AsTeRICS v3.0 Plugin Development StringFormatter Example

Martin Deinhofer, Department of Embedded Systems

FH University of Applied Sciences TECHNIKUM WIEN

academy

MA23 Wirtschaft, Arbeit Statistik

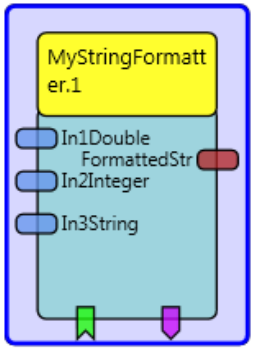
Stadt+Wien

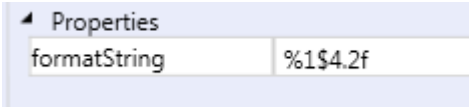
Agenda

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

AsTeRICS Plugin

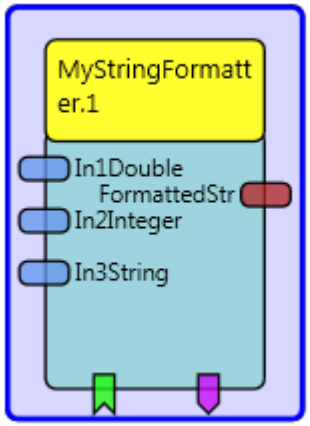
- Is a functional block (component) with defined
  - **input ports**: receives data to be processed (e.g. face tracked coordinates)
  - **output ports**: send processed data (e.g. formatted string)
  - **event listener**: receive event and execute assigned action (e.g. left mouse click)
  - **event trigger**: 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`**



