

TraceLab Component Breakdown

Revision 1

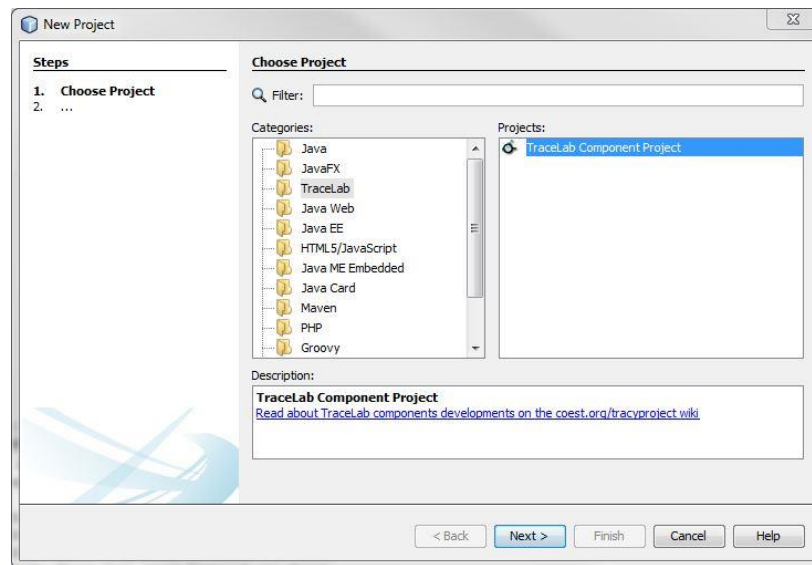
Date: 08/24/2016

Assumptions

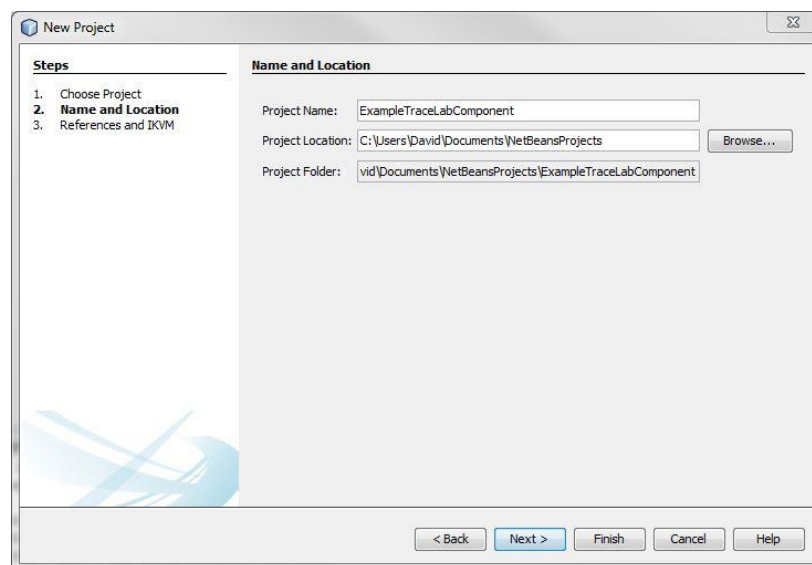
1. Assumption that the Netbeans TraceLab templates have been installed per the documentation provided at: http://coest.org/coest-projects/projects/tracelab/wiki/Installing_TraceLab_Templates_in_Netbeans

Create a TraceLab Component Project

2. Create a new project and select TraceLab under the Categories window. Next select the TraceLab Component Project under the Projects window. Click the “Next” button to progress to the next screen.



3. Enter the Project Name and verify the Project Location and Project Folder are set as desired. Click the “Next” button to progress to the next screen.

