

# AsTeRICS v3.0 Plugin Development StringFormatter Example

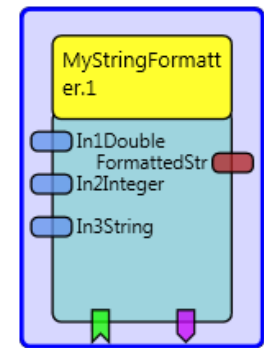
Martin Deinhofer, Department of Embedded Systems



# Agenda

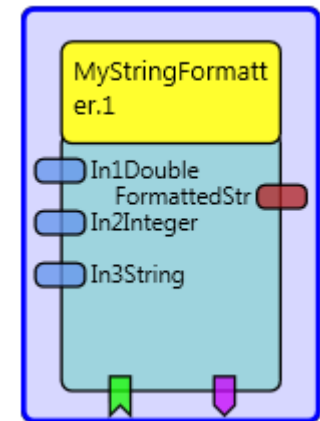
- Plugin characteristics
- Exercise definition
- Setup development environment
- ARE concept overview
- Plugin creation
- Build plugin
- Plugin activation and testing

- Is a functional block (component) with defined
  - **input ports:** receives data to be processed (e.g. face tracked coordinates)
  - **output ports:** send processed data (e.g. formatted string)
  - **event listener:** receive event and execute assigned action (e.g. left mouse click)
  - **event trigger:** send event to other functional blocks (e.g. time elapsed)
  - **properties:** Configure behaviour through property values



# Exercise definition

- Create AsTeRICS plugin:  
**MyStringFormatter**
- User can define string formats similar to printf function in C ([See Java-Class: Formatter](#))
- Input port values to be formatted
- Output port sends resulting formatted String
- Event Listener **sendFormattedStr**
  - formats and sends formatted string to port **formattedStr**
  - Triggers event **formattedStrSent**



Properties	
formatString	%1\$4.2f

# Exercise Example

- String format

„%1\$4.2f“

- Code example

```
String formatted=String.format("%1$4.2f",3.32643423);
```

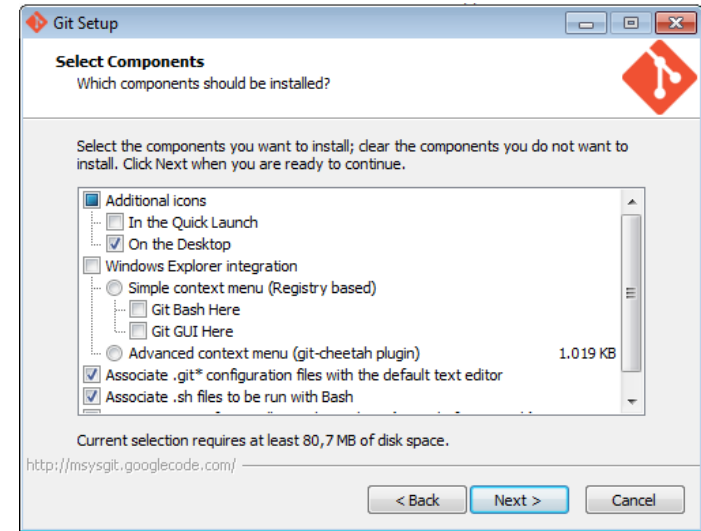
- Resulting output string

3.33

# Setup Development Environment git Installation

## 1. Install GIT for Windows\*

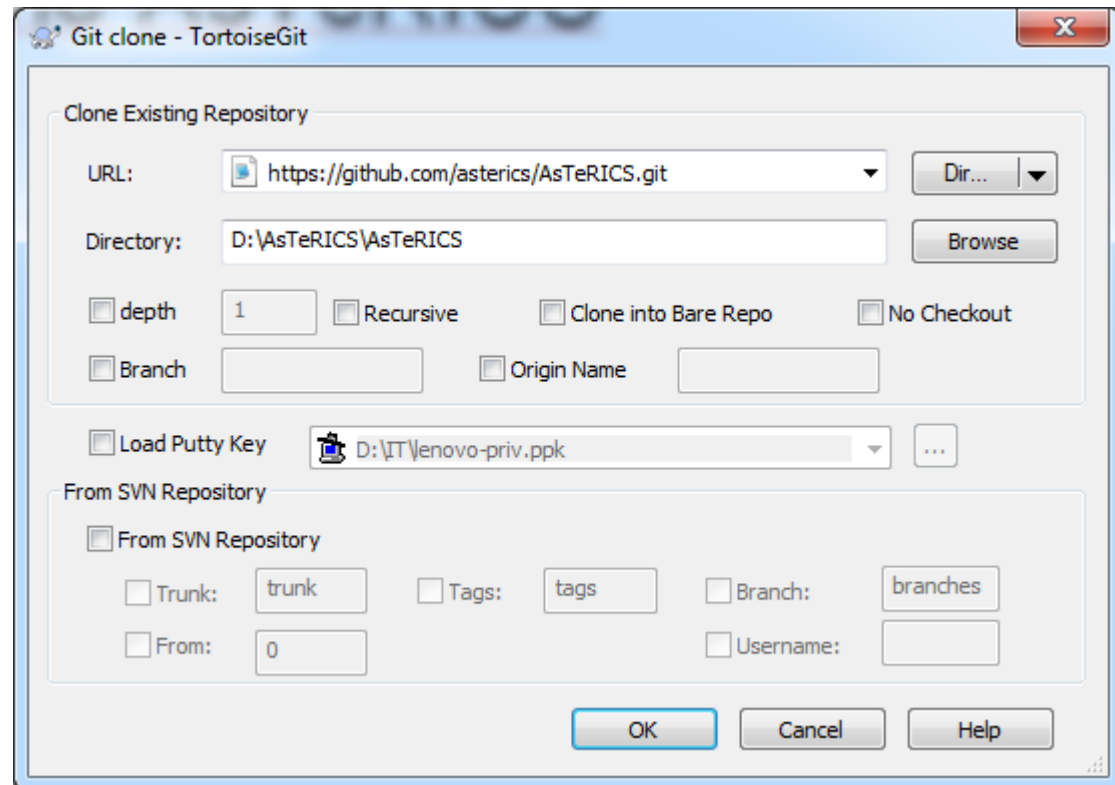
With the recommended settings in the screenshot (disable shell integration)