Exercise Example

- String format

„%1$4.2f“

- Code example

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

- Resulting output string

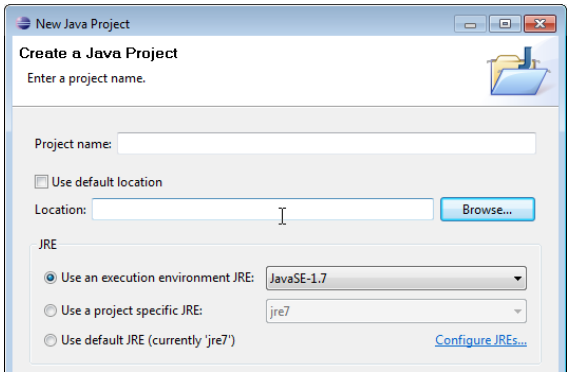
```
3.33
```

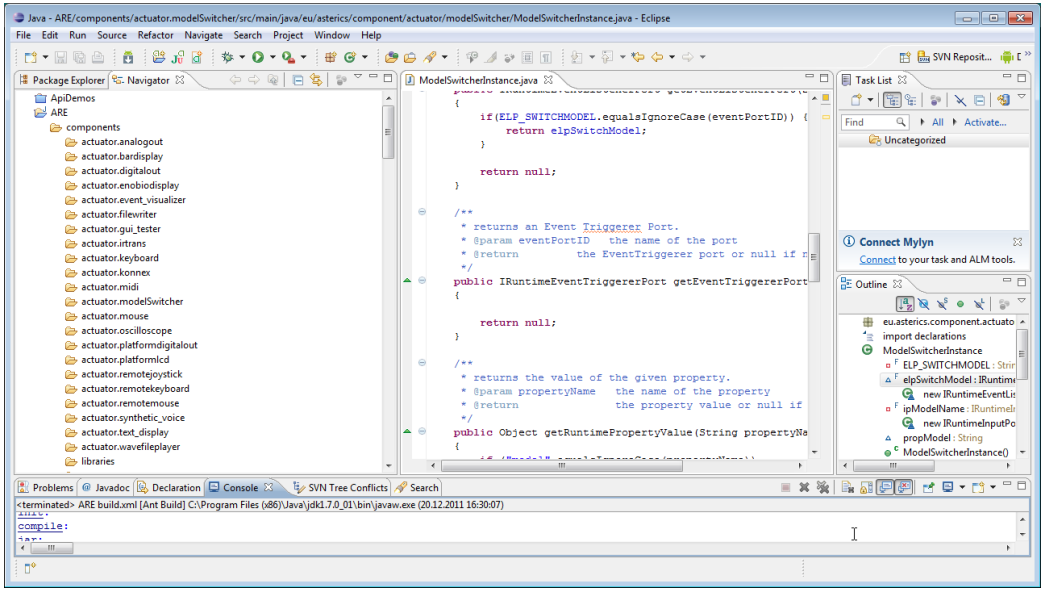
Setup Development Environment

1. Install the git cmd-line tool
2. Follow the build instructions of the AsTeRICS repository

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