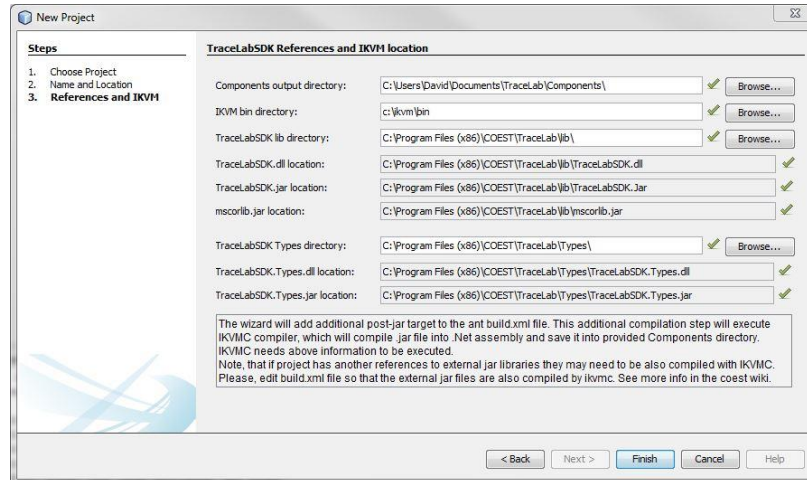


TraceLab Component Breakdown

Revision 1

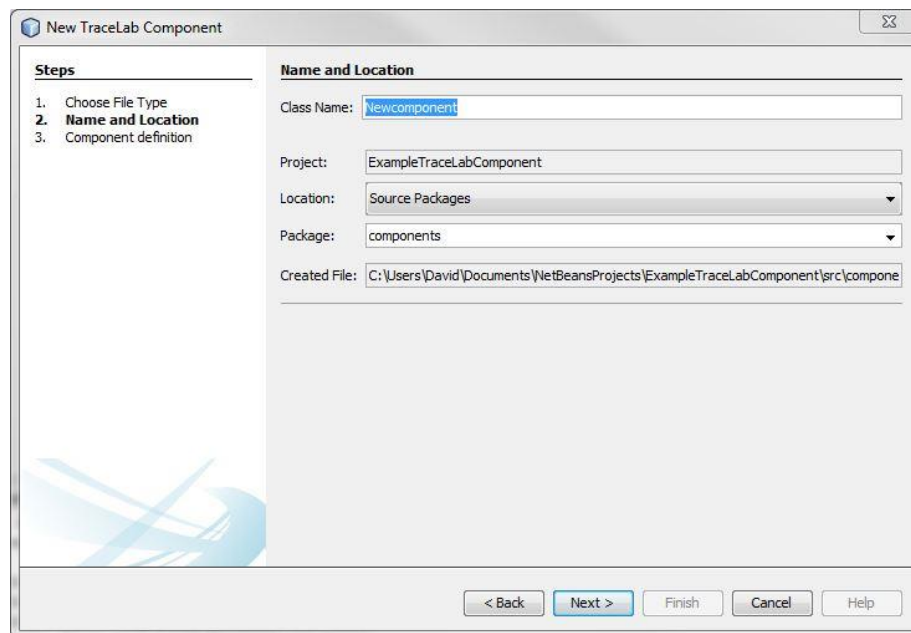
Date: 08/24/2016

4. Verify that the TraceLab References and IKVM location have been set. You should see green check marks next to each field. Click the “Finish” button to conclude the creation of the project.



Add TraceLab Component to Project

5. Create a new File. Under the Categories window select TraceLab. Under the File Types window select TraceLab Component. Click the “Next” button to progress to the next screen.
6. Fill in the Class Name field and set the Package to components.

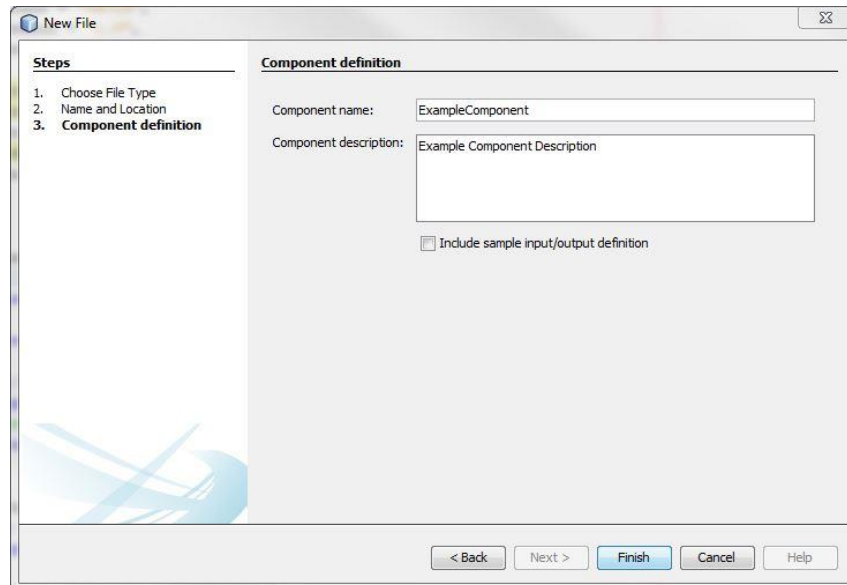


TraceLab Component Breakdown

Revision 1

Date: 08/24/2016

7. Fill in the Component Name and the Component Description with relevant information. This information will be displayed to the user in TraceLab. Click the "Finish" button to complete the creation of the template TraceLab component.