# Setup Development Environment

## Clone AsTeRICS

- Choose target folder for clone
- Rightclick in explorer folder and select
  - Git Clone



# Setup Development Environment JDK & Eclipse installation

1. Download and Install the [Java Development Kit](#) (JDK8, 32bit preferred)
2. Download and install/extract [Eclipse for Java developers](#)

In case of trouble, try to

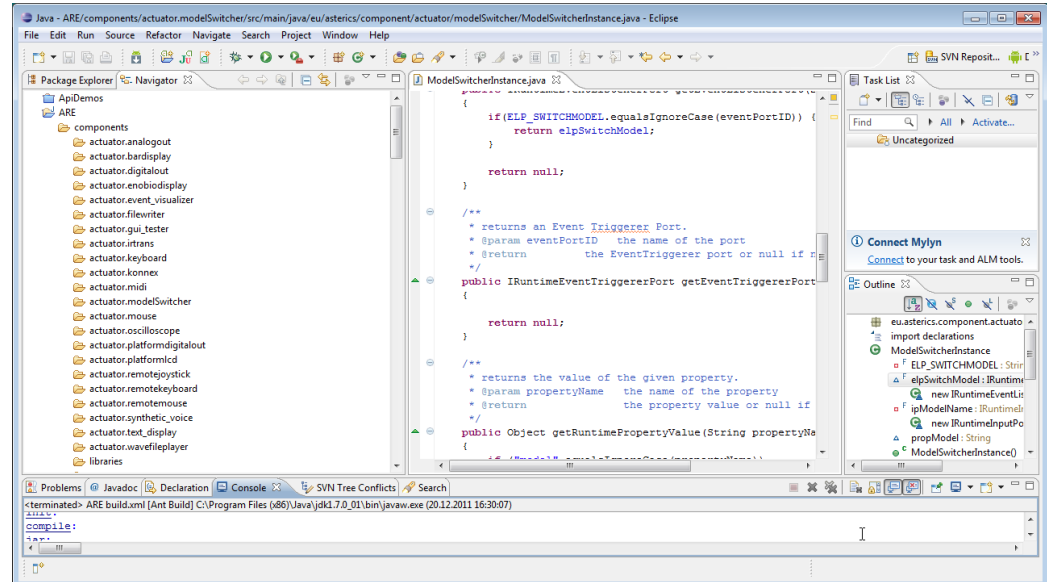
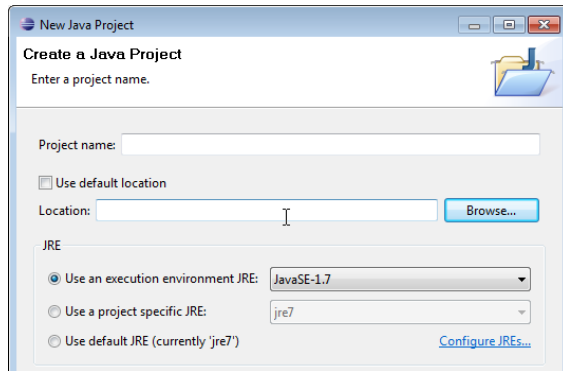
1. Set “**JAVA\_HOME**” to the JDK installation folder and add the JDK/bin path to the Environment Variable “**Path**”
2. and/or check the [Developer Manual](#)



# Setup Development Environment

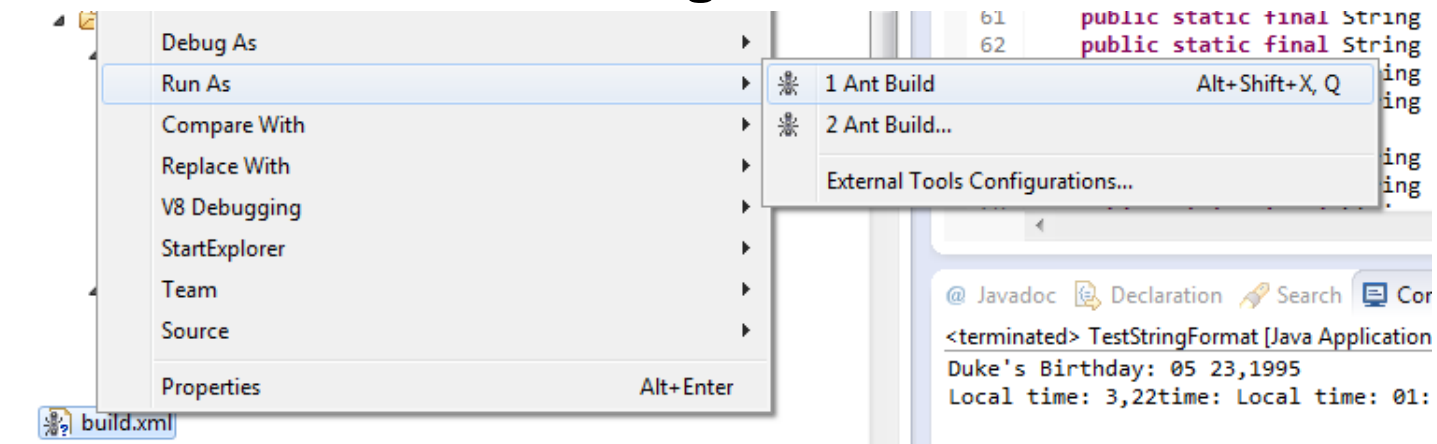
## Create Eclipse Project

- Start eclipse.exe
- Choose File -> New -> Java Project in the Eclipse main menu, disable the option “*Use default location*” and browse to the *ARE* subfolder
- You should get a project as shown in the right picture