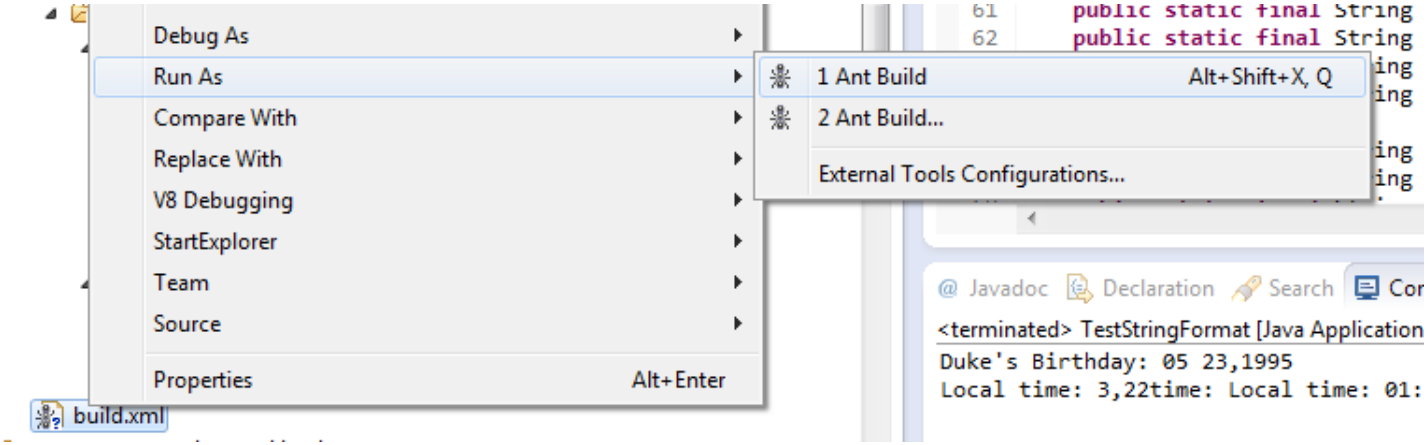


Build ARE

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
   ```

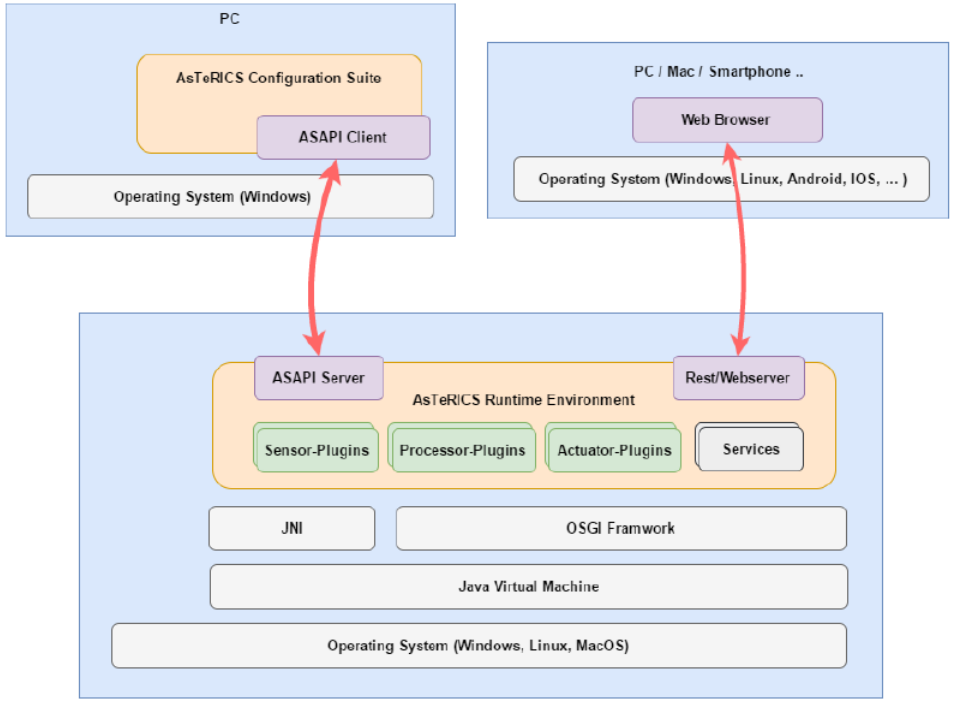

* Other start files are: ARE.exe or start.bat

ARE folder structure

- 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.

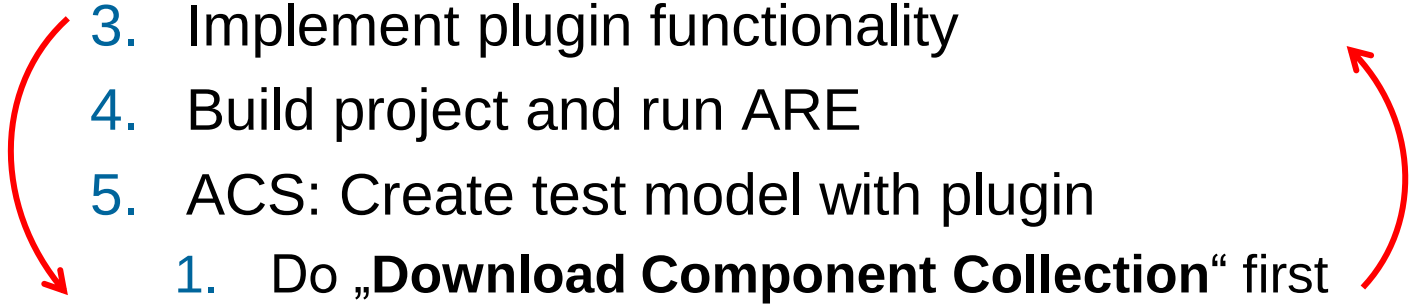
ARE concept

- 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)

