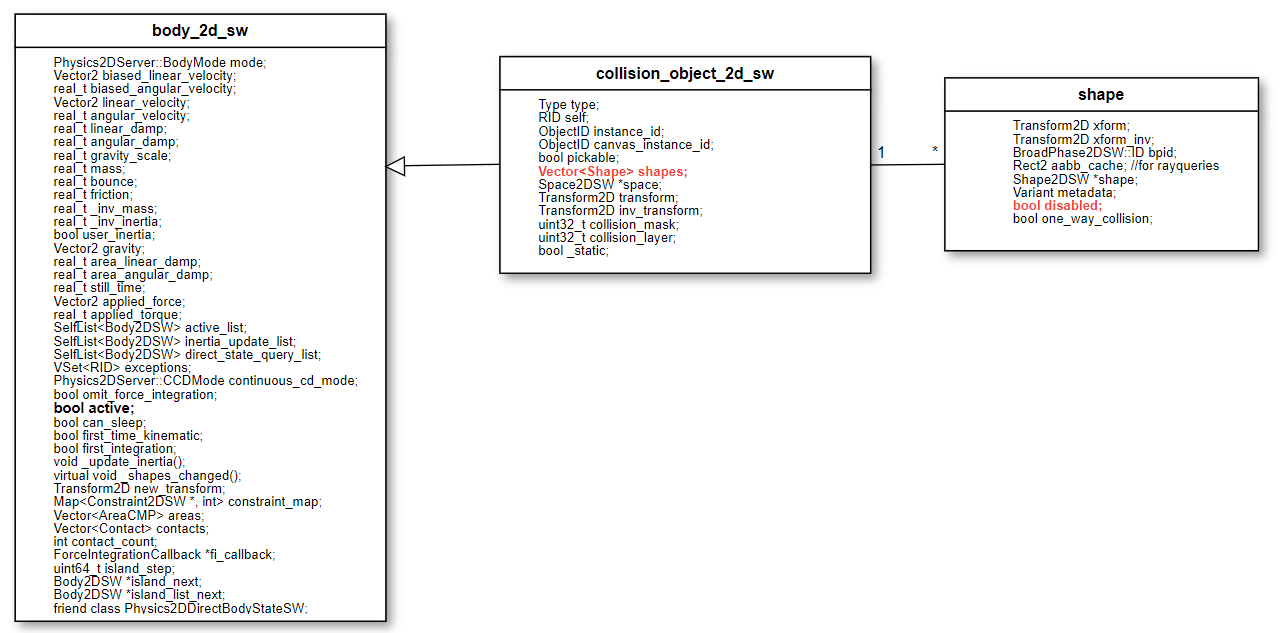
* Core
* Scene
* Servers
* Drivers

Each one is like a layer, the packages at the top use the packages below them. This issue is on Servers package. So it will affect the packages above, making it a serious issue.

## Fix design

Our issue is related with collisions between bodies in which of them has a disabled shape. So the first thing we thought was: we need to check if the body we are processing has a disabled shape, if it does it won’t be processed for collisions.

Figure 1 - class UML of the classes related to this issue



So, **collision\_object\_2d\_sw** inherits **body\_2d\_sw** methods, behaviors and properties. Has you can see on the class diagram on the top (element highlighted as bold red), **body\_2d\_sw** has an array of **shapes**. **Shapes** has a Boolean attribute ‘disabled’ which indicates if the shape is disabled or not.

S o, the solution for this is issue is right here, when calculating inertia we go through shapes, and the ones with this attribute set to false have to be ignored.

### Fix source code

We started by implementing a function to check if a certain shape is disabled or not in collision\_object\_2d\_sw.h.

\_FORCE\_INLINE\_ bool is\_shape\_set\_as\_disabled(int p\_idx) const {

CRASH\_BAD\_INDEX(p\_idx, shapes.size());

return shapes[p\_idx].disabled;

}

Then we modified function Body2DSW::update\_inertias() in source file body\_2d\_sw.cpp. We modified it so that it checks if the shape is disabled or not, using the previously described function, in a for loop that goes through all the shapes and calculates inertia.

…

for (int i = 0; i < get\_shape\_count(); i++) {

if (check\_if\_shape\_is\_not\_disabled(i)) { //it only adds to inertia if shape is not disabled

const Shape2DSW \*shape = get\_shape(i);

real\_t area = get\_shape\_aabb(i).get\_area();

real\_t mass = area \* this->mass / total\_area;

Transform2D mtx = get\_shape\_transform(i);

Vector2 scale = mtx.get\_scale();

\_inertia += shape->get\_moment\_of\_inertia(mass, scale) + mass \* mtx.get\_origin().length\_squared();

}

}

### Validate the fix

In order to validate the fix we checked if the code compiled. Then we loaded a [scene](https://github.com/godotengine/godot/files/2652606/24183-rigidbody-disabled.zip) made by another user that replicated this issue and checked if the inertia values were correctly computed. Without our fix, nothing on the scene would move, after the fix the scene was rendered correctly and an object with his shape disable falls to a static one.

## Submission of the fix

We submitted our pull request with a simple description of our changes. Unfortunately, other user also had submitted a fix to this issue, very similar to ours on day before we make our pull request, so our pull request was rejected for the following reason:

“Thanks for the contribution! The same fix has however already been proposed a few days ago: #24218“ – GitHub user [akien-mga](https://github.com/akien-mga).

We changed our solution from delivery #1 to #2 because we also changed the issue we proposed to do, because we couldn’t replicate the issue from the first delivery.