Right click on **bulid.xml** in the root folder of the ARE project  
**Run As -> Ant Build**

**Note: ARE must not run during build**



Or from commandline:  
`ant buildAllNoClean`

# Setup Development Environment

## Run ARE

### 1. Either

1. Go to `bin/ARE` folder
2. Start ARE with debug\* output  
`start_debug.bat`

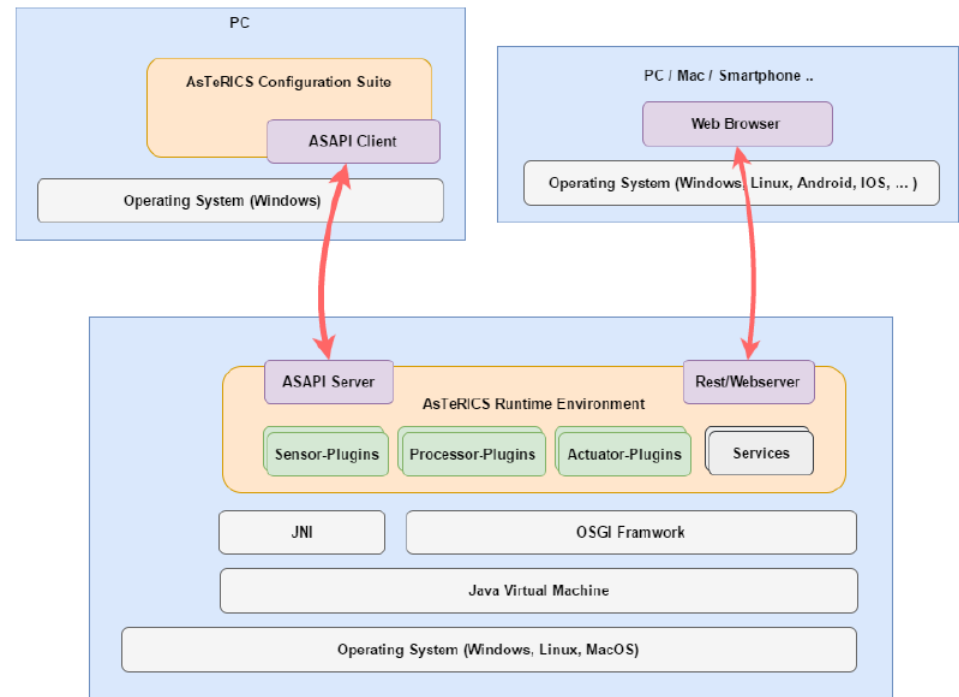
### 2. Or use ant target

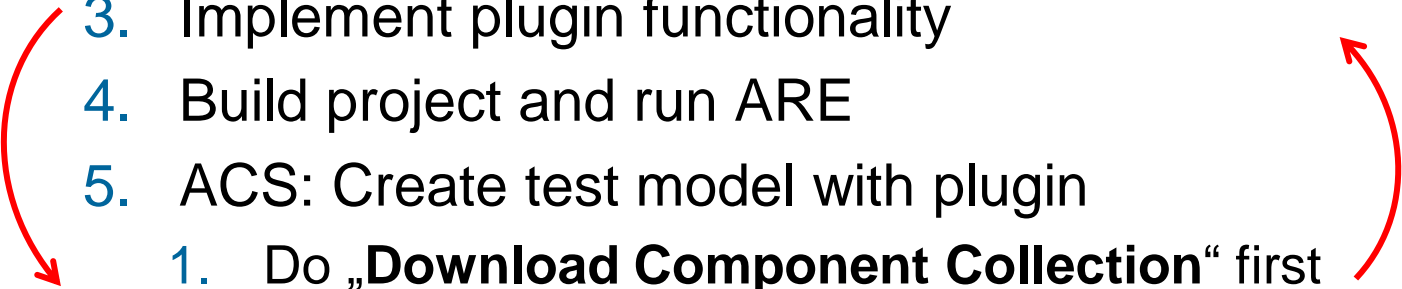
`ant run-debug`

- Important folders of source code repository
  - **ACS:** AsTeRICS Configuration Suite source code.
  - **ARE:** middleware and service layers and ARE components/plugins.
  - **ARE/components:** Source-code location of plugins, one folder per plugin. (**LICENSE** subfolder for involved licenses)
  - **APE:** tool and project template for creating standalone AsTeRICS-based SW packages
  - **bin:** subfolders where ARE, ACS and APE executable files are placed during the build flow.
  - **Documentation:** contains the User- and the Developer Manual, an OSKA manual and the license information for AsTeRICS source code and third party libraries.