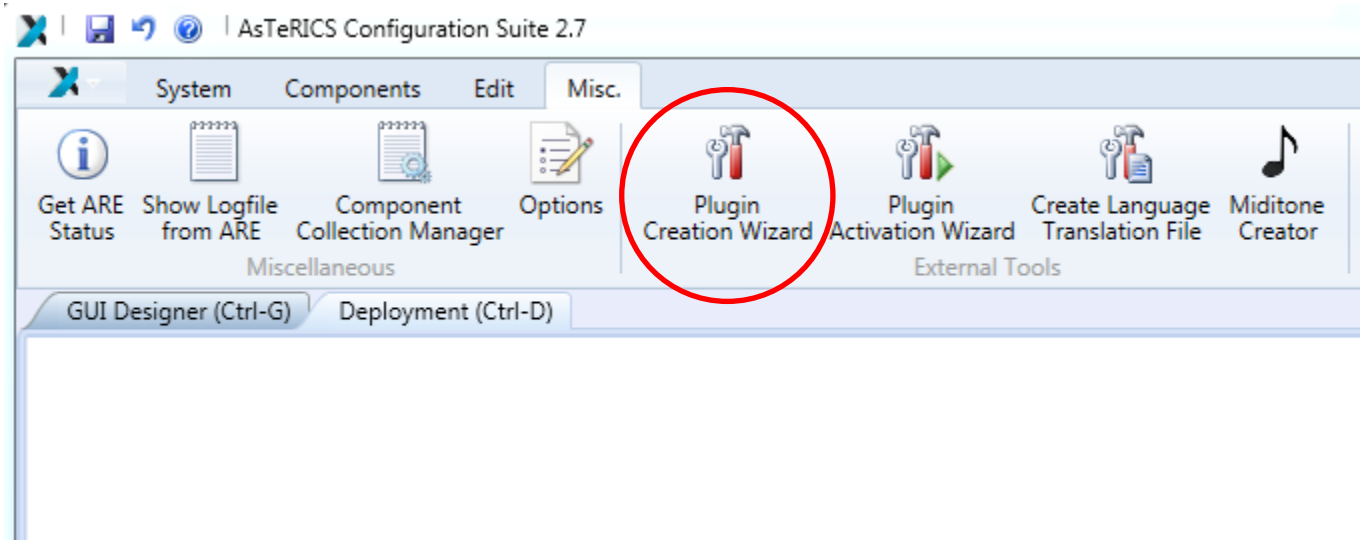


Plugin Development Workflow

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



Plugin Creation Wizard



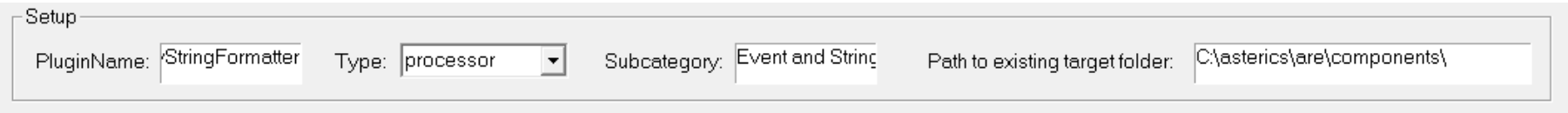
Plugin Creation Wizard

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

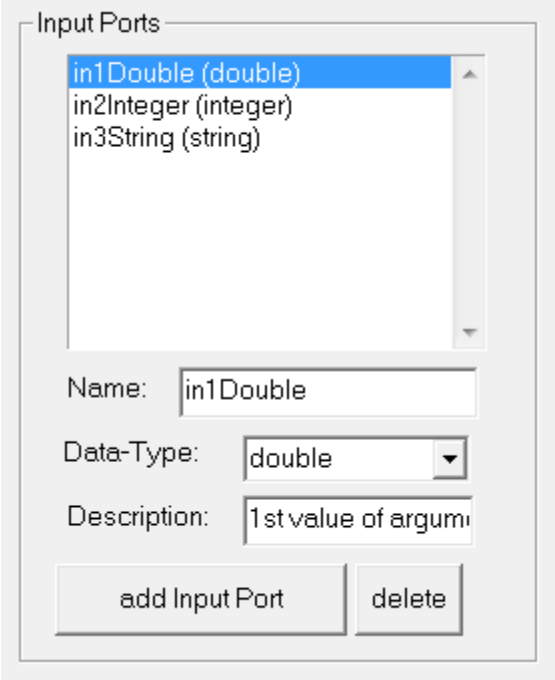
- Finally „CREATE PLUGIN“

Plugin Creation Wizard Common Parameter

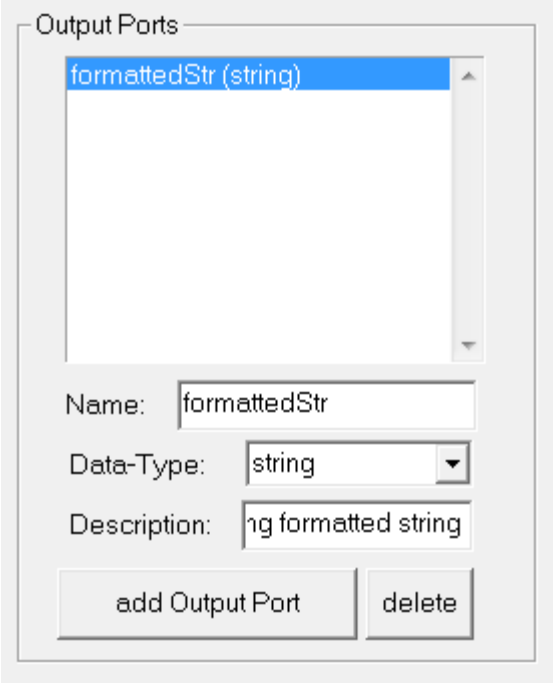