Component Class Sections

8. The `@ComponentAttribute.Annotation` section located near the top of the newly created TraceLab component file includes the information entered in the Component Definition window during the creation of the TraceLab component.

```
@ComponentAttribute.Annotation(  
    Name = "ExampleComponent",  
    Description = "Example Component Description",  
    Author = "David",  
    Version = "1.0")
```

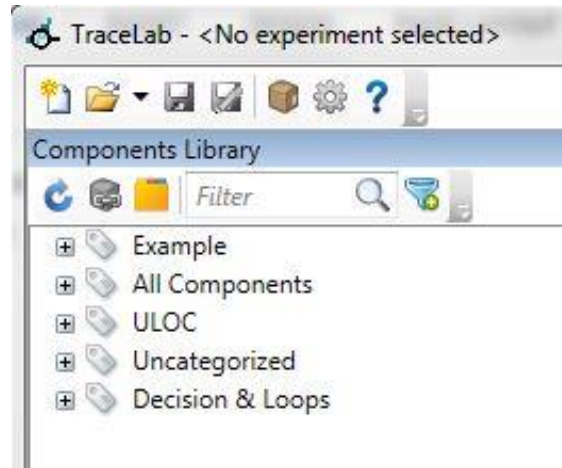
TraceLab Component Breakdown

Revision 1

Date: 08/24/2016

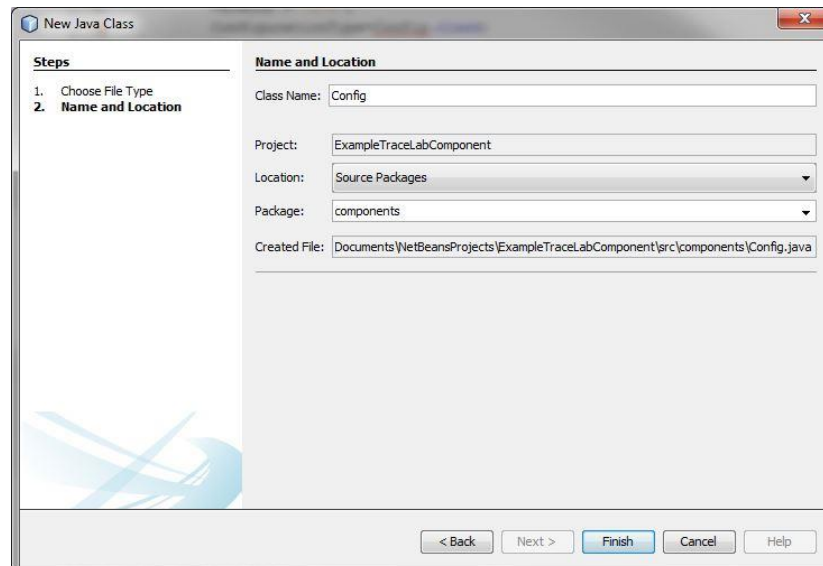
9. The text entered into the `@TagAttribute.Annotation()` call will set the category of the component in TraceLab. This allows for the organization of the components.

```
@TagAttribute.Annotation("Example")
```