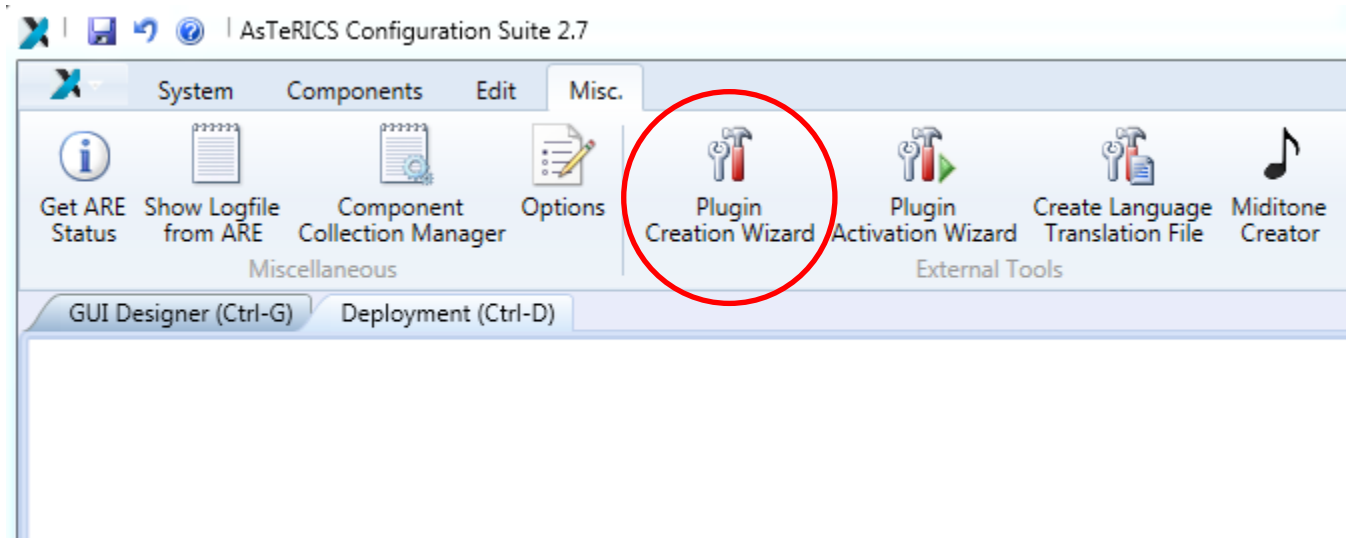
# ARE concept

- ARE middleware provides **Runtime Environment** for components/plugins
- Components based on **OSGi**
- Components can be
  - **Sensors:** Sense and create/send data (e.g. face tracker)
  - **Processors:** Process data (e.g. calculate moving average of data)
  - **Actuators:** Control environment (e.g. mouse cursor)

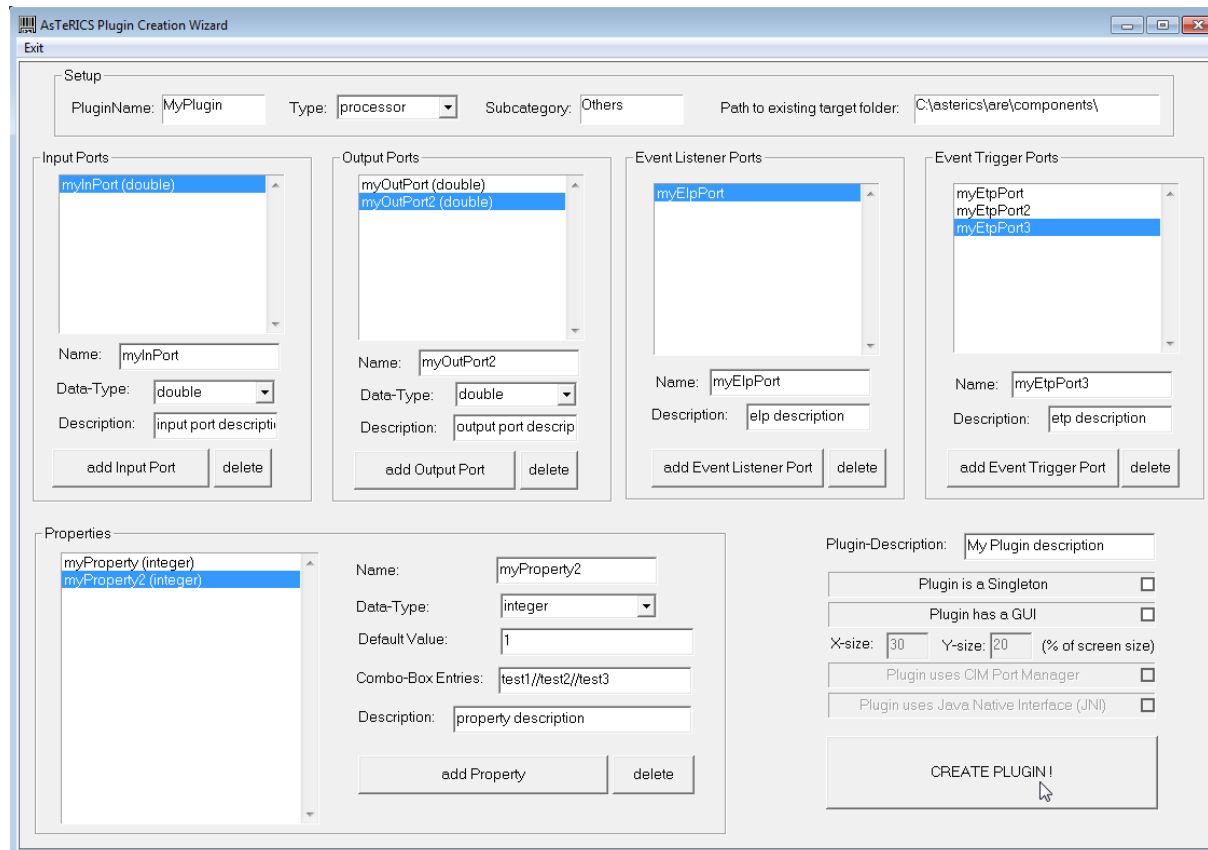


1. Plugin Creation Wizard → Generate plugin folder structure
    1. src folder
    2. ant build script
    3. Manifest file
    4. Bundle descriptor
    5. ...
  2. Add plugin to Eclipse project
  3. Implement plugin functionality
  4. Build project and run ARE
  5. ACS: Create test model with plugin
    1. Do „**Download Component Collection**“ first
- 
- Two red curved arrows are present. One starts at the left side of step 3 and points down towards the sub-step 1 of step 5. The other starts at the right side of step 5 and points up towards step 3, creating a loop around the sub-step.