- PluginName: „`MyStringFormatter`“ (CamelCase notation)
- Type: „`processor`“ (others: `sensor`, `actuator`)
- Subcategory: „`Event and String Processing`“ (See existing ACS components)
- Path to existing git target folder: `<AsTeRICS ROOT>/ARE/components`



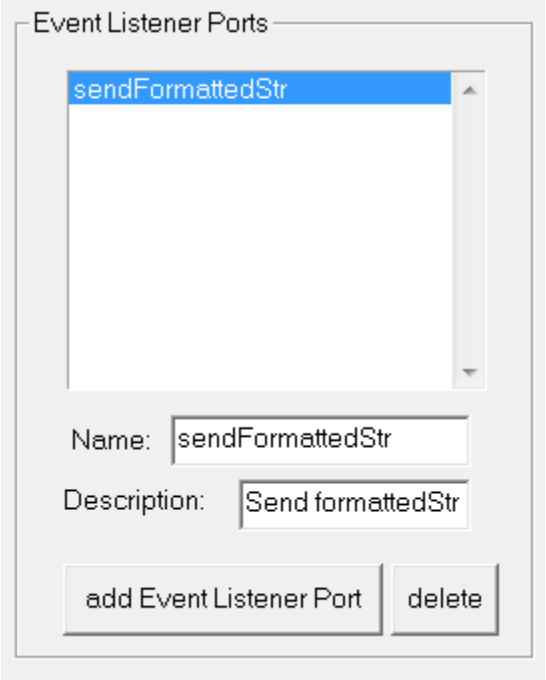
Plugin Creation Wizard
Input Ports



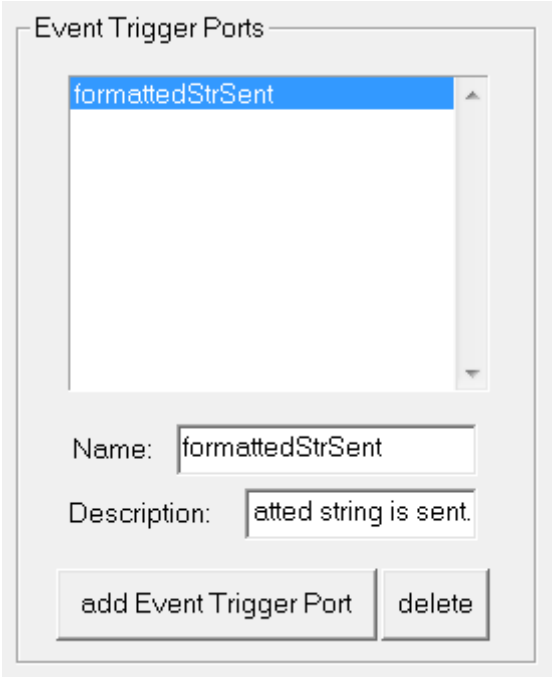
Plugin Creation Wizard Output Ports