Configuration File

10. Create a new Java Class file. The Java Class appears to generally named Config. Add to the components package.



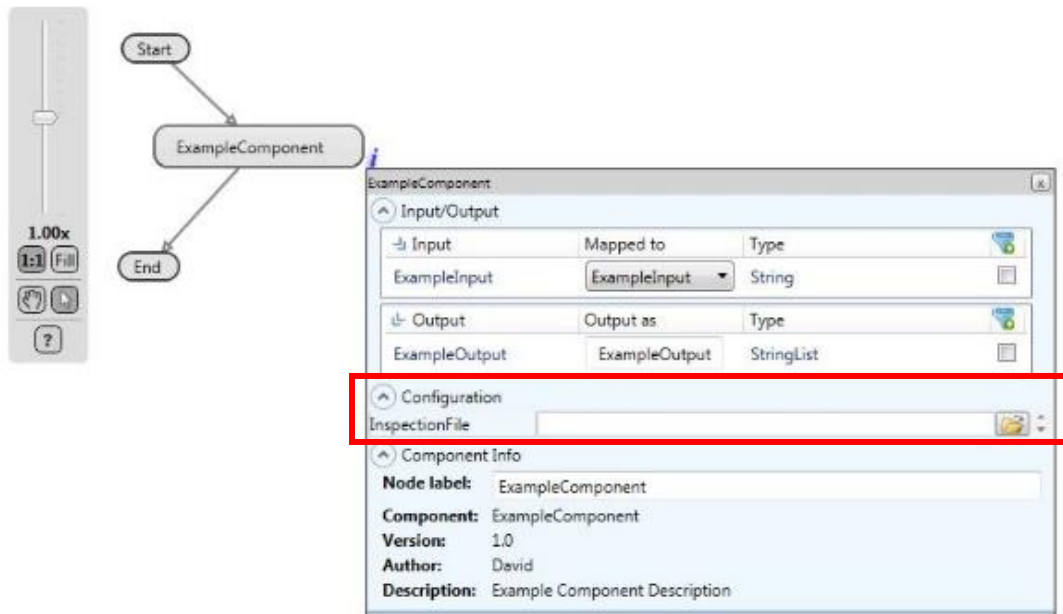
TraceLab Component Breakdown

Revision 1

Date: 08/24/2016

11. Below is an example of a configuration file. The configuration file creates a configuration entry in the generated TraceLab component. The configuration entry is named from the setter method in the configuration file, with “set” removed from the beginning of the setter method name. The below configuration entry will be a FilePath and the entry will be named InspectionFile.

```
1 package components;
2
3 import cli.TraceLabSDK.Component.Config.FilePath;
4
5 /**
6  * @author David
7  */
8 public class Config {
9     private FilePath inspectionFile;
10
11
12     public void setInspectionFile(FilePath inspectionFile) {
13         this.inspectionFile = inspectionFile;
14     }
15
16     public FilePath getInspectionFile() {
17         return inspectionFile;
18     }
19 }
20
```