# Plugin Creation Wizard



- Define characteristics of plugin
- Generate folder structure, source code stubs and build script



The screenshot shows the 'AsTeRICS Plugin Creation Wizard' window. It has a title bar with 'Exit' and standard window controls. The main area is divided into several sections:

- Setup:** Contains fields for 'PluginName' (MyPlugin), 'Type' (processor), 'Subcategory' (Others), and 'Path to existing target folder' (C:\asterics\are\components\).
- Input Ports:** A list box containing 'myInPort (double)'. Below it are fields for 'Name' (myInPort), 'Data-Type' (double), and 'Description' (input port description), along with 'add Input Port' and 'delete' buttons.
- Output Ports:** A list box containing 'myOutPort (double)' and 'myOutPort2 (double)'. Below it are fields for 'Name' (myOutPort2), 'Data-Type' (double), and 'Description' (output port description), along with 'add Output Port' and 'delete' buttons.
- Event Listener Ports:** A list box containing 'myElpPort'. Below it are fields for 'Name' (myElpPort), 'Description' (elp description), and 'add Event Listener Port' and 'delete' buttons.
- Event Trigger Ports:** A list box containing 'myEtpPort', 'myEtpPort2', and 'myEtpPort3'. Below it are fields for 'Name' (myEtpPort3), 'Description' (etp description), and 'add Event Trigger Port' and 'delete' buttons.
- Properties:** A list box containing 'myProperty (integer)' and 'myProperty2 (integer)'. Below it are fields for 'Name' (myProperty2), 'Data-Type' (integer), 'Default Value' (1), 'Combo-Box Entries' (test1/test2/test3), and 'Description' (property description), along with 'add Property' and 'delete' buttons.
- Plugin-Description:** A text field containing 'My Plugin description'.
- Options:** Four checkboxes: 'Plugin is a Singleton', 'Plugin has a GUI', 'Plugin uses CIM Port Manager', and 'Plugin uses Java Native Interface (JNI)'. The first two are unchecked, and the last two are checked.
- Buttons:** A large 'CREATE PLUGIN!' button at the bottom right.



# Plugin Creation Wizard

## Common Parameter

- PluginName: „MyStringFormatter“  
(CamelCase notation)
- Type: „processor“ (others: sensor, actuator)
- Subcategory: „Event and String Processing“  
(See existing ACS components)
- Path to existing target folder:  
<AsTeRICS ROOT>/ARE/components

Setup

PluginName:	<input type="text" value="StringFormatter"/>	Type:	<input type="text" value="processor"/>	Subcategory:	<input type="text" value="Event and String"/>	Path to existing target folder:	<input type="text" value="C:\asterics\are\components\"/>
-------------	--	-------	--	--------------	---	---------------------------------	--

# Plugin Creation Wizard

## Input Ports

Input Ports

in1Double (double)
in2Integer (integer)
in3String (string)

Name:

Data-Type:

Description:

# Plugin Creation Wizard

## Output Ports

Output Ports

formattedStr (string)

Name: formattedStr

Data-Type: string

Description: ng formatted string

add Output Port delete

# Plugin Creation Wizard

## Event Listener

Event Listener Ports

sendFormattedStr
------------------

Name:

Description:

# Plugin Creation Wizard

## Event Trigger

Event Trigger Ports

formattedStrSent
------------------

Name:

Description:

# Plugin Creation Wizard Properties

Properties

formatString (string)	Name:	<input type="text" value="formatString"/>
	Data-Type:	<input type="text" value="string"/>
	Default Value:	<input type="text" value="%1\$4.2f"/>
	Combo-Box Entries:	<input type="text"/>
	Description:	<input type="text" value="String defining format of input port val"/>
		<input type="button" value="add Property"/> <input type="button" value="delete"/>

# Plugin Creation Wizard

## Generate Plugin

Plugin-Description:

☐ Plugin is a Singleton

☐ Plugin has a GUI

X-size:  Y-size:  (% of screen size)