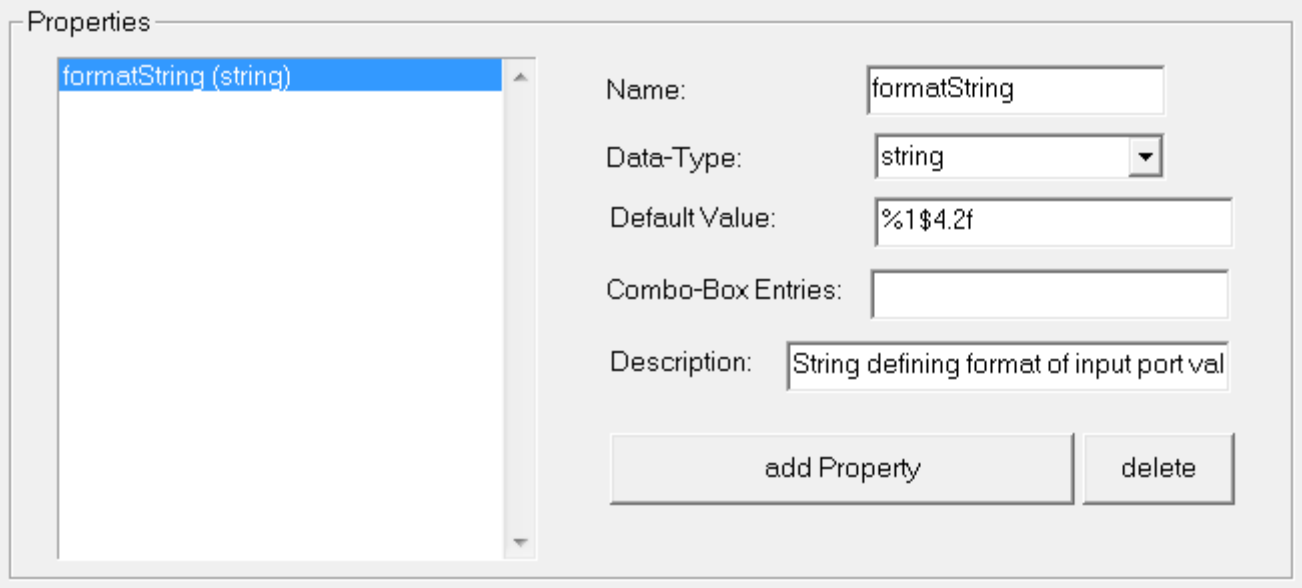
Plugin Creation Wizard
Event Listener



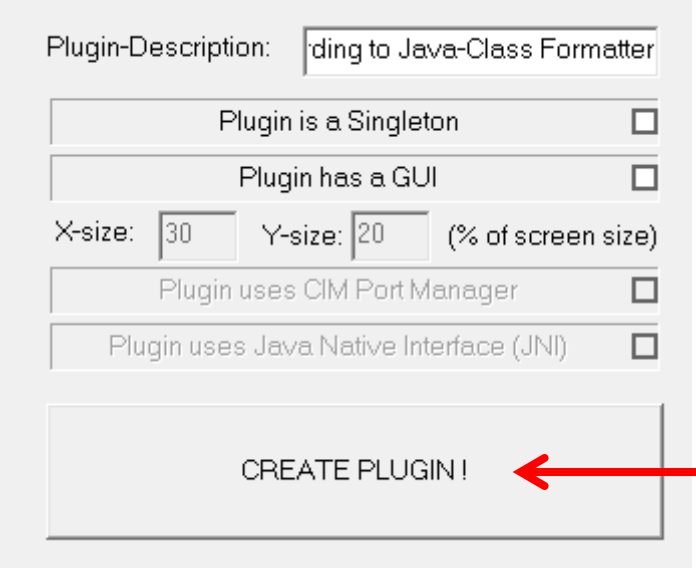
Plugin Creation Wizard
Event Trigger



Plugin Creation Wizard Properties

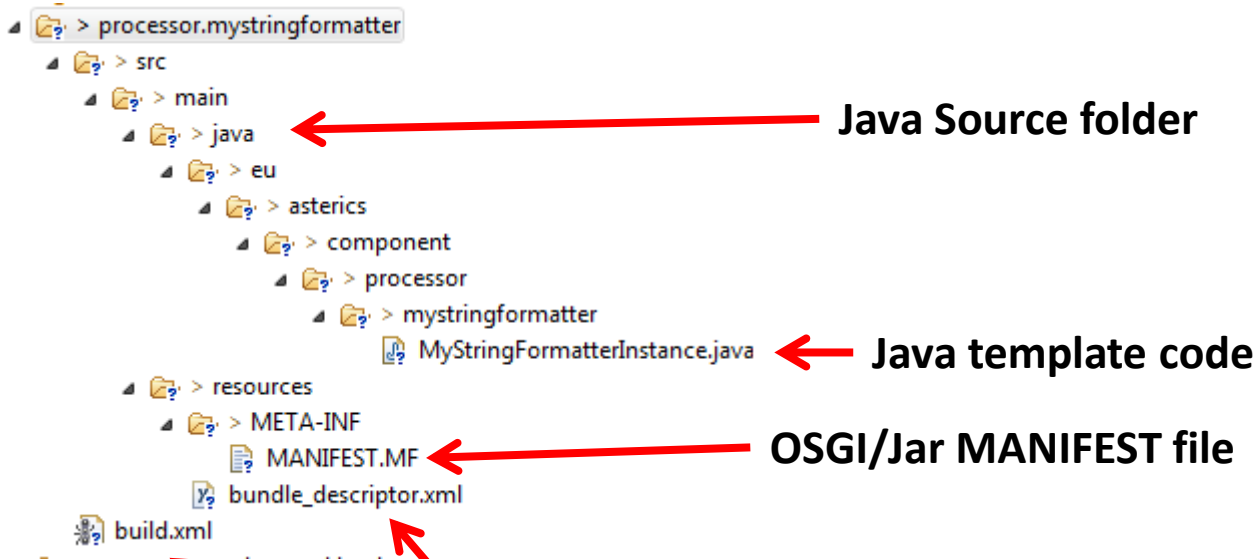


Plugin Creation Wizard
Generate Plugin



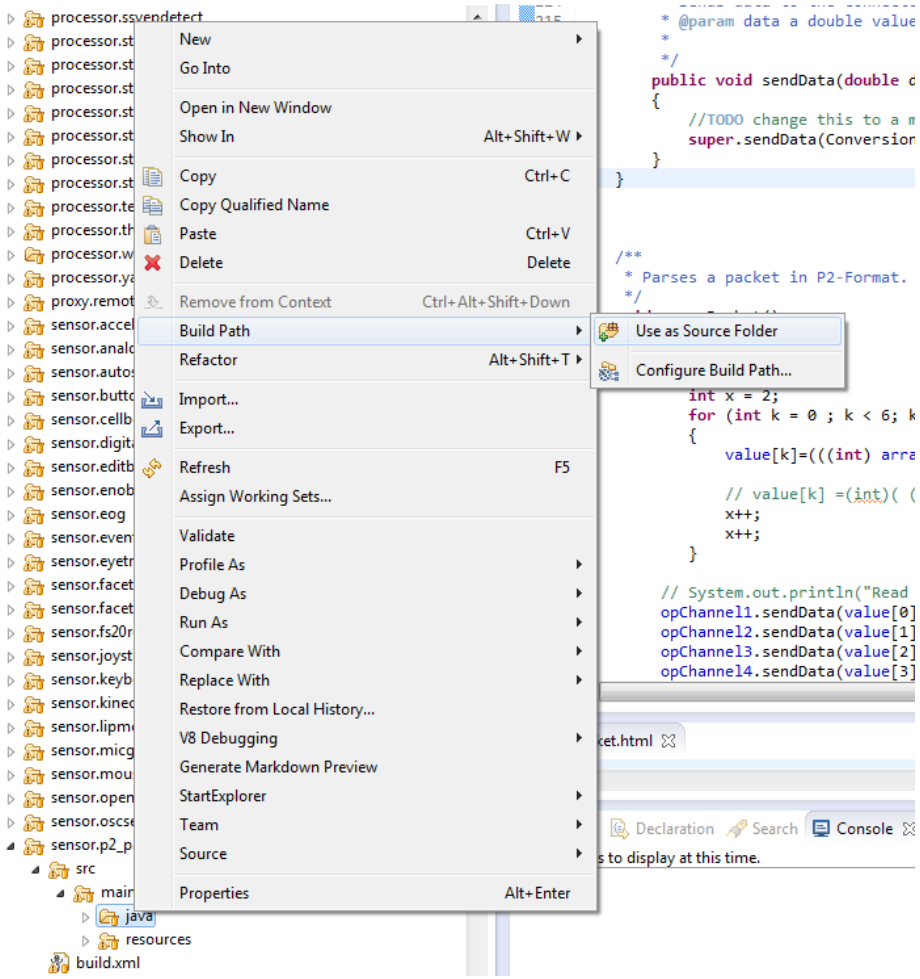
Click to generate

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