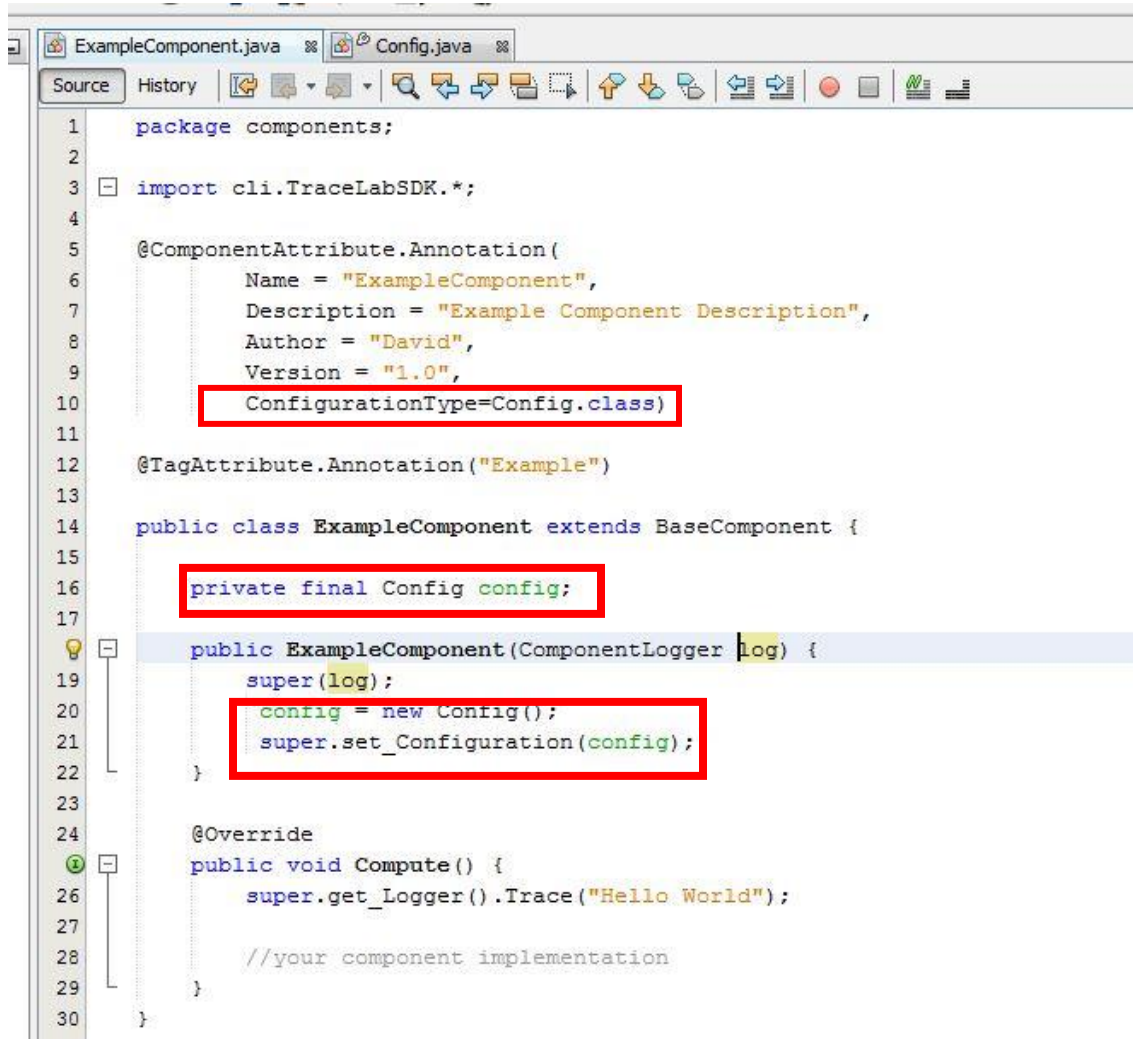


TraceLab Component Breakdown

Revision 1

Date: 08/24/2016

12. Below is an example of the modifications necessary to associate the configuration file with the TraceLab component.



```
1 package components;
2
3 import cli.TraceLabSDK.*;
4
5 @ComponentAttribute.Annotation(
6     Name = "ExampleComponent",
7     Description = "Example Component Description",
8     Author = "David",
9     Version = "1.0",
10    ConfigurationType=Config.class)
11
12 @TagAttribute.Annotation("Example")
13
14 public class ExampleComponent extends BaseComponent {
15
16    private final Config config;
17
18    public ExampleComponent(ComponentLogger log) {
19        super(log);
20        config = new Config();
21        super.set_Configuration(config);
22    }
23
24    @Override
25    public void Compute() {
26        super.get_Logger().Trace("Hello World");
27
28        //your component implementation
29    }
30 }
```