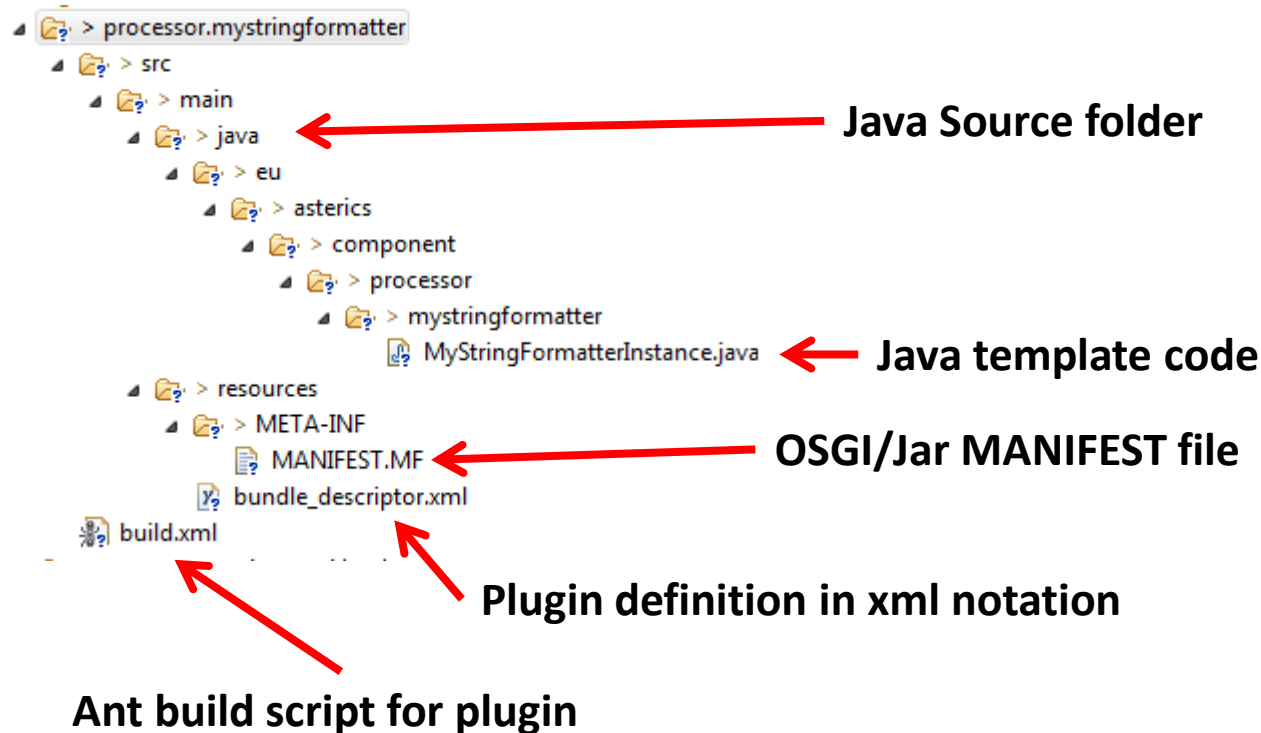
☐ Plugin uses CIM Port Manager

☐ Plugin uses Java Native Interface (JNI)

 **Click to generate**

# Plugin Creation Wizard

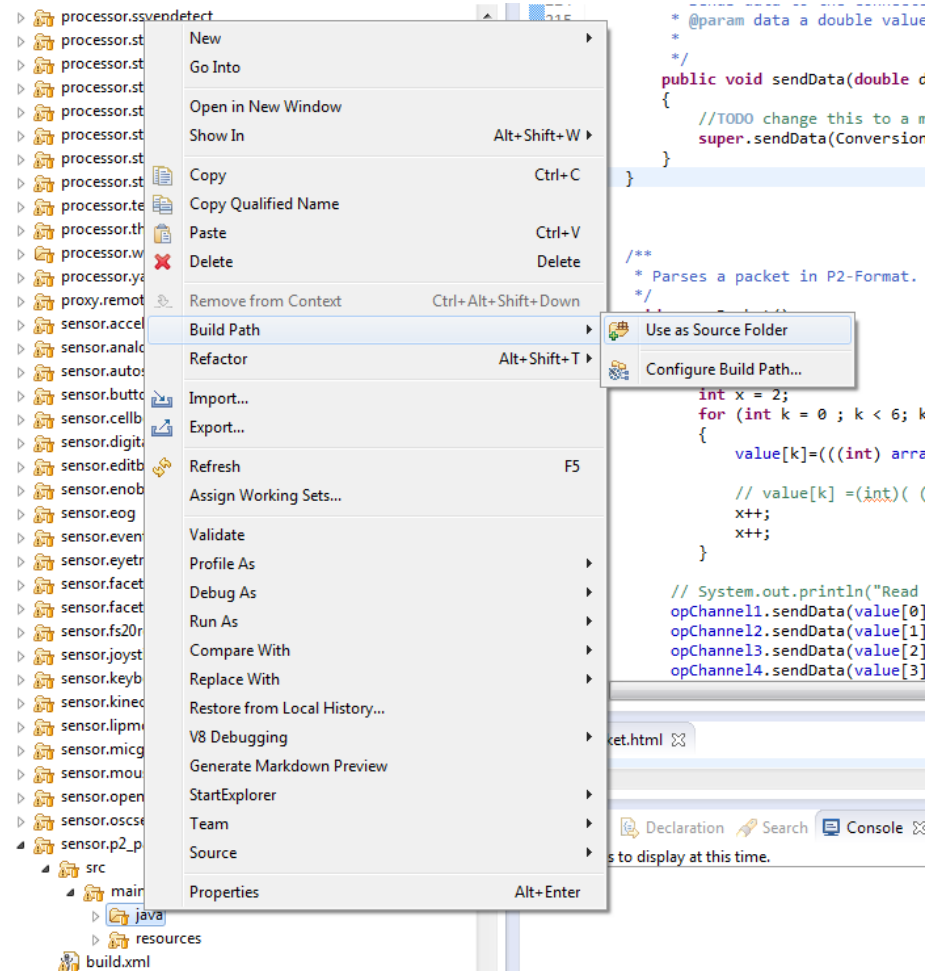
## Created folder structure





# Source folder to Build Path

Add Java source folder to Build Path



# Build Plugin and Run ARE

The target run-debug both builds the project and runs the ARE:

**ant run-debug**

**Note: Close running ARE before**

# Start ACS/WebACS and test plugin

1. Go to bin/ACS folder
2. Start ACS.exe or press F8 (to edit current model)
3. Click „Connect to ARE“
4. Click „Download Component Collection“
5. Add new plugin from Components tab

# Member variables

```
final IRuntimeOutputPort opFormattedStr = new  
DefaultRuntimeOutputPort();  
// Usage of an output port e.g.:  
opMyOutPort.sendData(ConversionUtils.intToBytes(10));
```

```
final IRuntimeEventTriggererPort etpFormattedStrSent = new  
DefaultRuntimeEventTriggererPort();  
// Usage of an event trigger port e.g.:  
etpMyEtPort.raiseEvent();
```

```
String propFormatString = "%1$4.2f";
```

```
// declare member variables here  
//Hold values of incoming input port data  
private Double in1Double;  
private Long in2Integer;  
private String in3String;
```