```java
/**
 * 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 incomding data to a String value.
  in3String=ConversionUtils.stringFromBytes(data);
}
};
```

Formatting & Sending Data

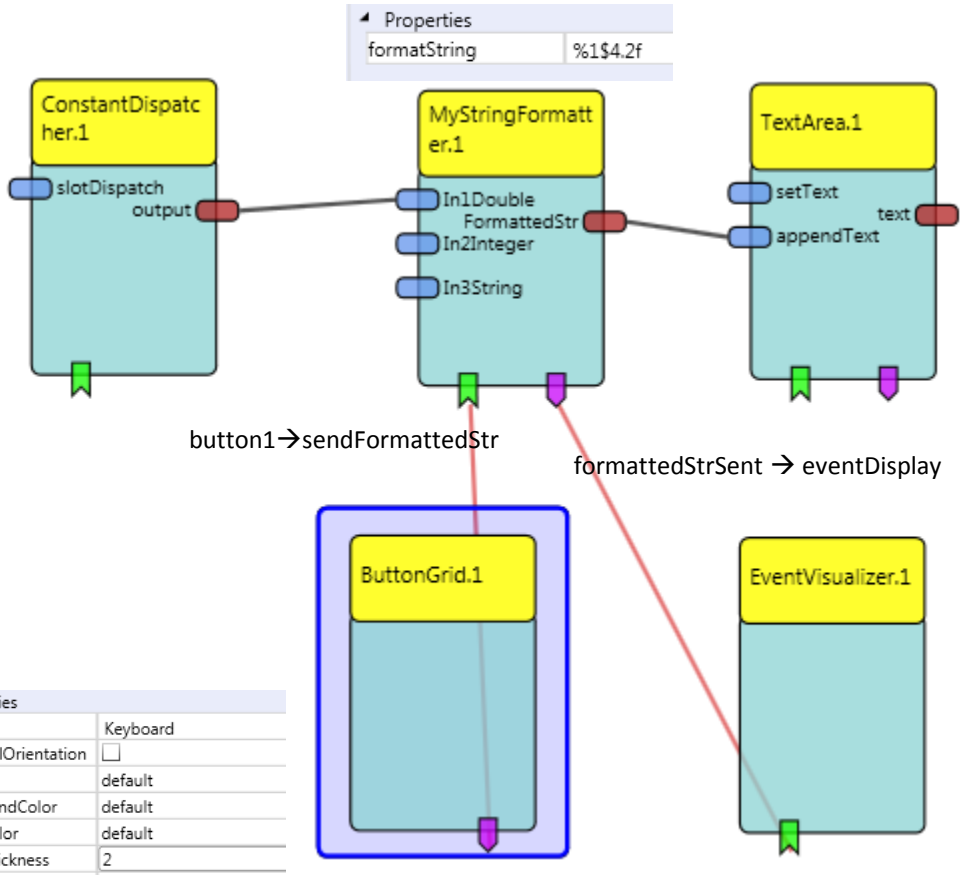
```
/**
  * Formats and sends the resulting foramtted string to the output port.
  */
 private void formatAndSendString() {
   //get current format string
   String curFormatString=(String)getRuntimePropertyValue("formatString");
   //Execute actual formatting of string
   String formattedString=String.format(curFormatString, 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 Listerner Ports.
  */
final IRuntimeEventListenerPort elpSendFormattedStr =