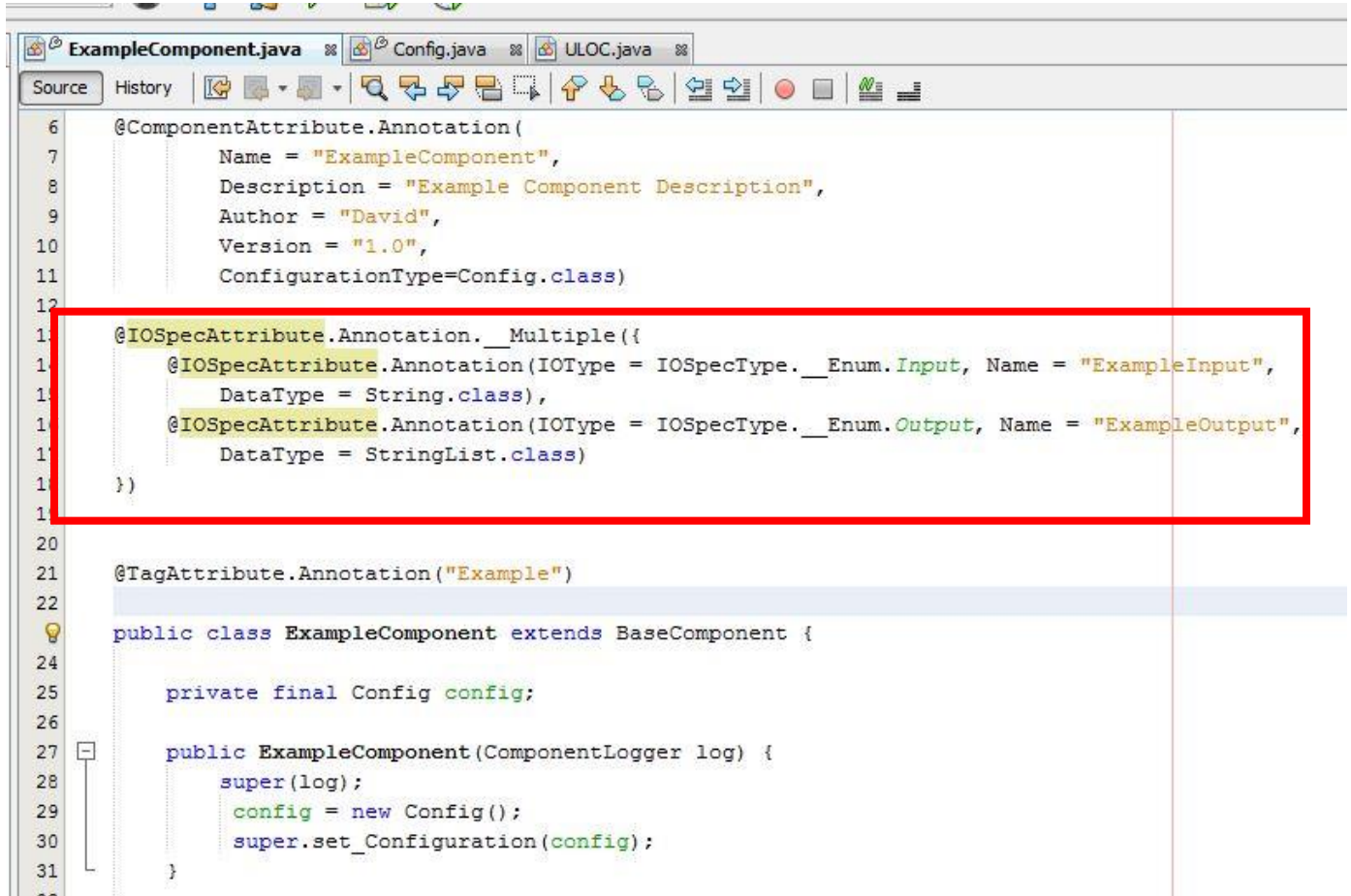
Component Inputs/Outputs

13. To create inputs and outputs to allow for storage and retrieval of data to the TraceLab workspace, add a section at the top of the TraceLab component wrapped in “@IOSpecAttribute.Annotation.__Multiple{“. Set the IOType, default input/output Name filed, and the DataType of the input/output. Below is an example of creating input and output fields of types String and StringList.

TraceLab Component Breakdown

Revision 1

Date: 08/24/2016



```
6 @ComponentAttribute.Annotation(  
7     Name = "ExampleComponent",  
8     Description = "Example Component Description",  
9     Author = "David",  
10    Version = "1.0",  
11    ConfigurationType=Config.class)  
12  
13    @IOSpecAttribute.Annotation.__Multiple(  
14        @IOSpecAttribute.Annotation(IOType = IOSpecType.__Enum.Input, Name = "ExampleInput",  
15            DataType = String.class),  
16        @IOSpecAttribute.Annotation(IOType = IOSpecType.__Enum.Output, Name = "ExampleOutput",  
17            DataType = StringList.class)  
18    })  
19  
20  
21    @TagAttribute.Annotation("Example")  
22  
23    public class ExampleComponent extends BaseComponent {  
24  
25        private final Config config;  
26  
27        public ExampleComponent(ComponentLogger log) {  
28            super(log);  
29            config = new Config();  
30            super.set_Configuration(config);  
31        }  
32    }
```


TraceLab Component Breakdown

Revision 1

Date: 08/24/2016

The screenshot displays the TraceLab software interface. On the left, the 'Components Library' pane shows a tree structure with 'ExampleComponent' selected. The main workspace shows a flow diagram with a 'Start' node, an 'ExampleComponent' node, and an 'End' node. A red rectangle highlights the 'ExampleComponent' configuration window. This window has two tabs: 'Input/Output' and 'Configuration'. The 'Input/Output' tab is active, showing a table with input and output mappings. The 'Configuration' tab shows fields for 'Node label', 'Component', 'Version', 'Author', and 'Description'. At the bottom, the 'Output' pane shows a log of messages.

Input	Mapped to	Type
ExampleInput	ExampleInput	String

Output	Output as	Type
ExampleOutput	ExampleOutput	StringList

Severity	Source	Message
Info	TraceLab.TraceLabApplication	Reading components from: C:\Program Files (x86)\COEST\TraceLab\Components\
Info	TraceLab.TraceLabApplication	Reading components from: C:\Users\David\Documents\TraceLab\Components\
Trace	ExampleComponent	Hello World