# Property setting & getting

```
/**
 * returns the value of the given property.
 * @param propertyName the name of the property
 * @return the property value or null if not found
 */
public Object getRuntimePropertyValue(String propertyName)
{
    if ("formatString".equalsIgnoreCase(propertyName))
    {
        return propFormatString;
    }
    return null;
}

/**
 * sets a new value for the given property.
 * @param propertyName the name of the property
 * @param newValue the desired property value or null if not found
 */
public Object setRuntimePropertyValue(String propertyName, Object newValue)
{
    if ("formatString".equalsIgnoreCase(propertyName))
    {
        final Object oldValue = propFormatString;
        propFormatString = (String)newValue;
        return oldValue;
    }

    return null;
}
```

# Converting incoming port data

```
/**
 * Input Ports for receiving values.
 */
private final IRuntimeInputPort ipIn1Double = new DefaultRuntimeInputPort()
{
    public void receiveData(byte[] data)
    {
        //Convert incoming data to a double value.
        in1Double = ConversionUtils.doubleFromBytes(data);
    }
};
private final IRuntimeInputPort ipIn2Integer = new DefaultRuntimeInputPort()
{
    public void receiveData(byte[] data)
    {
        //Convert incoming data to a Long (not int) value, because Formatter class expects Long value
        instead of int.
        in2Integer=new Long(ConversionUtils.intFromBytes(data));
    }
};
private final IRuntimeInputPort ipIn3String = new DefaultRuntimeInputPort()
{
    public void receiveData(byte[] data)
    {
        //Convert incoming data to a String value.
        in3String=ConversionUtils.stringFromBytes(data);
    }
};
```

# Formatting & Sending Data

```
/**
 * Formats and sends the resulting formatted string to the output port.
 */
private void formatAndSendString() {
    //get current format string
    String curFormatString=(String)getRuntimePropertyValue("formatString");
    //Execute actual formatting of string
    String formattedString=String.format(curFormatString, in1Double, in2Integer, in3String);
    //Convert formatted string to byte[] and send it to the output port
    opFormattedStr.sendData(ConversionUtils.stringToBytes(formattedString));
    //Inform others, trigger event
    etpFormattedStrSent.raiseEvent();
}
```

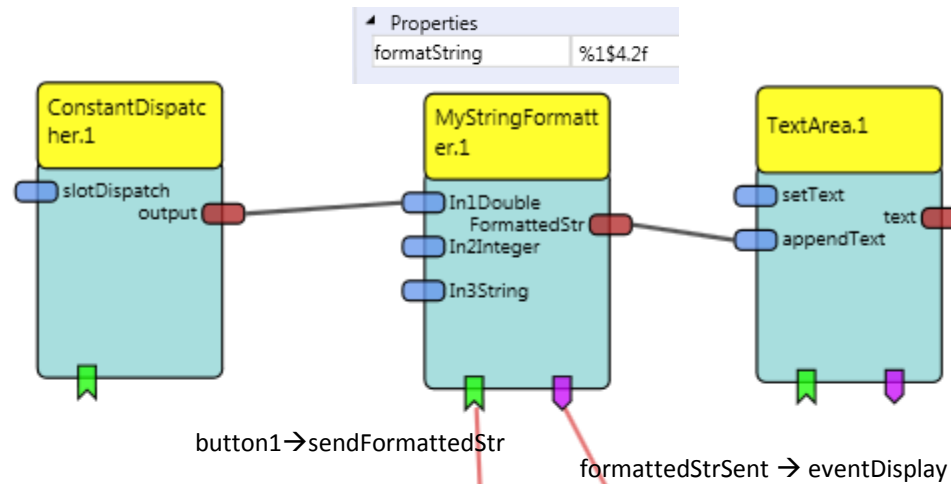
# Implement Event Listener

```
/**
 * Event Listener Ports.
 */
final IRuntimeEventListenerPort elpSendFormattedStr =
new IRuntimeEventListenerPort()
{
    public void receiveEvent(final String data)
    {
        formatAndSendString();
    }
};
```



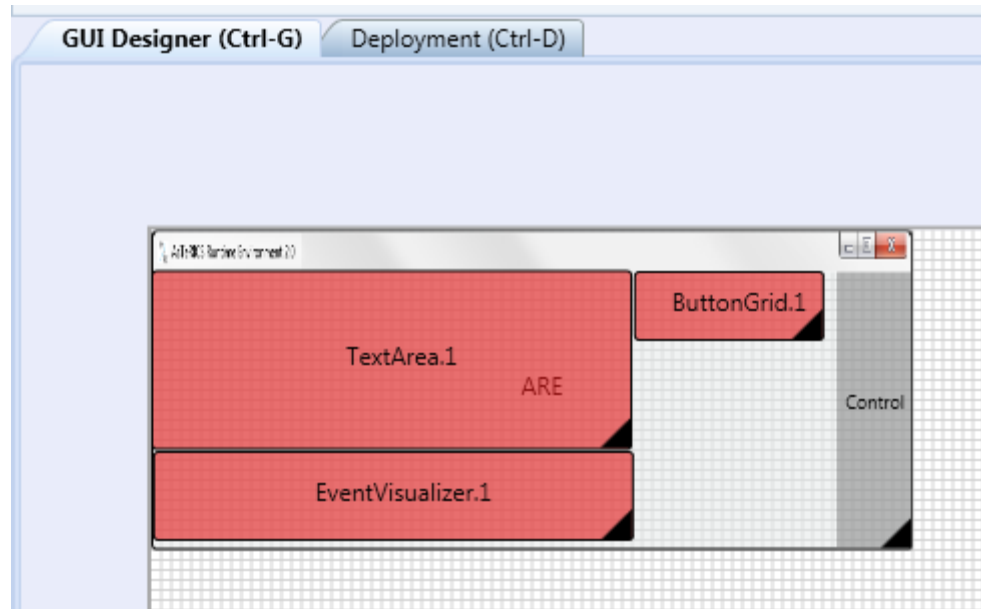
# Test model

Properties	
number	20
delay	0
slot1	3.32643423
slot2	0
slot3	0
slot4	0
slot5	0
slot6	0
slot7	0
slot8	0
slot9	0
slot10	0
slot11	0
slot12	0
slot13	0
slot14	0
slot15	0
slot16	0
slot17	0
slot18	0
slot19	0
slot20	0
autosendSlot	1