new IRuntimeEventListenerPort()
{
public void receiveEvent(final String data)
{
  formatAndSendString();
}
};
```

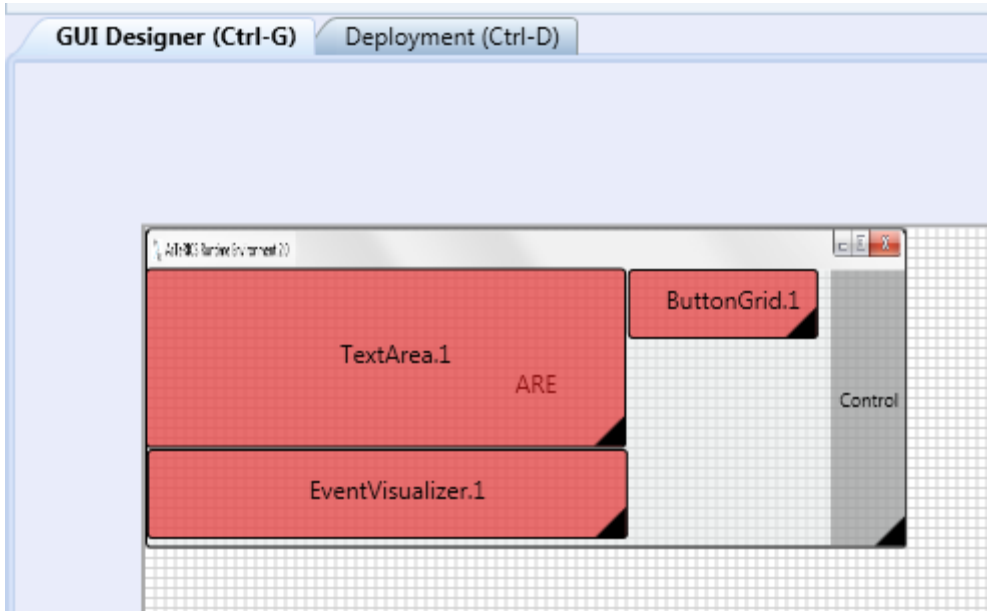
Test model

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



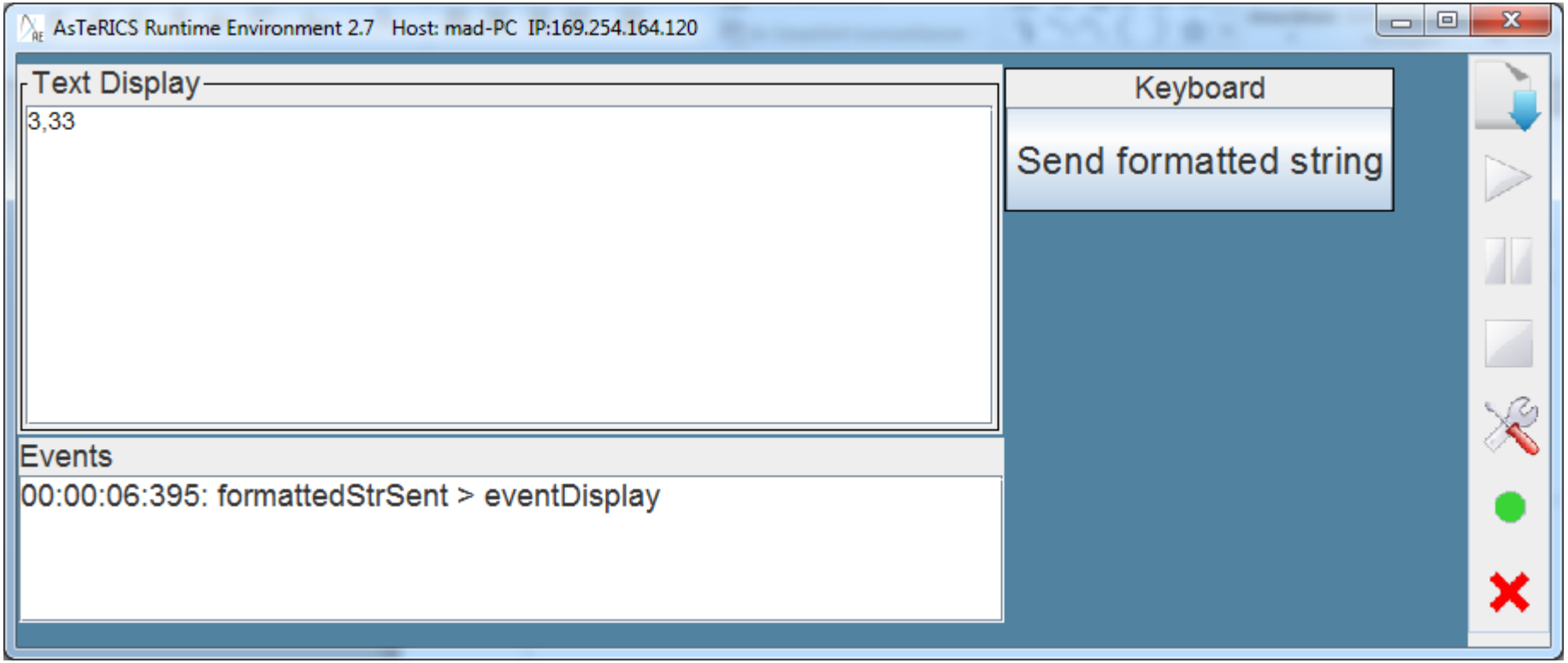
Properties	
caption	Keyboard
horizontalOrientation	☐
textColor	default
backgroundColor	default
borderColor	default
borderThickness	2
selectionFrameColor	default
selectionFrameThickne	4
displayGUI	☒
buttonCaption1	Send formatted string

Test model – GUI Designer



Model Result

Click on button „Send formatted string“



Plugins with Resources

- 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