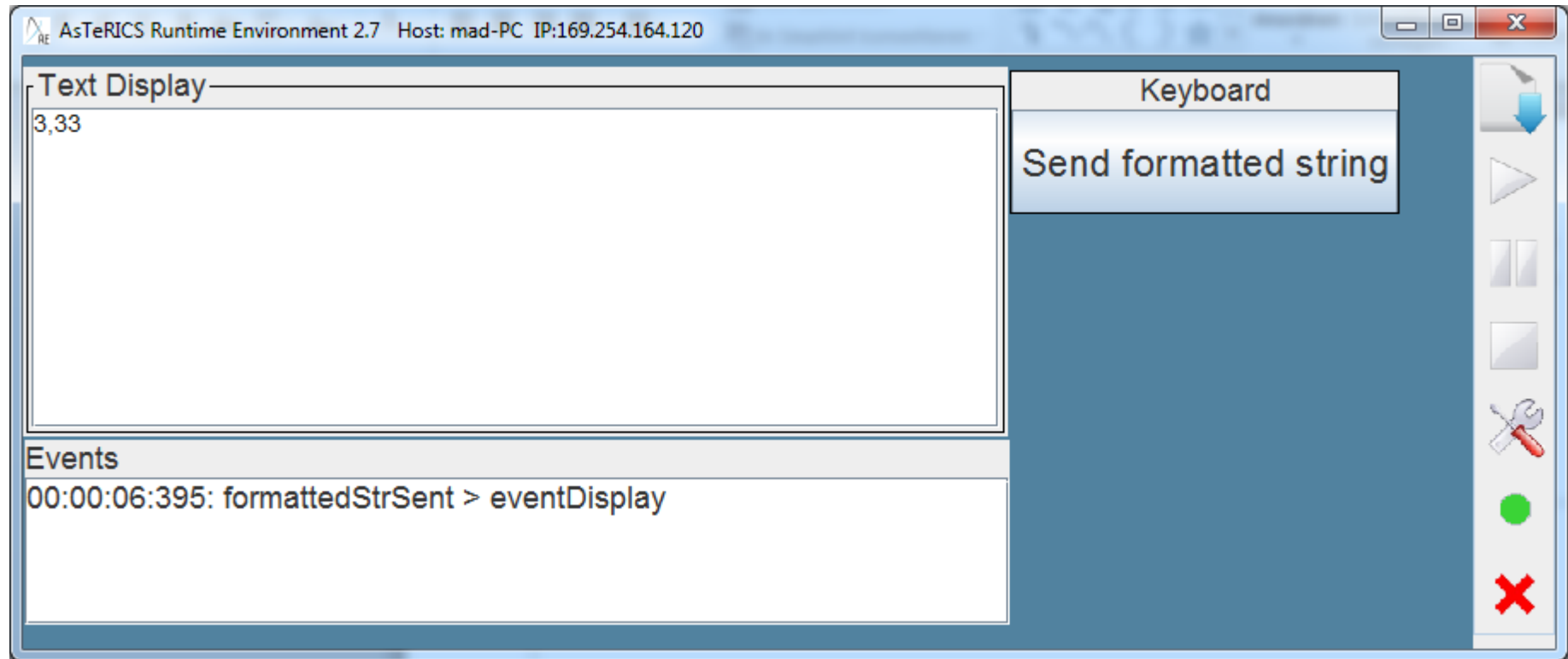
Properties	
caption	Keyboard
horizontalOrientation	<input type="checkbox"/>
textColor	default
backgroundColor	default
borderColor	default
borderThickness	2
selectionFrameColor	default
selectionFrameThickness	4
displayGUI	<input checked="" type="checkbox"/>
buttonCaption1	Send formatted string

# Test model – GUI Designer



# Model Result

Click on button „Send formatted string“



- For some plugins you might need to read the contents of some files (configuration files, images,...) or write to a file.
- Use [ResourceRegistry API](#)

## Example

- Get URI of image resource

URI myURI =

```
ResourceRegistry.getInstance().getResource("pictures/slide7.jpg", RES_TYPE.DATA);
```

- Get contents of text file in data/scripts folder

String contents =

```
ResourceRegistry.getInstance().getResourceContentAsString("scripts/script.js", RES_TYPE.DATA);
```

# Plugin License Files

- The author of a plugin must provide the license files of the plugin
- The license of self-authored code
- The license of used third-party code
- Put all the files into the folder

**ARE/components/<mycomponent>/LICENSE**

- Use this [file name convention](#)

# Plugin Help File

- Provide a help which is shown when pressing F1
- Copy another help file and save it to the appropriate location:

[Documentation/ACS-Help/HTML/Plugins](#)

- The name of the file must be the name of the plugin

# Modifying a Plugin

- You must edit the file

`AsTeRICS/ARE/components/<mycomponent>/src/main/  
resources/bundle_descriptor.xml`

- and update your code accordingly